Machine learning applied to the computation of chemical source terms in reacting flows

Xi Chen

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à MINES Paris

Apprentissage automatique appliqué au calcul des termes sources chimiques pour la simulation d'écoulements réactifs

Machine learning applied to the computation of chemical source terms in reacting flows

Soutenue par
Xi CHEN
Le 22 Jan 2024

Dirigée par
Florent DI MEGLIO

École doctorale n°621
ISMME- - Ingénierie des Systèmes, Matériaux, Mécanique, Énergétique

Spécialité
Mathématiques et automatique

Composition du jury :

Mathilde MOUGEOT Professeure, ENS-Paris Saclay	*Présidente du jury*
Alessandro PARENTE Professeur, Université Libre de Bruxelles	*Rapporteur*
Luc VERVISCH Professeur, CORIA-INSA Rouen	*Rapporteur*
Luis Fernando FIGUEIRA DA SILVA Chargé de recherche, Institut PPRIME	*Examinateur*
Cédric MEHL Ingénieur de recherche, IFPEN	*Examinateur*
Florent DI MEGLIO Maître de conférence, Mines Paris PSL	*Directeur de thèse*

MINES PARIS | PSL

Machine learning applied to the computation of chemical source terms in reacting flows

Xi CHEN

Director: Dr. Florent DI MEGLIO
Co-supervisor: Dr. Cédric MEHL
Co-supervisor: Dr. Thibault FANEY

Centre Automatique et Systèmes (CAS)
Mines Paris - Université PSL
IFP Energies Nouvelles
Paris, France

Abstract

This thesis deals with the acceleration of chemical kinetics calculations in CFD simulations by relying of machine learning methods. The principle is to replace the resolution of the chemistry in the calculations by an equivalent machine learning model, whose evaluation is much faster than the resolution of the original system of differential equations. The thesis work first focuses on the study of simplified combustion cases (0-dimensional system without transport terms). We propose a data processing framework enabling the construction of an accurate surrogate model. We provide an in-depth comparison between the baseline resolution and the resolution relying on the learned model. Then, models combined with dimension reduction technique and continuous flow learning are developed and applied to complex chemistry cases, providing a first step towards variant time steps prediction approaches. Finally, our approach is applied to a multidimensional case with transport terms, notably including turbulence.

The first contribution of this thesis is to extend the investigation of potential algorithms for predicting states in highly complex thermochemical systems. These algorithms are assessed, starting from basic benchmark cases such as 0D auto-ignition problems, and extended to multi-dimensional cases involving turbulent transport phenomena. We carry out the chemistry computation for fuels with different complexities. After the verification of a basic learning workflow, new deep learning models based on different construction are also investigated. Secondly, the reduced order models are used to carry out the learning of continuous latent system for complex chemistry mechanisms, with varying prediction time steps. The objective is to verify that if aligned results with respect to standard states prediction case can be obtained.

Résumé

Cette thèse porte sur l'accélération des calculs de cinétique chimique dans les simulations de CFD en s'appuyant sur des méthodes d'apprentissage automatique. Le principe consiste à remplacer la résolution de la chimie dans les calculs par un modèle d'apprentissage équivalent, dont l'évaluation est beaucoup plus rapide que la résolution du système original d'équations différentielles. Les travaux de thèse se concentrent d'abord sur l'étude de cas de combustion simplifiés (système 0-dimensionnel sans termes de transport). Nous proposons un cadre de traitement des données permettant la construction d'un modèle substitutif précis. Nous fournissons une comparaison approfondie entre la résolution de base et la résolution reposant sur le modèle appris. Ensuite, des modèles combinés à des techniques de réduction de dimension sont développés et appliqués à des cas de chimie complexe, constituant une première étape vers des approches de prédiction avec les pas de temps variables. Enfin, notre approche est appliquée à un cas multidimensionnel avec des termes de transport, incluant notamment la turbulence.

La première contribution de cette thèse est l'extension de l'étude sur les algorithmes potentiels pour la prédiction des états dans le système thermochimique largement complexe. Ces algorithmes sont évalués, premièrement vérifiés sur le cas basique de benchmark comme le problème d'auto-allumage de 0D, et ensuite étendus pour les cas multidimensionnels avec le phénomène de transport turbulent inclu. Nous avons sorti le calcul de chimie pour les carburants avec les complexités différentes. Après la vérification de prototype de base de l'apprentissage, nouveaux modèles de l'apprentissage basé sur les constructions différentes sont aussi étudiés. Deuxièmement, les modèles de réduction d'ordre sont proposé pour l'apprentissage de système continuous en espace latent pour les mecanismes complexes de chimie, en utilisant les pas de temps de prédiction variés. L'objectif est de vérifier si les résultats sont bien consistents par rapport aux cas de prédictions standard des états.

Remerciement

Je tiens à remercie d'abord tous les membres du jury pour leur révisions de ce manuscrit et pour leur participations de la soutenance finale. En particulier je remercie à Prof. Luc Vervisch et Prof. Alessandro Parente pour leur commentaires constructifs sur cette thèse et leur propositions précieuses sur les perspectives de ce sujet. Ensuite, je tiens encore plus remercie à Florent, Cédric et Thibault pour leur directions de ma thèse pendant ces trois ans, avec beaucoup d'intelligence et de propositions significatives. C'était pas vraiment facile de bien commencer la thèse comme un thésard non francophone, surtout en cas de confinement de Covid-19, mais c'était super sympa de travailler avec tous les trois encadrants qui ont toujours la passion pour la recherche. Merci aux tous les membres à IFPEN et CAS qui ont travaillé sur les sujets de thèse differents. Certains d'entre vous ont partagé plein de discussions avec moi. Merci à Romaric, Dib, Quan, et tous les autres collegues pour les amitiés internationales.

Et plus, je tiens en particulier remercie à mes parents, pour leur soutiens depuis dix ans pour moi en France à partir de 2014. Le plus important, merci à ma cherie Xueyang. Ca ne serait pas du tout facile de compléter la thèse sans toi, et maintenant ça sera à nous pour créer notre futur ensemble à la suite.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Résumé français

Le premier chapitre fais une introduction générale d'étude numérique de la combustion et les écoulements réactifs. La présentation de la contexte de recherche académique et industrielle est mise en place. En suite, une révision bibliographique dans ce domaine est faite.

1.1 Context

Challenges in the energy sector

Despite the increasing popularity of electric vehicles, combustion engines continue to play pivotal roles for many applications. To reach the goal regarding the reduction of carbon and pollutant emissions, EURO Standards have been progressively tightened over the years to regulate the limit of emissions (Figure 1.1). To comply with these standards, numerical simulation emerges as a powerful and essential tool for exploring the complex phenomena of the combustion process. It plays a crucial role in the design of engines or industrial processing machines. When considering the governing systems of physical models, it is important to investigate the complex convection-diffusion-reaction dynamics of combustion. This understanding is fundamental for optimizing designs across various processes.

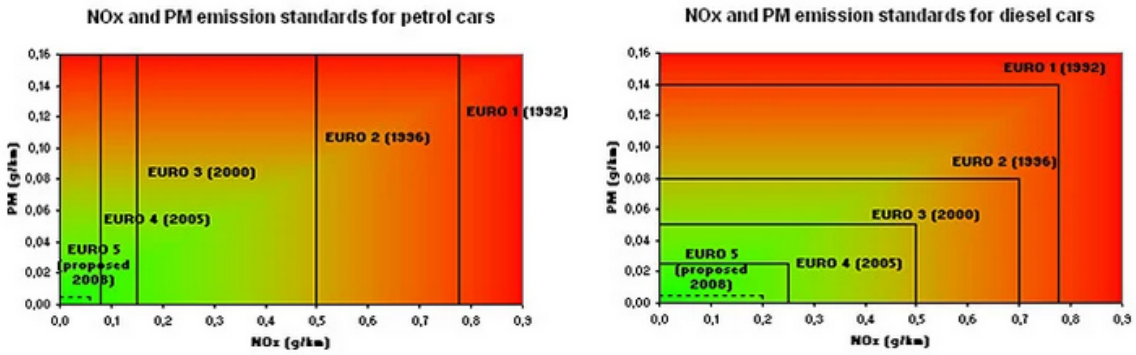


Figure 1.1: EURO Standards for the internal combustion engines from 1992 to 2008[1]

Computational modeling of reacting flows, including combustion, is a powerful tool to analyse and study the multi-physical phenomena within combustion systems, and has been a dynamic field of research over the past decades. It has found extensive applications in areas such as aeronautical propulsion, industrial processes, power generation, and so on. In today's context, as the challenge of environmental protection and emission reduction continues to become more complex, the study of combustion processes plays a crucial role in accurately predicting and mitigating pollutant emissions.

In this context, in-depth research into complex combustion chemistry is crucial. Simulations involving a large number of chemical species and complex chemical mechanisms, conducted within a extremely short period, prove to be valuable and applicable in industrial design. Furthermore, employing more intricate chemical mechanisms enables more accurate and comprehensive simulations of combustion physical processes (i.e. fluid dynamics), providing valuable physico-chemical insights.

Challenges for the simulation of reacting flows

With the growth of computing power, it is possible to simulate reacting flows in space and time for a wide range of scales. However, typical chemical processes involve hundreds of species and thousands of reactions, which are coupled with transport phenomena. It is therefore expensive to solve the fully coupled convection-diffusion-reaction equations, especially for industrial systems. Even when considering homogeneous combustion, therefore neglecting the convection-diffusion phenomena, chemical processes are modeled by non-linear and stiff ordinary differential equations (ODEs). It is expensive and impractical to solve these ODE systems, due to the multitude of physical evolution scales. The complexity of chemistry and the computation of the chemical reaction rates is an important issue. It is indeed often the bottleneck of the computation of chemical kinetics. Hence, additional techniques of computation for the thermochemical quantities during the reacting flow simulations are necessary. Several methods have been proposed to reduce the

computational cost of thermochemical states resolutions, including standard acceleration methods such as chemistry reductions and look-up tables, and machine learning based methods. A brief review of the main approaches is presented in the next section.

1.2 Chemistry integration methods

There are two different approaches for chemistry resolution. On one hand, the full chemical states can be computed by solving the transported partial differential equations (PDEs) for each chemical species, known as transported chemistry. On the other hand, one can simplify the full chemical states to a reduced number of variables, and these reduced variables are either directly integrated or tabulated in the case of complex chemistry.

1.2.1 Transported chemistry

The transported chemistry approach aims to solve all detailed chemical species involved in combustion systems.Nevertheless, there are still significant inconveniences. Firstly, for complex chemistry with hundreds to thousands of species, solving these full states is extremely expensive, and in some cases, impossible for large-scale complex combustion simulations. To enhance computational efficiency, direct integration can be combined with various chemistry reduction methods, such as reduced skeletal mechanisms or manifold generated reduced-order states. These reduction techniques aim to reduce the number of chemical species and reactions considered while retaining key information. Secondly, solving the fully coupled convection-diffusion-reaction Partial Differential Equations (PDEs) representing combustion dynamics, particularly for industrial systems, is computationally expensive. A commonly adopted approach is to address the chemical source terms separately using an operator-splitting method[12][13], where the resolution within a time step is split to sub-steps for transport terms (i.e. convection-diffusion mechanisms) and source terms (i.e. chemical reactions), respectively. The basic operator-splitting method is a first-order method, which is easy to implement. Other widely used operator-splitting methods, such as Strang-splitting [14] and Lu-splitting [15], are generally second-order accurate. Setting aside the resolution of the convection-diffusion part of the dynamics, we focus now on the chemical reactions mechanics, which consist of non-linear and stiff ordinary differential equations (ODEs).

Detailed chemistry

In detailed chemistry, simulations include multiple steps of chemical reactions and a larger number of minor chemical species and radicals, where their reactions occur on extremely fast timescales. This is necessary for tackling complex problems, such as determining

soot formation or NO_x in combustion, which are sensitive to the chemical mechanisms. Direct integration solvers, such as CVODE[16] and DASAC[17], offer a straightforward approach to solving chemistry states during simulations. These solvers rely on an array of variable-order, variable-step multistep methods to perform direct integration of the chemical kinetics, providing detailed chemistry information. This resolved chemistry data can then be coupled with the full set of equations governing the simulation. Within a chemical system, various chemical species operate on different chemical time scales. Minor chemical species with extremely fast chemical reactions and major chemical species with slower reaction time scales are identified in the system. A brief overview of the range of chemical time scales is provided in Figure 1.2. These intermediate reactions under many different time scales are extremely expensive to be solved directly. Therefore, additional strategies for managing thermochemical quantities during reacting flow simulations remain essential. To address these challenges, several methods have been proposed to reduce the computational cost associated with resolving thermochemical states. However, introducing a detailed chemistry process poses challenges for numerical resolutions, particularly in large-scale simulations with complex geometries in an industrial context. Direct numerical simulation with detailed chemistry remains restricted to small-scale academic problems. [18].

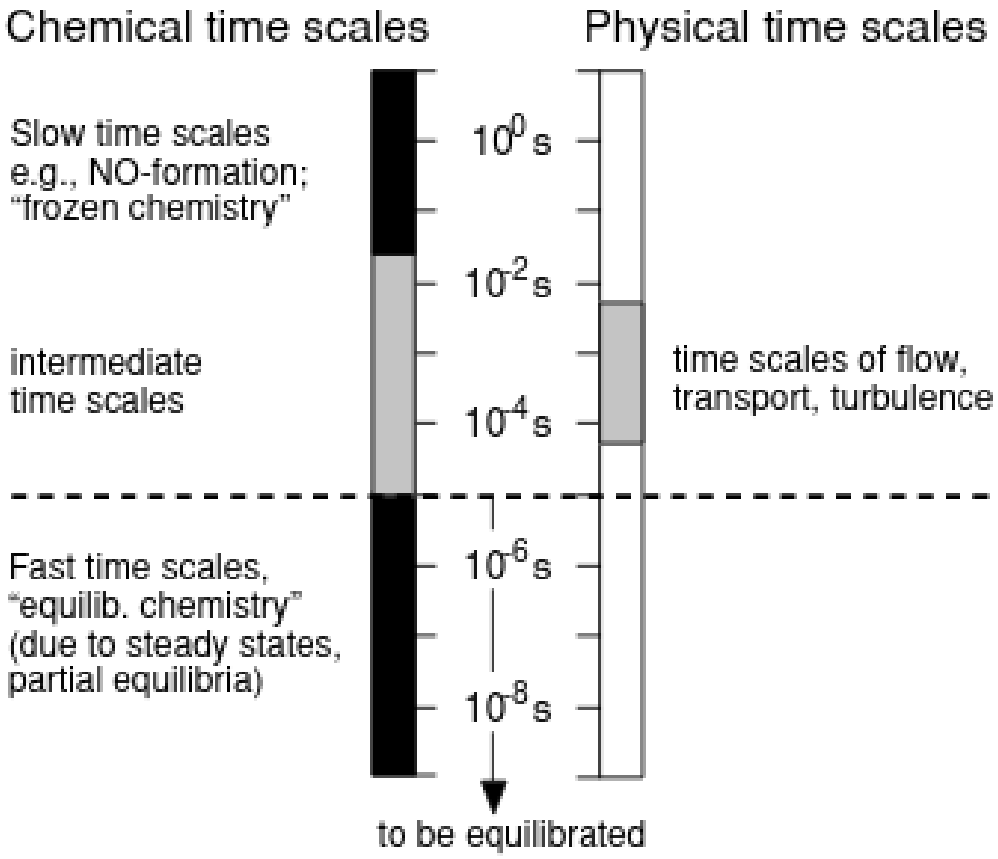


Figure 1.2: Range of chemical reaction time scales, where a group of chemical species under a scale threshold are considered as minor species, with chemical reactions in extremely fast time scales.[2]

Reduced chemistry

A popular strategy to accelerate chemistry integration involves the use of reduced chemical mechanisms [19, 20, 21, 22]. In a complex chemical system, three distinct categories of species are identified. Firstly, the important species, which are essential for representing the overall reaction system and are present in major reactants and products, must be retained. Next, the necessary species are preserved to capture essential phenomena within the process, such as ignition delays, flame propagation speed, and pollutants. Finally, the remaining species can be eliminated from the extensive chemical mechanisms. Under certain conditions, it is possible to compute only a few major chemical species that significantly influence the simulation results and represent the core reaction system. In such cases, minor chemical species that have negligible impact on the results are omitted. Various methods have been developed for deriving reduced mechanisms. Directed Relation Graph (DRG)[23] is used to develop skeletal mechanisms from full detailed chemistry. The Quasi-Steady State Approximation (QSSA)[24] is applied to derive the reduced chem-

istry from skeletal mechanisms. Moreover, Analytical Reduced Chemistry (ARC) [25] has been developed to reduce the chemistry and is utilized in computing turbulent spray combustion with soot, combined with direct integration. Alternative methods such as hybrid chemistry (HyChem)[26][27] or virtual chemistry[28] are proposed to build small mechanisms instead of reducing large mechanisms. To describe the pollutants accurately, these small optimized mechanisms need to be extended. In addition, methods such as error-propagation-based reduction [29] and automatic reduction and optimization from canonical problems [30] have been proposed and investigated. Nevertheless, the predictions of target pollutants, such as soots or NO_x, still require reduced mechanisms with more than 30 species including fast and slow chemical reactions. The direct integration approach still relies on traditional integration techniques for stiff ODE systems, limiting its suitability for simulating large hydrocarbon fuels with extensive chemical species and reactions based on reduced mechanisms.

Multi-zone method

Another method to accelerate chemistry integration involves a multi-zone model identified by physico-chemical properties or highly flexible clustering algorithms [31, 32, 33]. The resolution of detailed chemistry for combustion engines are typically treated on a cell-to-cell basis. However, solving chemistry for all cells requires a large number of calls, becoming expensive when there are a great number of cells. In this scenario, a group of cells is identified based on local composition, which depends on temperature and equivalent ratio, aiming to reduce the number of calls to the chemistry solver. Each cell within the group shares similar thermochemical conditions, thereby improving the efficiency of local chemistry resolutions.

1.2.2 Chemistry tabulation

Another approach to reduce computational costs in thermochemical simulations is to use the chemistry table method, often referred to as Look-Up Table (LUT) [34]. This method involves a pre-processing step where the necessary chemical terms are computed and stored in a discrete table that covers the composition space. During the simulation, relevant data from this table are retrieved and interpolated as needed.

Maas et al.[35] have demonstrated that within a complex chemical system, it is possible to identify an intrinsic low-dimensional manifold (ILDM). They propose a mathematical analysis based on Computational Singular Perturbations (CSP), which involves applying linear approximation to high-dimensional nonlinear chemistry systems. This process generates the eigensystem and identifies states with low or fast time scales by referencing the eigenvalues. Consequently, the reduced system includes the preserved major chemical species with slow reaction time scales and several progress variables generated from mani-

folds, which are used for chemistry tabulations. Moreover, applying the flamelet concept, turbulent flame is considered as an ensemble of thin and local 1D laminar flame, which is called flamelet. Progress variables generated from manifolds can also be determined using flamelets precomputed from detailed chemistry, applied for chemistry tabulations. This approach is more physically direct and pragmatic, often overcoming some of the limitations associated with the ILDM method. Elemental flamelets are pre-generated from abstract problems that encompass various phenomena, such as Flamelet Prolongation of IDLM (FPI)[36, 37], Homogeneous ignition[38, 39], Perfectly Stirred Reactor (PSR)[40], Steady non-premixed flamelets[41], flamelet/progress variables approach for quenching and re-ignition[42, 43, 44], and unsteady non-premixed flamelets[45, 46]. Another technique, known as In-Situ Adaptive Tabulation (ISAT) [47], employs multiple linear regressions to create a table during the simulation process and dynamically tabulate the composition space.

However, there are still significant limitations associated with these tabulation methods. When creating the table, it is essential to determine the input dimensionality of the table. For simple cases, tables can be generated with one to three dimensions. However, for complex scenarios, creating tables can be more costly and, in some cases, impractical. Additionally, in the context of parallel computing, these tables can consume a substantial amount of memory space.

1.2.3 Chemistry with Artificial Intelligence

Artificial Intelligence (AI) is a powerful tool that is reshaping the landscape of scientific research across diverse domains. Its ability to process vast amounts of data, automate tasks, and make predictions is enhancing our understanding of the natural world and driving innovation in numerous scientific fields. AI is widely applied to different scientific fields, such as molecular discovering techniques, computational materials [48, 49], and numerical simulations[50, 51]. Deep learning models can serve as surrogate models capable of fitting any non-linear model [52] [53]. In the realm of large and complex system simulations, the use of AI techniques to grapple with computational complexities constitutes a promising approach[54, 55].

A neural network-based surrogate model can replace traditional integration methods for chemical computations [56, 57]. After each transport resolution time step, the source terms are predicted directly using deep learning models. As a result, the utilization of stiff solvers for chemistry is no longer necessary, leading to improved computing efficiency. Further studies have extended the application of neural networks to even more complex chemical mechanisms , such as syngas combustion with fourteen- and sixteen-species mechanisms under turbulent premixed and non-premixed conditions[58] ; the direct numerical simulation of a syngas turbulent oxy-flame using 11-species reduced mechanism

with side-wall effects[3] (Figure 1.3) ; the simulation of turbulent Sandia flames incorporating CH_4 mechanism with thirty-two involved species[59] ; engine system turbulent flame simulations with reduced kerosene involving forty-one species and one hundred and thirty-two elemental reactions[4] (Figure 1.4); and the large eddy simulation of turbulent flameless combustion in the industrial furnace with fourteen chemical species [60]. The direct predictions of states can be carried out using the neural networks, not only for the detailed chemistry states, but also for states on reduced order manifolds of chemistry tabulations[61, 62, 63, 64, 65, 5]. An example of neural network model for states prediction to replace the chemistry tabulation is given by Figure 1.5. The dimension of states learned by models in these case are still limited to small numbers today, and it is supposed that the extension for more complex chemistry is necessary.

Deep learning-based tabulation methods have demonstrated their efficiency in thermochemical simulations. Most of these research studies demonstrate the efficiency of machine learning models in accelerating the computation of source terms compared to direct integration, and reveals the potential of employing neural networks to accelerate chemical resolution for complex fuels while keeping good agreement with exact solutions. The inference process is carried out by simple algebraic operations (matrix multiplications, etc...) which are generally faster than the expensive direct integration for complex chemistry. On the other hand, a notable advantage of utilizing deep neural networks (DNNs) over chemical look-up tables is their low memory requirement [66]. The number of parameters needed for a DNN is much smaller than the number of multi-dimensional points that would have to be stored in a table for a complex chemical reaction. A study by Blasco et al. [57] showed that Artificial Neural Networks (ANN) require 11 times less RAM storage space than direct integration, 831 times less than tabulated chemistry, and have similar CPU operating times compared to tabulated chemistry, while being 100-500 times smaller in CPU time usage than direct integration. Subsequent research has further evaluated the efficiency of deep learning methods in various computational contexts. Table 1.1 provides a summary of comparisons between the Deep Neural Network tabulation approach (DNN) and the direct integration approach (DI). Moreover, researchers have conducted quantitative analyses to define criteria for transitioning from tabulation to neural networks [67]. A recent study [11] proposed a general workflow to maintain both accuracy in chemistry computation and acceleration performance.

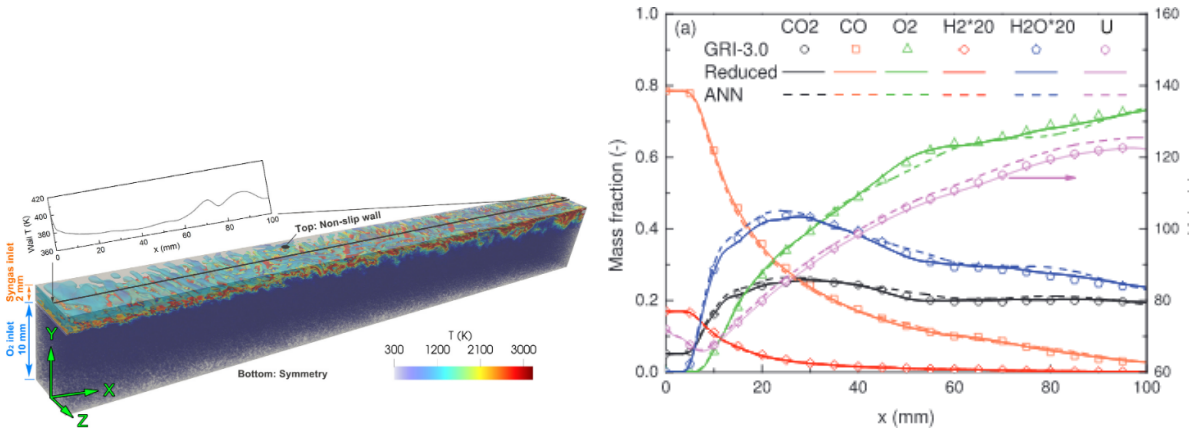


Figure 1.3: Direct numerical simulation of a syngas turbulent oxy-flame with side wall effects using artificial neural networks(ANN) for chemistry computation, (b) shows the comparison of averaged distributions along the planar fuel-jet centerline for chemical species. The simulation result with machine learning models is consistent with results with the standard GRI-3.0 chemical mechanisms and reduced mechanism.[3]

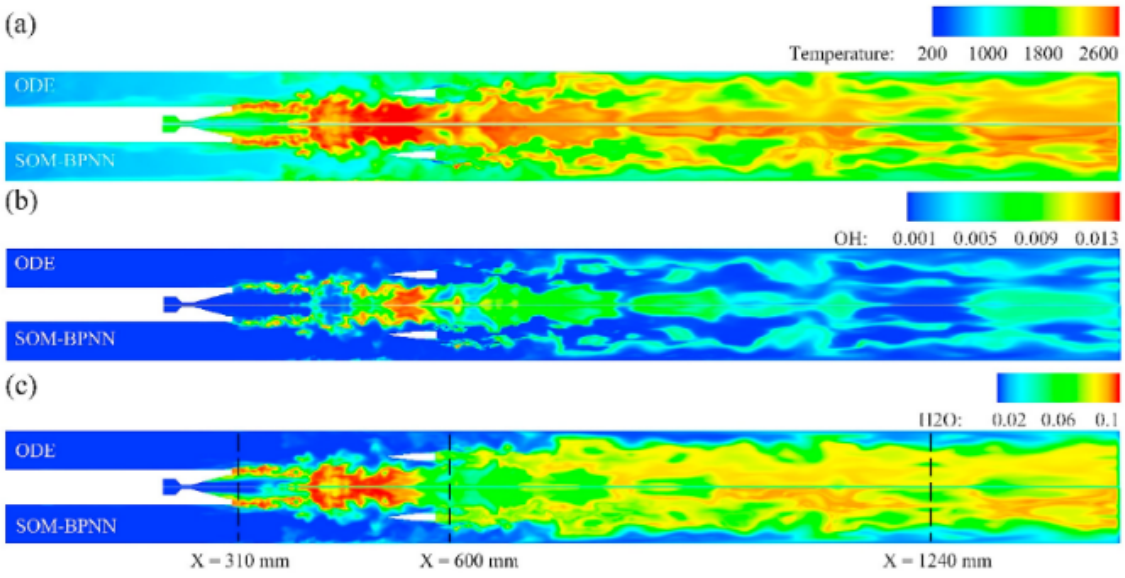


Figure 1.4: Results of a Rocket-based combined cycle (RBCC) engine system turbulent flame simulation with reduced kerosene using SOM-ANN model which globally aligns with the results of ODE based simulations, albeit with deviations in the magnitude of H_2O in the far region from the combustor. [4]

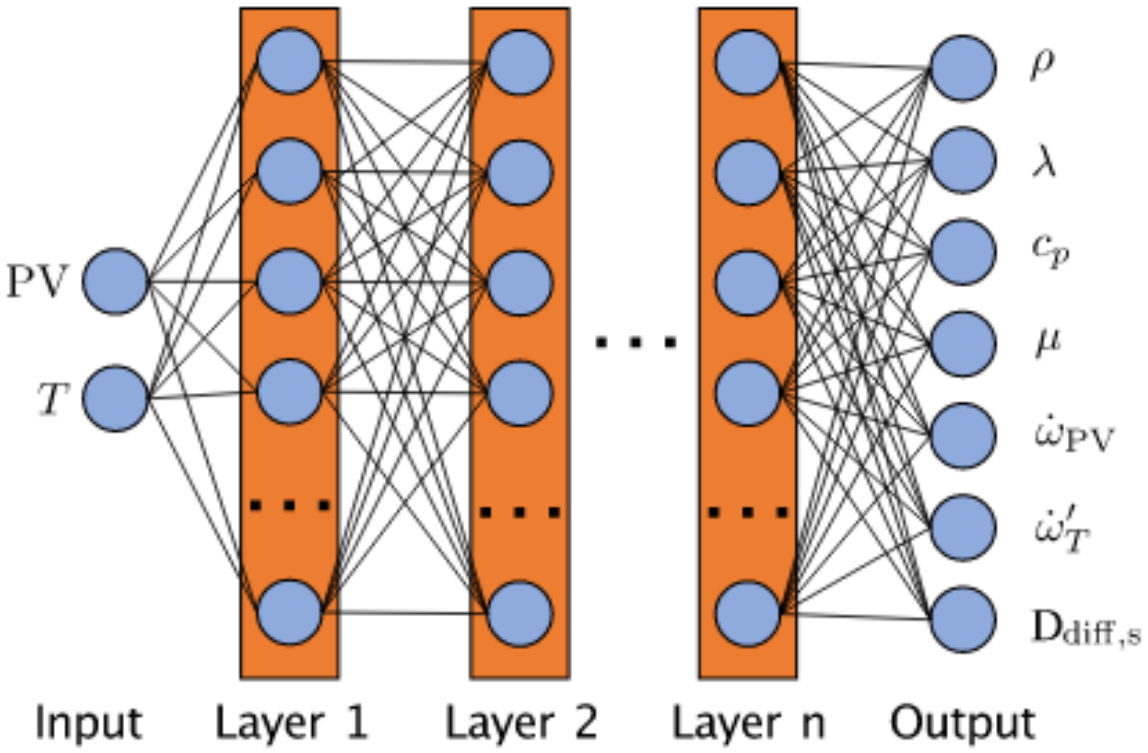


Figure 1.5: A neural network model to predict the state evolution on reduced order manifold-based system, with flamelet progress variable and temperature as inputs. [5]

Author	Acceleration(DNN vs DI)	Storage(DNN vs LUT)
Blasco et al.[57]	$165-511$	11
Blasco et al.[68]	2750	1000
Franke et al. [6]	4.6	Not mentioned (3.6MB for DNN)
Wan et al. [3]	25	Not mentioned
Ding et al. [59]	$10-20$	Not mentioned
Readshaw et al. [69]	17	Not mentioned
Nguyen et al. [70]	80	Not mentioned
Han et al. [71]	$2-12$	Not mentioned

Table 1.1: Literature review for efficiency of deep learning model

Dataset generation

The performance of ANN training results strongly relies on the quality of the dataset used. A proper generated dataset may improve the deep learning models generalization performance during the simulations. Generating an appropriate dataset that includes information from different time scales of various chemical reactions is a challenging task. A compact dataset for combustion simulation must include different target phenomena,

such as fast ignition, flame propagation and flame extinguishment for instance. In early studies of simple chemical mechanisms, researchers have employed random sampling of chemical species within predefined operating ranges of a 0D simulation [56, 57], where only the ignition phenomena is included within the dataset. A fixed small-scale sampling time step can also be used to ensure the capture of fast ignition and heat release processes [71].

As for multi-dimensional combustion processes with transport terms, more complex combustion phenomena must be included into the dataset. Training the ANN models directly on the data from target simulations is the most direct and intuitive choice for multi-dimensional combustion case. However, it is always expensive to obtain the data from high fidelity numerical simulations. Different canonical problems are defined and simulated to contain target information of combustion regimes without simulating the targeted configuration. These simplified problems help to generate the dataset with less computing costs. Firstly, the dataset can be constructed based on flamelet approaches. In [6], 1D igniting/extinguishing non-premixed flamelets under different operation conditions are simulated to generate samples. Then all data samples are shuffled. The trained ANN models based on this dataset are tested on non-premixed combustion case (i.e. turbulent jet of Sydney flame). Moreover,in [70, 60, 11], it is proposed to solve a particle system of stochastic mixing reactors model with additional mixing terms to emulate the transport phenomena and generate the target dataset. These different studies train models based on this dataset generation approach, applying the models under the case of direct numerical simulation (DNS) of non-premixed syngas turbulent oxy-flame simulations, or DNS of premixed hotspot ignition of H_2/air turbulent combustion. Besides, in early research the dataset is generated by simulations based on a stand alone Linear Eddy Model(LEM) simulation[58]. The simulations contains both premixed and non-premixed conditions. Then the dataset can be used to train ANN models for large eddy simulations of turbulent combustion, such as flame-turbulence-vortex interaction simulation.

Including input simulation noise during each resolution time step to generate a more robust dataset is a strategy to make the models more robust, for 0D problems [71] with only ignition process, or multidimensional problems with generated flamelets[69, 59]. Besides, in a study by Zhang et al. [72], different strategies for data-driven generative sampling methods are compared. The authors propose a new multi-scale sampling method for different groups of chemical species. In this approach, the species are classified into major and minor groups, and random sampling is performed at fixed intervals using either normal or logarithmic distributions. Each of these methods aims at ensuring that the dataset captures the essential features of the chemical processes at different scales.

All these approaches for the generation of datasets are under different specific scenarios. However, there are still no generalized methods to construct datasets for complex chemistry, particularly in an industrial setting.

Data clustering

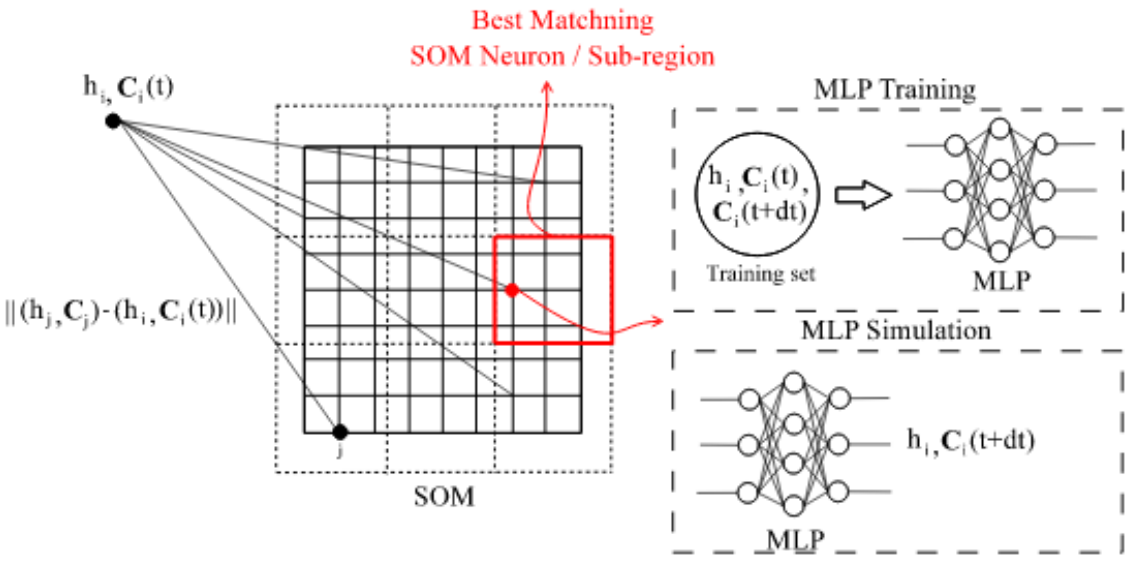


Figure 1.6: Self-organizing map model to partition the chemical states to different subdomains in a low dimensional representation [6], where the topological structure of the data are preserved

Most of the previously cited data generation methods result in a dataset that contains information about different combustion regimes. Therefore, predicting the chemistry states using a single ANN for the various dynamics investigated is difficult. Splitting the chemical manifold into several subdomains and training multiple ANNs can improve learning efficiencies by facilitating the training process and reducing local model complexity. Various methods have been employed to efficiently separate subdomains and enhance prediction accuracy. Physical descriptions of the combustion process, such as progress variables [73] or the rate of temperature increase [71], can effectively partition the subdomains and improve prediction accuracy. Another approach is to utilize data-driven classification algorithms. One such algorithm is the self-organizing map (SOM) with a neural network structure as shown in Figure 1.6). In this figure the SOM neural network preserves the topological structure of the data and produces a low-dimensional representation to partition input data points [4, 74, 6]. Another option is the K-Means partitioning algorithm. This algorithm partitions the data by calculating the distance between each data point and a fixed number of centroids, which represent different clusters. Modified K-Means algorithms are used to partition the full chemistry states into subdomains with similar reactive conditions, successfully accelerating the simulation time of combustion with high-dimensional full chemistry by using conventional numerical resolutions or artificial neural networks in engine environments [31, 75] or in detonation combustion waves simulation

[76]. These advances underscore the effectiveness of employing non-supervised partitioning algorithms in complex combustion simulations, enabling improved prediction accuracy and efficient handling of different combustion regimes.

Surrogate models

Up to now, most of the models used to learn chemical states were based on artificial neural networks. Previous studies start from the simplest state-to-state feed-forward neural network with outputs for all thermochemical states [56, 57, 58, 6]. Then, varying prediction time steps can be included as an input element, allowing the use of different sizes of prediction time steps during the inference simulation [74]. Later, the studies transition to different forms of learning workflows. For example, multiple layers with a single output for each dimension can be applied to improve the performance of models [69]. The author demonstrate that multiple neural networks for each dimension helps to improve the accuracy compared to the Self-Organizing Maps clustering based workflow by statistical analysis of training results. The reaction rate of each chemical species, rather than the output states, can be also predicted using deep neural networks[3, 70, 69, 59, 77]. Besides, stacked neural networks are applied recurrently during the training and inference process for multi-scale values learning, improving the prediction performance for small scales of chemical species within the system [59]. Moreover, an on-the-fly learning workflow without a priori training process has been proposed[73]. This method allows to train on-the-fly models based on datasets generated from previous ongoing simulation time steps, and thus the pre-training process is no longer needed. Physics information such as first principles can also be included in the learning model to improve the robustness of models following physical laws [78]. More recently, new topology of learning networks structure are proposed, including residual neural networks [63, 79], physical informed stiff kinetic neural networks [80] and multi-scale chemical kinetic neural networks [81]. Conventional machine learning method, such as gradient boost decision trees (GBDT), is also applied as a first approach to the simple 0D auto-ignition combustion problem for H_2 [82]. Nevertheless, these approaches are limited to simple cases with small number of states, and have never been extended to multidimensional simulations.

State-to-state prediction involves directly predicting the state of a dynamical system at a future time based on the system's state at a given time. Another approach have been developed to predict the effective states augmentation under the dynamical system approximation, building on concepts such as residual networks (ResNet) and dynamical systems modeling. This approach aims to improve the understanding and prediction of dynamical systems from different perspectives. Thie conception is based on learning the effective "increments" of the dynamical system states, following numerical schemes like effective increment learning [7, 83]. In this approach, researchers use recurrent residual blocks to approximate the continuous flow maps. Figure 1.7 shows the general ideal of

using recurrent residual blocks to learn the dynamical system evolution. These residual blocks can be implemented using feed-forward neural networks and are designed to mimic numerical integration schemes. The idea is to iteratively update the system's state by applying these recurrent blocks to approximate its evolution over time. Both approaches aim to capture the underlying dynamics of a system, but they use different techniques to achieve this goal. The choice between them often depends on the specific problem and the characteristics of the dynamical system being studied [84, 85, 86].

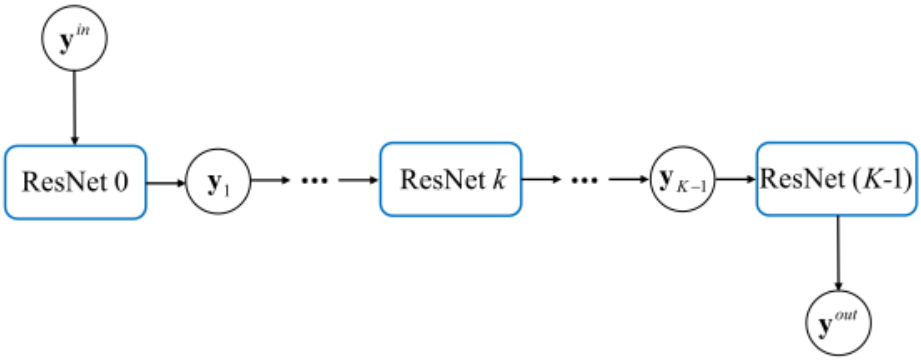


Figure 1.7: Deep residual blocks considered as the discrete approximation to learn multiple increments for the dynamical system[7]

Neural Ordinary Differential Equations (neuralODE) is another approach to learn continuous unknown dynamical systems under a different framework of learning process. It has been applied to various applications, including chemical species prediction[87, 88, 89]. The training process is based on the backward differentiation of adjoint states (Figure 1.8) [8]. However, neuralODE has some limitations. One of the drawbacks of neuralODE is the training time, especially for high-dimensional inputs. Training neuralODE models can be computationally expensive and time-consuming, making it less practical for large-scale problems. Additionally, the inference time for neuralODE models can also be slower compared to conventional deep neural networks (DNNs). This is because the inference process for neuralODE models involves solving ordinary differential equations using explicit adaptive solvers, which can be slower than the matrix multiplication used in DNNs. While there are studies suggesting that explicit time resolution in neuralODE models can accelerate computing time compared to implicit solvers like CVODE [88], more research is needed to further investigate and optimize the performance of neuralODE in various contexts. In summary, while neuralODE is a powerful approach for learning continuous flows, its computational costs and inference speed should be considered when applying it to different problems.

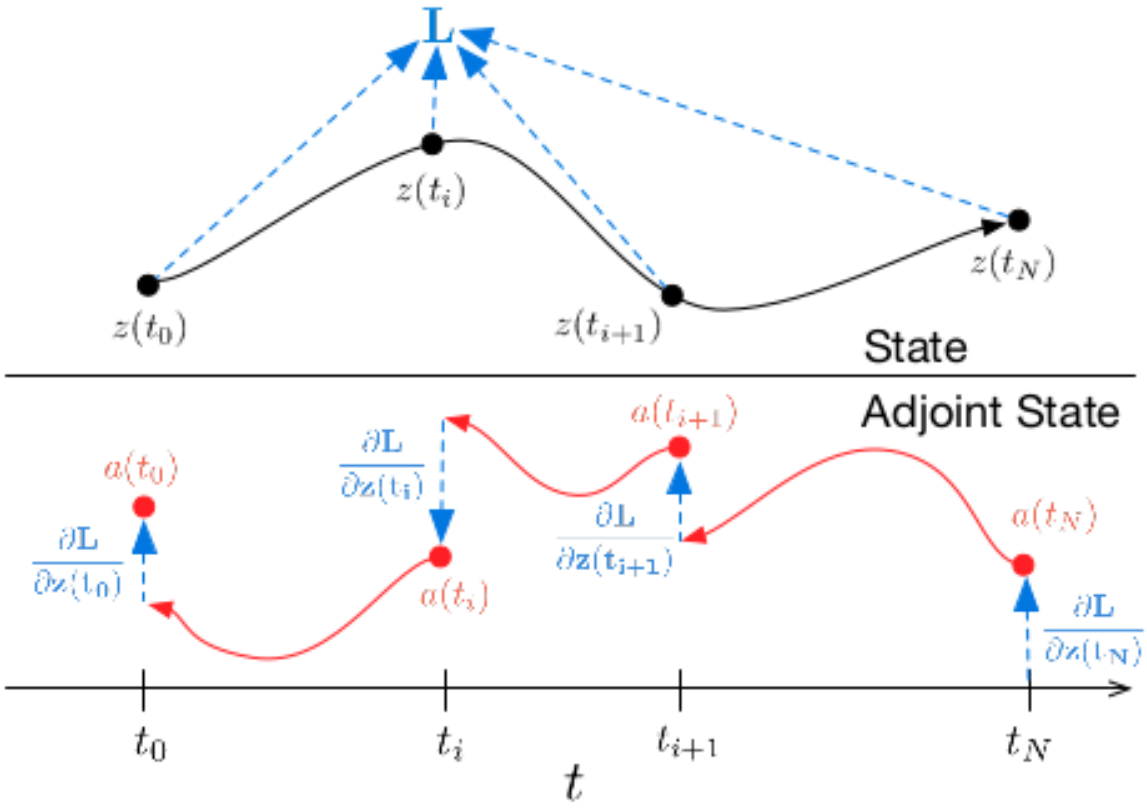


Figure 1.8: Neural Ordinary Differential Equation for the dynamical system learning, where the training algorithm is based on the backward adjoints computing[8]

Until now, more and more advanced deep learning models have been proposed and applied to different problems. These diverse models offer more possibilities for designing the learning workflow for predicting chemical states, potentially improving the efficiency of the computing process

Data-driven reduced order manifold generation

The manifold approaches described in Seection 1.2.2 possess a data-driven counterpart. Different approaches are applied to identify reduced latent dynamical systems, including linear models based on the matrix decomposition method and nonlinear models based on deep learning methods.

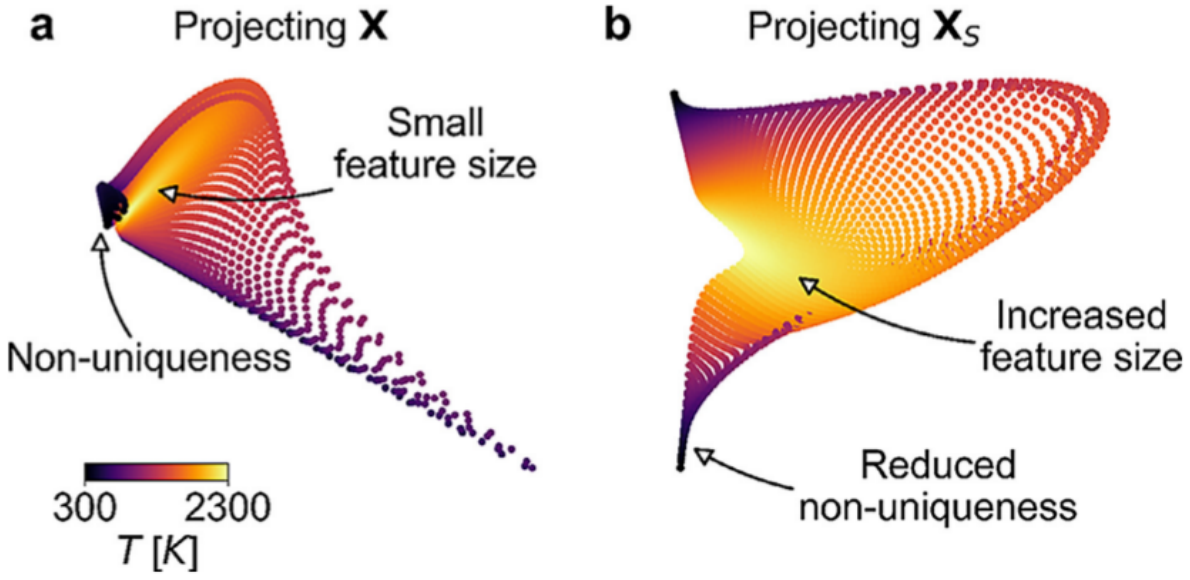


Figure 1.9: Data-driven projection of the chemical states using PCA method, by projecting the full states vectors (a) and optimal chosen subset of states vectors.[9]

Linear model of projection: A direct method for identifying low-dimensional states from high-dimensional systems is to apply Principal Component Analysis (PCA), which is based on singular value decomposition. PCA is a non-supervised learning algorithm, and the mathematical expression for principal components can be determined directly from matrix multiplications without involving parameters. The original proposal for using PCA to reduce systems in reacting flows comes from [90]. In this approach, the full chemistry within the system is reduced to a set of principal components, and the transport terms are linearly projected onto a system with reduced degrees of freedom through matrix multiplications. The original form of PCA dimension reduction is called the score-PCA approach, where transport equations are based on the reduced principal components. Another PCA approach is called manifold-generated PCA (MG-PCA) [91, 92], where a set of original chemical states is retained, and the rest of the states are eliminated. The reconstruction of the original states is based on the truncated approximation matrix. More recent research has revealed that a small number of original parameters can be optimally chosen for data-driven techniques to represent the complexity of reacting flows [9, 93]. These reduced-order states can be employed to simulate reacting flows in various contexts, in combination with fluid transport terms. The PCA algorithm is fast and mathematically intuitive. However, as it is a linear projection algorithm, the projected latent states may lose a significant amount of information stemming from correlated variables in the high-dimensional space. An improved technique to address this limitation involves using an extended local PCA algorithm to obtain local latent states with varying numbers of major components and to partition the dataset into subdomains [94, 95, 96, 97]. Since the total

dataset may exhibit different nonlinear correlations in various subdomains, this technique helps identify local latent spaces adaptively and can increase the reconstruction accuracy of the original system space. Additionally, the reconstruction of original chemical states can be carried out using nonlinear models like Gaussian regression or neural networks to preserve the nonlinear correlations. Further studies and extension to complex simulation scenario of this technique can be found in [98, 99, 100]. As shown in Figure 1.9, linear PCA reduction can identify the low-dimensional chemistry manifold that defines the input full states.

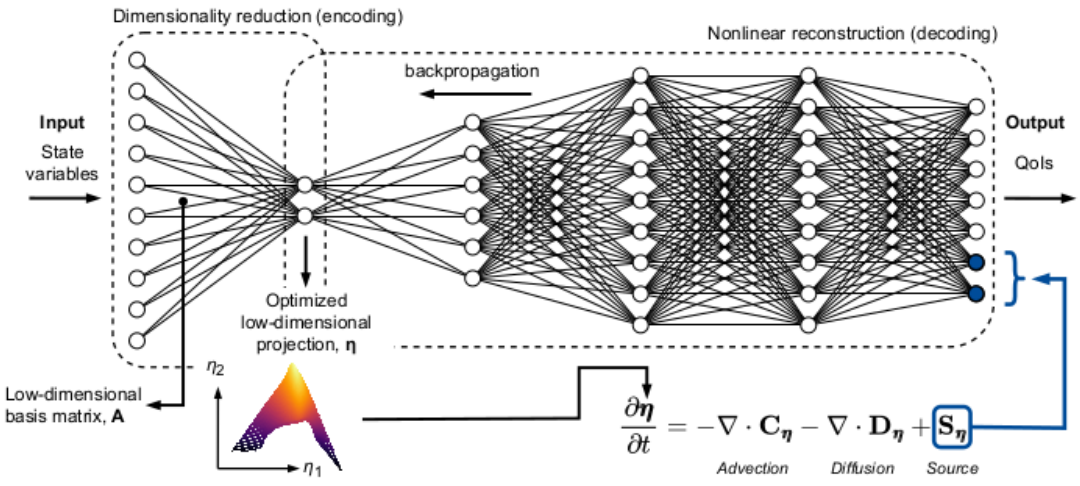


Figure 1.10: Nonlinear dimension reduction for full chemical states using QoI autoencoders, generating low dimensional manifolds.[10]

Nonlinear model of projection: Various approaches for nonlinear chemistry dimension reduction have been explored in the literature. Nonlinear PCA, such as kernel PCA, is an alternative choice for nonlinear chemistry reduction. Another popular nonlinear dimension reduction technique used to reduce the full states in a high-dimensional system is the autoencoder. Figure 1.10 shows an autoencoder model, which is a type of neural network designed to project a high-dimensional vector to a bottleneck hidden layer with nonlinear activation functions in each neuron. In fact, PCA is a specific form of an autoencoder where the nonlinear activation function is replaced with a linear one. An autoencoder consists of an encoder that projects the original states to latent states and a decoder that reconstructs the original states from latent states. The technique to project the transport terms to the nonlinear latent space during the numerical resolution of latent states is derived based on gradient matrix computations [101], or predicted by stand alone neural networks with input information from reduced latent states[102, 103].

1.3 Contribution of the work

Previous studies feature different approaches of machine learning for chemistry computations. However, most of these studies are based on simple or moderately complex chemical mechanisms. In the context of our research, following the foundation of prior studies, we leverage learning models to tackle the complexity of chemistry resolution within the framework of reacting flow simulations. Thus the first contribution of this thesis is to extend the investigation of potential algorithms for predicting states in highly complex thermochemical systems. These algorithms are assessed, starting from basic benchmark cases such as 0D auto-ignition problems, and extended to multi-dimensional cases involving turbulent transport phenomena. We carry out the chemistry computation for fuels with different complexities. After the verification of a basic learning workflow, new deep learning models based on different construction are also investigated. Secondly, the reduced order models are used to carry out the learning of continuous latent system for complex chemistry mechanisms, with varying prediction time steps. The objective is to verify that if aligned results with respect to standard states prediction case can be obtained.

The PhD thesis is structured as follows:

- Chapter 2 provides the general description of models that we will concentrate on and the methodologies that we will apply. These includes the basic theory of reacting flows and combustion chemistry calculations, and also supervised and unsupervised machine learning methods. Besides, the basic algorithm for chemical states prediction is proposed.

- Chapter 3 focus on 0D combustion case. The workflow is firstly tested on the 0D auto-ignition problems, where an adaptive data sampling method based on the adaptive CVODE resolution time steps is proposed. The algorithm and data sampling method have been tested on various chemistry cases, ranging from the simplest case of H_2 to the much more complex case of CH_4. A novel type of deep network models has been applied for predicting chemical states based on dynamical system approximation and numerical schemes for ODEs. The learning workflow is systematically verified for 0D cases, thus we pass the further verification for more complex multidimensional cases in Chapter 5

- Chapter 4 presents an extended workflow to achieve dimension reduction and to learn the evolution of the chemical system in latent space. This workflow combines a nonlinear autoencoder model with various approaches to latent learning models, including discrete ODE-type models and neural ordinary differential equations. The algorithm is applied to a CH_4 0D auto-ignition case.

- Following the 0D ignition test case, the proposed algorithm has been tested in Chapter 5 on a 2D simulation case involving both H_2 and C_2H_4, and the results are presented in chapter 5. We test three cases with a DNS sampled dataset and two datasets generated by stocastic micro-mixing reactors.

Chapter 2

Theoretical background

In this chapter, a generic model of reacting flow with chemical source terms, able to simulate combustion mechanisms, is introduced. We give details on the numerical integration of chemistry relying on operator splitting, and illustrate the behavior of combustion solvers during simulations. Then, we give an overview of some Machine Learning (ML) methods which are core elements of this thesis and serve as the basis of Chapters 3, 4 and 5.

Résumé français

Dans ce chapitre, un modèle générique d'écoulement réactif avec des termes de source chimique, capable de simuler les mécanismes de combustion, est introduit. Nous donnons des détails sur l'intégration numérique de la chimie en s'appuyant sur le fractionnement des opérateurs, et illustrons le comportement des solveurs de combustion lors de simulations. Ensuite, nous donnons un aperçu de certaines méthodes d'apprentissage automatique (ML) qui sont des éléments centraux de cette thèse et servent de base aux chapitres 3, 4 and 5.

2.1 Combustion modeling

Reacting flows feature multi-physical coupled dynamics involving convection-diffusion phenomena and chemical reactions, resulting in rapid changes in temperature and mixture composition. First principles for mass, momentum, chemical species, and energy conservation lead to the following governing equations

2.1.1 Conservation laws

$$\begin{aligned}
&\frac{\partial \rho}{\partial t} + \nabla \rho \mathbf{U} = 0 \\
&\frac{\partial \rho \mathbf{U}}{\partial t} + \nabla \rho(\mathbf{U}\mathbf{U}) = -\nabla p + \nabla \tau \\
&\frac{\partial \rho Y_k}{\partial t} + \nabla(\rho \mathbf{U} Y_k) = -\nabla \mathbf{J}_k + M_k \dot{\omega}_k \\
&\frac{\partial \rho T}{\partial t} + \nabla(\rho \mathbf{U} T) = \frac{1}{C_p}\nabla(\lambda \nabla T) - \frac{1}{C_p}\sum_{s=1}^{N} C_{p,k} J_k \nabla T - \frac{1}{C_p}\sum_{s=1}^{N} h_k \dot{\omega}_k
\end{aligned} \tag{2.1.1}$$

where

- ρ, $\mathbf{U}$, p, T, Y_k are variables to be resolved in the system, which represent density, velocity, pressure, temperature and mass fraction of chemical species respectively.
- τ is the stress tensor for momentum transport.
- C_p, and λ represent the molar heat capacity, and thermal conductivity, respectively.
- M_k, $C_{p,k}$, h_k, and $\mathbf{J}_k$ are the molar weight, molar heat capacity, molar enthalpy, and mass flux of species k, respectively.
- $M_k\dot{\omega}_k$ denotes the mass fraction source term of species k due to the chemical reactions.
- The mass flux $\mathbf{J}_k$ is expressed as $\mathbf{J}_k = \rho \mathbf{V}_k Y_k$, where ρ is the density and $\mathbf{V}_k$ is the diffusion velocity of species k. The diffusion velocity $\mathbf{V}_k$ can be computed using various diffusion models, which are not detailed here.

More details about conservation equations, more specifically in the context of combustion, can be found in [104]. In this thesis, our general objective is to calculate the source terms which are related to chemical reactions with consumption/production of chemical species.

2.1.2 Numerical integration of chemistry

In the conservation system, the convection-diffusion terms and the chemical reaction source terms are coupled, leading to a significant complication in solving the equations. The system of chemical reactions is known to be stiff, with time scales of chemical reactions typically much smaller than those of the convection and diffusion processes. To address this challenge, a widely used technique is to employ operator-splitting methods for the convection-diffusion-reaction equation.

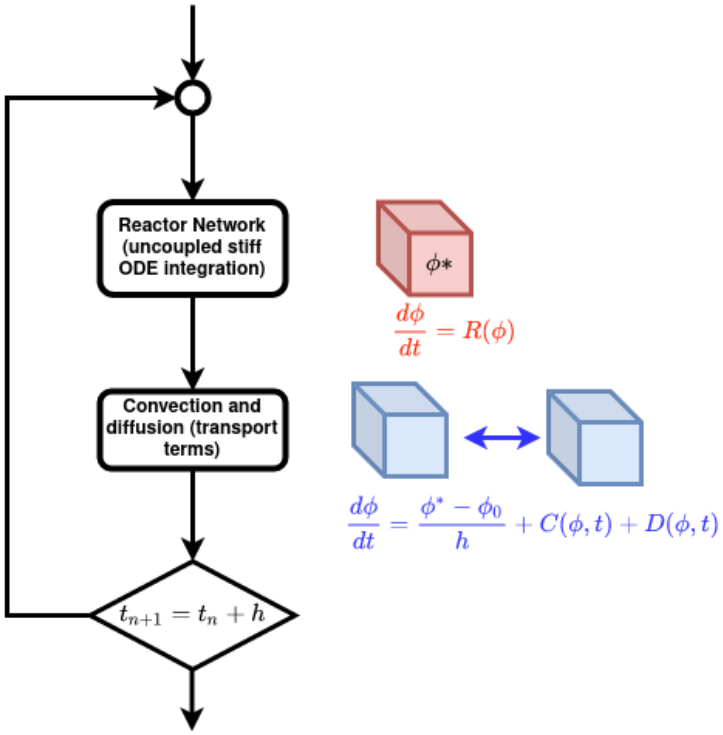


Figure 2.1: Operator-splitting technique with N cells for source term

First, we rewrite the system equations as

$$\frac{\mathrm{d}\phi}{\mathrm{d}t} = C(\phi, t) + D(\phi, t) + R(\phi) \tag{2.1.2}$$

In the above formulation, ϕ represents the state variable vector, while C, D, and R denote the convection terms, the diffusion terms, and the reaction source terms, respectively. The computational domain is discretized into N cells, and within each cell, the source term is computed by integrating the reaction rates. Meanwhile, the convection-diffusion term is computed using a Computational Fluid Dynamics (CFD) solver, employing a finite-volume method for space discretization and various time-discretization schemes for the time evolution. It is important to note that in real Computational Fluid Dynamics (CFD) simulations, if the time evolution is solved explicitly, the time steps must be fixed according to the Courant-Friedrichs-Lewy (CFL) criterion, which is coupled with the spatial discretization steps to ensure numerical stability.

A classical operator-splitting technique is the first-order operator-splitting method [12], that we now describe. This method is first-order accurate. It consists in integrating the following ODE

$$\frac{d\phi}{dt} = R(\phi) \tag{2.1.3}$$

over a horizon corresponding to one CFD time step h, typically using a stiff ODE solver. This yields the first step solution ϕ^*. A CFD solver then solves the following transport

equation

$$\frac{d\phi}{dt} = \frac{\phi^* - \phi_0}{h} + C(\phi, t) + D(\phi, t) \tag{2.1.4}$$

where ϕ_0 is the initial state values. The first-order decoupled computing process is illustrated in Figure 2.1, where the red cubes represent the cells of homogeneous reactors, and the blue arrows depict the convection-diffusion process between different cells. Only the first order operator-splitting method is considered in this research, as it is simple to be implemented.

Combustion thermochemistry equations

We now further detail equation (2.1.3). It is composed of the conservation equation for energy, as well as one mass conservation equation for each chemical species. This yields the following system

$$\begin{aligned}
&\frac{dY_s^*}{dt} = \frac{M_s}{\rho^*}\dot{\omega}_s^* \qquad \forall s \in \{1, ..., N\} \\
&\frac{dT^*}{dt} = -\frac{1}{\rho^* C_p^*}\sum_{s=1}^{N} h_s \dot{\omega}_s^* \\
&Y_s^*(0) = Y_s(\underline{\xi_m}, t_n) \\
&T^*(0) = T(\underline{\xi_m}, t_n)
\end{aligned} \tag{2.1.5}$$

The system described here is derived at a given time t_n and for a grid cell surrounding a point M located at coordinates $\underline{\xi_m}$. As mentioned above, the variables Y_s^*, h_s, M_s, and $\dot{\omega}_s^*$ represent the mass fraction, molar enthalpy, molar weight, and chemical reaction rate for species s, respectively. Additionally, T^*, C_p^*, and ρ^* denote the mean temperature, mean specific heat capacity at constant pressure, and density of the gas mixture in the cell, respectively. The star symbol is used to distinguish between the global variables, which are the solutions of the original governing equations, and the local variables utilized in the chemical fractional step of the operator splitting approach. The system is expressed as follows in a generic form

$$\begin{aligned}
&\dot{\mathbf{S}}(t) = \mathbf{f}(\mathbf{S}(t), \gamma) \qquad t \in [0, t_{n+1} - t_n] \\
&\mathbf{S}(t=0) = [T(t_n), Y_1(t_n), ..., Y_{N_s}(t_n)]^T
\end{aligned} \tag{2.1.6}$$

where $\mathbf{S}$ represents the states of temperature and each chemical species, γ is a variable regrouping all thermodynamic constants.

A common solver for solving the system described in (2.1.6) is CVODE [16]. This solver is particularly employed in CANTERA [105], which is a chemical computation piece of software used for calculating thermodynamic and chemical species terms in reactive flow systems, including the present work. CVODE is a multi-step solver with variable order and step sizes, and it utilizes dynamic adaptive time stepping. To approximate the solution

$\mathbf{S}(t_n) = \mathbf{S}^{(n)}$ at time t_n, CVODE solves the following algebraic equation:

$$\sum_{i=0}^{K_1} \alpha_{n,i} \mathbf{S}^{(n-i)} + h_n \sum_{i=0}^{K_2} \beta_{n,i} \mathbf{f}(\mathbf{S}^{(n-i)}) = 0 \tag{2.1.7}$$

where $h_n = t_{n+1} - t_n$ is the time step. To tackle stiff problems, Backward Differentiation Formulas (BDF) is specifically considered in fixed-leading coefficient (FLC) form [16], defined by $K_1 = q$ and $K_2 = 0$. The order q ranges from 1 and 5. The coefficients are fixed by the method type, its order, the recent history of the step sizes and the normalization $\alpha_{n,0} = -1$. The standard CVODE chemical solver in CANTERA effectively handles thermochemical states using implicit and combined stiff/non-stiff solvers [105].

2.1.3 Expression of chemical reaction rates

Within the system of equations 2.1.1, the temperature T and the chemical reaction rates for each species $\dot{\omega}_s$ need to be explicitly expressed in terms of the mass fractions of species Y_k and the temperature T.

Regarding chemical reactions, in the case of reversible reactions, the chemical species are both produced and consumed simultaneously. The rate at which a species is produced or consumed is determined by the forward and backward rate constants, respectively. When the forward and backward reaction rates are balanced, it indicates that the chemical reaction has reached an equilibrium state. The r^{th} reaction in the mixture can be expressed as follows

$$\sum_{s=1}^{N} \nu_{sr}^{'} X_s \underset{k_{br}}{\overset{k_{fr}}{\rightleftharpoons}} \sum_{s=1}^{N} \nu_{sr}^{''} X_s \tag{2.1.8}$$

X_s represents the chemical species and $\nu_{sr}^{'}$ and $\nu_{sr}^{''}$ represent the stoichiometric coefficients of species s in the forward and backward directions of reaction r, respectively. If a chemical species is not involved in the particular reaction r, the corresponding coefficients are set to zero. The rate constants in the forward and backward directions are denoted by k_{fr} and k_{br}, respectively. In the absence of reversible reactions, the value of k_{br} is zero. This formulation is applicable to any mixture reaction type.

In most combustion scenarios, k_{fr} is represented by a modified Arrhenius law:

$$k_{fr} = A_r T^{\beta_r} e^{\frac{-E_r}{RT}} \tag{2.1.9}$$

In this equation, A_r, β_r, and E_r represent the pre-exponential factor, the temperature exponent of reaction r, and the activation energy of the reaction, respectively. For the backward direction of the chemical reaction r, the backward rate constant, k_{br}, can be expressed in terms of the forward rate constant, k_{fr}, as follows:

$$\begin{cases} k_{br} = \frac{k_{fr}}{K_r} \\ K_r = \exp\left(\sum_{s=1}^{N} (\nu_{sr}^{''} - \nu_{sr}^{'}) \left(\frac{S_s^0}{R} - \frac{h_s^0}{RT} \right) \right) \left(\frac{p_0}{RT} \right)^{\sum_{s=1}^{N} (\nu_{sr}^{''} - \nu_{sr}^{'})} \end{cases} \tag{2.1.10}$$

Here, the constant K_r denotes the equilibrium constant of the reaction r, which is a function of the atmospheric pressure p_0, the molar absolute entropies S_s^0, and the species molar enthalpies h_s^0. The expressions for S_s^0 and h_s^0 are given by NASA polynomial fits:

$$\begin{cases} \frac{h_s^0}{R} = a_{k1}T + \frac{a_{k2}}{2}T^2 + \frac{a_{k3}}{3}T^3 + \frac{a_{k4}}{4}T^4 + \frac{a_{k5}}{5}T^5 + \frac{a_{k6}}{T} \\ \frac{S_s^0}{R} = a_{k1}\log(T) + a_{k2}T + a_{k3}T^2 + a_{k4}T^3 + a_{k5}T^4 - \frac{a_{k6}}{T} + a_{k7} \end{cases} \tag{2.1.11}$$

The molar production rate of species s can be written as the sum of the contributions from all reactions, which is:

$$\dot{\omega}_s = \sum_{r=1}^{R} \dot{\omega}_s^r = \sum_{r=1}^{R} \nu_{sr} q_r = \sum_{r=1}^{R} (\nu_{sr}^{''} - \nu_{sr}^{'}) q_r \tag{2.1.12}$$

where q_r and ν_{sr} are the rate of progress of reaction r and the stoichiometric coefficient of species s in the reaction. The term q_r can be expressed as:

$$q_r = k_{fr} \prod_{s=1}^{N} \left(\frac{\rho Y_s^*}{M_s} \right)^{\eta_{sr}^{'}} - k_{br} \prod_{s=1}^{N} \left(\frac{\rho Y_s}{M_s} \right)^{\eta_{sr}^{''}} \tag{2.1.13}$$

where $\eta_{sr}^{'}$ and $\eta_{sr}^{''}$ are constants representing the forward and backward rate exponents of species s in the reaction r, which depend on the considered chemical mechanism.

2.1.4 Expression of thermodynamic variables

In the 0D system, the density ρ is a time-dependent variable and can be computed using the ideal gas state equation:

$$\rho = \frac{P}{RT} \tag{2.1.14}$$

where R is the universal gas constant, and P and T are known quantities in this system. Since the system consists of mixtures of different gases, the mean specific heat of the mixtures, C_p, can be expressed using the molar heat capacities of all species:

$$C_p = \sum_{s=1}^{N} C_{p,s} Y_s = \sum_{s=1}^{N} C_{p,s}^m \frac{Y_s}{M_s} \tag{2.1.15}$$

where $C_{p,s}^m$ is the molar heat capacity of species s. This term can be determined and implemented using NASA polynomial fits, which represent the molar heat capacity as a function of the mixture's temperature:

$$C_{p,s}^{m} = R(a_k + a_{k2}T + a_{k3}T^2 + a_{k4}T^4) \tag{2.1.16}$$

The computation of the thermo-chemical variables can be easily performed using modern codes and libraries, such as Cantera for homogeneous auto-ignition 0D simulations or steady 1D flamelet simulations. Nowadays, different chemistry solvers for 2D/3D problems are implemented in academic or commercial solvers. As chemical reactions occur across various timescales—from extremely small scales, much shorter than CFD resolution time steps, to larger timescales, comparable or even exceeding CFD resolution time steps—traditional numerical chemistry solvers handle the resolution of chemical states at different time steps. However, this approach can be computationally expensive for complex chemistry problems. Therefore, our objective is to utilize a deep learning-based surrogate model to replace the multi-time-step resolution process of the chemistry solver, aiming to accelerate computations.

2.1.5 Adaptive chemistry resolution time steps

By adaptively resolving the stiff and non-stiff regions with different time steps, the CVODE solver 2.1.7 refines the step sizes in areas with rapid reaction rates. The absolute and relative numerical tolerances of CVODE are set to 1.0×10^{-9} and 1.0×10^{-15}, respectively. Figure 2.2 illustrates the simulations of temperature with dynamically adjusted time steps during the temporal evolution of H_2, C_2H_4, and CH_4 cases, where the x axis for time evolution is plotted under the logarithmic scale. The initial temperature and equivalence ratio are set to $(T_0, \phi) = (1700K, 1.0)$. It can be observed that the CVODE solver refines the time step dt_{cvode} in fast-reacting regions while using larger time steps in the starting and equilibrium regions. As for rapid reaction during fast ignition process, the chemistry resolution time steps are much smaller than the CFD resolution time step during multidimensional simulations. Therefore, the adaptive time steps can be applied to augment the dataset for machine learning to include more information of ignition process, thus to improve the robustness of dataset. We will discuss this data generation methodology in chapter 3.

Furthermore, due to the magnitude of chemical species varying at extremely small scales around zero, we incorporate certain data pre-processing techniques to capture the evolution of chemical states for the transformed data. In the next section, we will introduce the basic methodologies of machine learning and their application in predicting chemical states.

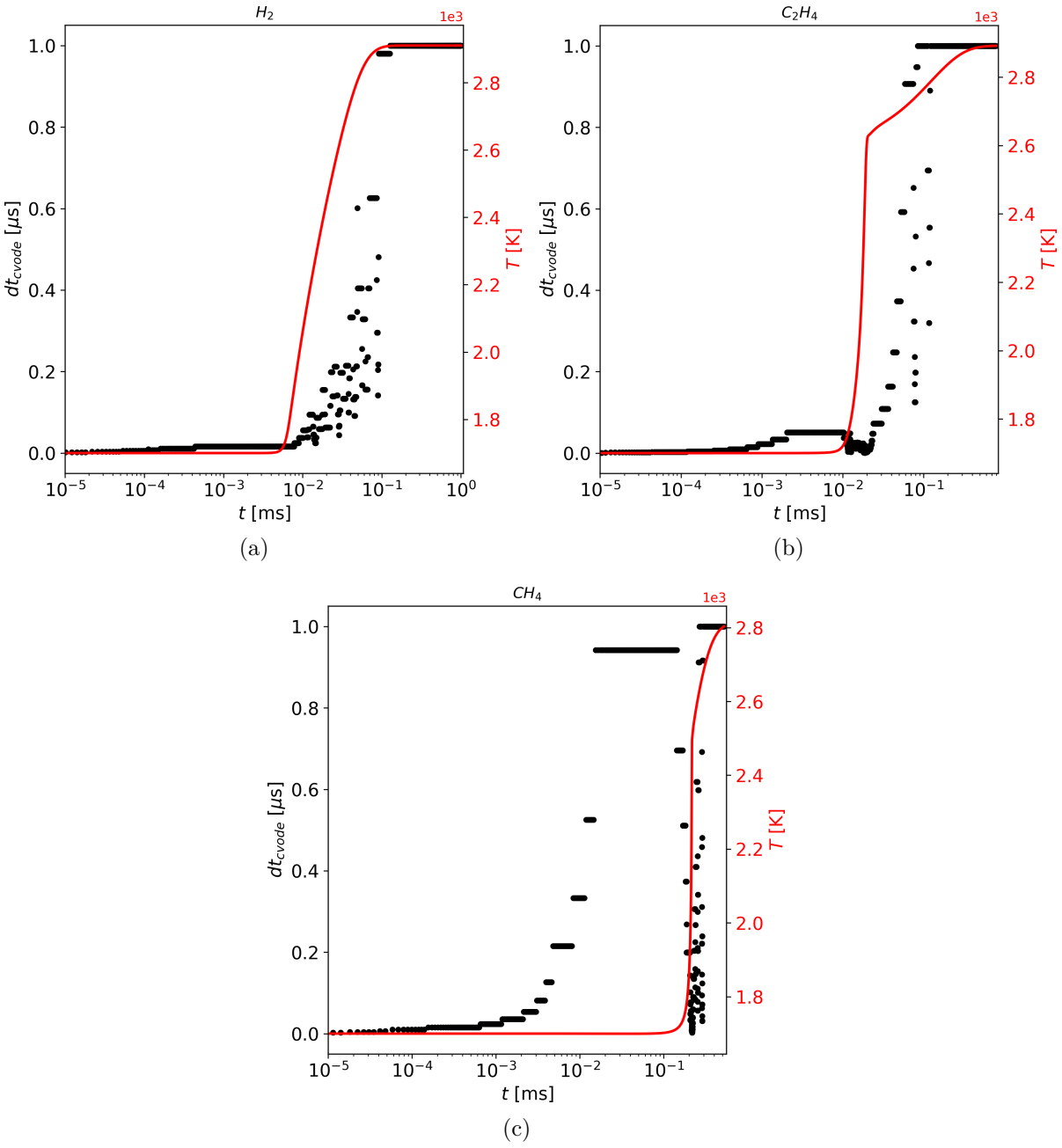


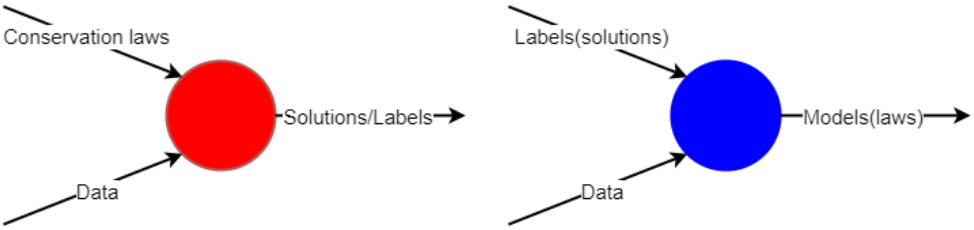
Figure 2.2: Dynamic adaptive time steps used by CVODE solver with (a)H_2/air, (b)C_2H_4/air, and (c)CH_4/air cases, where the simulations are set with $T_0 = 1700.0K$ and $\phi = 1.0$ The red lines represent the numerical solution of temperature, and the black dot points represent the local adaptive time steps given by CVODE solver

2.2 Machine learning and its applications for chemistry predictions

Machine learning is a modern technique to derive intrinsic laws of the system from data, thus being also referred to as data-driven science. Instead of determining universal laws from first principles, data scientists typically use surrogate models with trainable parame-

ters, and feed these models with data and labels to automatically explore the general rules within these pre-obtained data. The models with well-trained parameters can be used to interpolate unseen data labels. A brief comparison of paradigms using physics-based method and data-driven machine learning methods is represented in figure 2.3.

Different machine learning workflows can be used for different problems. In this work we apply supervised learning and unsupervised learning techniques. Supervised learning methods learn from a collection of labeled data $\{(x, y)_i\}_{i=1}^{N}$, predict outputs $\hat{\mathbf{y}}$ based on inputs $\mathbf{x}$ through a well-defined machine learning surrogate application $f(\mathbf{x}; \theta)$, where θ represents the model parameters. Problems such as regression and classification can be solved by applying supervised machine learning algorithm, such as linear regression, logistic regression, support vector machine (SVM), artificial neural network (ANN), etc. Unsupervised learning, on the other hand, extracts knowledge from unlabeled data $\{x_i\}_{i=1}^{N}$. Algorithms based on this workflow, such as Principle components analysis (PCA), KMeans, Gaussian mixture model (GMM), can be used for clustering and dimension reduction. One may note that different machine learning methodologies and algorithms are applied for different target problems, and some combinations of multiple machine learning approaches may dramatically improve the efficiency of a given learning workflow. In the following paragraphs of this sections, a general introduction of several machine learning methods used in this research is provided.



(a) Physics-based approach (b) Data-driven approach

Figure 2.3: Schematics for comparisons of knowledge discovering paradigm between (a) physics-based approach, (b) data-driven approach

2.2.1 Supervised regression learning with neural networks

Regression model

We consider n points of observation $\{x_1, x_2, ..., x_n\}$ and their target labels (or values to interpolate) $\{y_1, y_2, ..., y_n\}$ in the relative spaces $\mathcal{X}$ and $\mathcal{Y}$. These are considered as the training data to approximate a mapping ϕ from $\mathcal{X}$ to $\mathcal{Y}$ by a surrogate model $\mathcal{F}$, such that:

$$y_i = \phi(x_i) + \epsilon_i \qquad i \in \{1, ..., N\} \tag{2.2.1}$$

where ϵ is the observation noise of the surrogate model. The objective is to fit the surrogate model $\mathcal{F}$, that is to optimize the parameters of the surrogate model in order to better approximate the target mapping from $\mathcal{X}$ to $\mathcal{Y}$.

After the parameterized learning models are constructed, a loss function is defined to evaluate the ground-truth values and predicted values based on regression models. The loss function, mean squared error, which need to be optimized can be defined by:

$$\mathcal{L}_{MSE} = \frac{1}{K}\sum_{j=1}^{K}(\frac{1}{N}\sum_{i=1}^{N}(\hat{S}_i^j - S_i^j)^2) \tag{2.2.2}$$

where K and N are the numbers of data samples and species respectively, $\hat{S}_i^j$ and S_i^j represent the predicted and the reference output values in the j^{th} sample. The MSE loss function is widely used as a loss function in deep learning problem. The loss function can also be completed by additional information terms (i.e. physical informed soft constraint) to introduce the target information, or L_1 and L_2 regularization terms to tackle the overfitting problem. Thus the total loss function is denoted as:

$$L = L_{MSE} + L_{soft} + L_1(\theta) + L_2(\theta) \tag{2.2.3}$$

The objective of the learning procedure is to optimize the internal parameters based on the minimization of the loss function. It is also called "training" process. The most common optimization algorithm is the gradient descent, which computes the objective function's derivatives with respect to parameters $\mathbf{P}$ of the learning model:

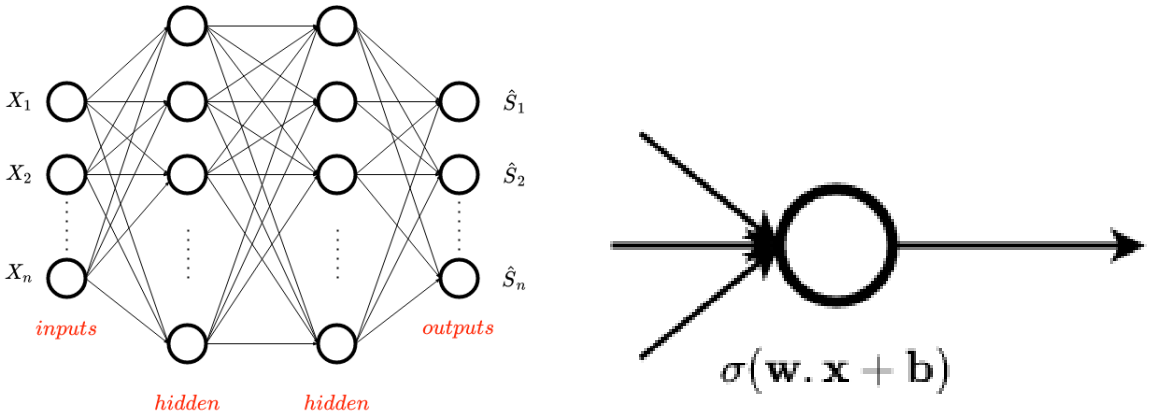
$$\mathbf{P}^{(t)} = \mathbf{P}^{(t-1)} - \eta\frac{\partial\mathcal{L}(X, \mathbf{P}^{(t-1)})}{\partial\mathbf{P}} \tag{2.2.4}$$

Here the η represents the learning rate, and X is the raw input data from each batch of training dataset. When utilizing gradient descent, the computational cost per iteration for each independent variable is $\mathcal{O}(N)$, resulting in a linear growth with respect to the dataset size N. Consequently, when dealing with larger training datasets, the expense associated with each gradient descent iteration becomes higher. Conventional methods to tackle this problem is to use mini-batch gradient descent algorithm or stochastic gradient descent algorithm (SGD) [106].

Neural network construction

Artificial Neural Networks (ANN) are at the very core of deep learning, which is powerful and suitable for complex machine learning tasks. In this study, the objective is to use ANNs as surrogate models for chemistry computations, in order to accelerate the solving. This amounts to performing a regression task, for which ANNs are well-suited. It is theoretically demonstrated that an ANN consisting of at least one hidden layer can approximate any given continuous function on any compact subset of $\mathbf{R}^n$ according to

the universal approximation theorem [52]. The ANN is composed of several layers and each layer includes a certain number of neurons. The general topology of ANN contains an input layer, several hidden layers, and an output layer.



(a) A 2 hidden layers topology of ANN model

(b) linear combination with an activation function for one neuron

Figure 2.4: Neural network structure

A typical ANN structure is represented in Figure 2.4(a), where the vector $\mathbf{X}(t) = [X_1, ..., X_n]$ represents the input variables, and $\hat{\mathbf{S}}(t) = [\hat{S}_1, ..., \hat{S}_n]$ are the output predicted variables. The ANN performs as a surrogate model to learn the regression function $f : \mathbf{X} \rightarrow \hat{\mathbf{S}}$. The training of ANN consists in forward and backward propagation. The forward propagation predicts the values based on the given data, and then computes the loss function (objective function) which describes the difference between the predicted and true values of output. Each neuron in the hidden layers are activated by the activation function as shown in Figure 2.4(b). Some different type of activation functions are shown in Figure 2.5, and the choice of activation functions for the model is also a key factor to be fixed during the training process.

Model parameters optimization

The backward propagation algorithm [106] is applied to compute the gradient of the loss function with respect to the weights and biases of ANN, which can then be updated. Up to now, there are much more advanced optimization algorithms which can efficiently find the optimized parameters without the gradient explosion and vanishing problem, such as gradient descent with momentum. One of the most popular optimization algorithm for deep learning is the Adam algorithm [107], which is used in our research case. The Adam algorithm is expressed as follows

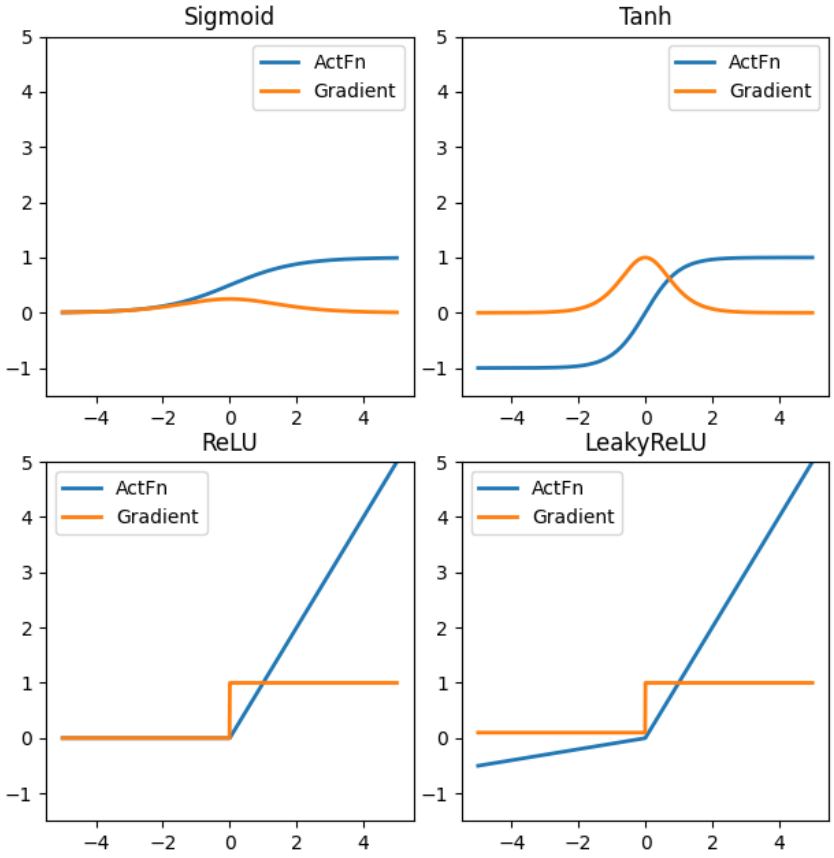


Figure 2.5: 4 different activation functions which are mostly used for neural networks, where the blue color represents the original functions and the orange color shows the gradients

$$
\begin{aligned}
m^{(t)} &= \beta_1 m^{(t-1)} + (1-\beta_1) g^{(t)} \\
v^{(t)} &= \beta_2 m^{(t-1)} + (1-\beta_2)(g^{(t)})^2 \\
\hat{m}^{(t)} &= \frac{m^{(t)}}{1-\beta_1^t}, \quad \hat{v}^{(t)} = \frac{v^{(t)}}{1-\beta_2^t} \\
\mathbf{P}^{(t)} &= \mathbf{P}^{(t-1)} - \frac{\eta}{\sqrt{\hat{v}^{(t)}} + \epsilon} \times \hat{m}^{(t)}
\end{aligned}
\tag{2.2.5}
$$

where $g^{(t)}$ is the gradient of loss function, m and v are the moment vector based on the momentum optimization theory. $\mathbf{P}$ is parameters of learning model. ϵ is a small constant, which is used to improve the numerical stability for extremely small gradient norms. More information about this algorithm and fundamental theory of momentum optimization can be found in [106].

2.2.2 Neural networks formulations

Discrete neural network

Neural networks, particularly multiple-layer perceptrons (MLP), are popular due to their simplicity and computational efficiency and are widely used for combustion chemistry simulations [54, 55]. However, when dealing with highly complex systems, accurate regression using MLPs may require a significant number of parameters. This can lead to optimization challenges, such as the vanishing gradient problem commonly encountered in machine learning applications [106]. The vanishing gradient problem refers to the issue where the gradients during backpropagation diminish as they propagate through many layers, making it difficult for the network to learn effectively. To address this challenge, an alternative approach used in this work is the application of residual layer structures with shortcut connections, known as ResNet [108]. Similar topologies have been utilized in various physico-chemistry computations, such as fluid flash computations [109] and flamelet progress variables tabulation [63]. The original ResNet architecture, which is called vanilla ResNet, denoted as "ResMLP" in this research, has been shown to mitigate the vanishing gradient problem and enable efficient learning for complex problems. Figure 2.6 provides an illustration of the ResMLP architecture.

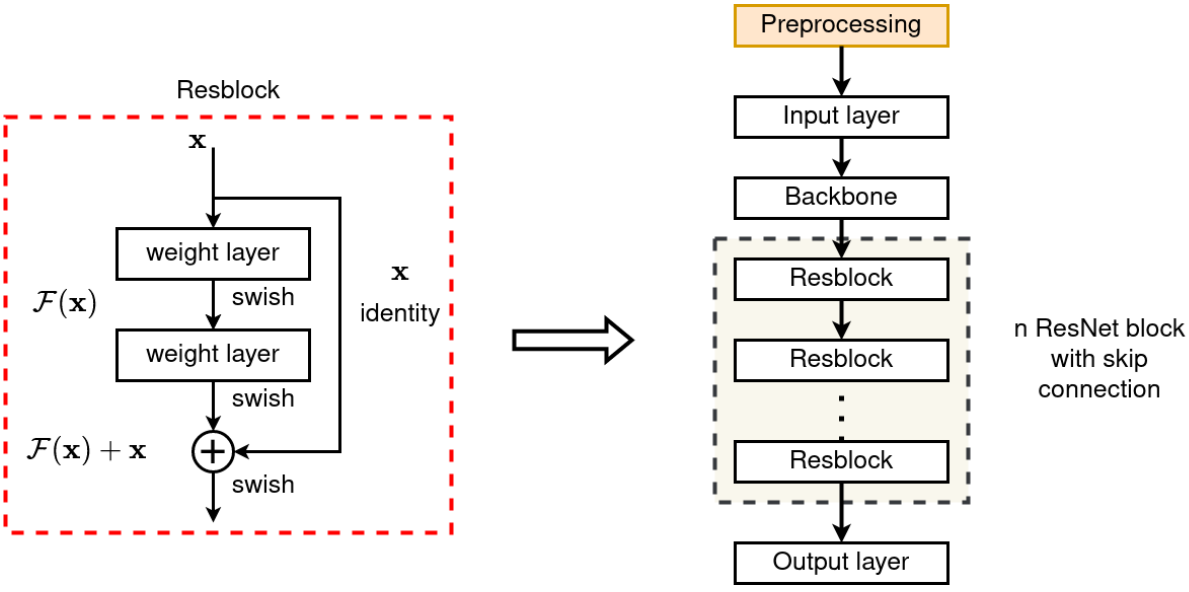


Figure 2.6: ResMLP model structure design

The ResNet architecture in our research consists of basic units called resblocks, which include two standard hidden layers. The key principle of ResNet is the inclusion of shortcut connections that bypass one or more layers, allowing each layer to predict an increment rather than a direct value. This enables the network to learn residual information, which facilitates the training process. If x is the input of a resblock, the output y is given by:

$$\mathbf{y} = \sigma(\mathcal{F}(\mathbf{x}) + \mathbf{x}) \tag{2.2.6}$$

where $\mathcal{F}(\mathbf{x})$ is the feed-forward neural network composed of the two hidden layers and σ represents the non-linear activation function.

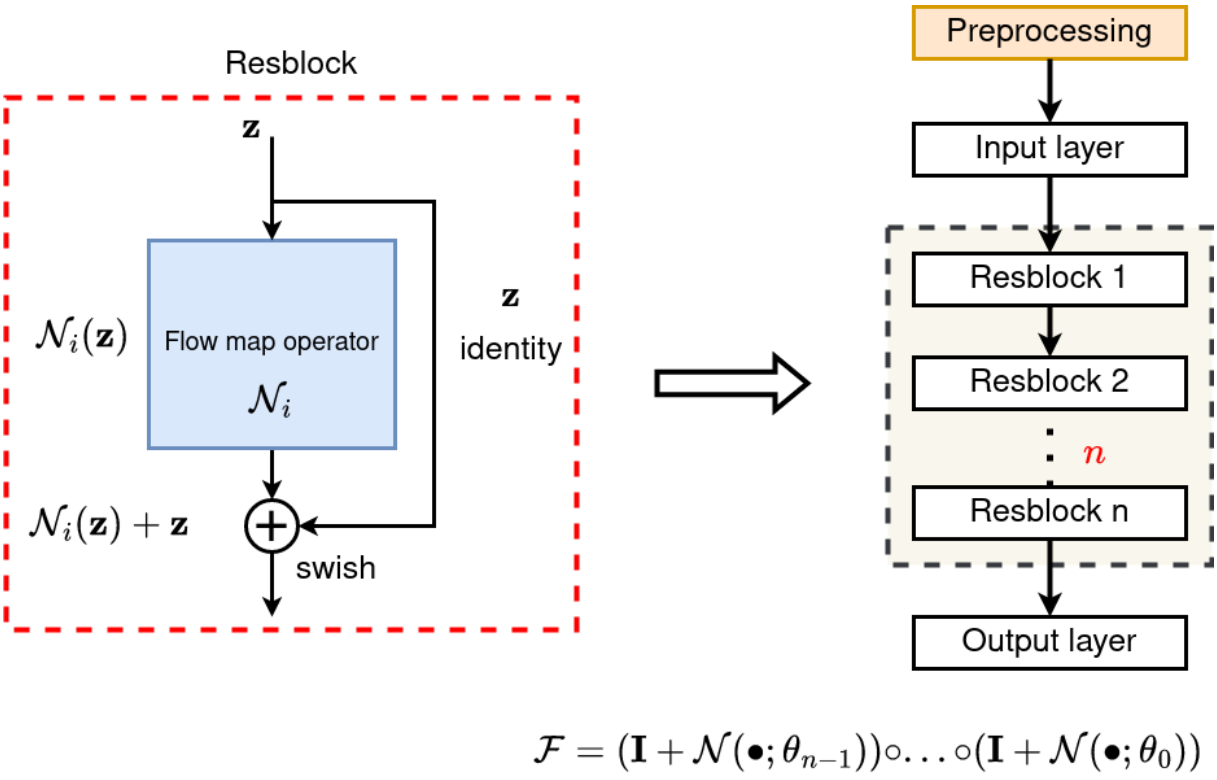


Figure 2.7: The basic structure of discrete residuals learning for flow maps

More recently, many studies have demonstrated that residual networks perform like the discrete approximation of continuous, finite-dimensional dynamical systems. The conception of the residual block comes from the intuition of explicit numerical schemes. The integral form of such an ODE reads

$$\begin{aligned} \mathbf{S}(t + \Delta t) &= \mathbf{S}(t) + \int_t^{t+\Delta t} h(\mathbf{S}(t), r)\mathrm{d}t \\ \mathbf{S}(t_0) &= \mathbf{S}_0 \end{aligned} \tag{2.2.7}$$

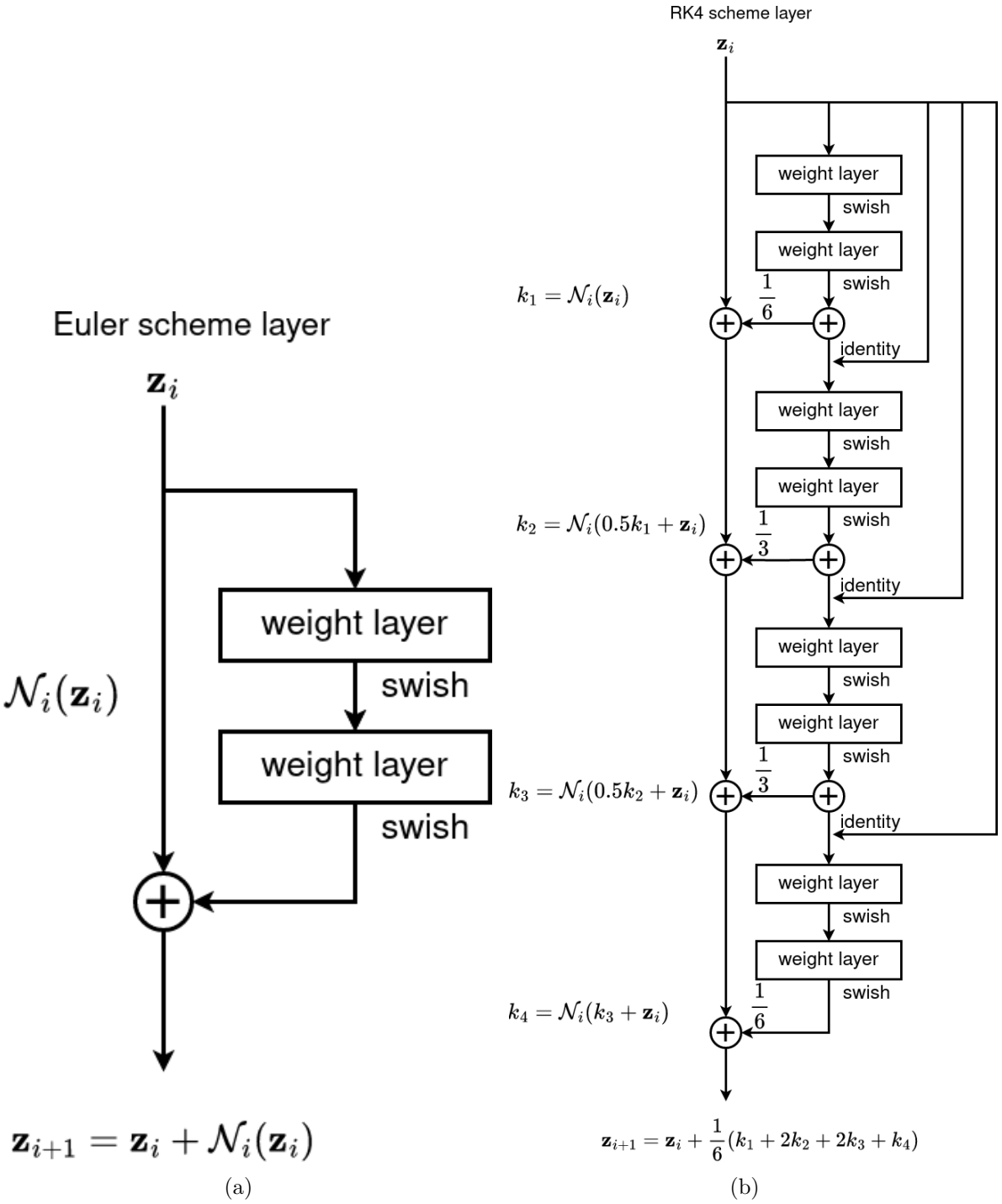


Figure 2.8: The Euler Network and the RK4 Network for a single residual block design

As a mathematical definition, the temporal mappings from state in t to state in $t+\Delta t$ is called flow maps [7, 110], hence the learning of this mapping is also called "flow map learning". Based on the integral form in Equation (2.2.7), the fundamental concept is to acquire knowledge of the effective discrete increments from $\mathbf{S}(t)$ to $\mathbf{S}(t+\Delta t)$ within the dynamical system. This is achieved by approximating the time integrals through effective increments along the trajectory of the dynamical system. For systems for which the flow map is locally Lipschitz continuous from t to $t+\Delta t$(which is typically the case

of combustion dynamics), one can approximate the dynamics by learning the discrete residuals over multiple n steps, denoted as:

$$\begin{aligned} &\mathbf{S}(t+\Delta t) \approx \mathcal{F}(\mathbf{S}(t)) \\ &\mathcal{F} = (\mathbf{I} + \mathcal{N}(\bullet; \theta_{n-1})) \circ ... \circ (\mathbf{I} + \mathcal{N}(\bullet; \theta_0)) \end{aligned} \tag{2.2.8}$$

Here, θ represents the parameter set of the neural networks. Each $\mathcal{N}_i$ denotes an element block of models, responsible for learning intermediate small time step increment δ_k for $k = 0, ..., n-1$. The general concept of discrete flow map learning is depicted in Figure 2.7, where the flow maps are segmented using an identity and an increment operator. This increment operator is discretized through elemental residual blocks. As the parameters are not shared across each element block, the intermediate time steps for small flow maps are not explicitly known and do not need to be of equal intervals. Therefore, this approach approximates the governing equations by learning the small effective increments within δ_k. Compared to the ResMLP model, in the case of each resblock, the input dimension number corresponds to the number of chemical input states, rather than the number of neurons in the hidden backbone layer.

Two different designs of the elemental residual block are provided based on the discrete flow map learning. Denoting the input state of an intermediate interval by z_i and the output state by z_{i+1}:

1. Euler scheme of first order:

$$z_{i+1} = z_i + \mathcal{N}_i(z_i; \theta_i) \tag{2.2.9}$$

2. Runge-Kutta scheme of fourth order:

$$\begin{aligned} k_1 &= \mathcal{N}_i(z_i; \theta_i) \\ k_2 &= \mathcal{N}_i(\frac{1}{2}k_1 + z_i; \theta_i) \\ k_3 &= \mathcal{N}_i(\frac{1}{2}k_2 + z_i; \theta_i) \\ k_4 &= \mathcal{N}_i(k_3 + z_i; \theta_i) \\ z_{i+1} &= z_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \tag{2.2.10}$$

$\mathcal{N}_i$ is the feed-forward neural networks with trainable parameters. Brief representations of the residual networks design are given by Figure 2.8. The networks based on the intuition of Euler scheme is denoted by Euler network, and the networks designed from Runge-Kutta fourth scheme is denoted by RK4 network in the rest of the thesis. Detailed theoretical analysis of error estimation for Euler network and RK4 network was originally investigated in [7][84].

For more complex cases, it is challenging to learn the global system with a single regression model. In fact, several techniques based on unsupervised learning can be applied to improve the performance of learning workflow, such as data clustering and dimension reduction. This is discussed further in Section 2.2.3.

Neural ordinary differential equation

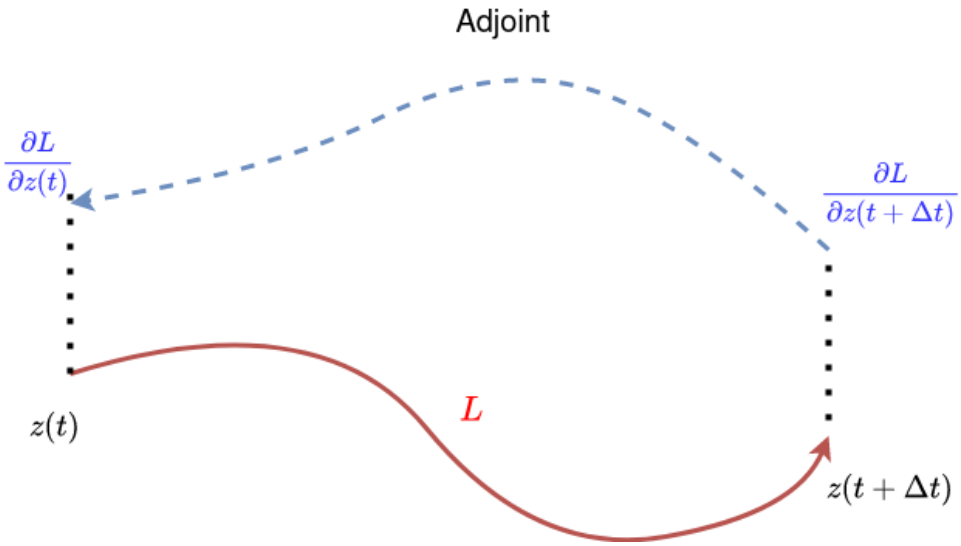


Figure 2.9: The adjoint sensitivity method for reverse-mode differentiation

In addition to learning the flow maps through increment approximation, an alternative approach involves approximating the continuous latent dynamic variable h using Neural Ordinary Differential Equations (NeuralODE) [8]. The fundamental concept behind Neural Ordinary Differential Equation is to train a surrogate dynamics model based on a neural network. This model approximates the right-hand side of the ODE in real dynamical systems. The continuous trajectory of dynamic is learned and approximated by NeuralODE models.The training of this surrogate continuous dynamic variable is based on reverse-mode differentiation, where gradients are computed using the adjoint sensitivity method.

As illustrated in Figure 2.9, let us denote a surrogate continuous dynamics variable as h_θ, which is constructed using a parameterized neural network. When considering the optimization of a loss function derived from a continuous autonomous flow, it can be formulated as follows:

$$L(\mathbf{z}(t+\Delta t)) = L\left(\mathbf{z}(t) + \int_t^{t+\Delta t} h_\theta(\mathbf{z}(t), \theta)\mathrm{d}t\right) \tag{2.2.11}$$

In this loss function $L()$, the original input is the continuous states along the dynamic flow. However, one can apply numerical methods to resolve the ODE. Thus, numerically the input can be the solution of an ODE solver from t to $t + \Delta t$:

$$L(\mathbf{z}(t+\Delta t)) = L(\text{ODESolve}(\mathbf{z}(t), h_\theta, t, t+\Delta t, \theta)) \tag{2.2.12}$$

From this formulation, it can be seen that the forward propagation is the direct resolution of parameterized dynamic (which is approximated by neural networks). Nevertheless, the backward differentiation mode is not the conventional backward propagation for neural networks. The gradients with respect to θ for loss function of continuous flow is computed based on the adjoint, which is denoted as:

$$\mathbf{a}(t) = -\frac{\partial L}{\partial \mathbf{z}(t)} \tag{2.2.13}$$

In fact, the adjoint dynamics can be derived from the system dynamics and reads:

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^T \frac{\partial h_\theta(\mathbf{z}(t), \theta)}{\partial \mathbf{z}(t)} \tag{2.2.14}$$

where the gradients with respect to input and output time are computed by solving this equation in reverse time, starting from time $t + \Delta t$. The gradient of loss function L with respect to θ is computed as:

$$\frac{\partial L}{\partial \theta} = -\int_{t+\Delta t}^{t} \mathbf{a}(t)^T \frac{\partial h_\theta(\mathbf{z}(t), \theta)}{\partial \theta} dt \tag{2.2.15}$$

The detailed reverse-mode differentiation which is proposed in [8] is introduced in Appendix A, where the computation of gradients of loss function with respect to model parameters, the input time states, the input time and the output time is based on the resolution of an augmented state dynamical system. The implementation of this algorithm is performed *torchdiffeq* library, which is also developed in [8]. In this work, the latent dynamic model with optimization algorithm is also implemented using this library.

2.2.3 Data clustering and dimension reduction

Unlike supervised learning, unsupervised learning algorithms learn patterns from unlabeled data. The objective of unsupervised learning is to partition the data, or extract the low dimensional latent features. In this thesis work, clustering and dimension reduction algorithms are used to construct the learning workflow. Clustering algorithms enable the separation of the dataset into subdomains, where local supervised regression models can be trained for each sub-dataset. Besides, dimension reduction techniques help us to find the principal latent components which intrinsically dominates the full system's characteristics. These techniques are used to improve the learning performance in this work, and we now briefly describe them.

KMeans algorithm

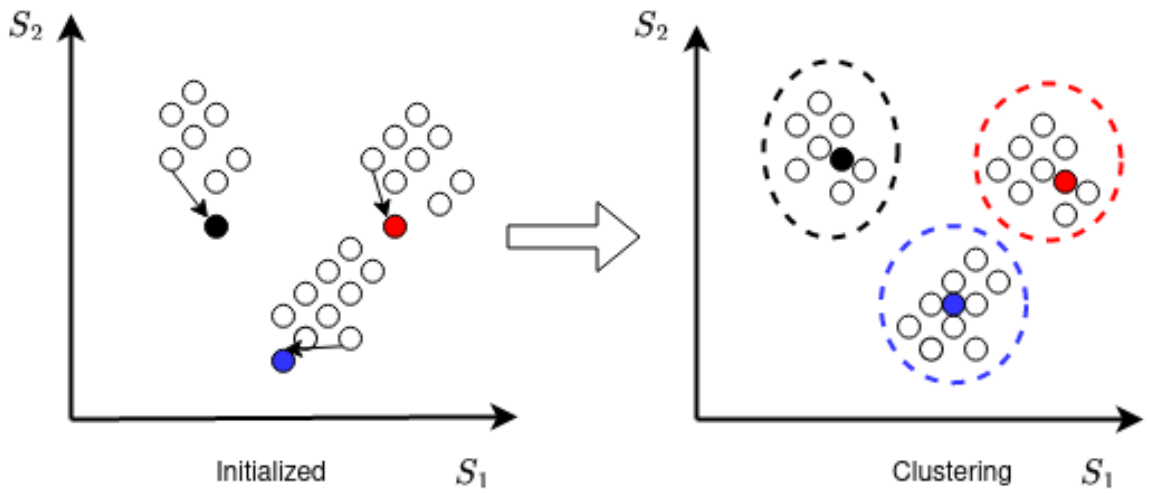


Figure 2.10: Clustering by K-Means algorithm

The K-Means algorithm, as schematically depicted on Figure 2.10, is a clustering algorithm that partitions a set of N sampling points based on their similarities, aiming to minimize the average squared distance between K centroids and the sampling points. Given a set of N observed sampling points $\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_N$, the algorithm iteratively minimizes the within-cluster sum of squares (WCSS) as the loss function. The WCSS is defined as the sum of the squared Euclidean distances between each sampling point and its assigned centroid, which is denoted as:

$$\mathcal{L}_{kmeans} = \sum_{i=1}^{K} \sum_{\mathbf{S} \in \Omega_i} \left\| \hat{\mathbf{S}}_l - \mathbf{d}_i \right\|_2 \tag{2.2.16}$$

where the $\mathbf{d}_i$ denotes the centroid of each cluster, $\hat{\mathbf{S}}$ represents the data normalization. The K-Means++ algorithm[111] is an improved version of the standard K-Means algorithm that addresses the issue of poor clusterings that can occur with the standard approach. It introduces a more effective initialization step for selecting the initial centroids. The steps of the centroids initialization by the K-Means++ algorithm are as follows:

- Initialize the first centroid by randomly selecting one data point from the dataset, while each points are as far apart from each other as possible.
- For each remaining data point, compute its distance to the nearest centroid.
- Select the next centroid from the remaining data points with a probability proportional to the square of the distance to the nearest centroid. This ensures that points further away from existing centroids are more likely to be selected as new centroids.

- Repeat steps 2 and 3 until K centroids are selected.

By using the K-Means++ algorithm for initialization, the K-Means clustering process starts with more representative initial centroids, leading to better overall clusterings. This helps to avoid situations where the algorithm gets stuck in suboptimal solutions or produces unbalanced clusters.

Singular value decomposition and Principle component analysis

Singular Value Decomposition (SVD) consists in factorizing a matrix A into the product of three matrices as follows

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T \in \mathbb{R}^{m\times n} \tag{2.2.17}$$

where $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ are orthogonal, and Σ is a rectangular, positive diagonal matrix. The diagonal entries of Σ are called singular values and are usually in decreasing order along the diagonal. One of the main application of SVD is to represent a matrix $\mathbf{A}$ as a sum of low-rank matrices $\mathbf{A}_i$, and this allows to reconstruct the original matrix by matrix approximations [112]. A matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ of rank r can be constructed as a sum of $\mathbf{A}_i$ as:

$$\mathbf{A} = \Sigma_{i=1}^{r}\sigma_i\mathbf{u}_i\mathbf{v}_i^T = \Sigma_{i=1}^{r}\sigma_i\mathbf{A}_i \tag{2.2.18}$$

As for rank $k < r$, a rank-k approximation for matrix $\mathbf{A}$ is:

$$\hat{\mathbf{A}}(k) = \Sigma_{i=1}^{k}\sigma_i\mathbf{A}_i \tag{2.2.19}$$

Based on this approximation, the Principle Component Analysis (PCA) algorithm can be derived. For a dataset $\mathbf{X} \in \mathbb{R}^{n\times r}$ with n samples of r labels, PCA can be used to approximate the original dataset using only $k(k < r)$ linear correlations between the r labels. The covariance matrix of samples is computed as:

$$\mathbf{S} = \frac{1}{n-1}\mathbf{X}^T\mathbf{X} \tag{2.2.20}$$

The covariance matrix is then decomposed by eigenvalues decomposition (ED), denoting by:

$$\mathbf{S} = \mathbf{D}\mathbf{L}\mathbf{D}^T \tag{2.2.21}$$

In this equation, the $\mathbf{D}$ is the eigenvectors of $\mathbf{S}$, and $\mathbf{L}$ is the diagonal matrix of eigenvalues of $\mathbf{S}$. The PCs are computed as:

$$\mathbf{Z} = \mathbf{X}\mathbf{D} \tag{2.2.22}$$

That is to say, the original dataset space can be constructed based on PCs, as $\mathbf{X} = \mathbf{Z}\mathbf{D}^{-1}$. If only first $k(k < r)$ PCs are used, the dataset can be reconstructed approximately as:

$$\mathbf{X} \approx \mathbf{Z}_k\mathbf{D}_k^{-1} \tag{2.2.23}$$

By this means, the original dataset with r labels can be reduced to k labels by evaluating the PC scores scale. The PC scores scale are also the same scale as eigenvalues/singular values. Nevertheless, PCA is a linear dimension reduction technique, and when the data variables is linearly projected to the low dimensions, it may lose non-linear information which is important in the original dataset. To tackle this problem, one can use varied form of PCA such as kernel PCA method. However, the most popular non-linear dimension reduction technique is the autoencoder model.

Autoencoder

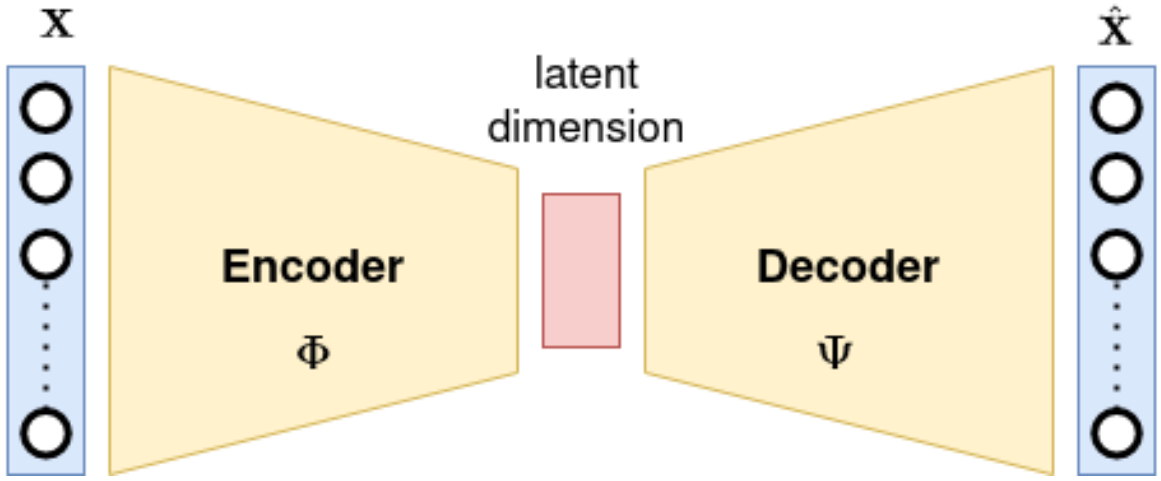


Figure 2.11: Autoencoder model structure

An Autoencoder is a non-linear dimension reduction model based on deep neural networks with non-linear activation functions. Compared to linear PCA, when using the same number of reduced dimension k, an autoencoder may preserve more non-linear information, and thus can better reconstruct the original dataset space. Figure 2.11 gives a general structure of an autoencoder. It contains an encoder Φ which is used to project the original vectors to latent vectors with low dimension non-linearly. Then, a decoder Ψ is used to re-project the latent space vectors to original space. The training of the autoencoder model by optimizing the parameters is similar to the training of the ANN model presented in the previous section. The loss function to be optimized is:

$$\mathcal{L}_{AE} = \sum_{i=1}^{n} \|\Psi(\Phi(\mathbf{X}_i)) - \mathbf{X}_i\|_2 \tag{2.2.24}$$

The purpose of the loss function is to minimize the error between the real data vectors and the reconstructed data vectors generated by the autoencoder. Once the autoencoder is trained, it can be employed in conjunction with other supervised learning algorithms to discern patterns in the latent space. We will incorporate this model for chemistry dimension reduction in Chapter 4.

2.2.4 Neural networks for chemical states predictions

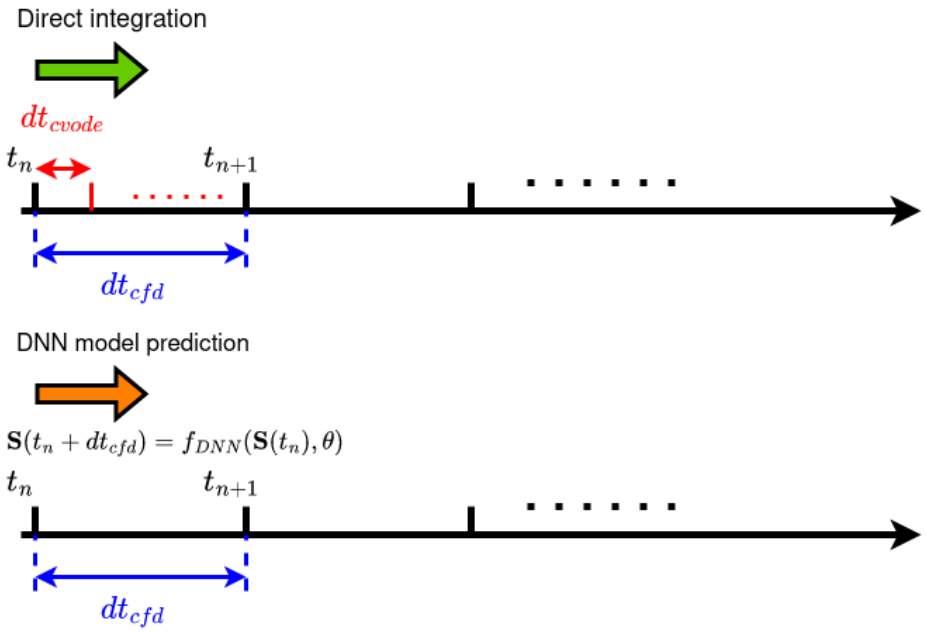


Figure 2.12: Workflow for the states prediction with time evolution

During a 3D simulation with operator splitting, the ODE solver typically performs numerous adaptive time steps within a single computational fluid dynamics (CFD) time step $dt_{cfd} = t_{n+1} - t_n$. The objective in this study is to replace the direct integration of reaction terms by deep ANN models (DNN) for each CFD time step resolution, as illustrated in 2.12. We want to predict the thermochemical states at time $t_n + dt_{cfd}$ as $\mathbf{S}(t_n + dt_{cfd}) = \mathcal{F}(\mathbf{S}(t_n); \theta)$, where θ represents the parameters of the learning model. $\mathbf{S}$ represents the system states defined in equation 2.1.6.

It must be noted that: 1) the temperature is included into the model output. For such systems the temperature might be directly estimated from species mass fractions and enthalpy. In an attempt to make the model more general and not solely tailored for constant pressure low Mach problems, the temperature is included as an additional output state of the DNN in the context of this research 2) the time step is constant. Neural network can be used to fit non-linear applications under different time step conditions.

Besides, for larger time steps, the model training for one time step prediction will be more complicated, as the stiffness for systems within one time step will be larger.

A summary of the DNN-based model architecture is provided in Fig. 2.13. The data samples are firstly partitioned to different clusters based on unsupervised KMeans clustering algorithm, and then pre-processed by nonlinear logarithmic transformation to be learned in the similar magnitude.

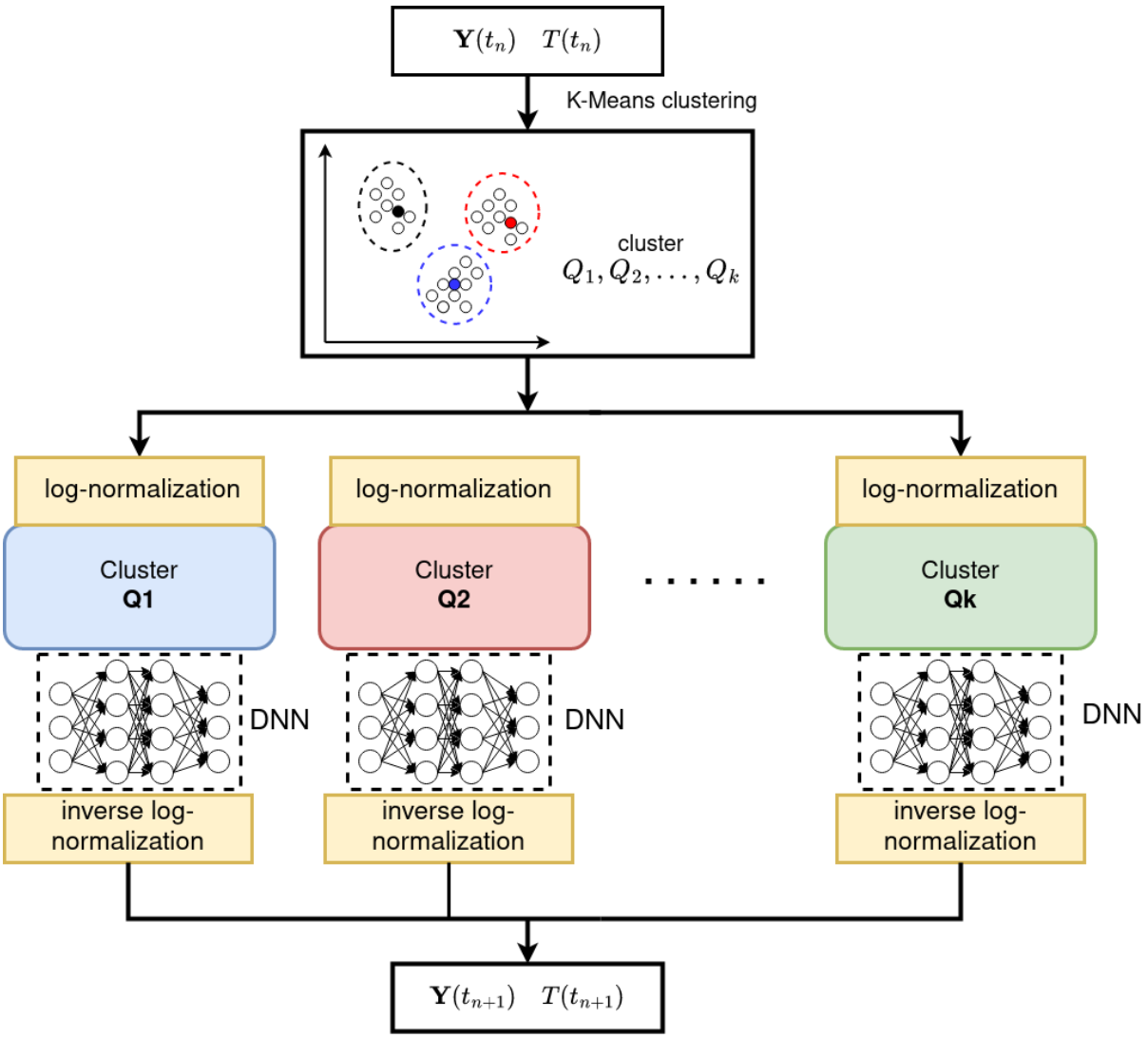


Figure 2.13: The global learning algorithm for the states prediction with the separation of composition space

2.3 Conclusion

In this chapter, the physical model of the combustion system were introduced, serving as the foundation for all simulation scenarios of the present work. Additionally, the necessary machine learning tools are presented, along with the basic principles of data-driven

methods. Based on these general methodologies, the workflow of predicting thermochemical states using machine learning techniques are introduced to replace the conventional numerical simulation of stiff ODE systems. Beginning with the next chapter, our focus will shift to the application of machine learning for predicting thermochemical states for 0D ignition model and 2D unsteady premixed flame model, and implementing dimension reduction techniques for highly complex chemistry mechanisms.

Chapter 3

Machine learning for 0D reactors

In this chapter, we introduce a workflow designed to train learning models using datasets derived from 0D homogeneous reactors. By omitting the convection-diffusion terms in the original 3D system, the governing equations for the 0D system characterize a homogeneous auto-ignition system. Originally this simplified model has been employed in studies for constructing chemical databases [38, 39]. The 0D system captures fast ignition phenomena with stiff chemical reactions. We explore three different chemical mechanisms and implement an adaptive sampling method to ensure a more balanced dataset. Additionally, various neural network architectures, introduced in Chapter 2, are tested for their efficacy.

Résumé français

Dans ce chapitre, nous présentons un flux de travail conçu pour entraîner des modèles d'apprentissage à l'aide de jeux de données dérivés de réacteurs homogènes 0D. En omettant les termes de convection-diffusion dans le système 3D d'origine, les équations gouvernantes du système 0D caractérisent un système d'auto-allumage homogène. À l'origine, ce modèle simplifié a été utilisé dans les études de construction de bases de données chimiques [38, 39]. Le système 0D capture les phénomènes d'inflammation rapide avec des réactions chimiques rigides. Nous explorons trois mécanismes chimiques différents et mettons en œuvre une méthode d'échantillonnage adaptative pour assurer un ensemble de données plus équilibré. De plus, diverses architectures de réseaux neuronaux, présentées dans le chapitre 2, sont testées pour leur efficacité.

3.1 Dataset generation

We consider a 0D homogeneous and adiabatic reactor. Its dynamics are described by Equation (2.1.5). In this work, we consider three different fuel combustion scenarios

involving the fuels H_2, C_2H_4 and CH_4, as outlined in table 3.1.The selection of these cases progressively increases the complexity of the thermochemical reaction system, as detailed in Table 3.1.

Mechanism	number of species	number of reactions
H_2 [113]	9	19
C_2H_4 [114]	32	206
CH_4 [115]	53	325

Table 3.1: Statistics for experiments of C_2H_4/air case

3.1.1 Generation of simulation trajectories

In the context of the 0D homogeneous reactors in this chapter, the resolution time step dt_{cfd} is set to a fixed value of $dt_{cfd} = 10^{-6}$ seconds. In simulations using operator splitting approaches, the time step is usually limited by stability limits on convection and diffusion (CFL and Fourier numbers are typically defined). The dt_{cfd} value selected here is a typical value found in large eddy simulations of systems such as gas turbines or engines. Of course, this value might vary depending on the mesh and flow conditions, and the sensitivity of the method in this research to dt_{cfd} needs to be understood in the future. As for machine learning regressions, neural network can be used to fit non-linear applications under different time step conditions. It must be noted that as for smaller time steps, the iteration number for the same simulation time will be larger and this may lead to more accumulated errors when the simulation time is large. Besides, for larger time steps, the model training for one time step prediction will be more complicated, as the stiffness for systems within one time step will be larger.

The design of experiments to generate the training dataset consists in specifying multiple initial conditions for the temperature and species mass fractions for the set of ODE 2.1.6. However, not all the initial species mass fractions are physically relevant: in this work, we assume that the initial composition is a pure mixture of fuel and air, where the mass fractions for all other species are set to zero. A common practice in combustion is to specify the initial mass fractions values with a more significant variable called the equivalence ratio ϕ, which is a measure of the excess of fuel in the mixture with respect to stoichiometry, and expressed as:

$$\phi = \frac{m_{fuel}/m_{ox}}{(m_{fuel}/m_{ox})_{st}} \tag{3.1.1}$$

where m are masses and n are numbers of moles, and the suffix st refers to stoichiometric conditions. The equivalence ratio provides a measure of the excess or deficiency of fuel in the mixture. By specifying the equivalence ratio, we can effectively control the initial mass fractions of the species in the combustion simulation.The range of initial conditions (IC)

for the simulations is limited. The pressure is kept constant at the standard atmospheric pressure of 1 atm. The temperature ranges from $1600K$ to $1800K$, while the equivalence ratio varies from 0.7 to 1.5. To generate the training dataset, a total of 1000 initial conditions are randomly selected within these intervals using Latin Hypercube Sampling (LHS), as illustrated in Figure 3.1. Each initial condition is then used to simulate a trajectory that represents the evolution of species mass fractions and temperature over time. The resulting database consists of pairs $(S(t), S(t + dt_{cfd}))$ sampled along these trajectories. Since our objective is to simulate over multiple time steps starting from a new given initial condition $(T_0, \phi)_{\text{new}}$, it is important to split the database based on trajectories rather than individual pairs across all trajectories. all pairs of data $(S(t), S(t + dt_{cfd}))$ from a specific training trajectory will be assigned to the training database. The total dataset is splitted with a ratio of 75%/15%/10% for train, validation, and test datasets, respectively, based on trajectories. A criterion is determined to end a given simulation by defining the following variable:

$$\tau = \left| \frac{T_{eq} - T(t)}{T_{eq}} \right| \tag{3.1.2}$$

Where $T(t)$ denotes the temperature of the local time step and T_{eq} denotes the temperature at the equilibrium state that is determined for given initial conditions (ϕ, T_0) by a simple thermodynamic equilibrium computation. The simulations are terminated when the total simulation time reaches $10^{-3}s$. At this point, it is assumed that all variables have reached their equilibrium states. Alternatively, if the simulation reaches convergence with a given tolerance τ, the simulation is also considered complete. In this work, the tolerance τ is set to 10^{-4} to ensure that all simulations from different initial conditions reach their final equilibrium states. In the next section, a novel strategy for generating the dataset is presented, which takes advantage of the time step adaptation feature of the CVODE solver.

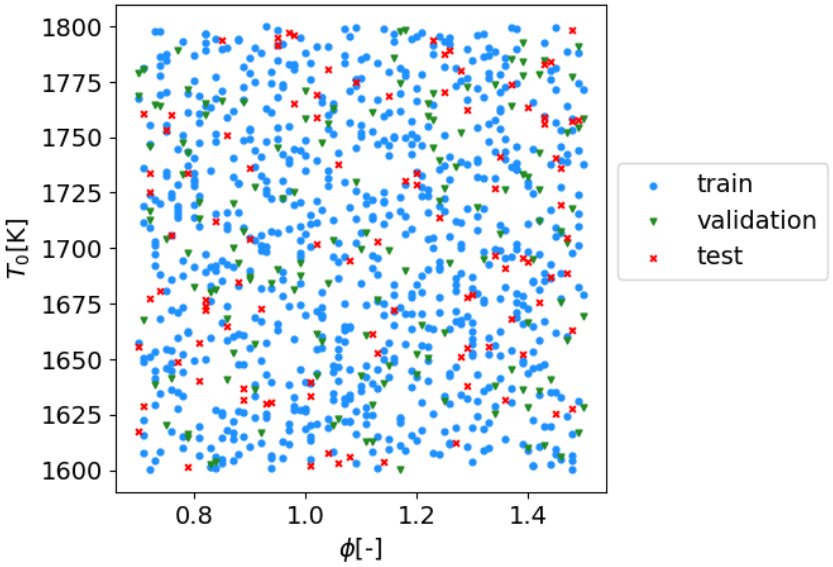


Figure 3.1: Initial conditions distribution

3.1.2 Data acquisition from simulation trajectories

The data pairs $(\mathbf{S}(t), \mathbf{S}(t + dt_{cfd}))$ are acquired within the generated trajectory zones. However, the states evolution rates vary with time. Roughly speaking, reactions are "slow" close to the equilibrium and "fast" after ignition. A naive strategy would consist in selecting sampling points from the target trajectory during the simulation with a regular time step, noted as dt_s. In other words, the dataset would contain pairs of the form $(\mathbf{S}(kdt_s), \mathbf{S}(kdt_s + dt_{cfd}))$, where dt_s denotes the uniform sampling time step.The straightforward approach of selecting data points from the target trajectory with a regular time step (dt_s) may result in an imbalanced dataset. This method would result in fewer data points in the ignition region, where chemical reactions occur rapidly. As a consequence, the dataset would be imbalanced with a skewed distribution of data points. This imbalance can be observed in the temperature and CO_2 mass fraction distributions, as shown in Figure 3.2 for the case of C_2H_4.

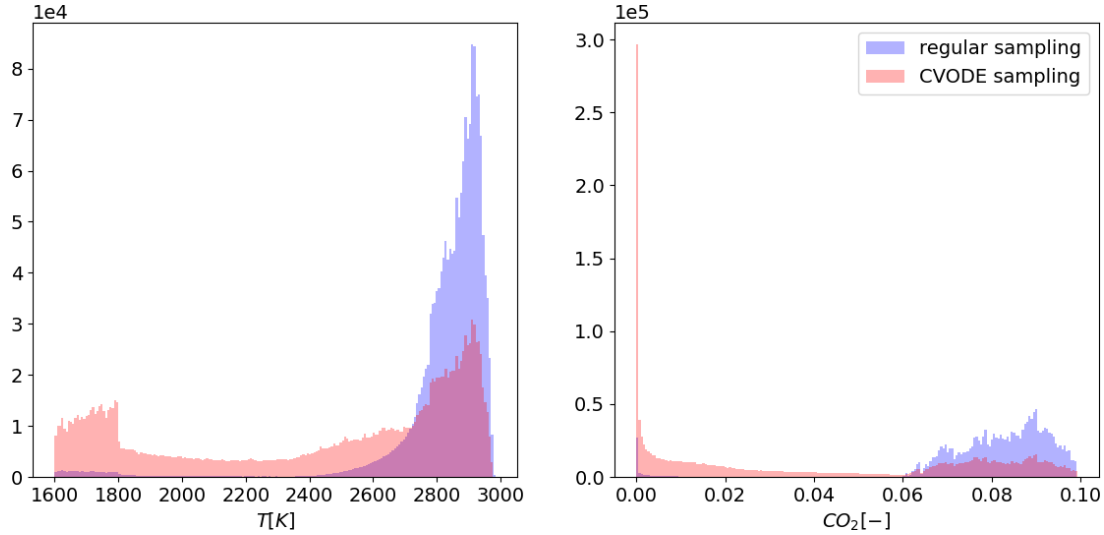


Figure 3.2: The sampling points distribution of (a) temperature and (b) CO_2 values generated by regular sampling method (blue) and CVODE sampling method (red) for the C_2H_4/air case. The total size of data points generated by two strategies is around 1.5×10^6.

In this research, the dynamic time step adaptation feature of the CVODE solver is utilized to generate the sampling sequence. This feature allows us to generate a more balanced dataset by leveraging the time step adaptation performed by CVODE during the simulation, as presented in section 2.1.2. To generate the dataset, a format of $(\mathbf{S}(t_{cvode}), \mathbf{S}(t_{cvode} + dt_{cfd}))$ is applied, where t_{cvode} represents the time sequence obtained from the CVODE solver's time step adaptation. To achieve this, the simulations by CVODE run twice at each time step: once to obtain the points $\mathbf{S}(t_{cvode})$, and a second time to obtain the sequence $\mathbf{S}(t_{cvode} + dt_{cfd})$. By taking adaptive time steps, this approach ensures that there are more sampled points in fast reaction regions, which is beneficial for training the model. Furthermore, the maximum CVODE evolution time step is limited to dt_{cfd} to ensure a sufficient resolution in the near-equilibrium region. This restriction helps maintain accuracy in those regions. The resulting dataset is better balanced compared to using a constant time step, as demonstrated in Figure 3.2. This balanced dataset is crucial for training neural networks effectively.

3.1.3 Data pre-processing

To address the issue of chemical species values being in different scales and having extremely small values skewed towards zero, a pre-treatment of the chemical species values is performed using a logarithmic transformation. The transformed state vector is denoted as $\mathbf{S}_l = [T, \ln(Y_j)]$, where $j = 1, ..., N_s$. This transformation ensures that the variables are scaled to similar ranges, which is particularly beneficial for improving the prediction accuracy of minor chemical species. However, before applying the logarithmic transformation, it is necessary to clip the zero values by a threshold. This is because the logarithmic

transformation is only applicable to positive values. Empirical thresholds are determined for each dataset, where the thresholds used for the H_2, C_2H_4, and CH_4 cases are 10^{-10}, 10^{-12}, and 10^{-28}, respectively. These thresholds have been chosen to preserve the accuracy of reactor computations. An empirical criterion to choose the threshold for each fuel case is based on two important elements:

- A reason for the difference in threshold magnitude is the difference in the species evolution time scales for each fuel. Species with lower time scales, and therefore lower absolute magnitudes for minor species, are present for the C_2H_4 and CH_4 cases. (10^{-8} to 10^{-40} from maximum to minimum scales)
- The thresholds need to be high enough to suppress numerical artifacts from CVODE. An example of numerical instabilities are shown in Figure 3.3 for a case of C_2H_4 simulation. A methodology could be based on 0-D simulations to detect these artifacts and therefore choose an appropriate value for the threshold.

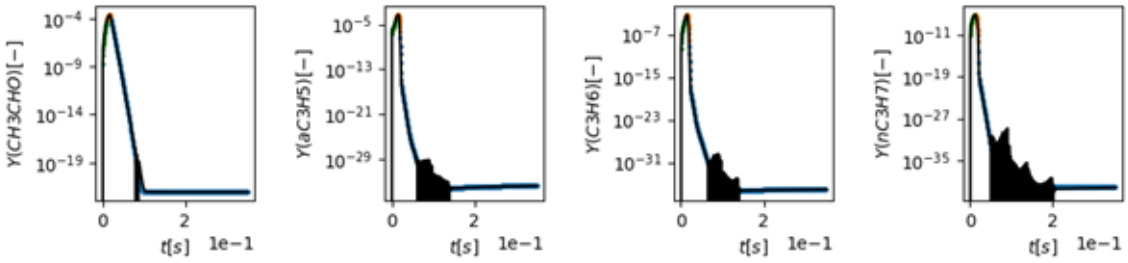


Figure 3.3: Simulation with numerical stabilities in extreme small scales, a case for C_2H_4

A comparative analysis of the impact of the threshold is performed further in this chapter, in Section 3.2.3, using C_2H_4 as an example. There still is no automatic workflow to determine the optimal threshold for each fuel case, and this might be an important extension in the future of this research.

In addition to the logarithmic transformation, a scaling of the data is applied to facilitate the training of the DNN models. A standard normalization technique is employed, which sets the mean of each feature to zero and the variance to one. This scaling helps in ensuring that all features contribute effectively to the learning process. Mathematically the normalization of data reads

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \mu}{\sigma} \tag{3.1.3}$$

where μ and σ represents the mean and variance of input data respectively. The standard normalization linearly transform the original data distribution and has no distortion to boundary outliers.

3.1.4 Data clustering

A strategy to simplify the learning process is to separate the composition space into several sub-domains. This division is achieved using the K-Means method mentioned in 2.2.3. Then, a distinct DNN model is trained for each sub-domain. This allows for more specialized and focused modeling within each region of the composition space.

3.1.5 Neural network model

We use the vanilla residual networks defined in 2.2.2, where a backbone layer is added before the hidden states pass to residual blocks. The number of residual blocks is denoted as n_r, and the neuron number in each hidden layer is represented as n_e. The swish activation function (also denoted $SiLU$) [116] is used throughout this work:

$$\text{swish}(x) = \frac{x}{1 + e^{-\beta x}} \tag{3.1.4}$$

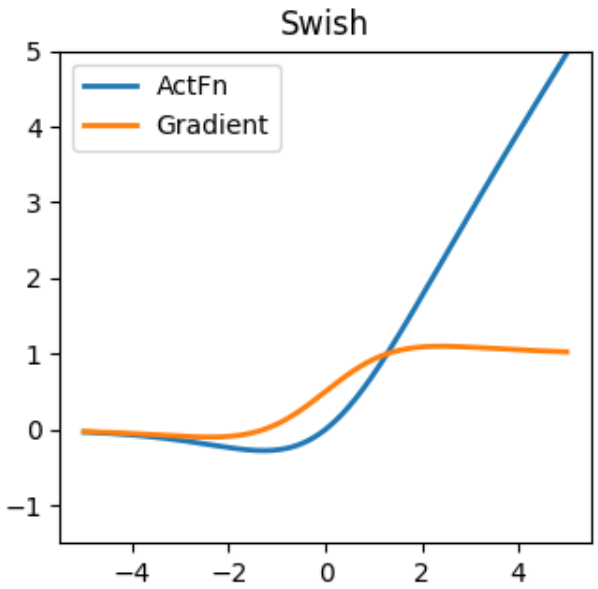


Figure 3.4: *swish* activation function and its gradient function

The profile of the swish activation function with its gradient function is provided in Figure 3.4. Compared with $ReLU$ function, swish is a smooth and continuous non monotonic function which tackles the problem of saturation for negative values[116]. It also reduces the number of 'dead' neurons which are inactive during the training. In practical applications, particularly for regression problems, the use of the *swish* activation function has been found to be more efficient for the optimization process compared to commonly used activation functions such as $ReLU$, *sigmoid*, or *tanh* [71, 109]. This observation has been verified through various experiments conducted in this study.

3.1.6 Model training

In the present work, the Mean Squared Error (MSE) is used as the loss function for model training. The DNN models are trained using the Tensorflow 2.10.1 framework [117]. For optimization, the Adam algorithm is chosen [107]. The model parameters are initialized using the Glorot Uniform method, and an exponential decay strategy is applied to the initial learning rate during the training epochs. In this research, the initial learning rate for all three cases is set to 0.005. A learning decay rate is employed to aid the optimization process [118], and it is defined as $r(k) = r_0 \times \eta^{\frac{k}{N}}$, where r_0, η, k, and N represent the initial learning rate, the decay rate, the k^{th} step, and the total number of decay steps, respectively. In this study, the decay rate is empirically set to 0.92 and the number of decay steps to 650.

3.1.7 Model performance evaluation

The DNN model serves as an operator for numerical time-stepping, denoted as $\mathbf{f}_{DL}$. By utilizing this learned operator, we can simulate the trajectory of a chemical reactor with a fixed initial condition by iteratively computing $\mathbf{S}(t_0 + k\Delta t) = \mathbf{f}_{DL}^{(k)}(\mathbf{S}(t_0 + (k-1)\Delta t); \theta)$. During the time evolution, the thermochemical states are computed iteratively in each time step resolution until the end of the simulation. Therefore, it is insufficient to evaluate the prediction performance for a single input-output pair of the models. The error may accumulate and lead to divergence after multiple iterations. Hence, it is crucial to assess the overall inference performance over multiple iterations for the entire simulation.

To compare the results with reference numerical simulations, a global accumulative logarithmic mean average percentage error $\mathcal{M}_i$ is used to evaluate the performance of chemical species trajectories under different initial conditions. Additionally, the normal mean average percentage error $\mathcal{M}_0$ is employed to evaluate the temperature prediction. It is important to note that since the models are trained using data in logarithmic space for chemical species, the logarithmic error metric is used for consistency. These introduced errors are denoted mathematically by:

$$\begin{aligned} \mathcal{M}_0(T_0, p_0, \phi) &= \frac{1}{N_{iter}} \sum_{k=1}^{N_{iter}} \left| \frac{T^{pred}(kdt) - T(kdt)}{T(kdt)} \right| \\ \mathcal{M}_i(T_0, p_0, \phi) &= \frac{1}{N_{iter}} \sum_{k=1}^{N_{iter}} \left| \frac{\ln(Y_i^{pred}(kdt)) - \ln(Y_i(kdt))}{\ln(Y_i(kdt))} \right| \quad i = 1, ..., N_s \end{aligned} \tag{3.1.5}$$

where i represents each state component (temperature and chemical species), and N_{iter} is the number of total iterations until the final time step prediction which is specific for each simulation with different initial conditions. Each $\mathcal{M}_i$ is computed for one trajectory with one initial condition, for one state component. The overall average errors $\mathcal{M}$ are

computed by averaging values of all dimensions, which is written as:

$$\mathcal{M}(T_0, p_0, \phi) = \frac{\mathcal{M}_0(T_0, p_0, \phi) + \sum_{i=1}^{N_s} \mathcal{M}_i(T_0, p_0, \phi)}{N_s + 1} \quad i = 1, ..., N_s \tag{3.1.6}$$

The global errors $\mathcal{M}$ and errors for each state component $\mathcal{M}_i$ are statistically evaluated for all 100 initial conditions in the test set. This metric accounts for the potential error accumulation that may arise when repeatedly using a neural network to predict the evolution of chemical states. By taking into account the complete range of initial conditions, the overall performance and reliability of the neural network model can be evaluated in accurately capturing the dynamics of the chemical reactions.

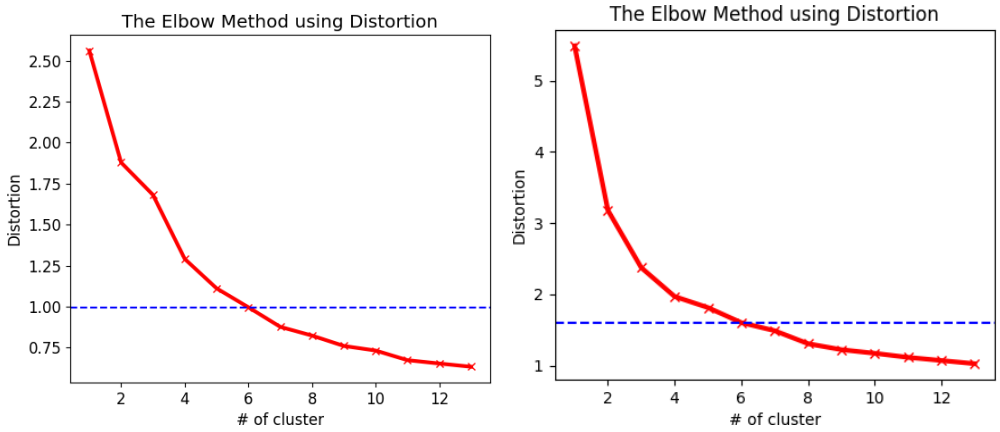
3.2 Analysis of results

3.2.1 Analysis of data clustering

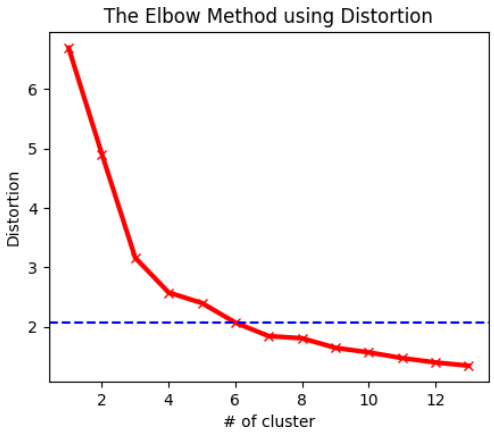
The number of K-means clusters needs to be selected beforehand as there is no method for systematic selection. A practical approach to evaluate the clustering efficiency is to compute the distortion $\mathcal{D}$, which corresponds to the squared Euclidean distance between the data points and the centroids:

$$\mathcal{D} = \sum_{i=1}^{K} \sum_{\mathbf{S} \in \Omega_i} \left\| \hat{\mathbf{S}}_l - \mathbf{d}_i \right\|^2 \tag{3.2.1}$$

where $\mathbf{S}_l$ is the logarithm of the state vector and $\hat{\mathbf{S}}_l$ its normalized counter-part. $\mathbf{d}_i$ represent the coordinates of the K centroids. The evolution of the distortion with the number of clusters is shown in Figure 3.5 for H_2, C_2H_4 and CH_4. It can be observed that as the number of clusters increases, the distortion decreases monotonically, with a steeper slope initially and a smoother evolution thereafter. In this study, the optimal number of clusters for each fuel is determined empirically within a predefined range by evaluating inference simulations. It should be noted that while the data is separated into subdomains with a converged distortion value, the local data distribution may not be optimal for model training, which can lead to inference instabilities during iterative predictions. Models are evaluated with up to six clusters, as the distortion value does not significantly decrease for larger numbers of clusters, indicating that all sampling points have already been optimally partitioned around local centroids. Moreover, with an increasing number of clusters after six, some subdomains may have insufficient data for effective local model training. After partitioning the dataset, individual learning models are trained and evaluated for each subdomain.



(a) H_2 (b) C_2H_4

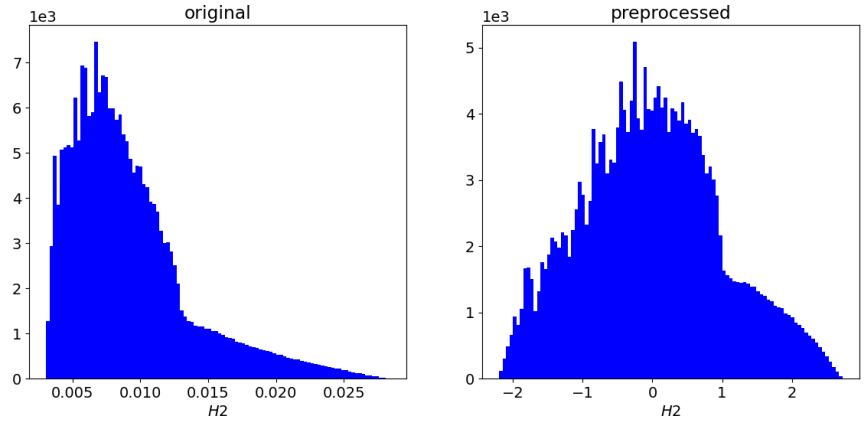


(c) CH_4

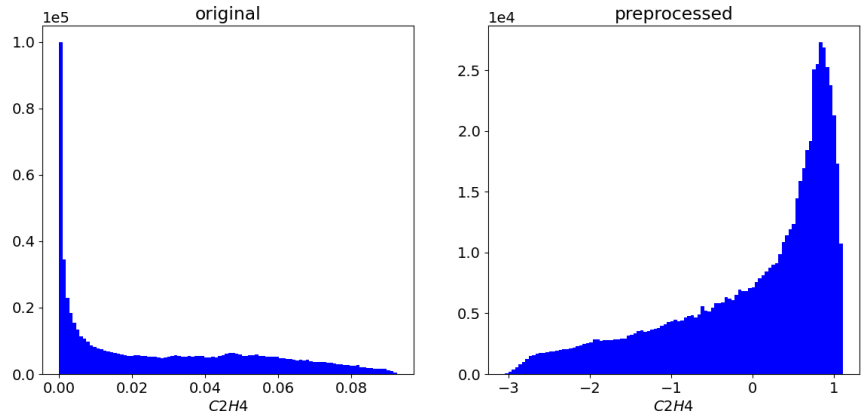
Figure 3.5: distortion values over cluster numbers for (a) H_2/air, (b) C_2H_4/air, and (c) CH_4/air cases

3.2.2 Model training evaluations

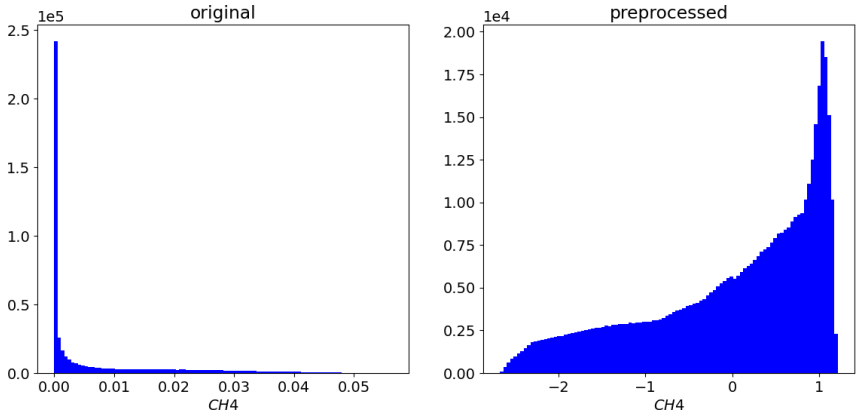
Several model training strategies are crucial for achieving optimal performance in this research. Firstly, trajectory learning under the logarithmic scale is necessary to address the prediction of extremely small values. After the data preprocessing, the rescaled data for each subdomain model in different cases are shown in Figure 3.6. The logarithmic transformation helps to transform the original data distribution into a more homogeneous distribution and removes extreme values skewed towards zero [119]. Data standardization, on the other hand, rescales the original data to have a mean of zero and a unit variance.



(a)



(b)



(c)

Figure 3.6: Distribution of training data from reactive subdomains before and after the preprocessing with nonlinear logarithmic transformation: (a)H_2, (b)C_2H_4, and (c)CH_4.

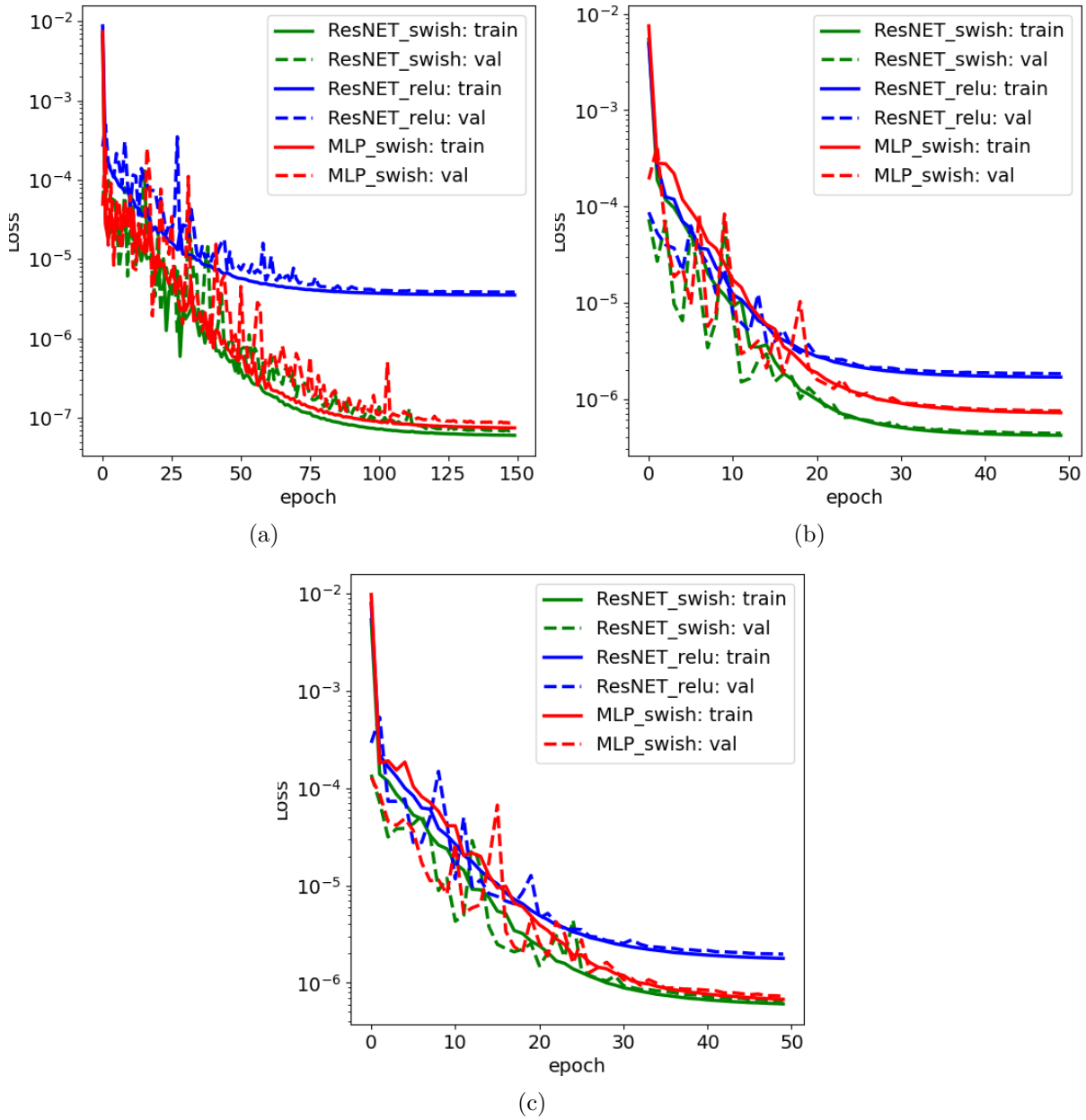


Figure 3.7: Training and validation loss function values(MSE) over epoch number for models of reactive subdomains for 3 cases (a)H_2/air, (b)C_2H_4/air, and (c)CH_4/air.

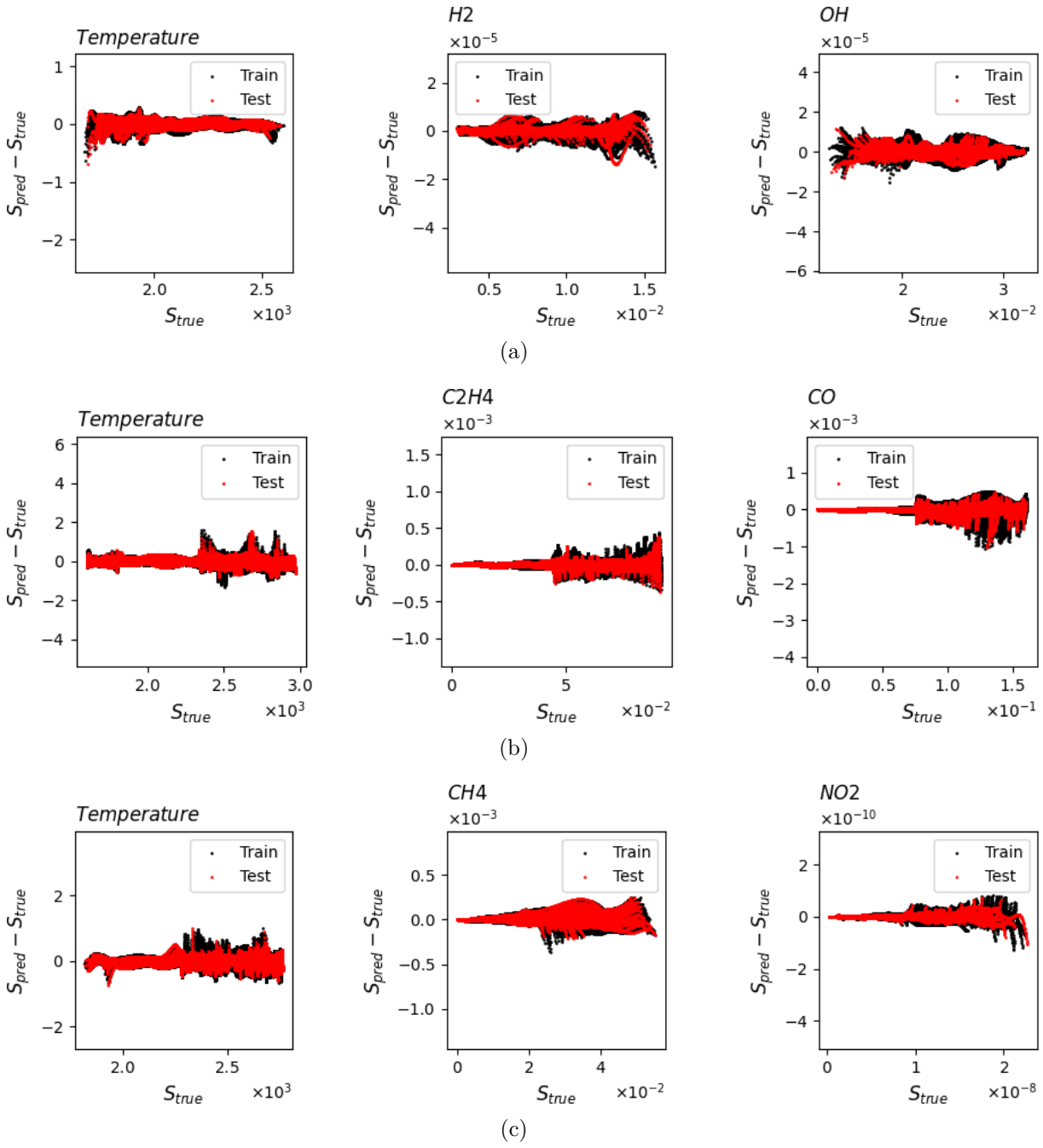


Figure 3.8: Parity plots of true output values and predicted errors for (a)H_2/air, (b)C_2H_4/air, and (c)CH_4/air. The chosen plot elements here are temperature, the fuel and a radical minor species, which are predicted by models of a reactive subdomain.

Training with a deeper neural network structure, including residual shortcuts, is more efficient compared to traditional multiple-layer perceptrons (MLP). This is demonstrated by comparing a standard MLP with the same number of hidden layers and neurons in each layer to our optimal trained ResNet models. Additionally, the activation function *swish* is expected to improve the optimization process compared to the commonly used *ReLU* activation function in previous research papers [3][70]. The evolution of the loss function over training epochs for models in a cluster containing strongly reactive states is

depicted in Figure 3.7. It can be observed that the *swish* activation function significantly promotes the optimization process, leading to smaller loss function values compared to the *ReLU* activation function. Furthermore, the residual networks with skip-connection structures also improve the optimization results for reactive zones. By employing the selected strategy in this work, which is a residual neural network model with the *swish* activation function, we achieve the lowest loss values during the optimization compared to other strategies. Additionally, there is no significant difference between training and validation losses, indicating that there is no overfitting during the training process. Similar conclusions regarding the absence of overfitting can also be drawn from Figure 3.8, which presents the parity plots of the true output values $\mathbf{S}_{true}$ and the errors between the true and predicted values $\mathbf{S}_{pred} - \mathbf{S}_{true}$. Most of error values between the true and predicted states are much smaller than the state values under physical scales for each dimension.

Tables 3.2, 3.3, and 3.4 provide statistics of $\mathcal{M}$, including the average, minimum, and maximum values, for inference simulations of 100 initial conditions in the test set. The total sizes of the trained models are 330 kB, 3.2 MB, and 3.8 MB for the H_2, C_2H_4, and CH_4 cases, respectively. Box plots in Figures 3.9, 3.10, and 3.11 illustrate the statistical distribution of temperature and species mass fractions. It must be noted that that for such systems the temperature might be directly estimated from species mass fractions and enthalpy. In an attempt to make the model more general and not solely tailored for constant pressure low Mach problems, temperature is added as an input of the DNN. The boxplot style charts represent the statistical results of global accumulative log mean absolute percentage error for each chemical species and normal mean absolute percentage error for temperature. The errors are computed for 100 test simulations as we mentioned in our manuscript and the y axis denotes the error values. The box shows the quartiles of the value set, and the whiskers extend to show the rest of the distribution. The black dots are particularly plotted to be outliers which represent the highest values of error for each case. From the overall statistical results, it is observed that methods with multiple models are more efficient compared to the baseline method with only one cluster. However, in some cases where the mean error $\mathcal{M}$ is low, there are still a few extreme error values. This observation highlights the issue of trajectory divergence from the correct path during the simulation, which can be attributed to the limited robustness of the local models. The optimal number of clusters leading to the best performance, with the lowest $\mathcal{M}$, in the predefined workflow is found to be 6, 4, and 5 for the H_2, C_2H_4, and CH_4 cases, respectively. The data is divided into different subdomains, including preheated mixing zones with extreme values skewed towards zero, stiff reactive zones, and burn-up zones. The best models for each fuel will be considered in the subsequent analysis. Figure 3.12 demonstrates the partitioned clusters represented by different colors on the manifold of mass fractions of O_2 and H_2O in logarithmic scale over the progress variable. The progress variable is defined based on the evolution of states, and in the 0D case it is derived from

temperature values, denoted by:

$$c = \frac{T - T_0}{T_{\text{eq}} - T_0}. \tag{3.2.2}$$

It is evident that the total manifolds are piecewise separated into subdomains. Each subdomain exhibits reduced complexity, enabling the construction of learning models with simpler structures and fewer parameters for all subdomains.

Cluster number	n_r	n_e	mean $\mathcal{M}(\%)$	minimum $\mathcal{M}(\%)$	maximum $\mathcal{M}\%)$
1	1	120	0.301	0.043	2.913
2	1	85	0.097	0.016	1.446
3	1	70	0.117	0.020	1.056
4	1	60	0.068	0.010	3.571
5	1	54	0.081	0.023	11.87
6	1	49	0.068	0.012	0.643

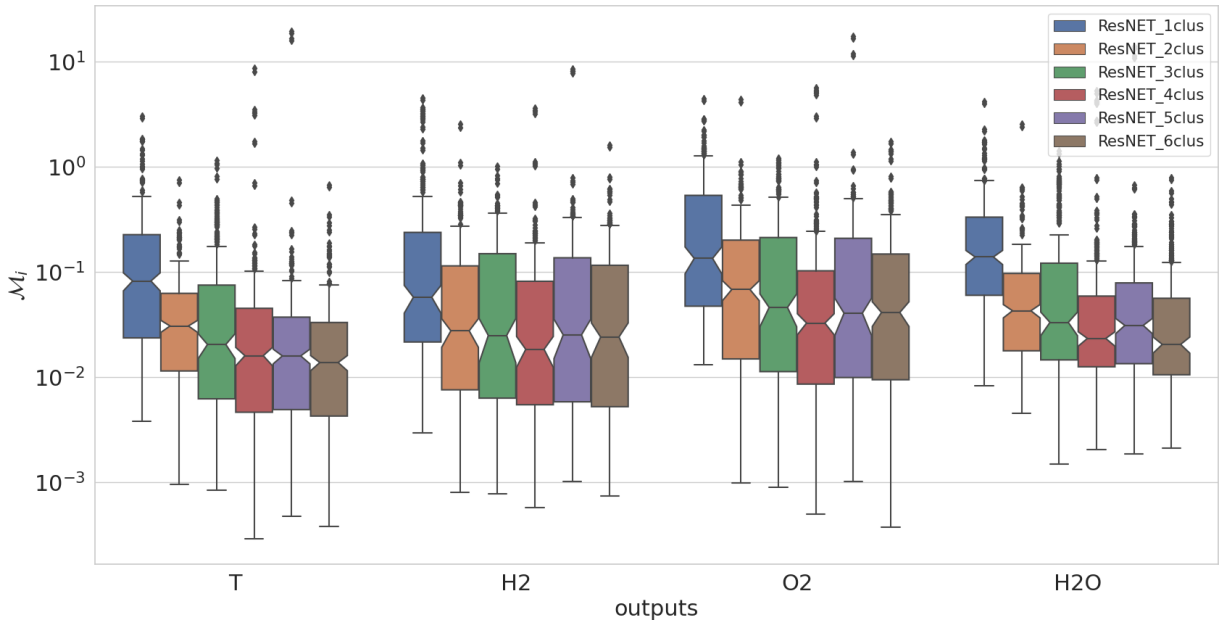
Table 3.2: Statistics for experiments of H_2/air case

Cluster number	n_r	n_e	mean $\mathcal{M}(\%)$	minimum $\mathcal{M}(\%)$	maximum $\mathcal{M}(\%)$
1	2	300	0.971	0.047	45.30
2	2	212	0.366	0.070	3.571
3	2	174	0.039	0.009	187.94
4	2	150	0.033	0.010	0.249
5	2	135	0.040	0.010	0.312
6	2	123	0.043	0.010	0.179

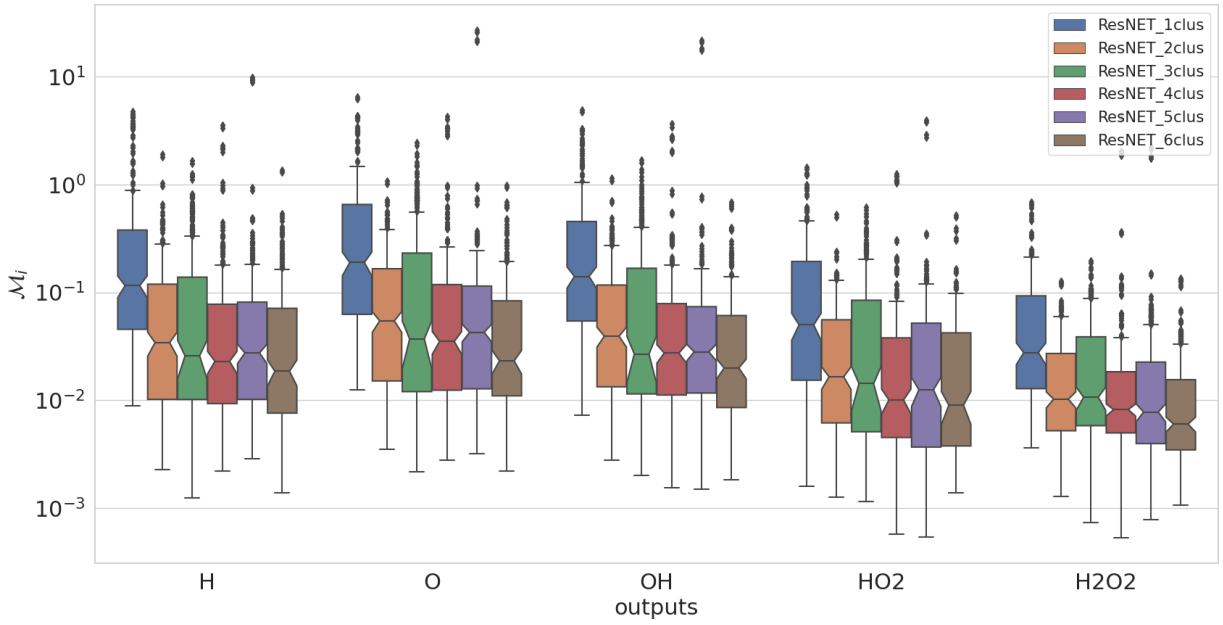
Table 3.3: Statistics for experiments of C_2H_4/air case

Cluster number	n_r	n_e	mean $\mathcal{M}(\%)$	minimum $\mathcal{M}(\%)$	maximum $\mathcal{M}(\%)$
1	2	350	0.947	0.199	6.907
2	2	248	0.882	0.138	9.391
3	2	202	0.332	0.036	4.421
4	2	175	0.153	0.042	2.159
5	2	157	0.152	0.033	1.102
6	2	143	0.173	0.030	1.583

Table 3.4: Statistics for experiments of CH_4/air case

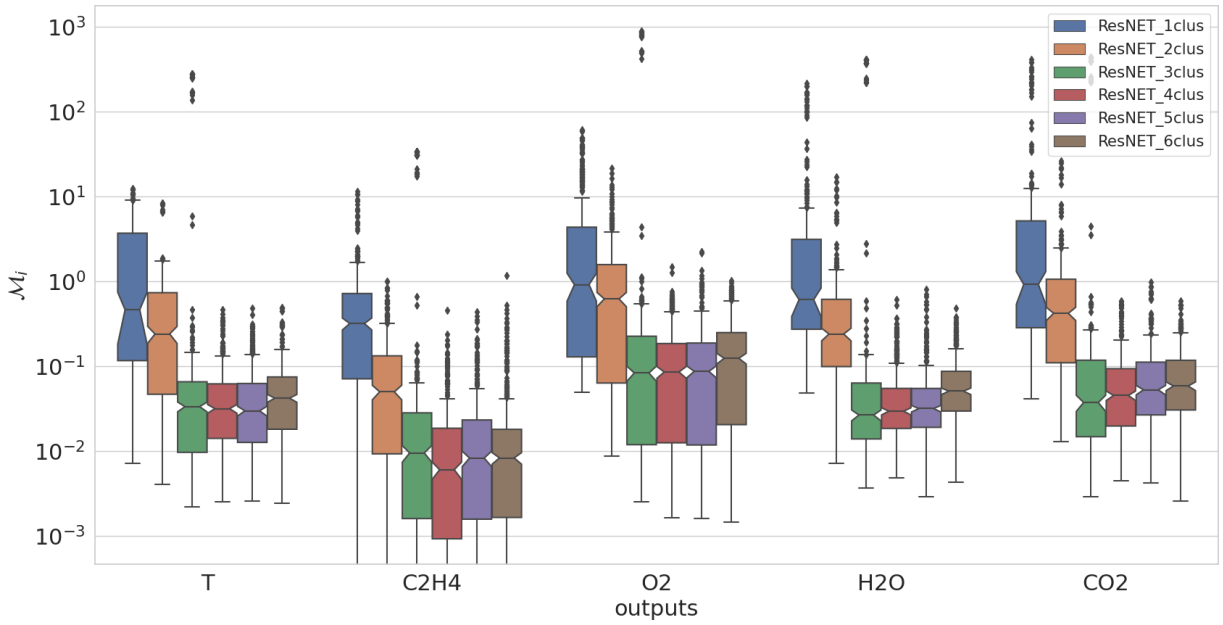


(a)

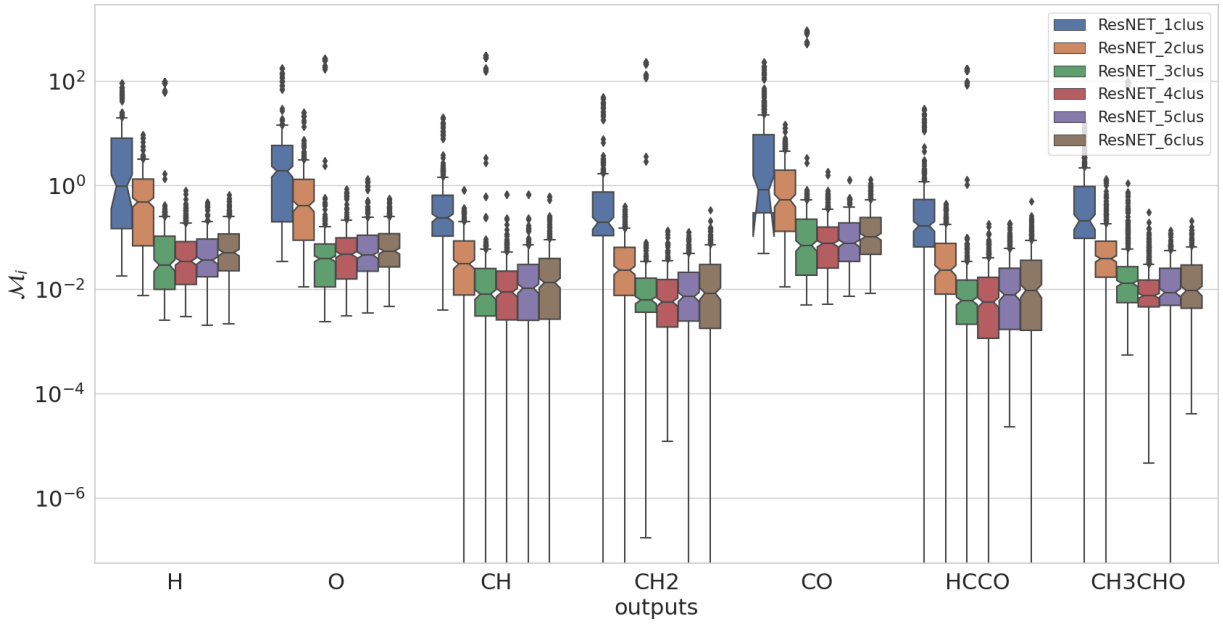


(b)

Figure 3.9: Box plot of statistical logMAPE errors for 100 a posteriori test simulations of H_2/air case with (a)temperature and major chemical species and (b)several minor chemical species

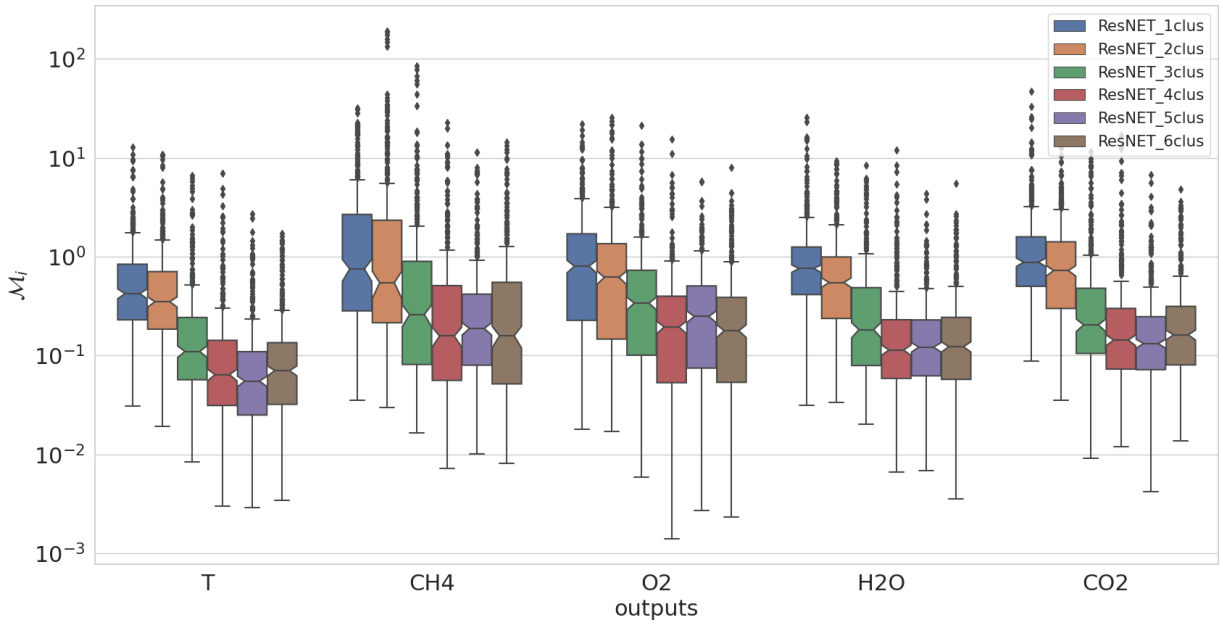


(a)

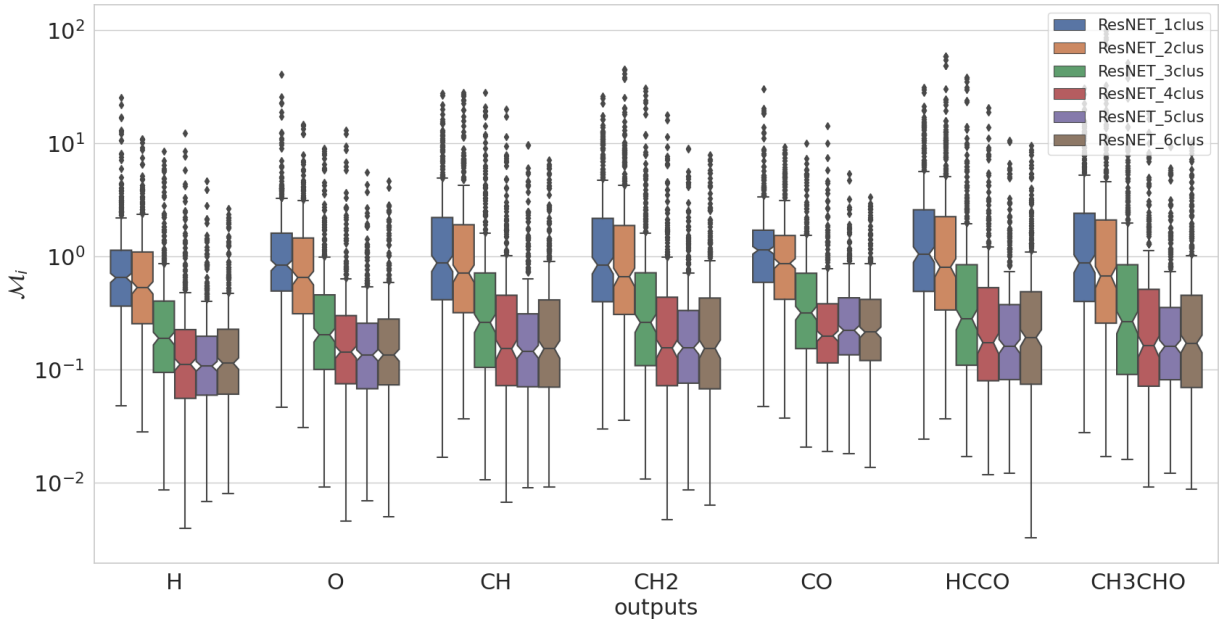


(b)

Figure 3.10: Box plot of statistical logMAPE errors for 100 a posteriori test simulations of C_2H_4/air case with (a)temperature and major chemical species and (b)several minor chemical species

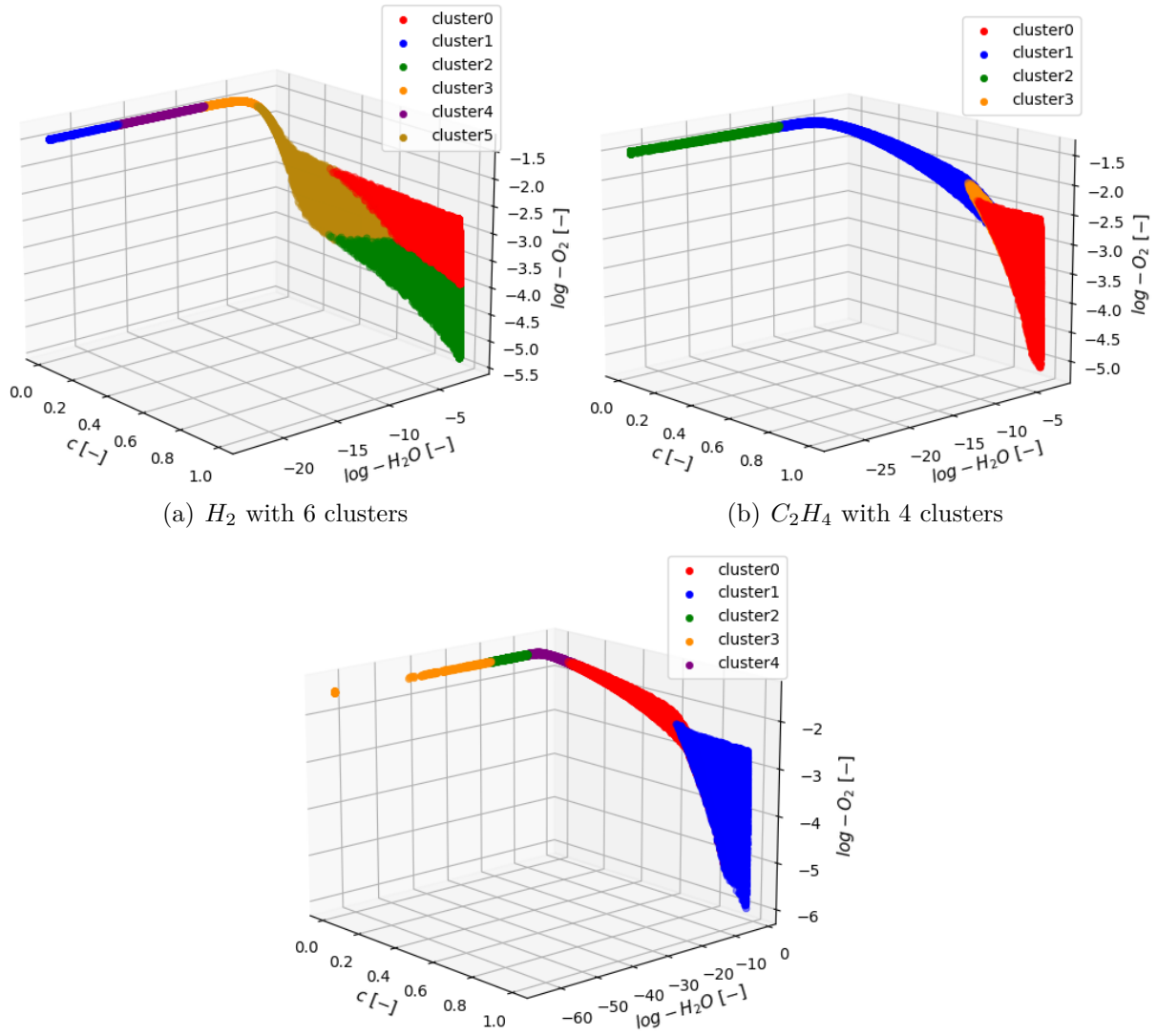


(a)



(b)

Figure 3.11: Box plot of statistical logMAPE errors for 100 a posteriori test simulations of CH_4/air case with (a)temperature and major chemical species and (b)several minor chemical species



(a) H_2 with 6 clusters

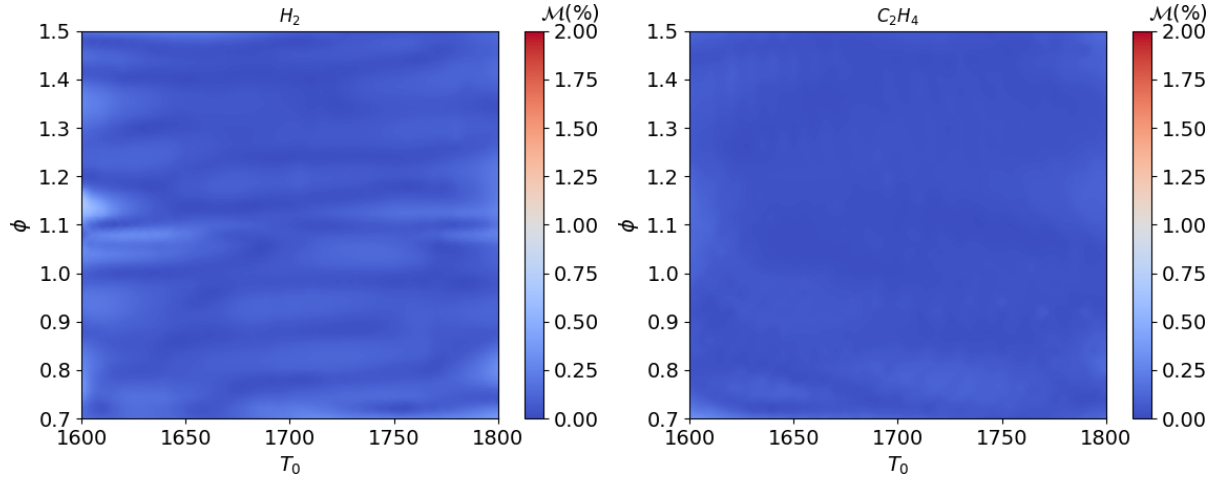
(b) C_2H_4 with 4 clusters

(c) CH_4 with 5 clusters

Figure 3.12: 3D clustering plots of manifolds for mass fractions of O_2 and H_2O over the evolution of progress variable: (a)H_2/air, (b)C_2H_4/air, and (c)CH_4/air.

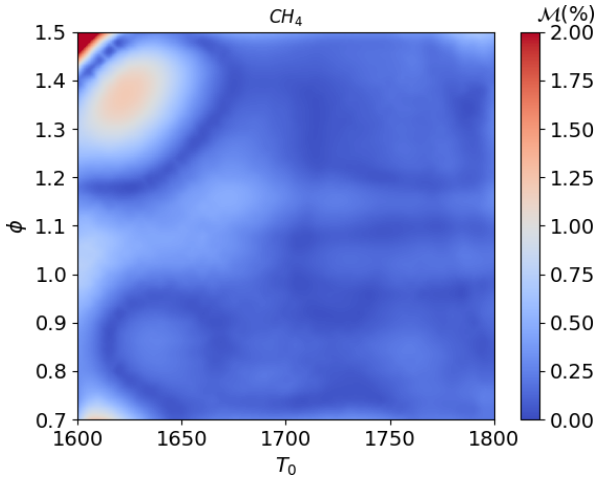
Furthermore, additional evaluations of iterative predictions are conducted by increasing the number of simulations within the predefined range of initial conditions. In this case, 1680 simulations based on model predictions are generated, and the overall accumulative logarithmic Mean Average Percentage Error (logMAPE) (4.2.3) is computed for each simulation. Cubic interpolation is utilized to generalize the 2D error distribution functions. The design of the experiment with a large number of initial conditions is fixed within the same predefined range that was used for training the models, with regular samplings of T_0 and ϕ at intervals of $\Delta T_0 = 5K$ and $\Delta\phi = 0.01$. The distribution of overall average MAPE errors is depicted in Figure 3.13. It can be observed that simulations based on the best models exhibit overall average MAPE errors lower than 1%, with a slight decrease in accuracy near the boundaries of the domain. Due to its higher complexity, the CH_4/air case exhibits larger overall logMAPE errors, but they are still

within an acceptable range.



(a) H_2

(b) C_2H_4



(c) CH_4

Figure 3.13: 2D distribution of mean relative errors of all dimensions as a function of the initial condition for (a)H_2/air, (b)C_2H_4/air, and (c)CH_4/air cases within designed initial condition zone, using refined initial conditions sampling and cubic interpolation

3.2.3 Analysis of threshold effects

	cluster 0	cluster 1	cluster 2	cluster 3
threshold 10^{-10}				
training loss	2.18×10^{-6}	3.05×10^{-7}	3.04×10^{-6}	2.48×10^{-6}
validation loss	2.46×10^{-6}	2.98×10^{-7}	3.46×10^{-6}	2.57×10^{-6}
test loss	2.46×10^{-6}	3.44×10^{-7}	3.36×10^{-6}	2.65×10^{-6}
threshold 10^{-12}				
training loss	1.87×10^{-6}	3.43×10^{-7}	3.44×10^{-7}	3.43×10^{-6}
validation loss	1.91×10^{-6}	3.67×10^{-7}	3.31×10^{-7}	3.55×10^{-6}
test loss	2.01×10^{-6}	3.70×10^{-7}	3.88×10^{-7}	3.55×10^{-6}
threshold 10^{-15}				
training loss	9.64×10^{-1}	7.20×10^{-7}	6.03×10^{-8}	1.13×10^{-6}
validation loss	5.00×10^{-3}	8.63×10^{-7}	6.35×10^{-8}	1.46×10^{-6}
test loss	6.50×10^{-2}	7.30×10^{-7}	6.85×10^{-8}	4.87×10^{-6}
threshold 10^{-20}				
training loss	3.30×10^{-2}	1.76×10^{-5}	5.16×10^{-8}	8.62×10^{-6}
validation loss	4.90×10^{-2}	7.71×10^{-6}	5.37×10^{-8}	1.49×10^{-6}
test loss	1.35×10^{-1}	9.97×10^{-6}	5.80×10^{-8}	2.81×10^{-5}

Table 3.5: Loss values obtained for cases of C_2H_4 training with different threshold in datasets

Thresholding is necessary in the current workflow as the logarithm transformation is not defined for zero values. The threshold needs to be sufficiently small not to affect the resolution of the chemistry. This can be verified by running a 0-D CANTERA simulation while clipping species mass fractions at each time step. In other words, its magnitude needs to be physically negligible. On the other hand, we have observed that the threshold cannot be too small because the CVODE algorithm generates oscillations for small species mass fractions magnitude. These are artifacts generated by the numerical scheme which have a detrimental impact on the NN model learning. As for an empirically tuning of threshold, an example of C_2H_4 case is analysed with different thresholds ($\epsilon = 10^{-10}$, $\epsilon = 10^{-12}$, $\epsilon = 10^{-15}$, $\epsilon = 10^{-20}$) which largely affect the training qualities, as shown in 3.5. It is shown that as for cluster 0 (which is the burn-up region), when threshold is larger, the trainings are failed, this is because of the numerical instabilities which are not physically meaningful. As for thresholds $\epsilon = 10^{-10}$ and $\epsilon = 10^{-12}$, the threshold $\epsilon = 10^{-12}$ may lead to the best training results where the mean squared error in cluster 2 is much less than that for threshold $\epsilon = 10^{-10}$.

3.2.4 Comparison of sampling strategies

	H_2	C_2H_4	CH_4
Regular sampling			
mean $\mathcal{M}(\%)$	0.170	0.051	0.332
maximum $\mathcal{M}(\%)$	1103.2	0.434	1.880
minimum $\mathcal{M}(\%)$	0.018	0.019	0.063
CVODE sampling			
mean $\mathcal{M}(\%)$	0.069	0.033	0.152
maximum $\mathcal{M}(\%)$	0.643	0.249	1.102
minimum $\mathcal{M}(\%)$	0.012	0.010	0.034

Table 3.6: Statistics for experiments of H_2/air, C_2H_4/air and CH_4/air case using regular sampling method and CVODE sampling method

The results of comparing predictions using regular sampling and the CVODE-based sampling method are shown in Table 3.6. The chosen clusters are based on the optimal clusters for each case, which is 6, 4 and 5 for H_2, C_2H_4 and CH_4. The results demonstrate that the CVODE-based sampling method improves the overall prediction performance, as indicated by smaller mean, minimum, and maximum errors ($\mathcal{M}$) for all three cases. Regular sampling can lead to unbalanced clustering and uneven data distribution, resulting in unstable iterative predictions, as observed in Table 3.6 for the H_2 case. By employing adaptive time steps, the CVODE-based method generates more data points from fast reaction regions, leading to a more robust unsupervised clustering model with a balanced distribution of data points.

3.2.5 Simulation of 0D reactors using DNN

To illustrate the prediction of specific trajectories, two simulations with different initial conditions near the boundaries of the dataset were performed. One simulation was conducted with a low temperature of $1620.0K$ and an equivalence ratio of 0.75, while the other simulation had a high temperature of $1790.0K$ and an equivalence ratio of 1.45. The results for temperature are shown in Figure 3.14 for the three fuels. It can be observed that the predictions generated by the models closely match the results of the direct numerical simulations. Additionally, Figures 3.15, 3.16, and 3.17 show the predictions for a subset of the chemical species. The DNN results exhibit good agreement with the direct numerical simulations in both the auto-ignition zones and equilibrium zones, compared to the exact numerical simulations. Figures 3.18, 3.19, and 3.20 display the predictions in logarithmic predictive space, with clustering zones marked by different colors. The cluster index of states is predicted after unsupervised classification by the clustering algorithm, resulting in states belonging to different subdomains at different time steps. The predictions in

logarithmic space are accurate within each partitioned subdomain of states, and the state values skewed toward zero are accurately predicted at the start of the simulations.

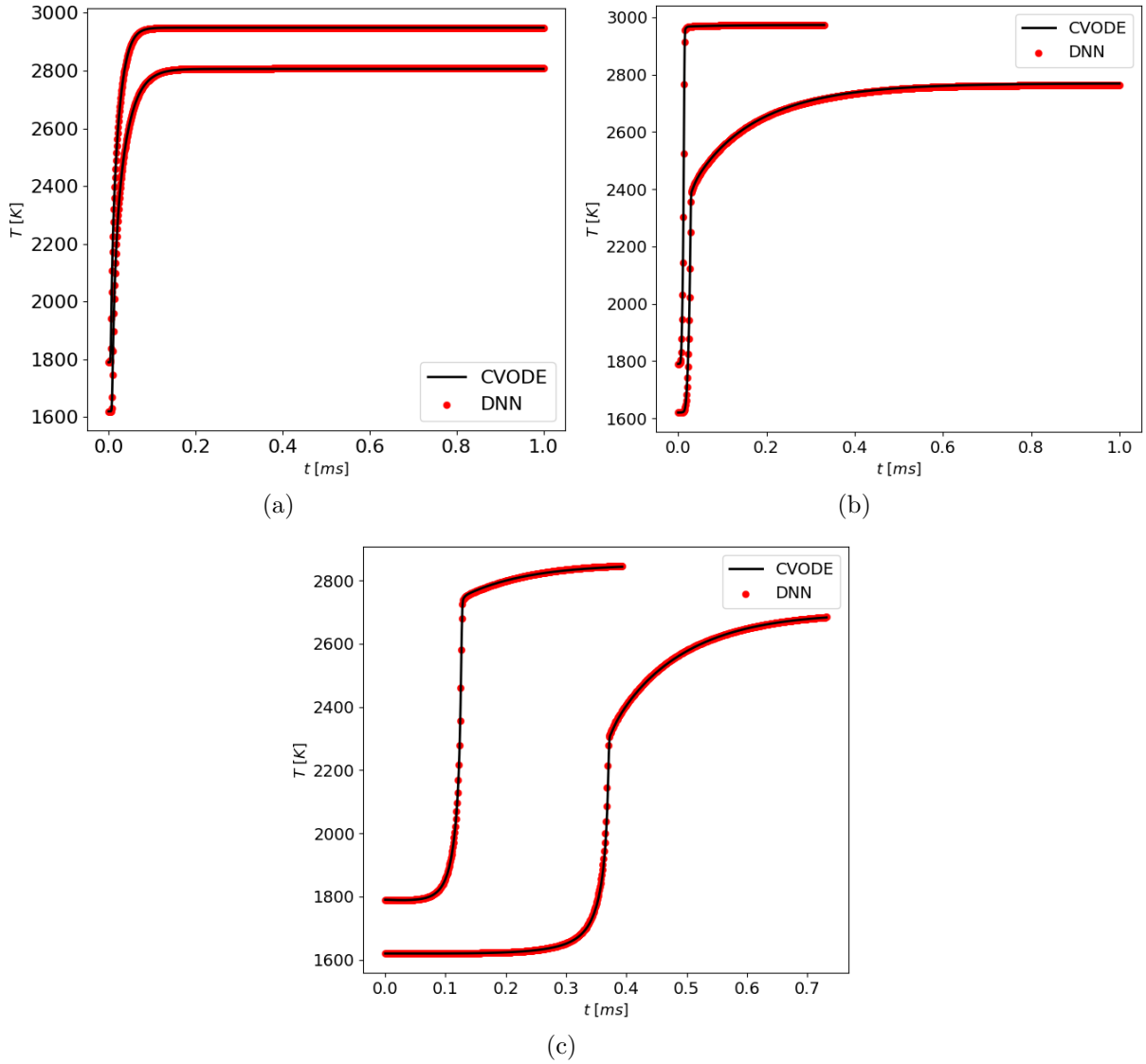
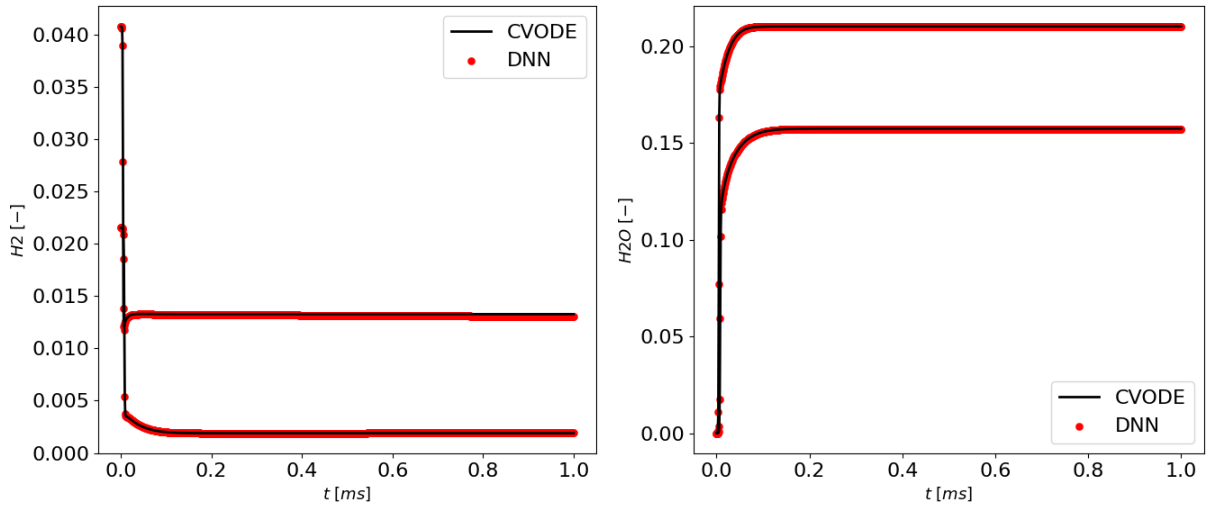
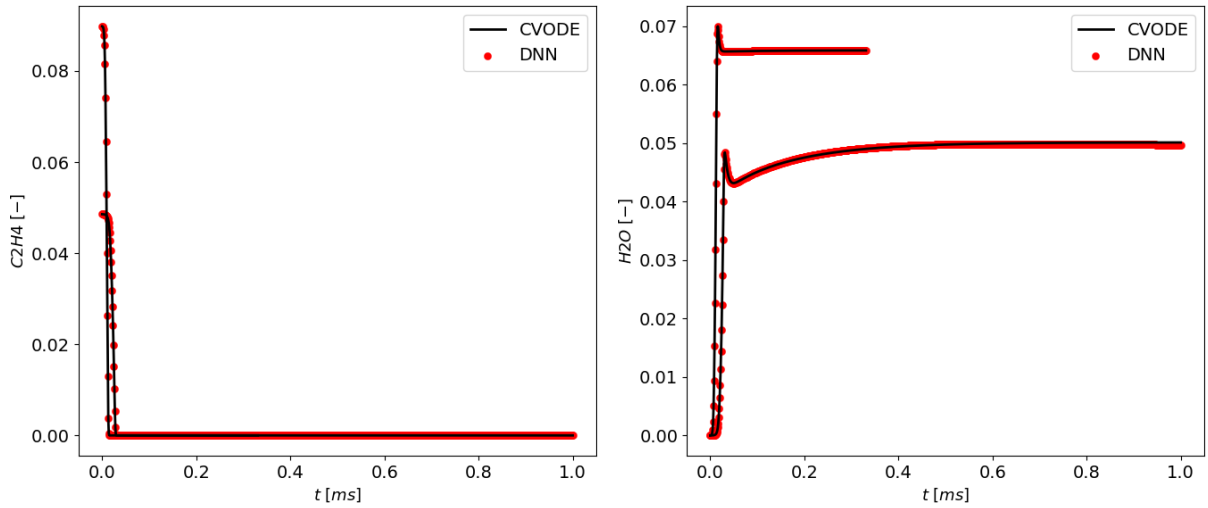


Figure 3.14: Continuous inference simulation of (a)H_2/air, (b)C_2H_4/air, and (c)CH_4/air cases for temperature with 2 trajectories: $T_{01} = 1620.0K$,$\phi_1 = 0.75$ and $T_{02} = 1790.0K$,$\phi_2 = 1.45$.



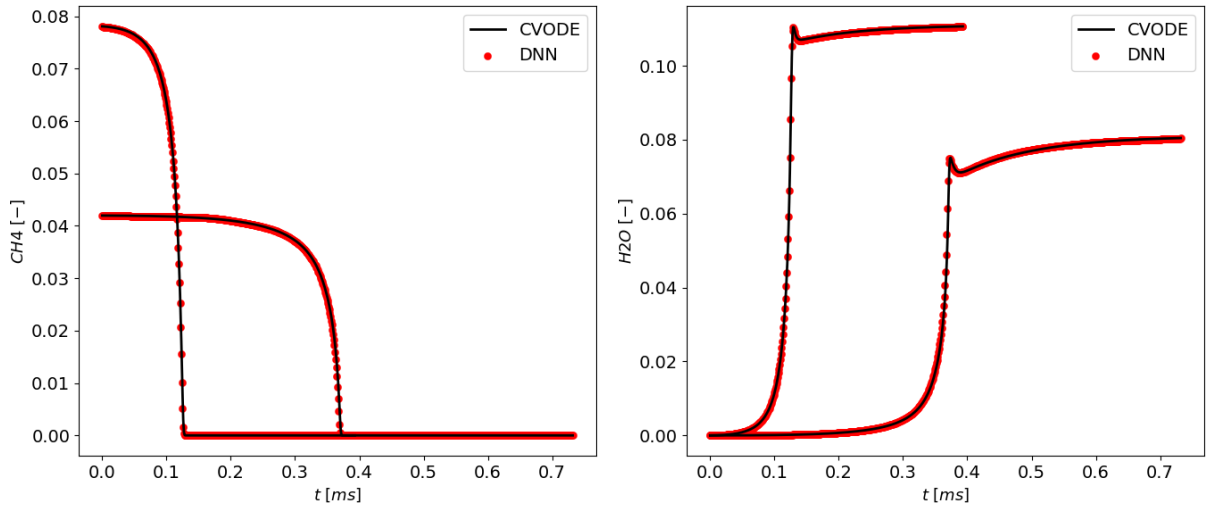
(a) H_2 (b) H_2O

Figure 3.15: Continuous inference simulation for H_2/air case with 2 trajectories: $T_{01} = 1620.0K$,$\phi_1 = 0.75$ and $T_{02} = 1790.0K$,$\phi_2 = 1.45$.



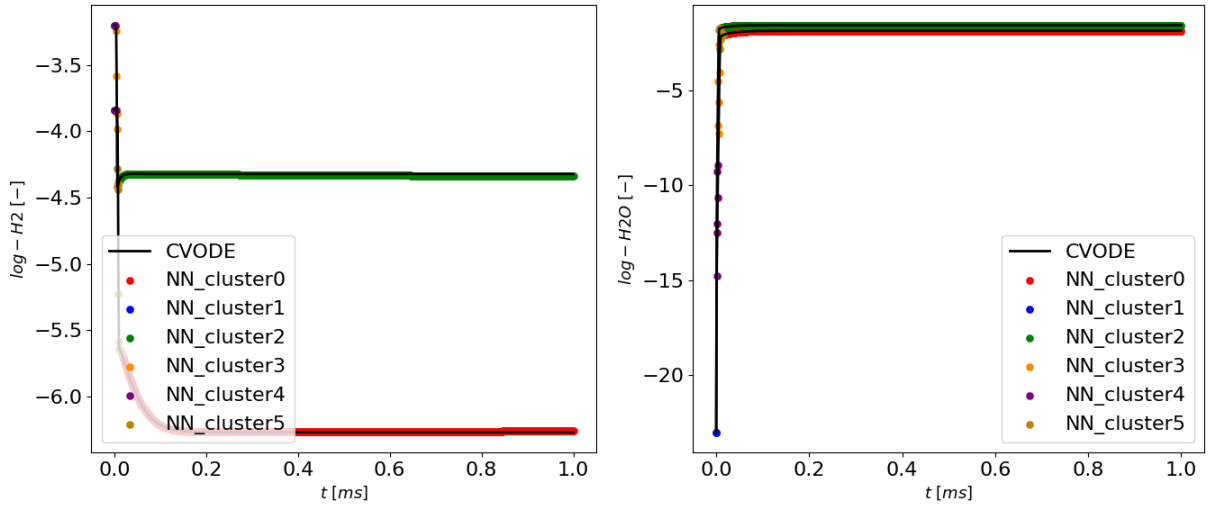
(a) C_2H_4 (b) H_2O

Figure 3.16: Continuous inference simulation for C_2H_4/air case with 2 trajectories: $T_{01} = 1620.0K$,$\phi_1 = 0.75$ and $T_{02} = 1790.0K$,$\phi_2 = 1.45$.



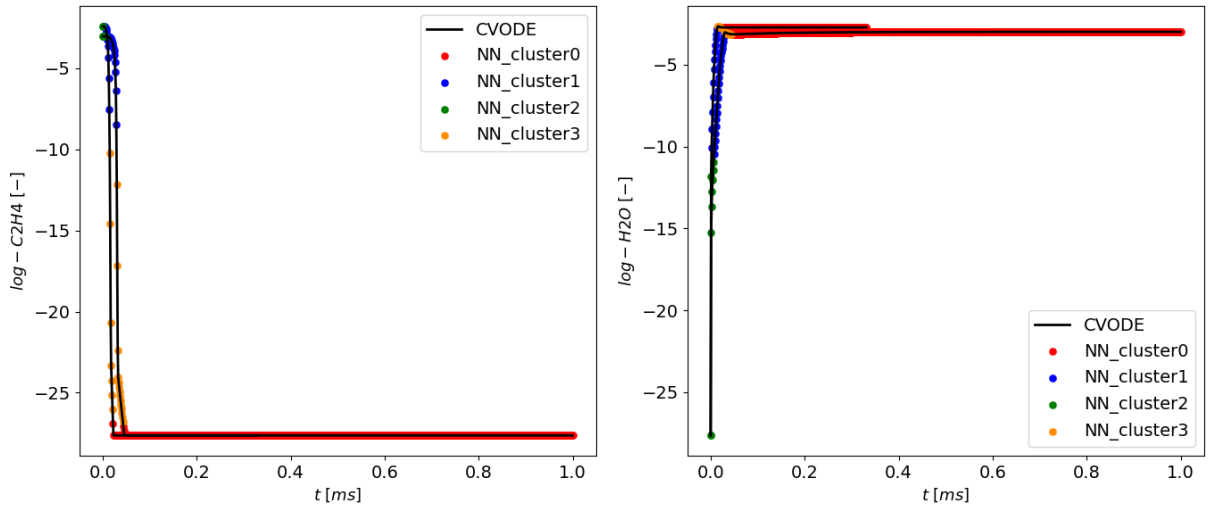
(a) CH_4 (b) H_2O

Figure 3.17: Continuous inference simulation for CH_4/air case with 2 trajectories: $T_{01} = 1620.0K$,$\phi_1 = 0.75$ and $T_{02} = 1790.0K$,$\phi_2 = 1.45$.



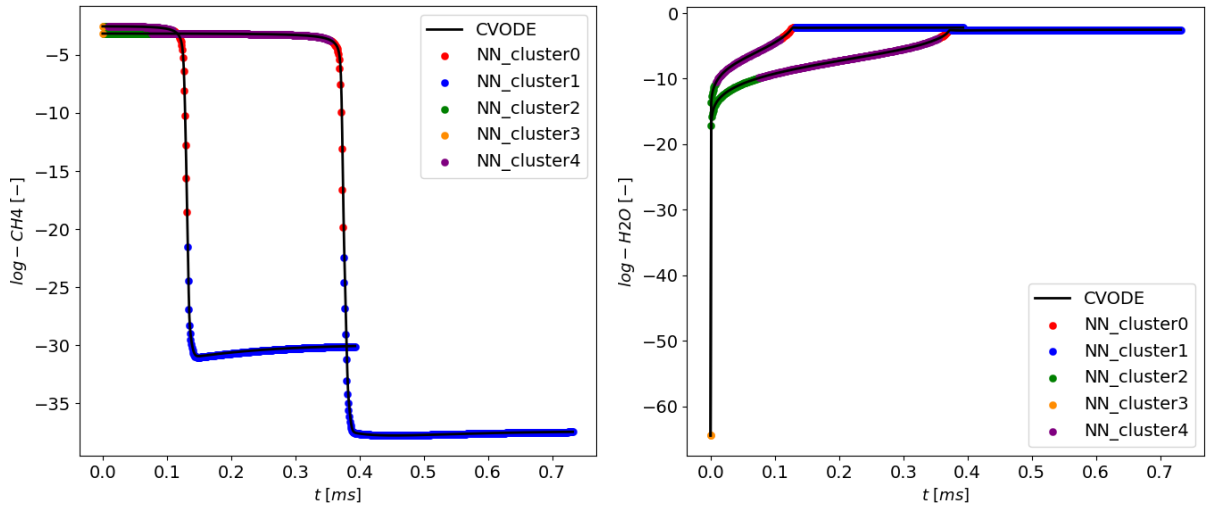
(a) H_2 (b) H_2O

Figure 3.18: Continuous inference simulation in logarithmic scale for H_2/air case with 2 trajectories: $T_{01} = 1620.0K$,$\phi_1 = 0.75$ and $T_{02} = 1790.0K$,$\phi_2 = 1.45$.



(a) CO_2 (b) H_2O

Figure 3.19: Continuous inference simulation in logarithmic scale for C_2H_4/air case with 2 trajectories: $T_{01} = 1620.0K$,$\phi_1 = 0.75$ and $T_{02} = 1790.0K$,$\phi_2 = 1.45$.



(a) CO_2 (b) H_2O

Figure 3.20: Continuous inference simulation in logarithmic scale for CH_4/air case with 2 trajectories: $T_{01} = 1620.0K$,$\phi_1 = 0.75$ and $T_{02} = 1790.0K$,$\phi_2 = 1.45$.

3.2.6 Computing performance

Direct acceleration under the solver

The performance of the DNNs in terms of computational cost is assessed in this section. A single 0-D reactor is computed and the cost of the simulation is analyzed. The Python framework used in the above DNN study is not adapted to performance analysis as the Tensorflow DNN inference involves overhead costs. Tests on computational speed are therefore performed using the NNICE[120] library, which is an in-house C++ code

providing lightweight DNN inference capabilities without relying on Machine Learning libraries C++ APIs. The NNICE library is easily coupled to any CFD code and is relevant for estimating gains which can be expected by replacing a direct integrator with DNNs in this context.

To evaluate the computing time for each time step resolution, a comparison is made between CVODE integration and ANN prediction. Figure 3.21 displays curves representing the acceleration ratio $t_{cvode} : t_{DNN}$ for each iteration during the simulation of C_2H_4. The initial condition of this test simulation is $T_0 = 1700.0K$ and $\phi = 1.0$, and the learning workflow uses the optimal cluster (i.e., 4). Three DNN models with different sizes are tested while preserving the simulation resolution accuracy, and general information about these models is provided in Table 3.7. The values in the figure represent the resolution time cost for each iteration with the evolution of time.

DNN model	n_r	n_e	model size for each cluster
DNN_1	1	150	462.7 kB
DNN_2	2	120	554.7 kB
DNN_3	2	150	830.9 kB

Table 3.7: Models information for DNN performance testing

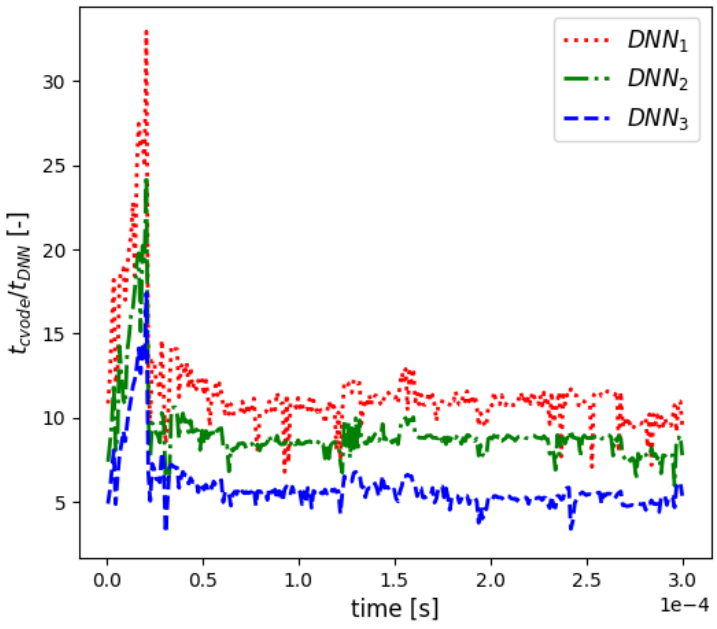


Figure 3.21: The acceleration ratio t_{cvode}/t_{DNN} between CVODE resolution and DNN prediction resolution for each time step, curves for three models with different sizes.

The results clearly demonstrate that when solving thermochemical states, predictions made by the DNN workflow are over 5 to 30 times faster than those obtained using

CVODE resolutions. Notably, DNN predictions for both stiff (from $0.0s$ to $4 \times 10^{-5}s$) and non-stiff regions exhibit the same computing efficiency, whereas the CVODE solver incurs higher costs due to the use of implicit schemes involving Jacobian matrix computations. This difference is particularly prominent at the starting time with stiff chemistry, where the acceleration ratio is much larger. Despite a slight increase in computing time with larger model sizes, the DNN workflow still offers significant computational gains over CVODE integration.

3.3 Results with different network design

This section explores the studies conducted on the neural networks introduced in Chapter 2. Three distinct networks—ResMLP with a primary backbone layer, EulerNET designed as an Euler explicit scheme, and RK4NET designed as a Runge-Kutta 4 explicit scheme—are evaluated under the 0D combustion simulation scenario.

3.3.1 Network parameters and test case

As mentioned in 2.2.2, the design of neural networks is extended to a topology as ODE schemes for flow map learning. The dataset and preprocessing is based on the same method. To assess model performance and compare it with the ResMLP model, experiments were conducted for 0D auto-ignition simulations involving both C_2H_4 and CH_4 cases. The experimental setup mirrors that of the 0D case with the ResMLP model. Given that increasing the cluster number does not lead to improved overall model performance, we opted for a cluster number of 4 for these two cases. This time, all models were trained using the PyTorch 1.1.10 library, another deep learning library known for its flexibility in implementing user-defined models. Various models with different lengths of residual blocks (n_r) were tested for the discrete ODE network models. To maintain a similar structure for comparison with the discrete ODE models, we selected 4 resblocks, each consisting of two hidden layers, following the backbone layer. As a result, the total layer count for the ResMLP models is 10. Tables 3.8 and 3.9 provide general information about the models used in the experiments. All models were designed to have similar sizes, although the discrete ODE network models have a deeper structure.

Model name	n_r	layers number	model size
ResMLP	4	9	1.0MB
EulerNET-6block	6	12	825.0kB
RK4NET-6block	6	12	825.0kB

Table 3.8: Models used for experiments: C_2H_4 case

Model name	n_r	layers number	model size
ResMLP	4	9	2.2MB
EulerNET-8block	8	16	2.0MB
RK4NET-8block	8	16	2.0MB

Table 3.9: Models used for experiments: CH_4 case

3.3.2 Training performance

Tables 3.10 and 3.11 provide the Mean Absolute Errors (MAE) on the validation dataset after the model training process. Overall, the training performance of the three models is quite similar. In some clusters, the ODE-type network models may yield slightly smaller validation MAE errors, while in others, the ResMLP model produces smaller values. It's noteworthy that the ODE-type network models consistently maintain similar training performance throughout our experiments.

Model name	cluster 0	cluster 1	cluster 2	cluster 3
ResMLP	9.07×10^{-5}	9.24×10^{-5}	4.85×10^{-4}	5.51×10^{-4}
EulerNET-6block	1.28×10^{-4}	9.31×10^{-5}	4.00×10^{-4}	6.43×10^{-4}
RK4NET-6block	1.47×10^{-4}	6.47×10^{-5}	5.47×10^{-4}	6.56×10^{-4}

Table 3.10: Validation MAE errors for C_2H_4 case

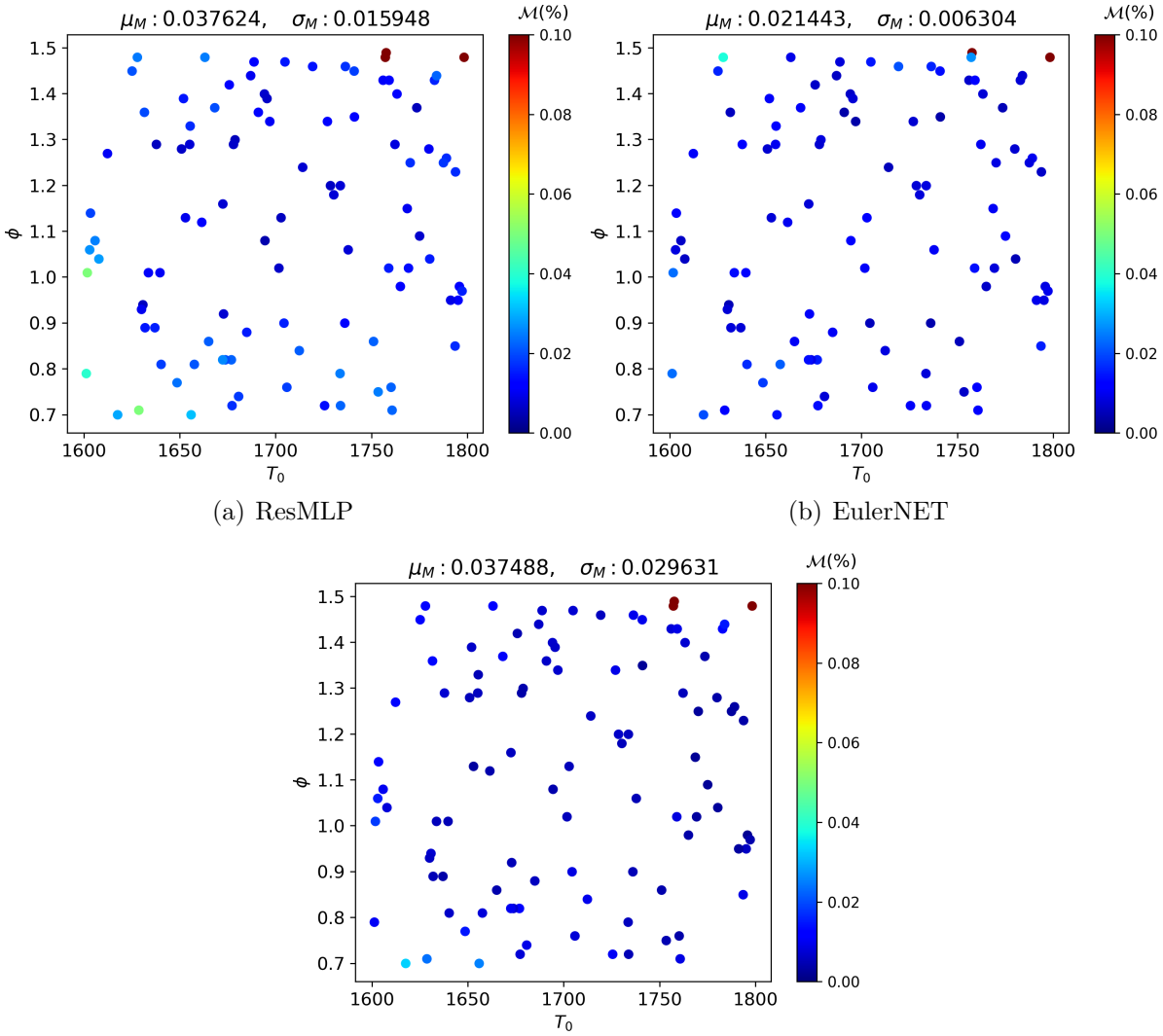
Model name	cluster 0	cluster 1	cluster 2	cluster 3
ResMLP	2.09×10^{-4}	2.88×10^{-4}	4.97×10^{-4}	1.20×10^{-4}
EulerNET-8block	9.80×10^{-5}	2.48×10^{-4}	6.29×10^{-4}	1.25×10^{-4}
RK4NET-8block	1.20×10^{-4}	2.32×10^{-4}	8.19×10^{-4}	1.44×10^{-4}

Table 3.11: Validation MAE errors for CH_4 case

3.3.3 Generalization performance

Figure 3.23 illustrates the cumulative average logMAPE error for inference simulations using initial conditions from the test dataset. The iterative predictions exhibit overall strong performance, with small logMAPE errors consistently below 0.1% for the C_2H_4 case and approximately 2.0% for the CH_4 case. Moreover, for the C_2H_4 case, all three models demonstrate good generalization performance, characterized by fewer extreme outliers in the logMAPE errors at the boundaries of the predefined experiment region, and EulerNET produces the smallest variance of generalization inference. In contrast, for the CH_4 case, both EulerNET and RK4NET exhibit better generalization performance regarding to the mean and variance of generalization inference. Most of the outliers are concentrated in regions of low temperature and low equivalent ratio, where simulations

with these initial conditions necessitate a larger number of iterations to reach the final time step, leading to larger accumulated model prediction errors.

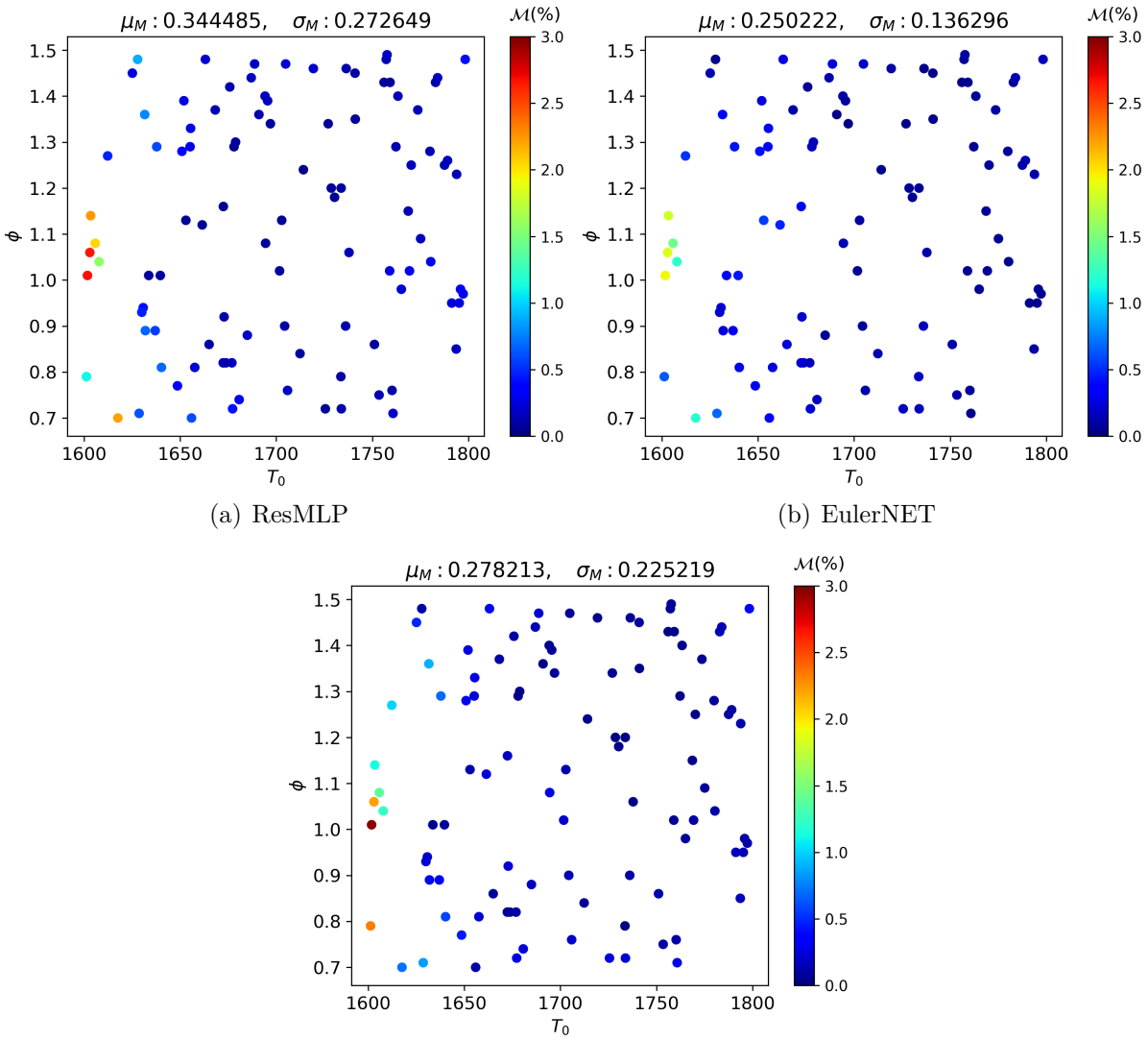


(a) ResMLP

(b) EulerNET

(c) RK4NET

Figure 3.22: The accumulative average logMAPE error of all states for initial conditions in the test set for C_2H_4, where (a) ResMLP, (b)EulerNET and (c)RK4NET



(a) ResMLP

(b) EulerNET

(c) RK4NET

Figure 3.23: The accumulative average logMAPE error of all states for initial conditions in the test set, for CH_4, where (a) ResMLP, (b)EulerNET and (c)RK4NET

3.4 Conclusion

In this research, a deep learning surrogate model has been successfully trained and applied to predict 0D combustion simulations using different fuels with increasing complexity. Several key strategies have been employed, including a novel sampling method based on adaptive time steps of the CVODE solver to achieve a balanced data distribution, the utilization of a non-supervised K-Means algorithm to partition the dataset into subdomains within logarithmic space, and the application of residual networks to enhance optimization during the training process.The analysis of the inference results demonstrates that by segregating the sampling points into subdomains, the overall complexity of the model prediction process can be reduced while achieving optimal results. The proper selection of hyperparameters plays a crucial role in obtaining accurate iterative predictions.

As an extension of this work, the proposed sampling method based on implicit numerical stiff solvers can prove valuable for incorporating ignition phenomena in the construction of datasets for multi-dimensional simulations involving other combustion regimes. This would enable the extension of the learning workflow to tackle more complex problems that involve convection and diffusion in computational fluid dynamics. Furthermore, the proposed clustering workflow and discrete ODE-type models can be further explored for predicting complex chemical states in combination with dimension reduction techniques. These aspects will be detailed in the next chapter.

Chapter 4

Adaptive latent dynamic learning for complex chemistry systems

As presented and analyzed in Chapter 3, neural network models combined with non-supervised clustering algorithms can be applied to predict the thermochemistry of combustion systems for various fuels, ranging from simple to complex chemistry. In this chapter, we extend our study of thermochemical state prediction to a more advanced workflow. This workflow involves the application of dimension reduction techniques to project the original states into latent space states, followed by learning the dynamics directly in the latent space. We apply our method to the CH_4 combustion chemical system, which consists of 53 species with complex chemical reaction mechanisms.

Résumé français

Comme présenté et analysé dans le chapitre 3, les modèles de réseaux neuronaux combinés à des algorithmes de clustering non supervisés peuvent être appliqués pour prédire la thermochimie des systèmes de combustion de divers carburants, de la chimie simple vers la chimie complexe. Dans ce chapitre, nous étendons notre étude de la prédiction de l'état thermochimique à un prototype plus avancé. Ce prototype implique l'application de techniques de réduction de dimension pour projeter les états d'origine dans des états d'espace latents, suivis de l'apprentissage de la dynamique directement dans l'espace latent. Nous appliquons notre méthode au système chimique de combustion CH_4, qui se compose de 53 espèces avec des mécanismes de réaction chimique complexes.

4.1 General methodology

4.1.1 Autonomous dynamical system

Using the operator splitting method, the computation of the chemical system and transport system is separated and carried out individually. The chemical dynamical system, after one step of CFD resolution, is described as an autonomous system. This autonomous system is described as in 2.1.6, and the integral form of the system is given by 2.2.7. Recall that the variable $\mathbf{S}(t) = [\mathbf{Y}(t), T(t)]$ denotes the thermochemical states with temperature and mass fraction of chemical species, and $\mathbf{S}_0$ denotes the initial state. For the chemical states prediction, the resolution time step is denoted by $\Delta t = dt_{cfd}$ as in Section 3.1. During the combustion simulation, starting from an initial condition, one can collect piece-wise dynamic flow maps from t_i to $t_i + \Delta t$, which is a one-step resolution from input states to output states. The data samples in this case, therefore, is constructed by:

$$\mathcal{D} = \{(\mathbf{S}(t_k), \mathbf{S}(t_k + \Delta t))|, \mathbf{S}(t_k) \in \Omega\} \tag{4.1.1}$$

where Ω denotes the data sampling space. The problem is defined exactly as in Chapter 3, where the dynamic evolution from input time steps to output time steps are learned, and the dataset is constructed based on the data couples from trajectories.

4.1.2 Reduced latent space learning

The primary reasons for employing the dimension reduction technique is to simplify the overall chemistry system of reacting flows, reducing the number of states within the governing system to a more manageable size, making it significantly less computationally expensive to solve. This reveals instrumental when implementing continuous neuralODE models, where the training process becomes exceedingly time-consuming and resource-intensive when dealing with a large number of input states. In this chapter, we leverage dimension reduction techniques to enable the use of neuralODEs for complex chemistry learning. The fundamental latent space learning workflow is depicted in Figure 4.1 and detailed in the following sections.

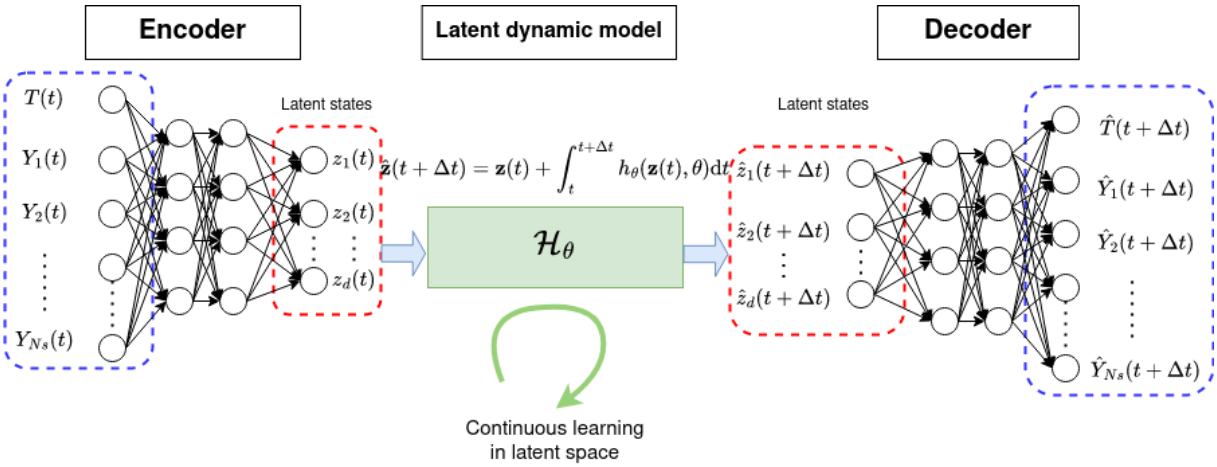


Figure 4.1: The basic latent space learning workflow, with the encoder, decoder and latent space dynamics learning model

4.1.3 Autoencoder

The aim of dimension reduction is to project the original state space onto a latent space, i.e. find a suitable function $E : \mathbb{R}^n \rightarrow \mathbb{R}^d$. Assuming that $\mathbf{S}$ represents the high-dimensional original chemistry states, one can reduce these high-dimensional states to latent space states using this reduced-order operator:

$$\mathbf{z}(t) = E(\mathbf{S}(t)) \tag{4.1.2}$$

where E denotes the linear or nonlinear reduced-order operator, also called the encoder. After the nonlinear projection, the latent dynamical system is described using the latent states $\mathbf{z}$, and the latent ODE system is denoted as:

$$\frac{d\mathbf{z}(t)}{dt} = h(\mathbf{z}(t), r) \tag{4.1.3}$$

where h is the latent dynamic, and r is the reduced system parameters. The latent space states in reduced system can be resolved knowing the reduce dynamic h which is identified by data-driven techniques[90, 102], or propagated directly by data-driven models. After propagating the latent dynamics, the resulting reduced states can be mapped back to the original state-space as follows

$$\mathbf{S}(t) = D(\mathbf{z}(t)) \tag{4.1.4}$$

where D is referred to as the decoder. It can be also linear or nonlinear. The reconstruction error of original state space is a crucial standard to evaluate the performance of reduced-order models. Nonlinear encoder-decoders, compared to linear ones, have the potential to preserve more features of the original space, thus leading to lower reconstruc-

tion errors, for the same latent space dimension d. This, of course, comes at the price of a more complex optimization problem to solve to find suitable model parameters.

The choice of the latent space dimension d is a non-trivial problem, as a too small dimension will lose much information from original state-space. To identify the importance of the basis for a data space, a popular method is rely on Singular Value Decomposition(SVD). The basic theory of matrix approximation is introduced in Section 2.2.3. The reconstruction percentage is a measure of how well the original full system can be approximated using a group of major singular values while eliminating the minor ones below a specified threshold. This percentage provides insight into how much information is retained in the reduced representation of the system. It is calculated as the fraction between the summation of singular values that are preserved and the total summation of all singular values:

$$\sum_{i=1}^{k} \sigma_i = \epsilon \sum_{i=1}^{r} \sigma_i \tag{4.1.5}$$

In this workflow, a pre-defined reconstruction ratio (in percentage), denoted as ϵ, is set. The summation of singular values is clipped to reach the reconstruction percentage, and the rested minor singular values are eliminated. Consequently, the reduced dimension is determined by the number of major singular values that are preserved. This SVD reconstruction process is applied adaptively for each subdomain after the data has been partitioned. Each subdomain has its own reduced latent dimension within this workflow, based on the ϵ ratio.

4.1.4 Local latent dynamic learning

After the dimension reduction, the reduced-order latent space dynamic in integral form is expressed as:

$$\begin{aligned} &\mathbf{z}(t+\Delta t) = \mathbf{z}(t) + \int_{t}^{t+\Delta t} h(\mathbf{z}(t), r)\mathrm{d}t \\ &\mathbf{z}(t_0) = \mathbf{z}_0 \end{aligned} \tag{4.1.6}$$

In some scenarios, the latent state dynamics are known: this is e.g. the case when the reduced-order models is obtained from linear Galerkin projection. However, in this research, the reduced-order manifold is constructed from the data. This implies that the latent state dynamics themselves must be learnt, for each subdomain defined by the clustering algorithm.

To this end, two different approaches are applied in this work. The first approach involves learning the "flow maps" of dynamic evolution. This workflow aims to predict the value of the latent states at the next time step $z(t+\Delta t)$ directly from $z(t)$, all along the trajectory, using a simple feed-forward neural network model as introduced in Chapter

3 or a series of residual network operators. The second approach relies on Neural ODEs, which are described in Section 2.2.2. It aims at learning a suitable function $h(\cdot, r)$ which, when integrated using an adaptive ODE scheme, yields an accurate prediction of the thermochemical states.

4.1.5 Model conception and training

The proposed framework thus involves training several neural networks: one for the encoder, one for the latent space dynamics (either for direct prediction or neural ODE), and one for the decoder. In this work, a single optimization problem enabling the learning of parameters for all three models is formulated. To derive the loss function, several components are considered, including reconstruction errors, errors in latent dynamic predictions, projection consistency, and the conservation of physically informed elements, as we now detail.

Reconstruction loss function

The reconstruction error is designed for the autoencoder, to guarantee that the original state vectors can be projected to latent space and inversely projected to original space without including large deviations. It is formulated as a non-supervised learning problem, not requiring the labelling of data. Denoting the encoder by E_θ and the decoder by D_θ, the reconstruction loss function is expressed as:

$$\mathcal{L}_{rec} = ||\mathbf{S}(t) - E_\theta * D_\theta * \mathbf{S}(t)|| \tag{4.1.7}$$

where the $\mathbf{S}(t)$ is the input time states in t. This formulation shows that the error is evaluated between the ground truth original states and the states after the original states pass to the encoder and decoder model.

Latent dynamic loss function

The latent dynamic prediction error is based on the output states in latent space, and is written as:

$$\mathcal{L}_{dyn} = ||\hat{\mathbf{z}}(t + \Delta t) - E_\theta * \mathbf{S}(t + \Delta t)|| \tag{4.1.8}$$

The ground truth of predicted states in latent space is the projection of the ground truth of output states in original space. As the encoder and decoder model do not share the same group of parameters, an extra loss function terms which is served as the preservation of consistency is included:

$$\mathcal{L}_{consist} = ||\mathbf{S}(t + \Delta t) - D_\theta * \hat{\mathbf{z}}(t + \Delta t)|| \tag{4.1.9}$$

From this term, it can be seen that the consistency error is the same as the prediction error of the output states in the original space. This is the inverse of the $\mathcal{L}_{dyn}$, and the predicted states in latent space is inversely projected to original space. This process ensures that the autoencoder has a good consistency of projection and inverse projection during the latent dynamic learning process.

Finally, the overall loss function is the weighted sum of all above terms, which is formulated as:

$$\mathcal{L} = \mathcal{L}_{dyn} + \alpha(\mathcal{L}_{rec} + \mathcal{L}_{consist}) \tag{4.1.10}$$

Regarding the optimization loss function, one of the major objectives is to learn the latent dynamic, yielding the loss term with states in the latent space. Besides, the composition of reconstruction and consistency terms makes sure that the autoencoder is also well trained. The parameter α is a crucial parameter to balance the weights between the latent dynamic learning and the reduced order model learning, and needs to be tuned to find a equilibrium between the model reconstruction error and the dynamic states prediction error.

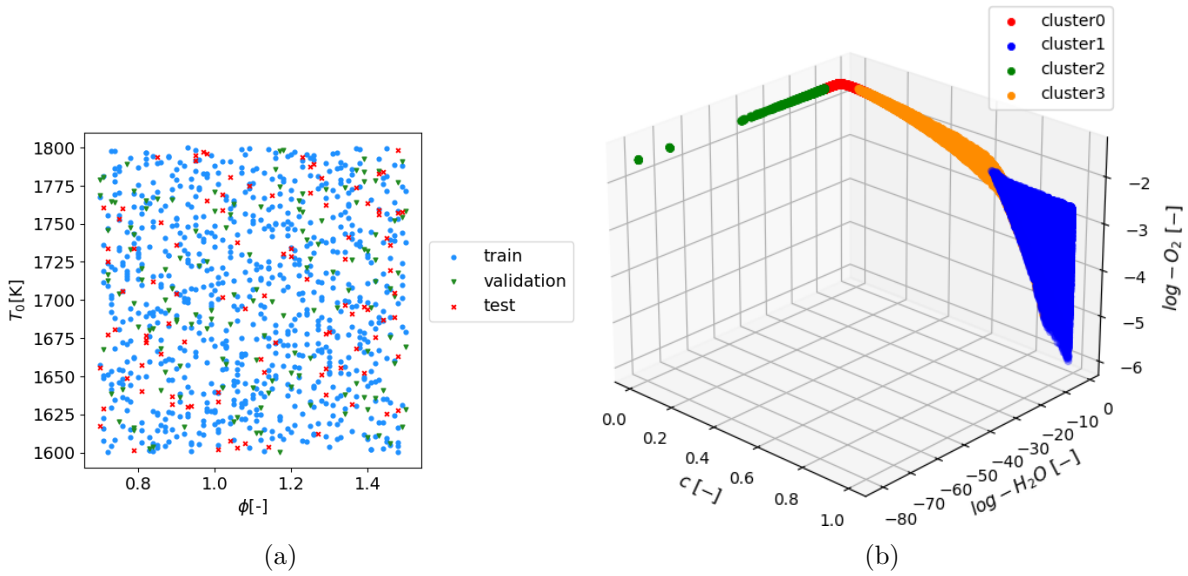
By defining the overall loss function, the general workflow for the learning system is constructed. For a versatile learning process, the original data samples extracted from combustion simulation by a predefined problem are passed to the data clustering step, then once they are partitioned to subdomains, the data from each subdomain are preprocessed, and projected to local latent space by adaptive encoder models. The local dynamics are learned and the output states are predicted by latent dynamics models, then the latent states are inversely projected and transformed to expected output states.

4.2 Application to high dimensional complex chemistry

The test case for this workflow is based on the study of 0D combustion for CH_4, which features 53 thermochemical states and 325 chemical reactions.

4.2.1 Dataset generation

The dataset is generated as in Chapter 3. Samples are constructed as input-output pairs, where the output is computed by the chemistry solver CVODE by integrating the dynamics over one prediction time step $\Delta t = \Delta t_{cfd}$ from the input state. The sampling sequence is generated by high-order adaptive CVODE resolution time steps as introduced in Section 2.1.2. A dataset with multiple initial conditions is generated, which contains 0D resolution states and includes more samples of reactive states of homogeneous autoignition by using our adaptive sampling method. The range of initial conditions is fixed



(a) (b)

Figure 4.2: The generated dataset of simulation for 0D auto-ignition problem. (a) Sampled initial conditions. (b) Simulation trajectories

as in Chapter 3, where $T \in [1600K, 1800K]$, and $\phi \in [0.7, 1.5]$. The separation of training and test data is the same process, where 1000 initial conditions are randomly chosen by Latin Hypercube Sampling (LHS), and 75% initial conditions are used to launch the simulation trajectories of training states, 15% initial conditions are used as validation set to evaluate the metrics of model training performance after each epoch, and the rest 10% initial conditions are kept to evaluate the a posteriori performance of inference simulations. Each single simulation is performed until the tolerance (as presented in chapter 3) reaches 10^{-4} or the simulation time arrives at $10^{-3}s$. The generated data is shown in Figure 4.2, where (a) represents the sampled initial conditions on the (T_0, ϕ) phase plot, and (b) is the simulations generated data sampling manifold in (H_2O, O_2, c) phase plot, and c represents the progress variable defined by the evolution of T.

The data pre-processing is also based on the same logic. The original dataset is clipped by a threshold 10^{-28} which is used to eliminate the extremely small and physically non meaningful values. The data samples are normalized by standard normalization:

$$\hat{\mathbf{S}} = \frac{\ln(\mathbf{S}) - \mu}{\sigma} \tag{4.2.1}$$

Where μ and σ denotes the mean and variance of data samples after logarithmic transformation. After that, the dataset is partitioned into 4 different clusters using the K-Means algorithm.

4.2.2 Hyper-parameters setting

The autoencoder structure is fixed with two hidden layers and 150 neurons for each layer, with the same number of parameters for the encoder and the decoder. The total size of the autoencoder model is 510.2kB, requiring a low amount of memory to be stored.

The elemental block of discrete ODE network and the neural network for continuous neural ODE are designed by the same structure, while for the discrete ODE network the number of multiple residual blocks needs to be fixed. A summary of the different models for latent dynamic learning is provided by table 4.1. The structure design of neural network for each model is kept by the same number of residual blocks and neurons in each layer. As for the discrete ODE network models, the number of residual blocks is fixed to 10. Obviously the neural ODE model has less memory storage requirements due to a lesser number of parameters: it only needs a single neural network to approximate the continuous dynamics.

Model name	Hidden layers number	Neurons in layer	Residual blocks number	model size
EulerNET	2	150	10	2.0-2.2 MB
RK4NET	2	150	10	2.0-2.2 MB
NODE	2	150	/	198-217 KB

Table 4.1: A summary of different designed models with hyper-parameters setting

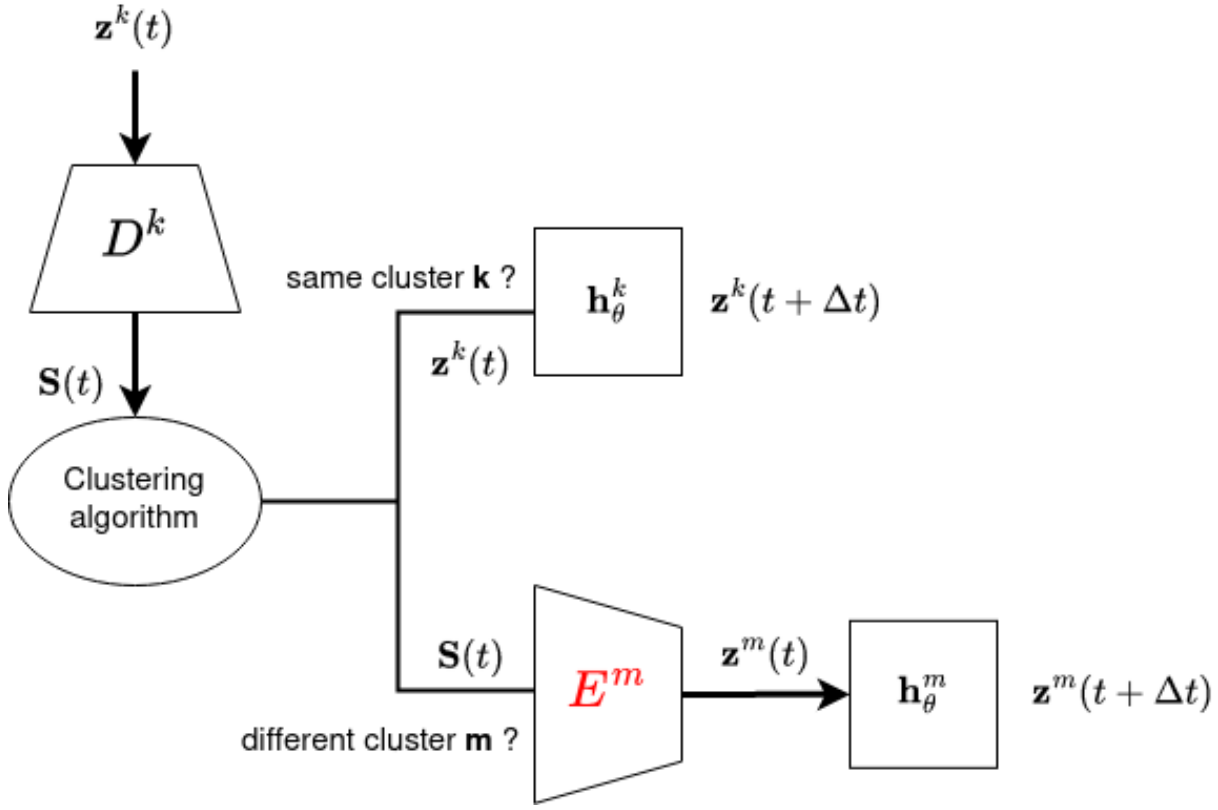


Figure 4.3: The general inference procedure for each time step prediction

The activation function of the neural networks used in this study is the *swish* function[116], and the initializer of the models is the Glorot uniform initialzer. The optimization algorithm for model training is the Adam algorithm, with an initial learning rate of 0.005. Learning rate decay is also applied with a decay rate of 0.92 and a decay step of 650 epochs.

4.2.3 Inference simulation and performance evaluation

The inference simulations are based on new initial conditions that are not included in the training dataset. The general inference algorithm is outlined in Figure 4.3. For each time step prediction, if the cluster index does not change, the prediction will continue to be carried out in the same latent space without additional encoder-decoder processing. However, if the cluster index changes, the input will be switched to the new latent space using the corresponding encoder model.

The criterion to evaluate the continuous inference is the global accumulative logarithmic mean average percentage error (logMAPE) $\mathcal{M}_i$ as introduced in chapter 3:

$$\begin{aligned}\mathcal{M}_0(T_0, p_0, \phi) &= \frac{1}{N_{iter}} \sum_{k=1}^{N_{iter}} \left| \frac{T^{pred}(kdt) - T(kdt)}{T(kdt)} \right| \\ \mathcal{M}_i(T_0, p_0, \phi) &= \frac{1}{N_{iter}} \sum_{k=1}^{N_{iter}} \left| \frac{\ln(Y_i^{pred}(kdt)) - \ln(Y_i(kdt))}{\ln(Y_i(kdt))} \right| \quad i = 1, ..., N_s\end{aligned} \tag{4.2.2}$$

where i represents each state component (temperature and chemical species), and N_{iter} is the number of total iterations until the final time step prediction which is specific for each simulation with different initial conditions. The overall average errors $\mathcal{M}$ are computed by averaging values of all dimensions, following:

$$\mathcal{M}(T_0, p_0, \phi) = \frac{\mathcal{M}_0(T_0, p_0, \phi) + \sum_{i=1}^{N_s} \mathcal{M}_i(T_0, p_0, \phi)}{N_s + 1} \quad i = 1, ..., N_s \tag{4.2.3}$$

The global errors $\mathcal{M}$ and errors for each state component $\mathcal{M}_i$ are evaluated for all 100 initial conditions in the test set. The potential error accumulation, which may arise when repeatedly predicting the evolution of chemical states until the final simulation time steps, is properly represented by this metric.

4.2.4 Model parameters selection

To systematically evaluate model performance, several critical factors are analyzed, including the α factor within the loss function, the latent model design and the dimension of the latent space in each cluster.

Latent space dimension selection

The latent space dimension d, is determined from the reconstruction ratio ϵ relative to the approximate reconstruction percentage of the total system, as detailed in Section 4.1.3. In this manner, the number of preserved major singular values becomes the latent space dimension, which is different for each subdomain. Figure 4.4 plots the latent space dimension as a function of the reconstruction ratio. It is assumed that a 95% reconstruction ratio retains most of the system information. The figure indicates that only 5 to 14 latent states, depending on the cluster, are required to obtain this accuracy. This corresponds to less than one-fourth of the original state dimension.

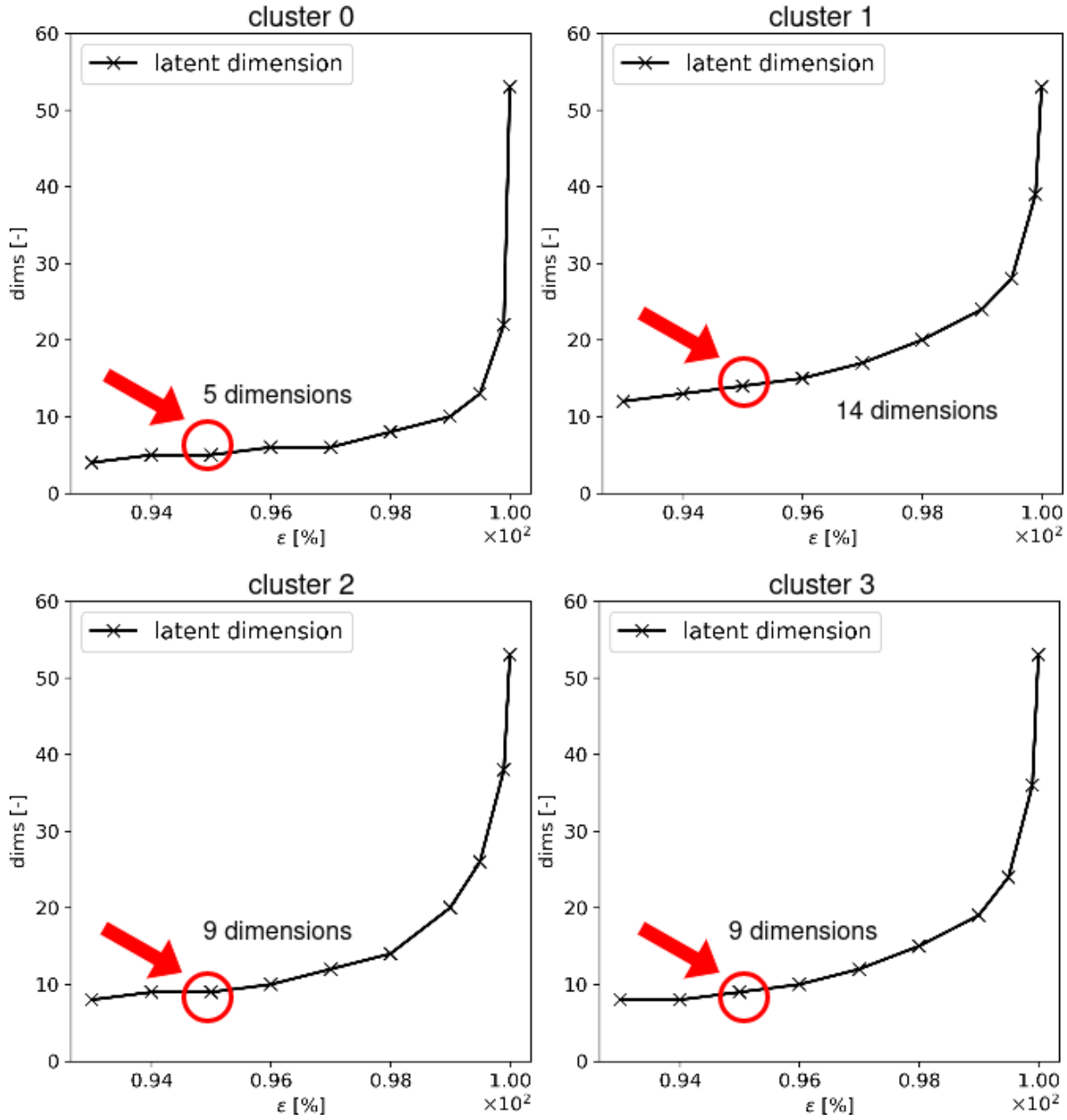
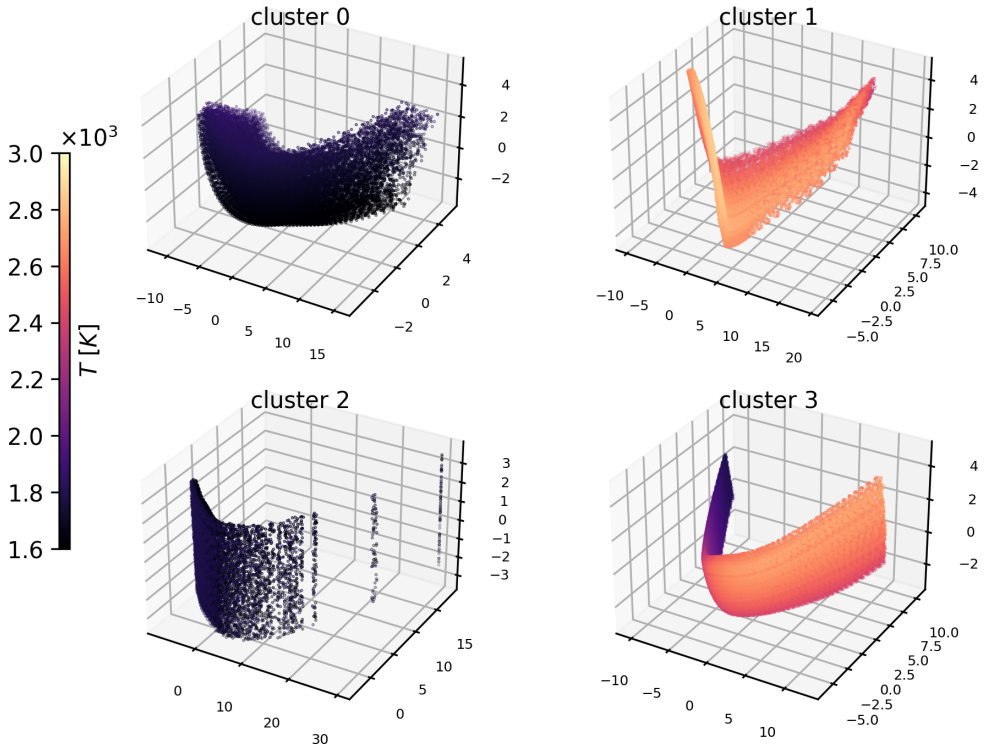
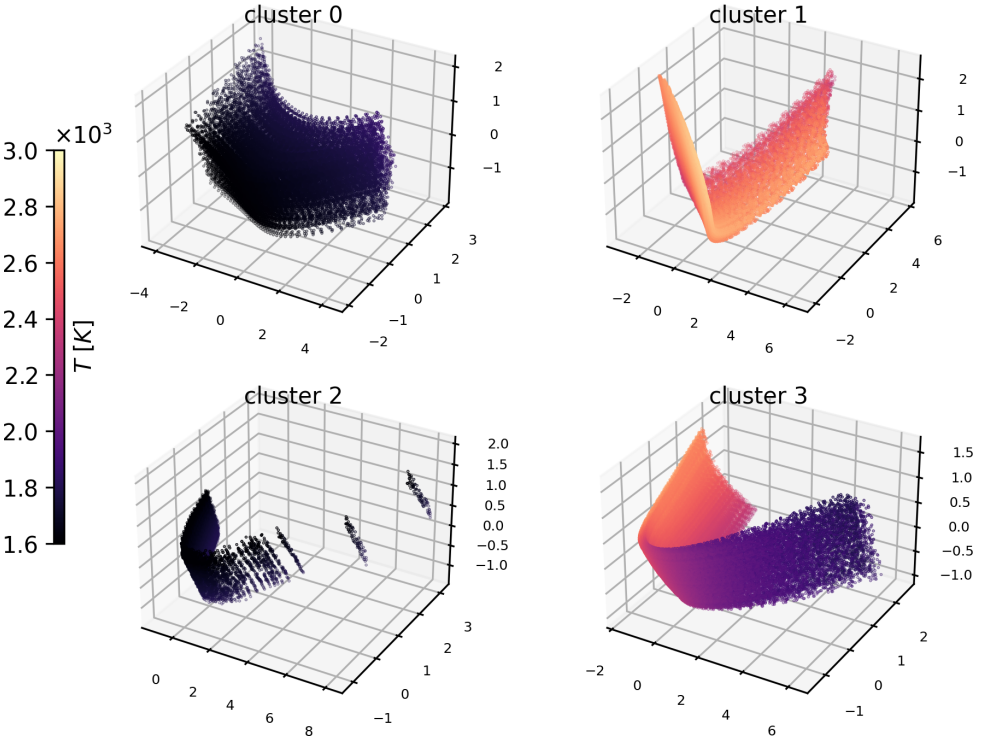


Figure 4.4: The singular values distribution from the maximum value σ_{max} to the minimum value σ_{min}



(a) full states $\mathbf{S}$



(b) latent states $\mathbf{z}$

Figure 4.5: The 3D visualization of states manifolds projected by PCA in 3D space of (a) original full states $\mathbf{S}$ and (b) latent states reduced by encoders $\mathbf{z}$ for each cluster. The visualization is produced by applying PCA analysis and picking up 3 principal components

Figure 4.5 depicts a projection of the dataset on a 3D space, for states from original and latent coordinates. More precisely, a PCA is applied to both the original full states

S and their encoded image z for data visualization. Each dot on the plots of Figure 4.5 corresponds to the first three principal components of one point in the dataset. It is worth noting that when autoencoders are employed for dimension reduction, the overall topologies of the manifold are preserved across all clusters, albeit reduced to smaller variation state regions. This preservation ensures that the major information within the original dataset is retained.

Value of the α factor for loss function

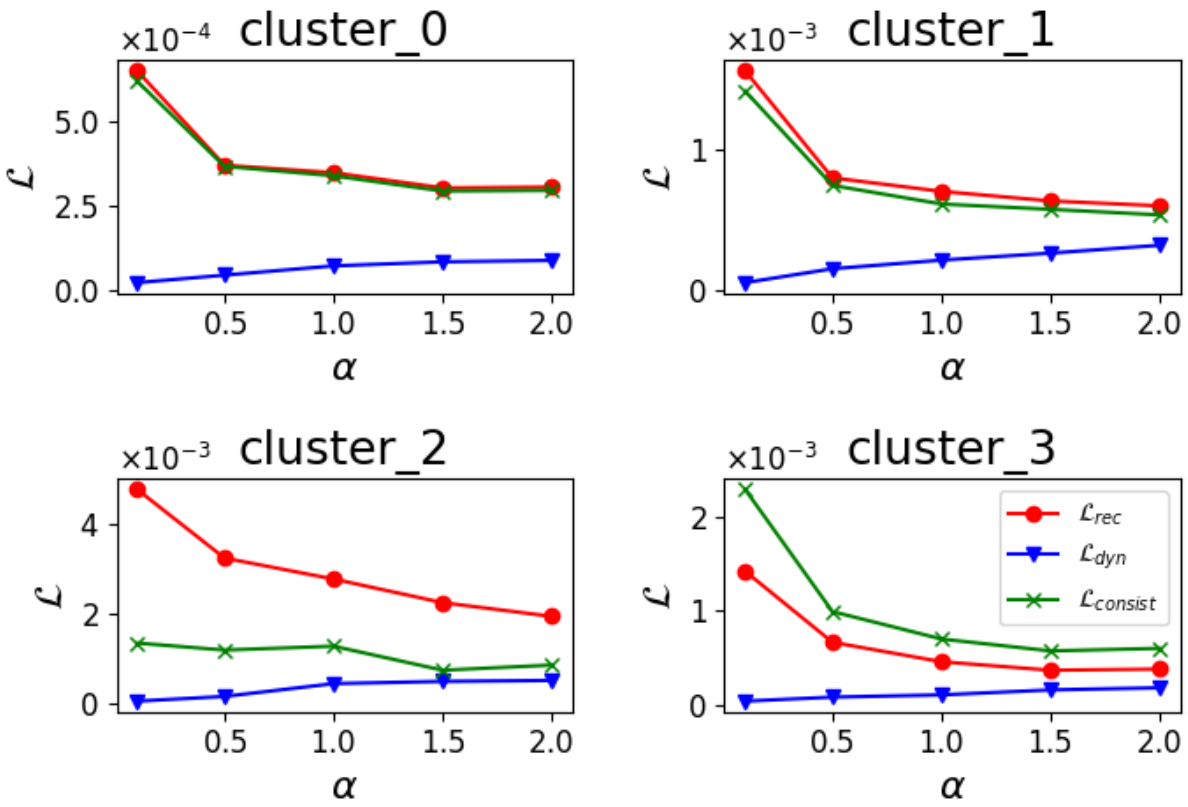


Figure 4.6: Loss terms distribution on the validation dataset using different α factors.

To determine the most suitable value for the α factor in the loss function (4.1.10), multiple training experiments were conducted with different α, by adjusting α with different values, ranging from 0.1 to 2.0, using the NODE model with a reconstruction ratio of $\epsilon = 95\%$. Figure 4.6 plots the values of the three terms of the loss as a function of α. The values of the reconstruction and consistency loss tend to decrease as α increases, while the dynamic prediction loss increases. This trend is attributed to the fact that a larger α assigns greater weight to the reconstruction and consistency loss terms within the total loss function during the multi-objective optimization process. As a balanced choice, the α factor is fixed at 1.5 for the remainder of this study.

Furthermore, it can be observed that for cluster 0 and cluster 1, the reconstruction loss and the consistency loss exhibit almost identical values for all values of α. In the case of cluster 3, there are slight differences, while for cluster 2, these two terms differ significantly. This discrepancy suggests that the dynamics in cluster 2 are more challenging to reduce. Figure 4.2(b) further supports this by indicating that cluster 2 exhibits the highest variability within the overall system. Figure 4.5 also shows that the manifolds generated by samples in logarithmic space for cluster 2 are sparse. The transformation of original states into logarithmic states serves to normalize extremely small values, which originally span different scales, resulting in states with significant variance.

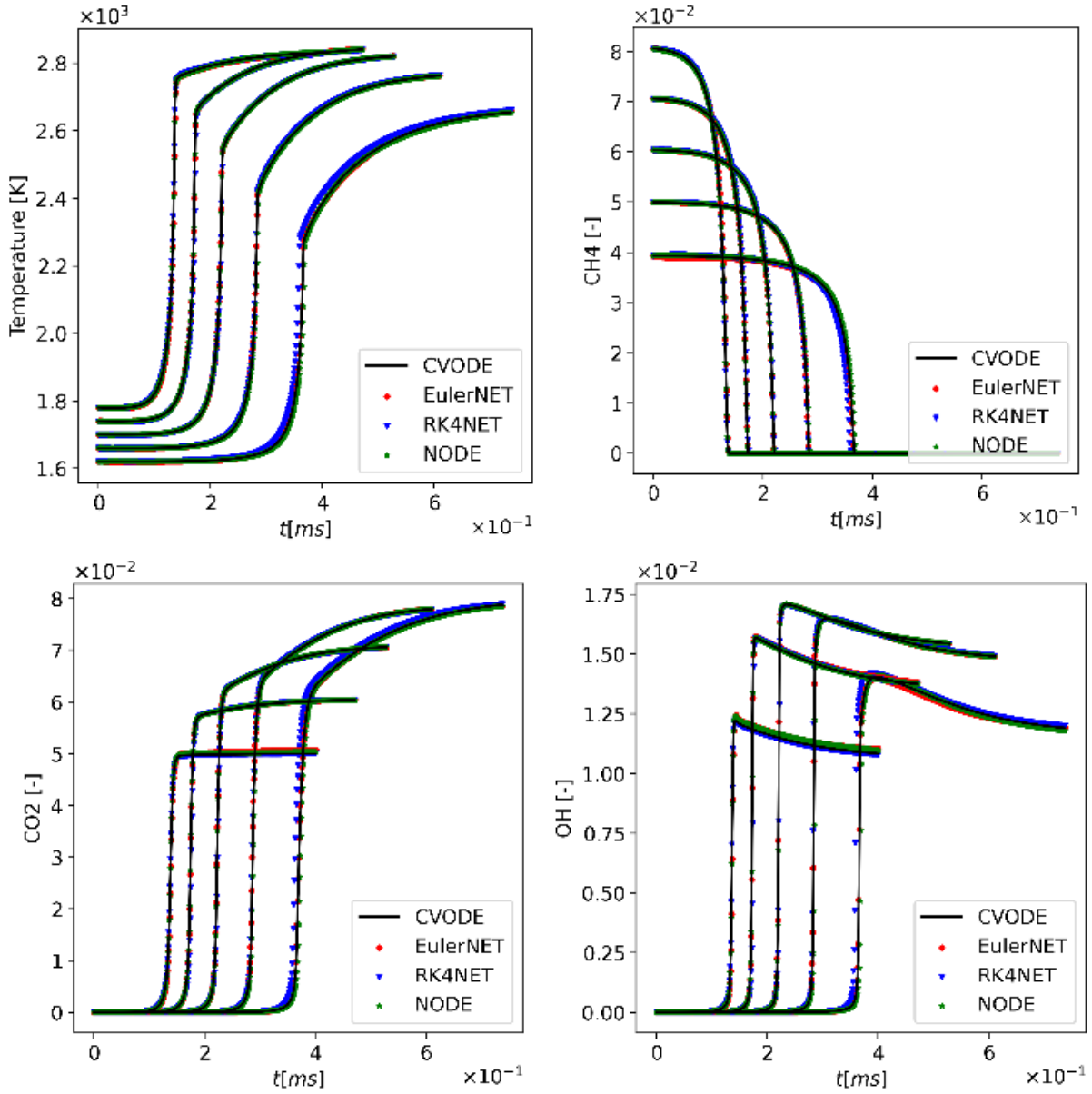
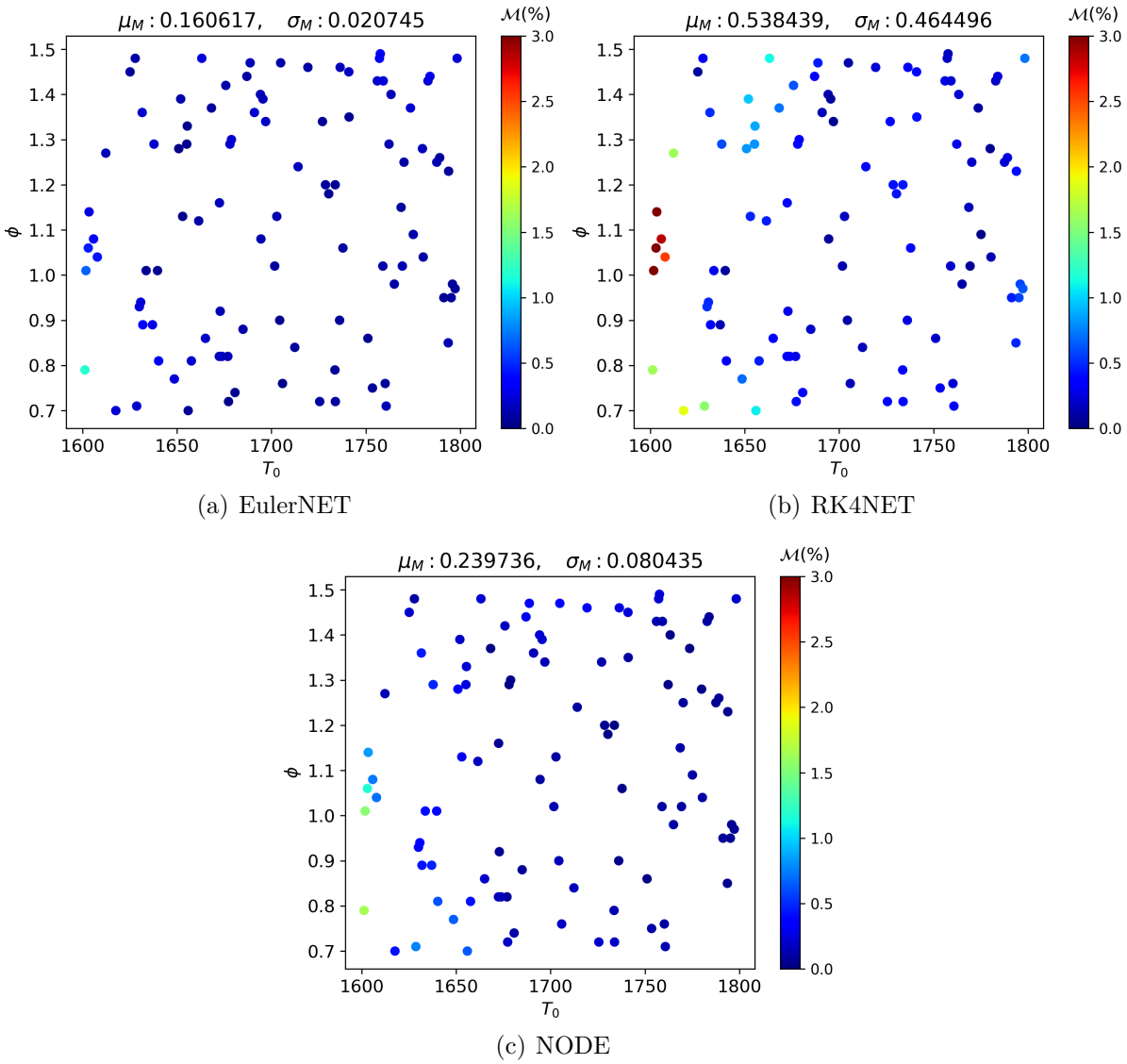


Figure 4.7: Numerical simulation of 0D auto-ignition, with 5 initial conditions $T = 1620.0K, 1660.0K, 1700.0K, 1740.0K, 1780.0K$, $\phi = 0.7, 0.9, 1.1, 1.3, 1.5$. 3 cases with different models are tested using $\epsilon = 95\%$ for the choice of latent dimension numbers

Results

Table 4.2 provides the validation dynamic prediction loss of each cluster in the three different model scenarios. In terms of the dynamic prediction loss values, it is observed that the discrete ODE network EulerNET and RK4NET exhibits slightly smaller values for clusters 1, 2 and 3, while continuous NODE models show smaller values for cluster 0. It is important to note that the precise final loss may vary with each training session. On a global scale, all three models demonstrate similar training performance, maintaining the dynamic prediction loss within the 10^{-4} range.



(a) EulerNET

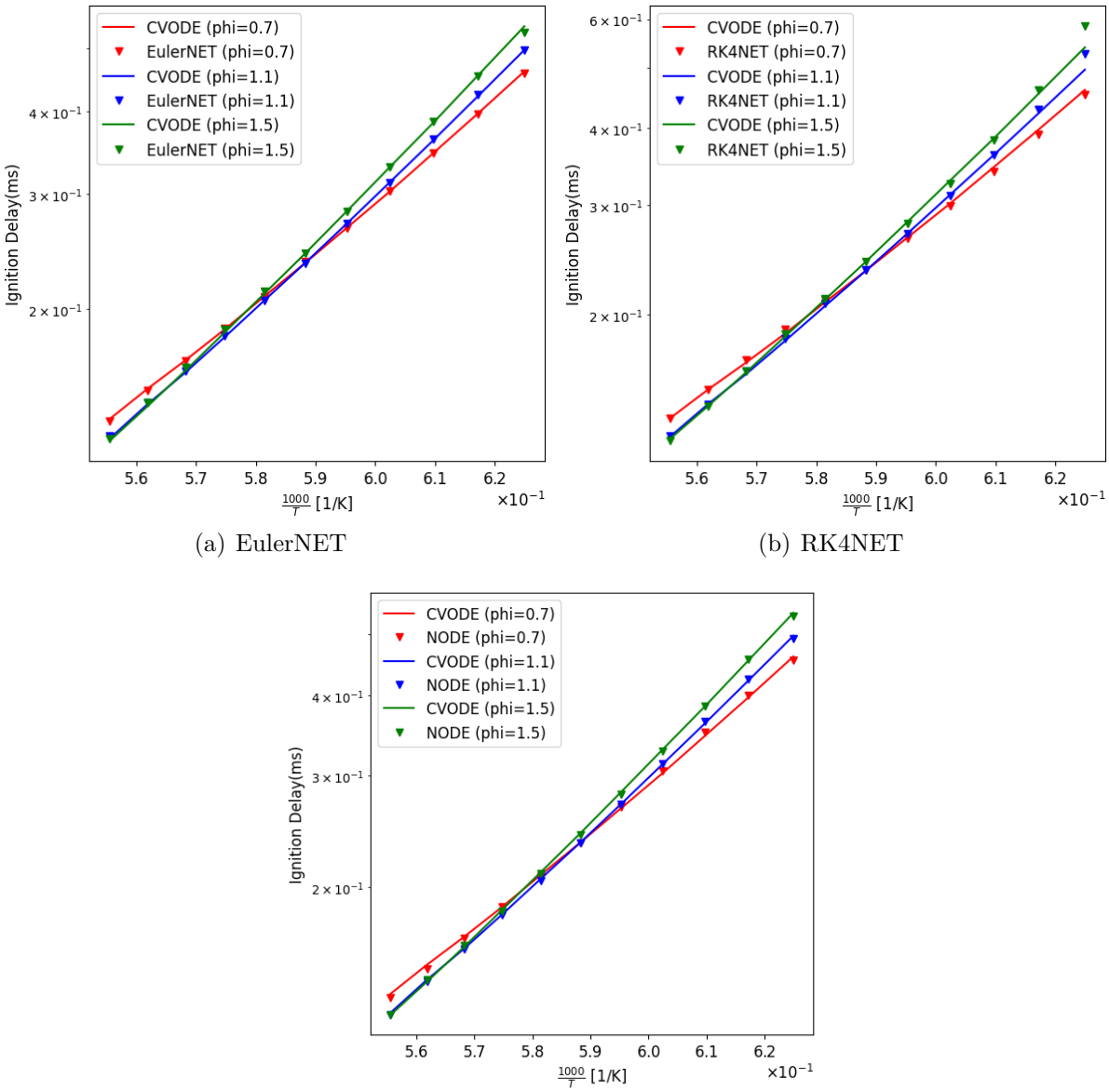
(b) RK4NET

(c) NODE

Figure 4.8: The accumulative average logMAPE error of all states for initial conditions in the test set, for (a):EulerNET, (b):RK4NET and (c): NODE

Model	Cluster 0	Cluster 1	Cluster 2	Cluster 3
EulerNET	9.44×10^{-5}	2.22×10^{-4}	2.45×10^{-4}	1.43×10^{-4}
RK4NET	9.97×10^{-5}	2.26×10^{-4}	2.32×10^{-4}	1.36×10^{-4}
NODE	8.19×10^{-5}	2.61×10^{-4}	4.89×10^{-4}	1.59×10^{-4}

Table 4.2: Validation dynamic prediction loss of each cluster in the three different model scenarios



(a) EulerNET

(b) RK4NET

(c) NODE

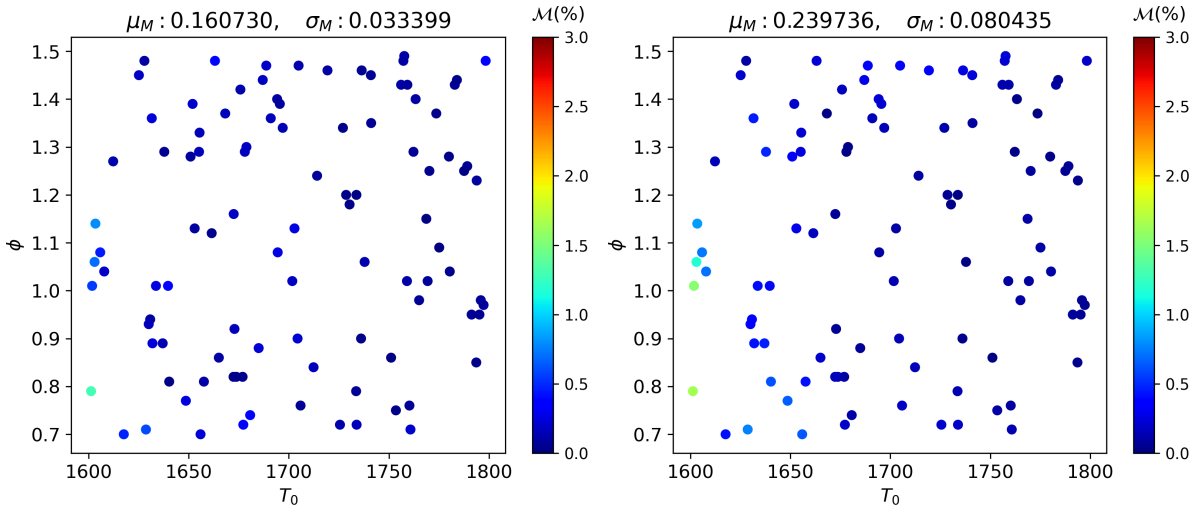
Figure 4.9: The ignition delays between the CVODE simulations and learning model predictions, for (a):EulerNET, (b):RK4NET and (c): NODE

The overall performance throughout the entire simulations in the test set is assessed using appropriate error functions. Figure 4.7 shows the comparison of simulations for 0D auto-ignition under varying initial conditions, where three different dynamic learning

models are used. All three simulations corresponding to these initial conditions align well with the CVODE simulations. However, it is worth noting that only the simulation with an initial condition near the boundary of the pre-defined experimental region produces slightly inaccurate results with ignition delay errors, particularly for the RK4NET model case. This discrepancy is also evident in Figure 4.8, which provides the accumulated average logMAPE error of all states for initial conditions in the test set, and in Figure 4.9, which displays the ignition delays between the CVODE simulations and learning model predictions within the pre-defined experimental region. The μ_M and σ_M denote the mean and variance of logMAPE errors for all 100 initial conditions simulations. Overall, based on the generalization inference results, it can be concluded that in these three experiments, the NODE model and EulerNET models demonstrate better generalization performance in comparison to RK4NET discrete models. Figures showing the prediction of latent dynamics of each clusters under three different models are also provided in Appendix B.

Comparison with direct state-to-state model

A complete comparison is carried out between the NODE reduced-order dynamic learning model (NODE) and the direct state-to-state learning model for full chemical states developed in Chapter 3 (ResMLP). Figure 4.10 provides the accumulative logMAPE errors distribution for 100 initial conditions in test set after using these two models for simulations. As a result, all test inference errors for two cases are below 2.0%. No significant difference of mean logMAPE presents between two cases. The reduced order learning model has the capability of reducing the number of degrees of freedom of the system while keeping a good prediction accuracy.

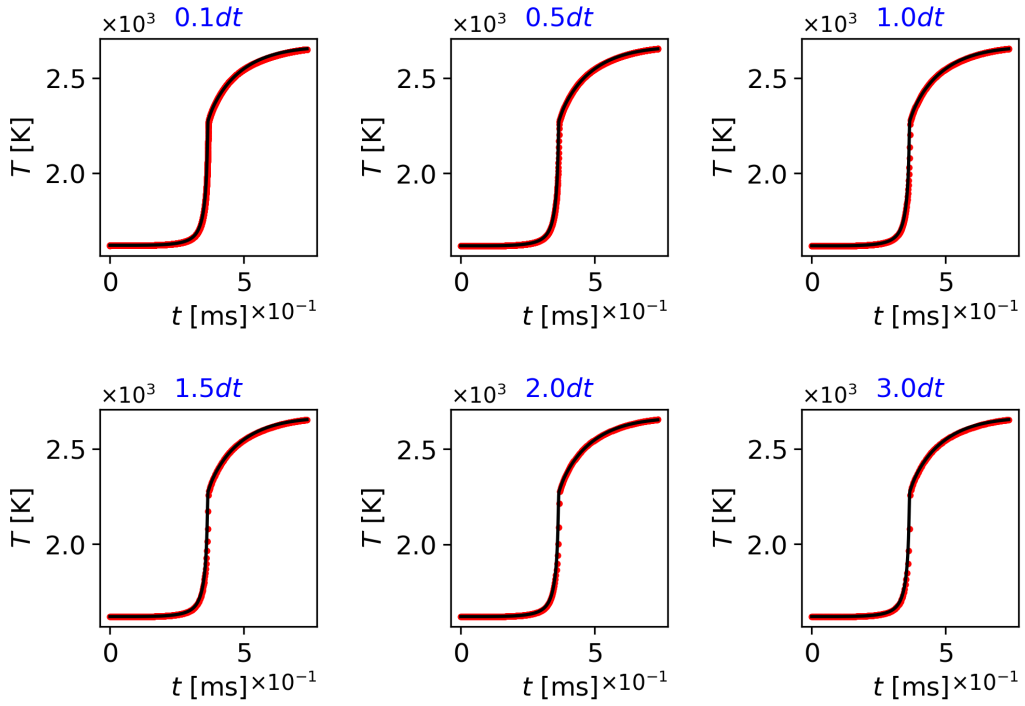


(a) ResMLP(full states)

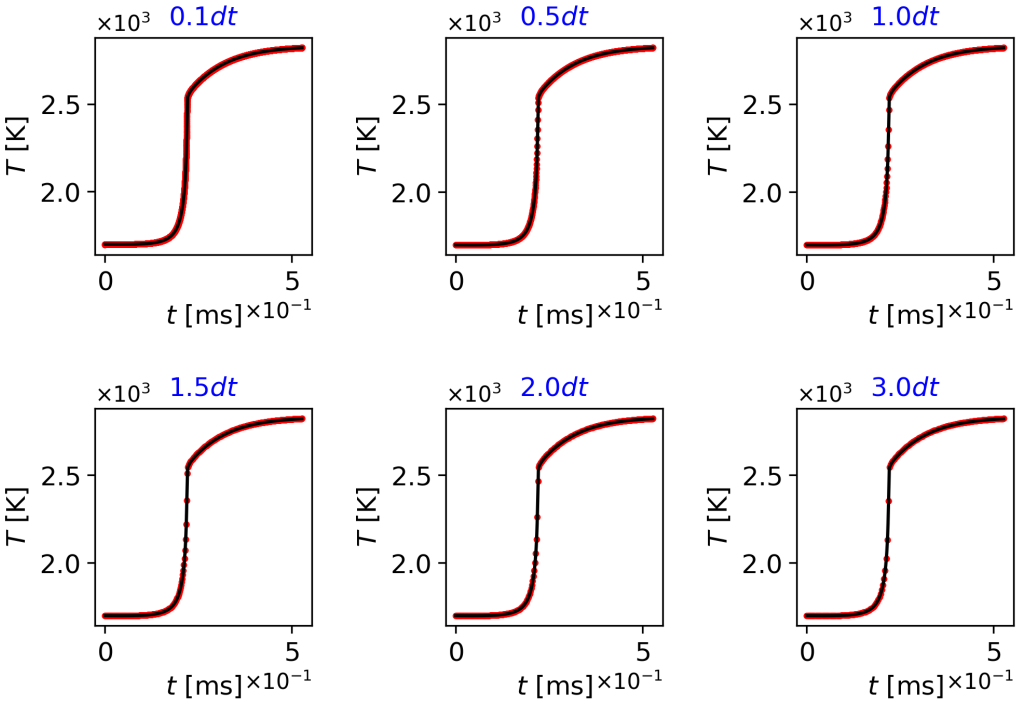
(b) NODE(reduced order states)

Figure 4.10: The accumulative average logMAPE error of all states for initial conditions in the test set, for (a):ResMLP, (b): NODE

Varying prediction time step for NODE

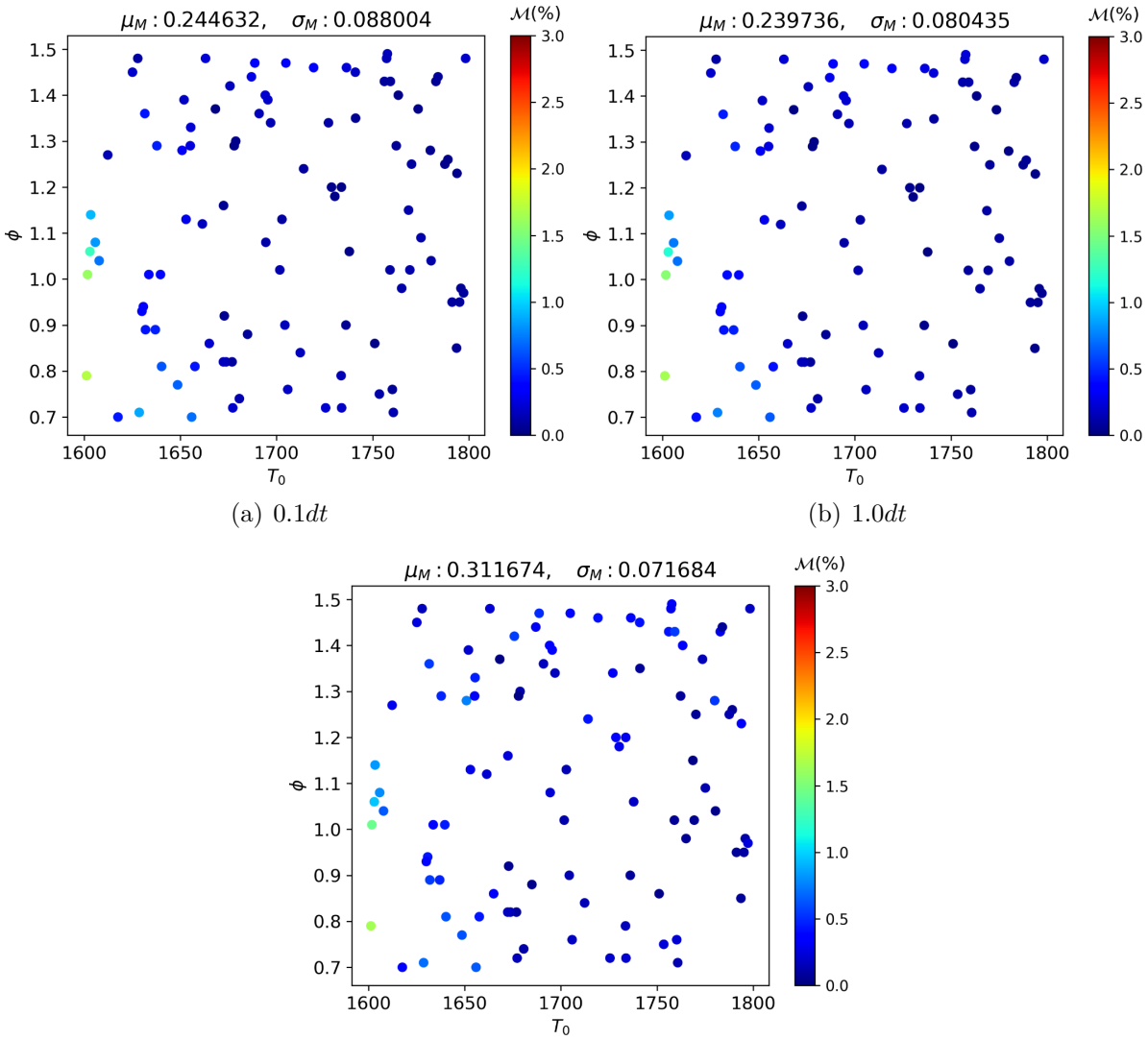


(a) $T_0 = 1620K, \phi = 0.7$



(b) $T_0 = 1700K, \phi = 1.1$

Figure 4.11: The simulation results using the NODE model with various prediction time steps, for (a):$T_0 = 1620K, \phi = 0.7$, (b): $T_0 = 1700K, \phi = 1.1$



(a) $0.1dt$ (b) $1.0dt$

(c) $3.0dt$

Figure 4.12: The simulation results by performing 100 test simulations on the test set with varying prediction time steps

Since the NODE model learns the continuous dynamics in latent space by solving the differential equations defined by the neural network, it can capture and represent the dynamic behavior of hidden states through piecewise surrogate differential equations. This allows it to predict future states with varying time steps. Figure 4.11 illustrates the simulation results using the NODE model with various prediction time steps, demonstrating a consistent level of accuracy in the simulations. Surprisingly, the time step can be both increased or decreased from the value that was used for training. More completely, this is also verified by performing 100 test simulations on the test set with varying prediction time steps $0.1dt$, $1.0dt$ and $3.0dt$, showing by Figure. All three figures demonstrates enough small prediction errors lower than 0.5%.

This feature of Neural ODEs is an important difference with the direct prediction models, which, by essence, only work with a fixed time step. An interpretation is that

the Neural ODE model replaces the original dynamics, which require complex numerical schemes to be integrated, with local, approximate dynamics which yield the same results when integrated using simpler and faster numerical schemes.

4.3 Conclusion

In this chapter, we extend our methodology for predicting chemical states by applying an autoencoder, a nonlinear dimension reduction model, to identify latent state vectors. The autoencoder is trained in conjunction with latent dynamic state prediction models, and we employ three different models in our experiments for the CH_4 0D auto-ignition case. These three models are based on two distinct dynamic learning approaches: the discrete ODE approach and the continuous NODE approach. Latent state vectors are identified within each cluster, and a clustering algorithm is applied to the original thermochemical states. Our results demonstrate that, in general, this workflow yields accurate prediction results that correspond closely to CVODE simulations. The local latent state dimension numbers in each cluster can be adaptively determined using the reconstruction percentage ratio ϵ, which is applied to filter out minor singular values. Additionally, in these experiments, while all dynamic prediction loss values fall within a similar range, the NODE model and the EulerNET model produce better generalization performance on the test set simulations. In particular, the NODE model is capable to predict the continuous latent dynamics with different size of prediction time steps, which is more flexible and useful for the unsteady problems. Besides, the prediction performance for reduced-order states prediction model is similar to the results of full states prediction model. In general, it is demonstrated that there exists latent dynamic within the detailed chemistry system. The latent states can be learned directly using data-driven reduced-order techniques, combining with different dynamic learning models, while the mathematical form of ODEs for the numerical resolution for latent states is more complicated to be derived.

The future perspective of this study comprises two main aspects. Firstly, the development of a reduced-order model for the total reacting flow system with a high degree of freedom is necessary to complete this methodology. Methods for projecting transport terms onto the reduced-order latent space need to be devised. The latent states dynamic also needs to be learned under the reduced order space containing the information of transport terms. Multiple-dimensional combustion simulations with convection-diffusion terms should be conducted to further advance this methodology. Besides, The clustering algorithm in this study is carried out in the original physical space, as discussed in Chapter 3. It is expected that a new clustering algorithm will be developed in the future for better latent space learning. Furthermore, a detailed analysis of the manifolds generated by the latent states needs to be conducted, offering a more physically intuitive explanation.

Chapter 5

Machine learning for multidimensional simulations

After successful validation for 0D homogeneous reactors, demonstrating encouraging results, it is essential to test the proposed workflow in this thesis to a multidimensional simulation case, integrating operator splitting methods. In this case, the transport terms are included into the governing equations, which increase the complexity of the system. This chapter showcases the outcomes for 2D simulation cases using the workflow proposed in chapters 2 and 3. The testing is performed for an autoigniting premixed combustion configuration, focusing on H_2 and C_2H_4 fuels. The training is based on two different generated datasets, where one is based on data sparsely sampled from direct numerical simulations, and the other is based on a canonical problem, the stochastic mixing reactors model, which mimics the transport terms.

Résumé français

Après une validation réussie pour des réacteurs homogènes 0D, démontrant des résultats encourageants, il est essentiel de tester le flux de travail proposé dans cette thèse dans un cas de simulation multidimensionnelle, intégrant des méthodes de fractionnement d'opérateurs. Dans ce cas, les termes de transport sont inclus dans les équations de la conservation, ce qui augmente la complexité du système. Ce chapitre présente les résultats des cas de simulation 2D à l'aide du prototype proposé dans les chapitres 2 et 3. Les essais sont effectués pour une configuration de combustion prémélangée à allumage automatique, en se concentrant sur les carburants H_2 et C_2H_4. L'apprentissage est basé sur deux ensembles de données générés différents, où l'un est basé sur des données creuses échantillonnées à partir de simulations numériques directes, et l'autre est basé sur un problème canonique, le modèle stochastique des réacteurs à mélange, qui imite les termes de transport.

5.1 Physical problem description

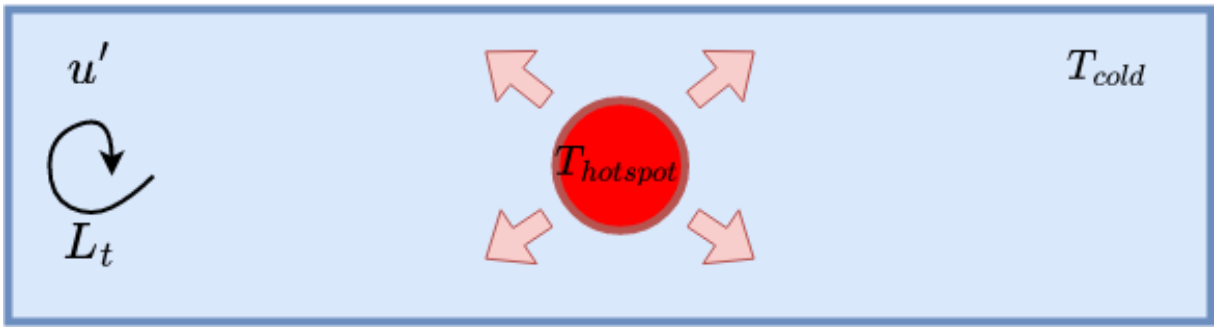


Figure 5.1: Illustration of the 2D hotspot simulation case with a prescribed turbulent spectrum

The objective is to conduct Direction Numerical Simulation (DNS) of hotspot auto-ignition, considering both fast ignition and premixed flame propagation. The rectangular 2D domain, as depicted in Figure 5.1, contains fuel/air mixture under atmospheric conditions. A hotspot circle with burned gas under high temperature is initialized in the center of simulation field. The radius of the circle corresponds to $3\delta_l^0$, where δ_l^0 represents the flame thickness for each fuel case under atmospheric conditions.

The ignition process takes place in a Homogeneous Isotropic Turbulent (HIT) field, with given turbulent integral length scale L_t and turbulent fluctuation intensity u'. To obtain the initial turbulent field, the Inverse Fast Fourier Transformation algorithm (IFFT) is employed, utilizing the analytical Passot-Pouquet turbulent spectrum [121]. Before ignition occurs, the simulation runs for a single time step of $\frac{L_t}{u'}$ to achieve a physically converged turbulent field. The boundary conditions in the resolution region are set to outlets. Moreover, a unit Lewis number is used to model diffusion processes for all chemical species.

An example of one of the studied cases, a DNS simulation of a H_2 hotspot under the turbulent field using CVODE direct integration, is presented in Figure 5.2. In this simulation, the hotspot ignites at an ignition delay of $\tau_{ign} = 5.2 \times 10^{-5}$ s, and the temperature increases up to $1900K$. Subsequently, the flame continues to spread and propagate as a thin front. Notably, this simulation case exhibits two distinct combustion states: ignition and flame propagation. As a result, it represents a more complex and comprehensive scenario compared to 0D reactor simulations studied in Chapter 3. To validate the learning workflow thoroughly, two different fuels with varying complexity, H_2 and C_2H_4, are studied in this work. Table 5.1 provides the initial conditions for each of these cases. The simulation time step is set to a fixed value of $dt_{cfd} = 5 \times 10^{-7}$ s for both the H_2 and C_2H_4 cases. This time step is small enough to adequately capture the ignition and flame propagation processes. However, in the case of C_2H_4, due to its significantly stiffer

chemistry, it becomes challenging to capture the start of fast ignition of the hot kernel using a fixed resolution time step. Since the currrent learning workflow is for fixed time steps, as a compromise to the research objective, the simulation starts later, after the hot kernel ignition has begun, and continues to propagate from that point onward.

Mechanism	$T_{hotspot}$	T_{cold}	ϕ	L_t	u'
H_2 [113]	$1200K$	$300K$	0.4	$6.3 \times 10^{-4}m$	$7.78m.s^{-1}$
C_2H_4 [114]	$1700K$	$300K$	1.0	$6.3 \times 10^{-4}m$	$7.78m.s^{-1}$

Table 5.1: Summary of initial conditions for two different studied fuel cases in 2D simulations

The thermochemical states resolution is coupled with convection and diffusion terms using first order operator-splitting method as presented in 2.1.2 and recalled here:

$$\frac{d\phi}{dt} = R(\phi) + C(\phi, t) + D(\phi, t) \tag{5.1.1}$$

The chemical source terms, denoted as $R(\phi)$, will be predicted by the machine learning workflow or solved directly using the CVODE solver. For the transport equations, second-order spatial and temporal numerical schemes are employed. The CONVERGE CFD solver is used in this simulation work.

Regarding domain discretization, a uniform mesh with a size of $\Delta_x = 2.5 \times 10^{-5}$ m is used to adequately resolve both the flame front and the large-scale turbulent structure. The mesh contains 12.5 points in the flame front and 25 points in the large-scale turbulent structures.

5.2 Training database generation

5.2.1 Sparse time step sampling from DNS simulations

As in 0-D combustion learning cases, the data samples are generated directly from direct numerical simulations. The most simple and direct strategy for 2-D combustion case is also to generate the data directly from DNS simulations. Generating datasets from each resolution time step results in an enormous number of samples. Therefore, it is not necessary to generate all the simulation data. Sparse sampling can be applied to reduce the total dataset size. Samples within several time steps often exhibit similar chemical states. Once the state space are covered, the a posteriori simulation may be successfully run with the generalization performance of learning models. Figure 5.3 illustrates the strategy for sparse snapshot sampling from the 2D time step resolution trajectory after several resolution time steps. The time step Δt_c for snapshots can be integer multiples of the resolution time step dt_{cfd}.

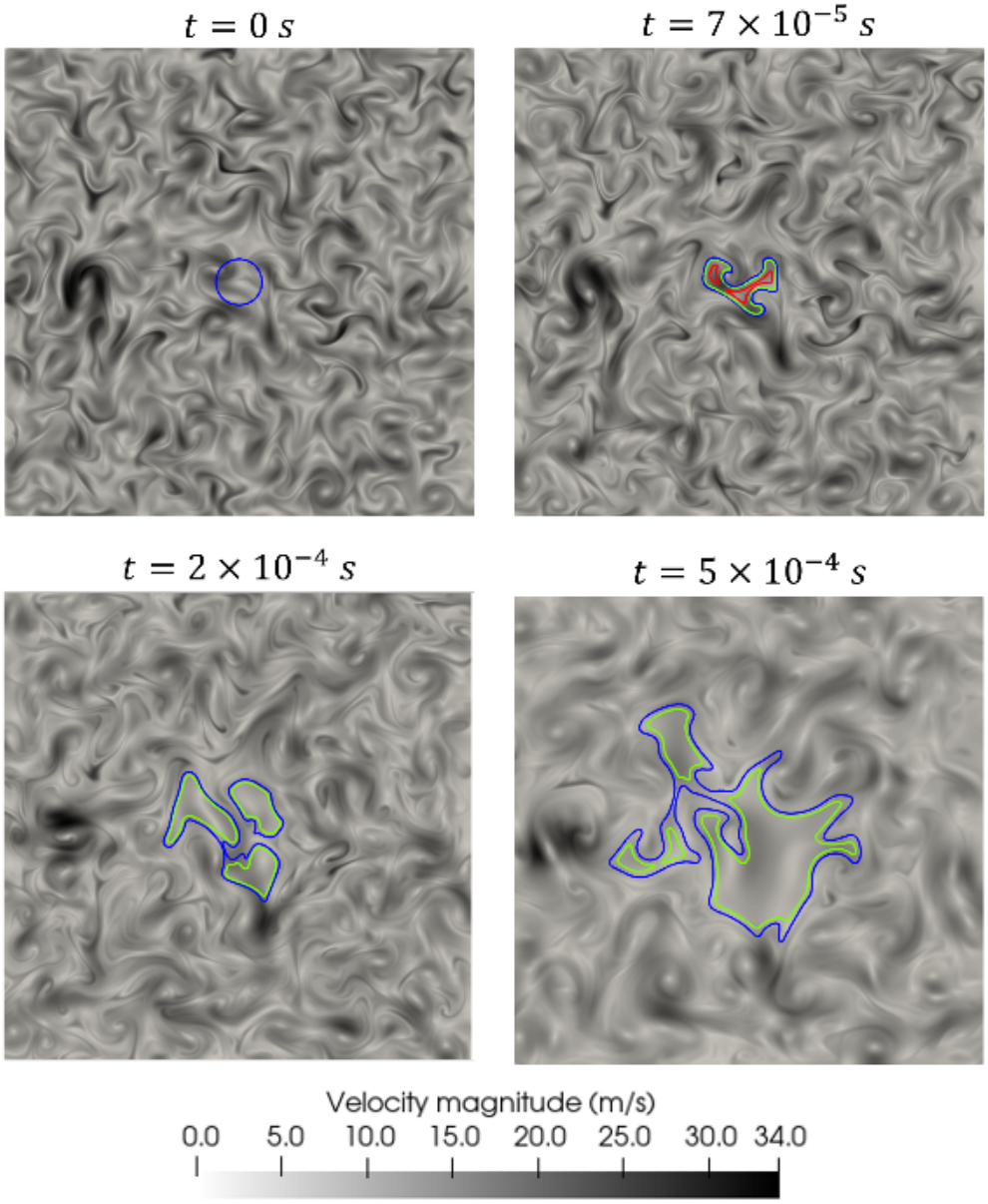
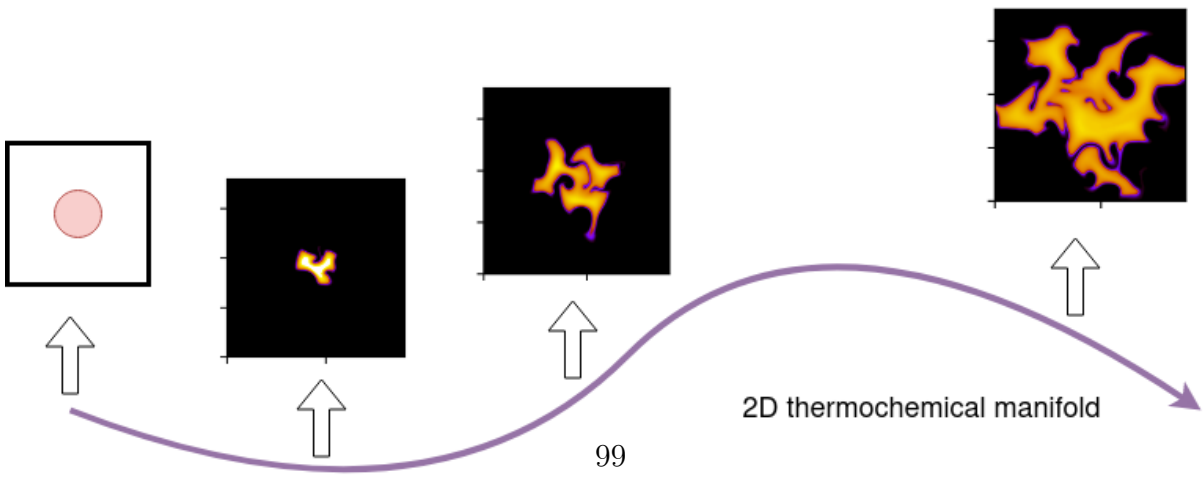


Figure 5.2: A 2D hotspot DNS simulation case using CVODE solver for H_2 case, the lines denote the iso-contours of temperature(Green:$T = 1000K$, Blue:$T = 1400K$, Red:$T = 1800K$)[11]



5.2.2 Stochastic micro-mixing reactors model (SMR)

The core objective of this study is to accelerate chemistry resolutions. However, generating the dataset from direct numerical simulation still involves expensive numerical integration of chemical states. It is expected to avoid the utilization of expensive resolutions. An alternative choice is to construct simplified abstract models based on canonical problems, which includes the target chemical states we want to simulate. Indeed, having a more informative and generalized dataset that includes a broader range of different combustion states is essential (for instance: 0-D and 1-D combustion configurations). Such a dataset can be obtained from simplified physical models, eliminating the need to run expensive DNS simulations. As indicated in literature review of this chapter, previous works have presented different approaches to generate datasets for machine learning model training in the context of multidimensional combustion simulations. These strategies vary depending on the specific objectives and complexities of the combustion simulation problem. In this work, the stochastic micro-mixing reactor model is applied as a simplified canocial problem[30, 3, 70, 60].

To construct the stochastic mixing reactors system, multiple time-evolving reactors are defined as stochastic particles, encompassing the pre-defined space of compositions. Each particle, denoted as p, is characterized by the mass fraction of chemical species Y_k^p, and the sensible enthalpy h_s^p. The simulation involves a total of N_{tot} particles, which remains constant. This approach allows to model the mixing process with discrete particles, representing different compositions, while maintaining a fixed number of particles throughout the simulation. The initial states of the particles are carefully selected to be representative of the initial and boundary conditions of the system to be simulated. This model was first proposed for the automatic reduction and optimization of chemical mechanisms, covering the chemical response along representative evolution of chemistry from fresh to burnt gases [30]. As for machine learning for chemistry computations, Wan et al. [3] constructed the stochastic mixing model for a non-premixed oxygen flame, where particles are separated by fuel and oxygen compositions, with an additional sink term to treat the heat loss in the wall region. More recently, this method was applied for large eddy simulation of non-premixed flameless combustion, which occurs in an industrial furnace [60].

In the present work, to mimic the premixed hotspot ignition system, the particles are divided into two groups which represent different states:

- **Cold fresh air region**: It consists of particles with cold temperatures at T_{cold} and air composition, which represent the fresh air region surrounding the hot kernel.
- **Hotspot region**: The remaining particles represent the fuel-filled hotspot region. These particles are initialized at a higher temperature $T_{hotspot}$ and the composition

of the fuel mixture. This region represents the actual hotspot where combustion is initiated.

By dividing the particles into these two distinct groups, the stochastic mixing reactors system effectively simulates the conditions around the hot kernel and captures the ignition process in a realistic manner. The governing equations of stochastic reactors can be expressed as:

$$\begin{aligned} \frac{dY_k^p}{dt} &= \dot{\omega}_k^p + \mathrm{MIX}_k^p(\tau_T) \qquad \forall k \in \{1, ..., N_s\} \\ \frac{dh_s^p}{dt} &= \dot{\omega}_{h_s}^p + \mathrm{MIX}_{h_s}^p(\tau_T) \end{aligned} \tag{5.2.1}$$

In this system, $\dot{\omega}_k^p$ and $\dot{\omega}_{h_s}^p$ represent the source terms for mass fractions and specific enthalpy, respectively, within the governing system.MIX_k^p and $\mathrm{MIX}_{h_s}^p$ represent the stochastic micro-mixing closure terms for diffusive terms within the system. τ_T denotes the characteristic mixing time scale of turbulent micro-mixing process. The mixing terms in 5.2.1 represent the micro-mixing process of stochastic particles. To approximate these terms, a mixing model must be selected. In this research, there are two mixing approximation models tested, the pair-wise modified Curl model [122] and the Euclidean-Minimum-Spanning-Tree (EMST) model [123].

Pair-wise modified Curl model: The fundamental concept of pair-wise modified Curl model is to randomly select particle pairs (p, q) at each time step. To update the states of mixing particles, they are computed under following rules:

$$\begin{aligned} \psi^p &= \psi_0^p + \frac{1}{2}\xi(\psi_0^q - \psi_0^p) \\ \psi^q &= \psi_0^q + \frac{1}{2}\xi(\psi_0^p - \psi_0^q) \end{aligned} \tag{5.2.2}$$

where $\psi \in \{h_s, Y_k\}$. $\xi \in [0, 1]$ is a random factor, which is used to vary the mixing force for each particle pair. It introduces variability into the mixing process, reflecting the random interactions and fluctuations between the particles. Molecular differential diffusion effects are not involved in this approach. The number of particles selected at each time step, denoted as N_p, is determined by the mixing time scale τ_m. It is calculated as $N_p = \frac{N_{\mathrm{tot}} \cdot dt}{\tau_m}$, where N_{tot} represents the total number of particles in the system, and dt is the time step size of resolution.

EMST model: The EMST mixing model is a particle-interaction model which is derived to overcome the deficiency of simple mixing models. The change in particle composition is determined by particle interactions along the edges of a Euclidean minimum spanning tree constructed in composition space. The mixing is continuous in the mixture fraction space. Such continuity is not accounted for in the modified Curl approach,

making the EMST approach more consistent with respect to the real physical process. This model is recently used in the scenario of machine learning chemistry computation for non-premixed combustion[3, 70, 60]

The stochastic micro-mixing reactors system is implemented and solved using an in-house code [124] which is developed at IFPEN.

Simulation of reactors

In the stochastic mixing reactor system, the hot particles will first auto-ignite, reaching high temperatures. Subsequently, they mix with the unburned particles, leading them to ignite, resembling the convection-diffusion phenomena in real combustion systems. The mixing time is set to $\tau_m = 4 \times 10^{-4}$s, which is in the order of magnitude of both the flame time and the turbulent mixing time. As noted in [11], this value guarantees that the targeted state space is well contained in the training database. The simulation of stochastic reactors is run for $10\tau_m$, which is $4 \times 10^{-3} s$, as used in [11]. After the simulation of the system, the particles with temperature lower than $600K$ will be removed, since no chemical reactions occur within this region. Moreover, the threshold is fixed by 10^{-10} for H_2 case and 10^{-12} for C_2H_4 case to clip the extremely small values which have no physical meanings. As mentioned in chapter 2, there are still no standard criteria to set the threshold in all cases.

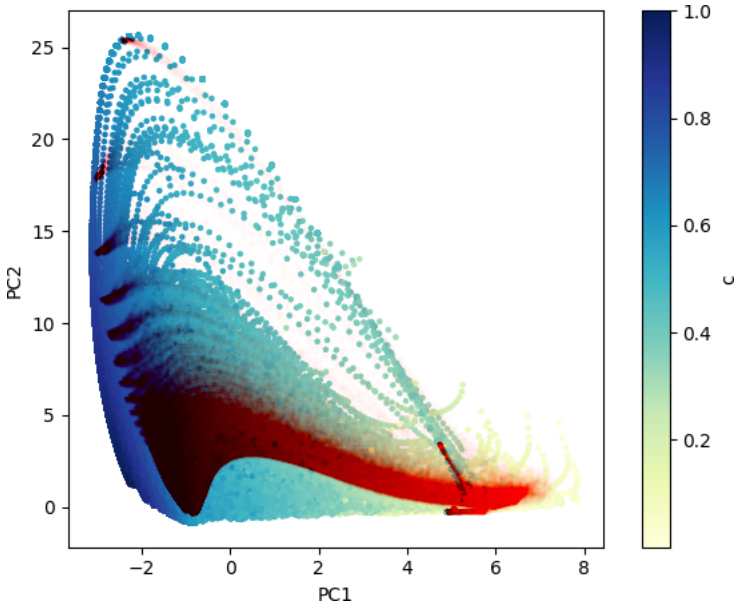


Figure 5.4: The comparison between the data points from real simulation and SMR system for H_2 case. The red color represent the points from real simulations, and the colorbar from blue to yellow represent the data from SMR simulations, the variation of colors denotes the evolution of progress variable of temperature c.

The comparison between the data points from the DNS simulation and SMR system

for H_2 case are shown in Figure 5.4 to demonstrate the adequacy of the workflow to generate the dataset. It is shown that the data from real simulation are located in the distribution of data generated from SMR approach. The points are projected on the reduced 2 dimensions PC_1 and PC_2 using PCA algorithm, based on the dataset of SMR particles. The red color represent the points from real simulations, and the colorbar from blue to yellow represent the data from SMR simulations, the variation of colors denotes the evolution of progress variable based on temperature $c = \frac{T-T_0}{T_f-T_0}$.

Model inference

Similar to the 0D simulation case, the inference for 2D case is implemented in the Converge solver using an in-house code-NNICE [120]. This code does not depend on third-party deep learning inference libraries and is based on the C++ programming language.

5.3 Results for H_2 case

5.3.1 Data distribution

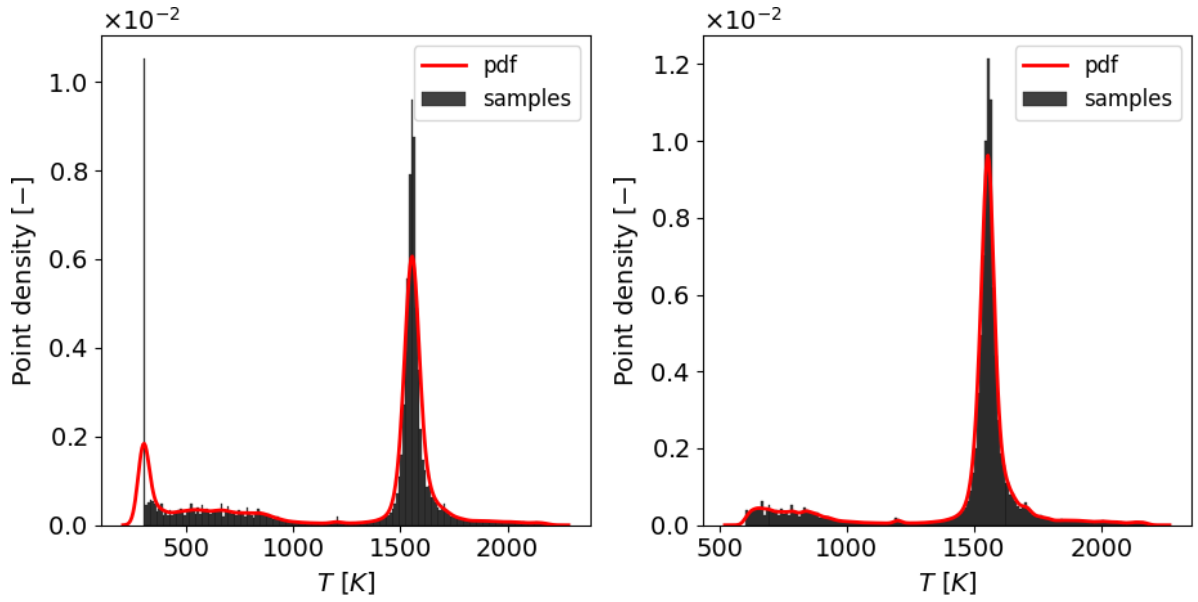
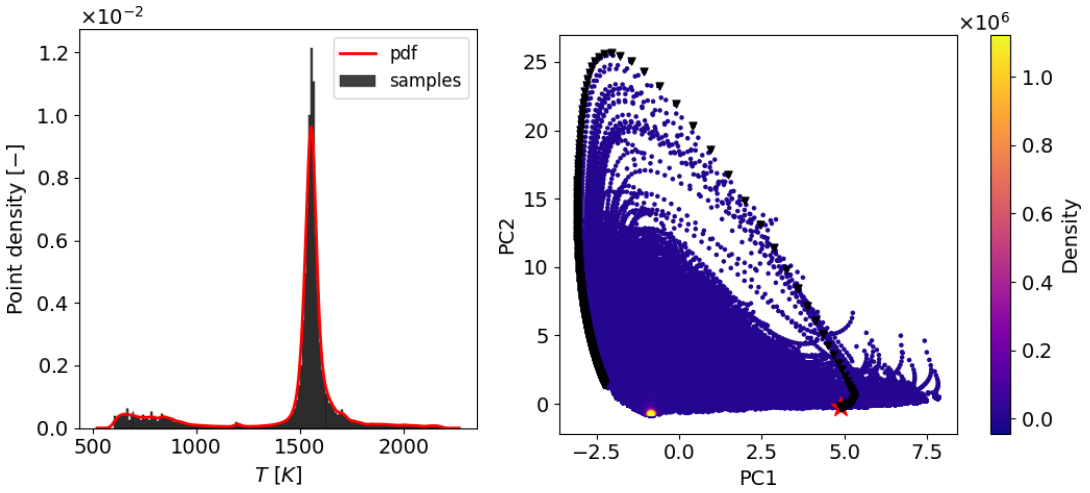


Figure 5.5: Temperature distributions of data samples, where: (a) the values of temperature smaller than $600K$ are not removed, with a peak of low temperature region around $300K$. (b) the values of temperature smaller than $600K$ are removed.



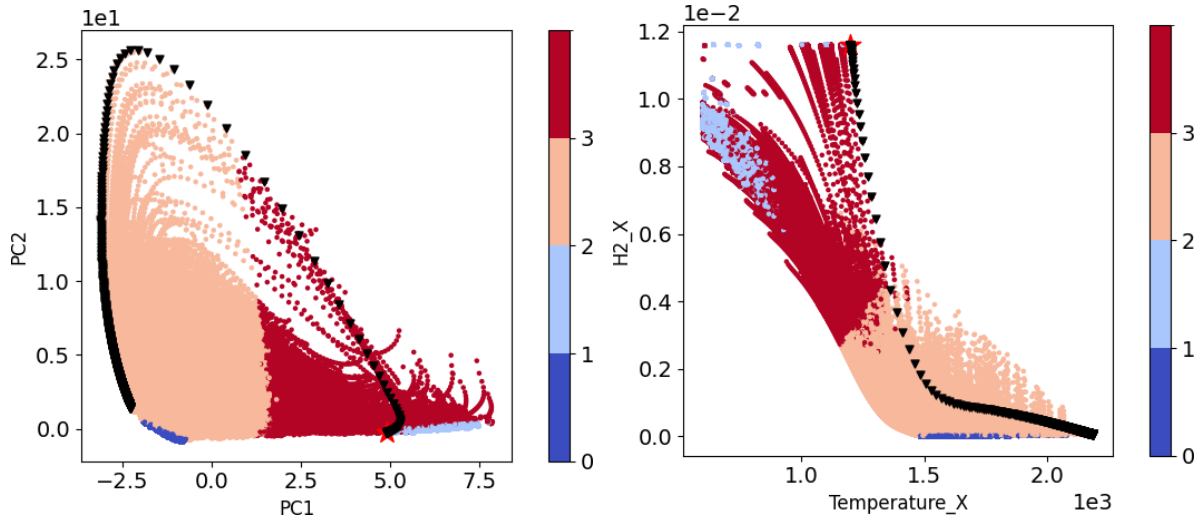
(a) (b)

Figure 5.6: (a): Temperature distributions of data samples. The red line represents the probability density function. (b) The scatter plots of data samples in projected PCA space. The colorbar denotes the density of samples. The black triangle with a red star initial point represents the samples from 0D simulation, which is also recovered by the total dataset

The training process with the H_2 combustion case is directly carried out based on the dataset with stochastic mixing reactor models. As presented in [11], this dataset is successfully used for H_2 case, and it is used to firstly test the learning workflow in this work. After conducting extensive simulations of stochastic reactors and filtering out samples with temperatures below 600K, we have generated a comprehensive dataset for training. The suppression of temperatures below 600K eliminates the distribution peak around 300K, as shown in Figure 5.5. The temperature distribution of these data samples is illustrated in Figure 5.6(a), with the probability density function overlaid on the figure. As for this dataset, most of samples are located in high temperature region. Figure 5.6(b) displays scatter plots of the data samples projected into the PCA space. The 0D fast ignition simulation samples are also included into the dataset. These samples cover a range of combustion states, spanning from low to high temperatures within the system. However, it is notable that there is a significant accumulation of data points in the high-temperature zone, representing the premixed burnt gas state. Conversely, there are fewer data points for the reacting states.

Cluster index	0	1	2	3
Samples number	1.58×10^6	1.64×10^5	2.30×10^5	8.34×10^5

Table 5.2: data samples number in each cluster index for H_2 case

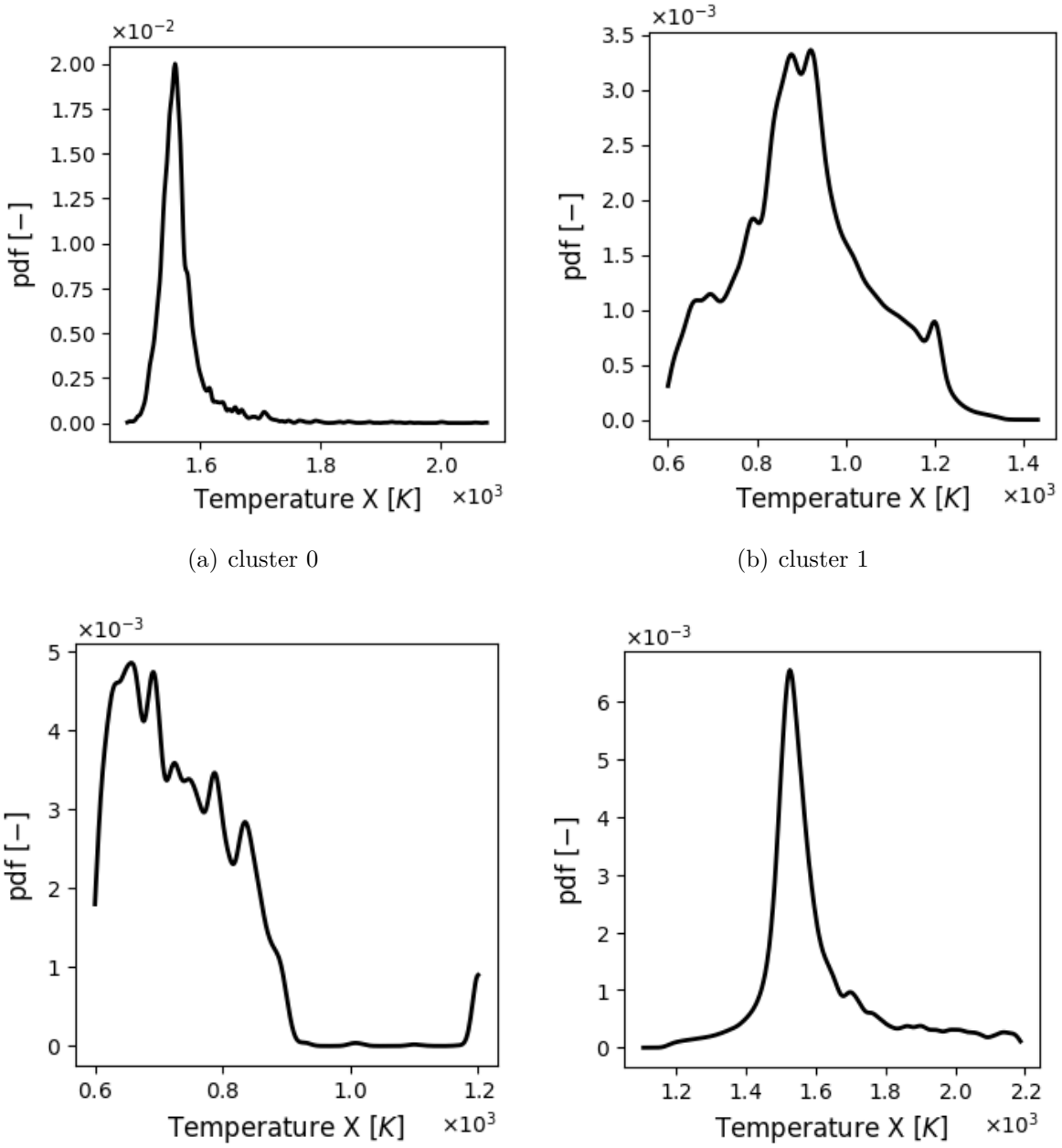


(a) (b)

Figure 5.7: (a): Clustering for the dataset in projected PCA space. The black triangle with a red star initial point represents the 0D ignition simulation trajectory which does not include transport phenomena (b) Clustering for the dataset in $T - H_2$ phase plan. The black triangle with a red star initial point represents the initial condition point

5.3.2 Data clustering and pre-processing

As discussed in chapter 2, KMeans clustering is employed for data clustering in the logarithmic space. The logarithmic transformation helps handle extremely small values. A fixed cluster number of 4 is chosen, since using a larger number of clusters leads to some clusters with an insufficient number of samples (less than 1000 samples) for training. Table 5.2 provides the number of data samples in each subdomain, ensuring that each subdomain contains an adequate number of samples for training purposes. Figure 5.7 displays the results of clustering for the H_2 dataset, with clusters denoted by different colors. It is evident that the dataset is segmented into subdomains corresponding to different combustion behaviors, including the initial pre-ignition zone, the burned-up zone and the flame propagation states. Figure 5.8 illustrates the temperature distribution of training data for each cluster. Remarkably, KMeans partitions the global system and generate simplified subdomains, similar to its performance in 0D cases of H_2 combustion. For each cluster, the distribution of data samples is more homogeneous. Figure 5.9 provides an example of the training data distribution for H_2 before and after data pre-processing for cluster 3. The log transformation and data standardization are necessary steps to achieve a more balanced distribution of processed data before model training.



(a) cluster 0

(b) cluster 1

(c) cluster 2

(d) cluster 3

Figure 5.8: Distributions of temperature in the training data for each cluster

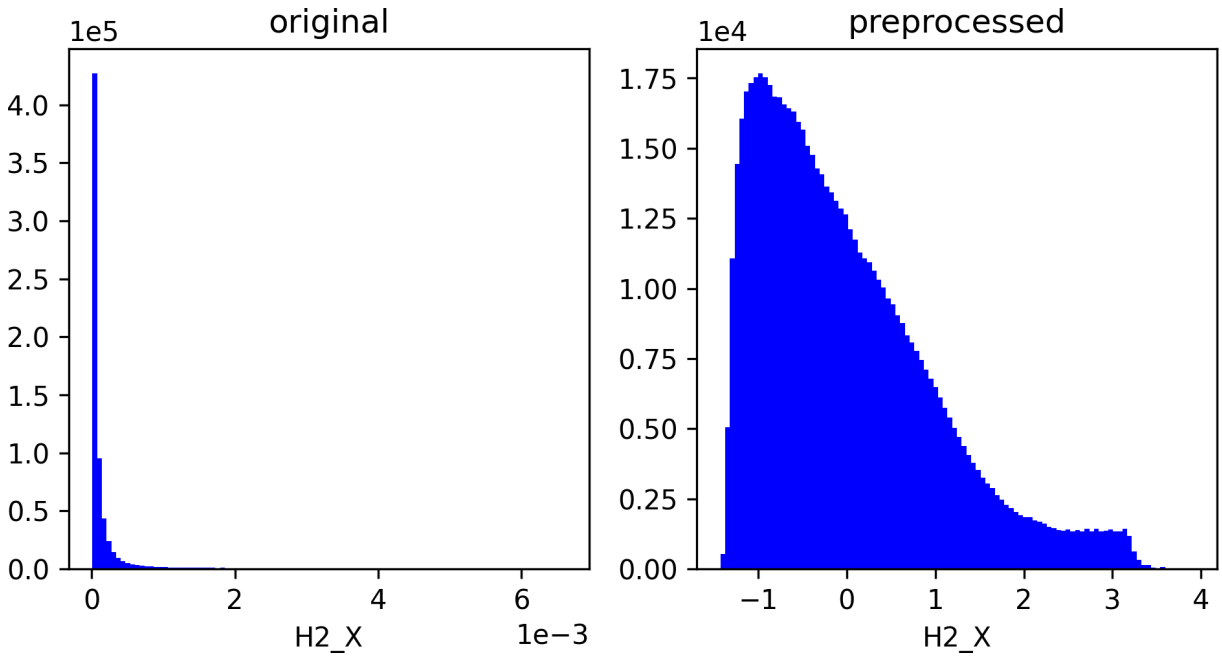


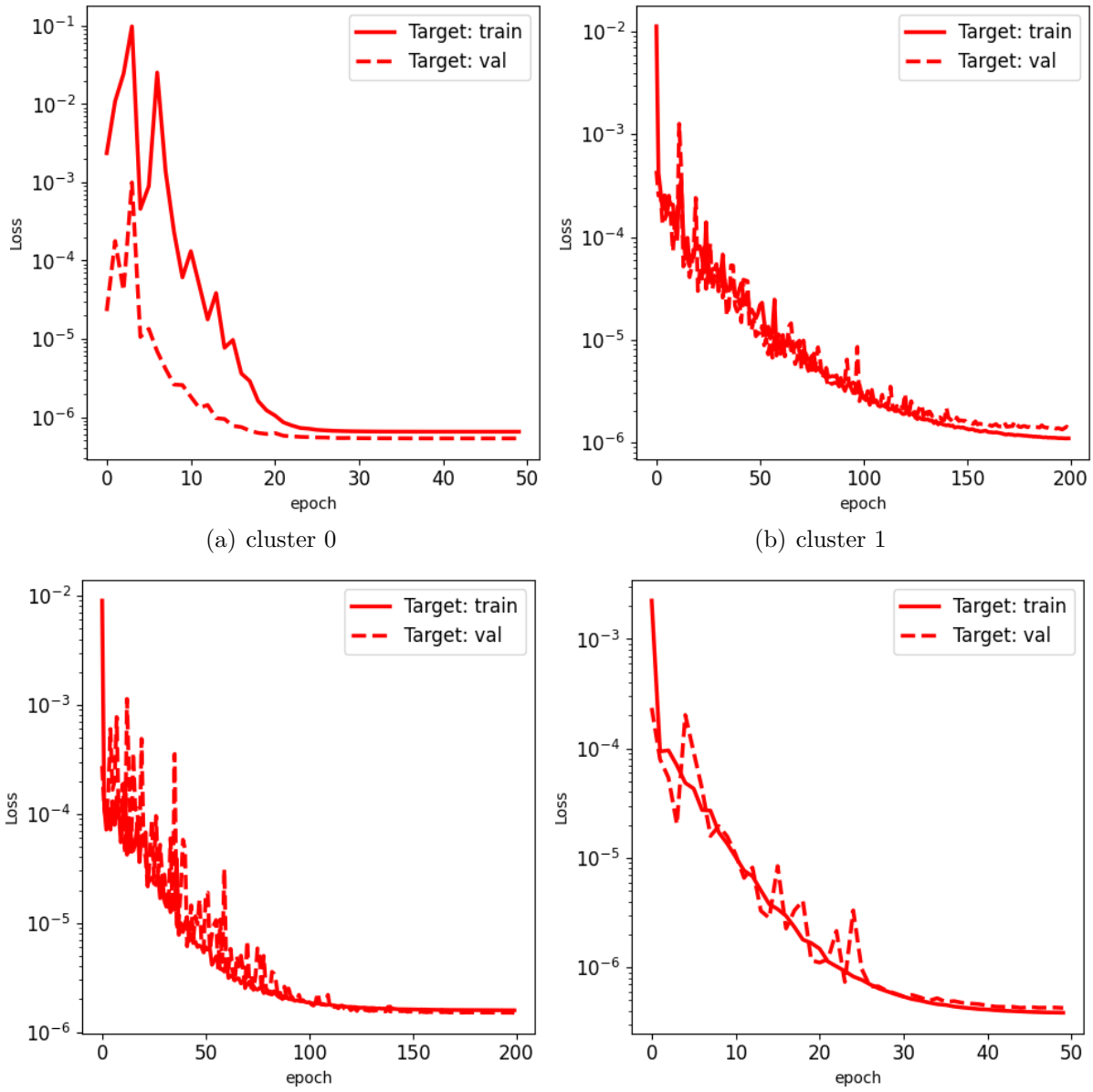
Figure 5.9: The distribution of training data for H_2 before and after data pre-processing with logarithmic transformation for cluster 3

5.3.3 Model training

Model name	Block number	Total hidden layer number	Neurons number	Model size
DNN_40	1	3	40	$52.1kB$
DNN_90	1	3	90	$164.9kB$
DNN_120	1	3	120	$269.8kB$

Table 5.3: data samples number in each cluster index

In the neural network architecture selected for the present case, temperature and N_2 are not directly predicted as outputs. Temperature is computed using the conservation of enthalpy, and the N_2 values in the H_2 case remains constant during the chemical reactions as NOx chemistry is not included in the chosen mechanism. We select three different ANN architectures to evaluate the sensibility of predictions to the neural network size. The table in Figure 5.3 provides a summary of the selected networks. The network architecture and the total number of layers remain unchanged, with only variations in the number of neurons per layer. The models are designed to have a low memory footprint (less than 1MB) and to keep the model parameters number small, thus achieving better code acceleration. The activation function used is the "swish" function, as in the 0D cases. The loss function employed is the standard mean squared error (MSE) function, and optimization is carried out using the Adam optimization algorithm with an initial learning rate set to 0.01.



(a) cluster 0

(b) cluster 1

(c) cluster 2

(d) cluster 3

Figure 5.10: Training loss functions evolution with the number of epochs for each cluster

Figure 5.10 shows the evolution of the loss functions for each cluster during training for model DNN_90. The loss functions steadily decrease and converge to small, nearly equal values, indicating that the models have been effectively trained for all clusters. Both the training and validation mean squared errors (MSE) values for normalized data are within the same scale, suggesting that there is no significant over-fitting occurring during the training process.

Besides, we employ the mean absolute error in logarithmic transformation space ($logMAE(j)$) for each dimension for the prediction of chemical species, and the total mean values for all dimensions $logMAE$ to evaluate the general performance of the model training before the data normalization. These two kind of metrics are denoted as:

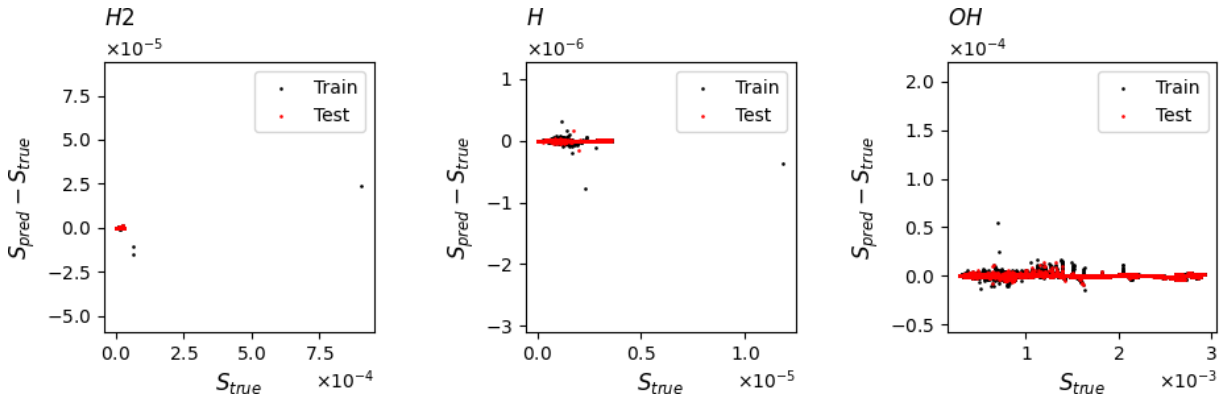
$$\begin{aligned} \mathcal{L}_{loss} &= \frac{1}{K}\sum_{j=1}^{K}(\frac{1}{N}\sum_{i=1}^{N}(\hat{S}_i^j - S_i^j)^2) \\ \mathcal{L}_{logMAE}(j) &= \frac{1}{N}\sum_{i=1}^{N}|log(\hat{Y}_i^j) - log(Y_i^j)| \end{aligned} \tag{5.3.1}$$

Here the S denotes the normalized data before training, and Y is the original chemical species states. Table 5.4 gives loss values and logMAE metrics for each cluster after the training process, for the case DNN_90. The loss function values are around $10^{-6} - 10^{-7}$ for each cluster. The mean logMAE for all chemical species in each subdomain are all in 10^{-4} scale.

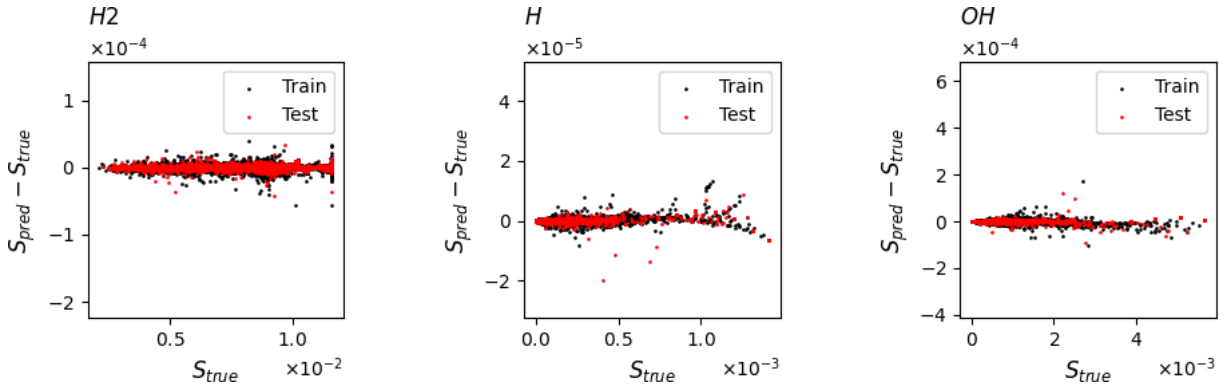
Cluster index	Loss	logMAE(H_2)	logMAE(O_2)	logMAE(H_2O)	logMAE(H)	logMAE(OH)	logMAE
0	4.11×10^{-7}	1.42×10^{-4}	1.40×10^{-7}	2.35×10^{-7}	1.6×10^{-4}	5.34×10^{-5}	9.32×10^{-5}
1	1.55×10^{-6}	1.86×10^{-5}	6.75×10^{-6}	3.97×10^{-4}	1.45×10^{-3}	1.27×10^{-3}	6.73×10^{-4}
2	4.36×10^{-7}	1.72×10^{-4}	3.66×10^{-6}	6.97×10^{-6}	2.00×10^{-4}	8.81×10^{-5}	1.25×10^{-4}
3	1.33×10^{-6}	8.14×10^{-5}	2.38×10^{-5}	2.75×10^{-4}	9.86×10^{-4}	1.19×10^{-3}	6.07×10^{-4}

Table 5.4: Loss values and logMAE metrics for each dimension, for the H_2 case

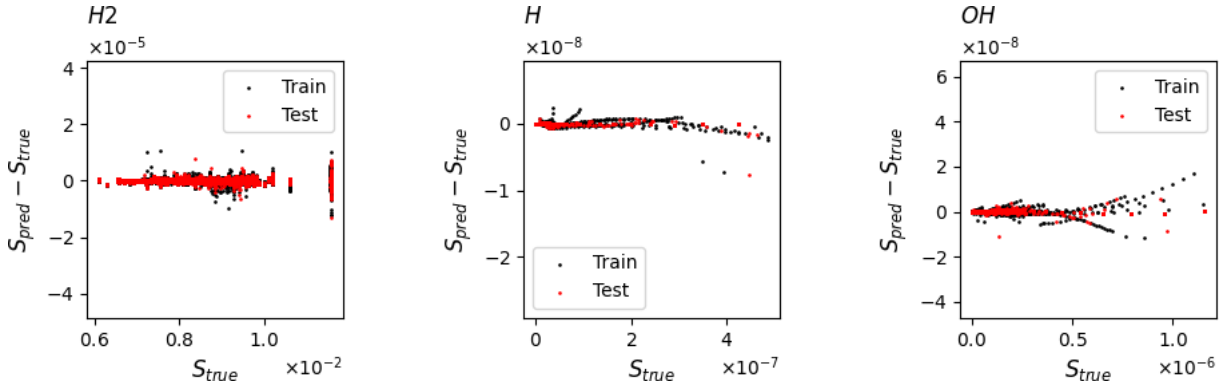
The analysis of the training results is visualized using parity plots, which compare the true values with the predicted values for each cluster's model, focusing on point-to-point predictions for a single time step. Figure 5.11 displays the parity plots between true values and predicted values for H_2, H, and OH, for the model DNN_90 of each cluster. The performance is evaluated using the absolute error, represented as $Y_{pred} - Y_{true}$. The error for the majority of training and test samples is significantly limited around $0.1\% - 1\%$ of ground-truth values. However, there are some scattered samples that lie far from the central distribution zone in cluster 0. This suggests that the clustering algorithm may not have achieved a perfect partitioning of the normalized data, as these outliers in cluster 0 are still present. Fortunately, as demonstrated in the subsequent analysis, these outliers do not pose issues during the actual inference in the CFD solver for the H_2 case. Nevertheless, future research may benefit from the development of more advanced and efficient clustering algorithms to address this problem.



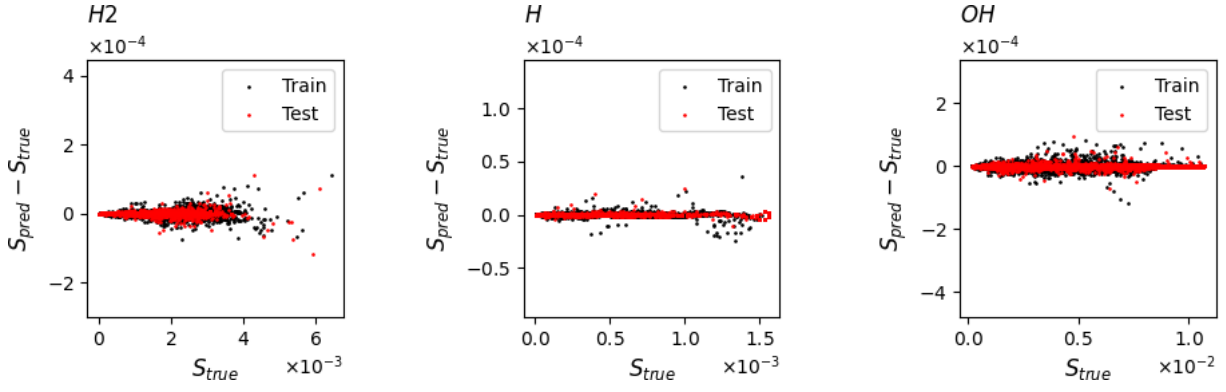
(a) cluster 0



(b) cluster 1



(c) cluster 2



(d) cluster 3

Figure 5.11: The parity plots for real and predicted output values for one time step prediction evaluation.

5.3.4 0D inference performance

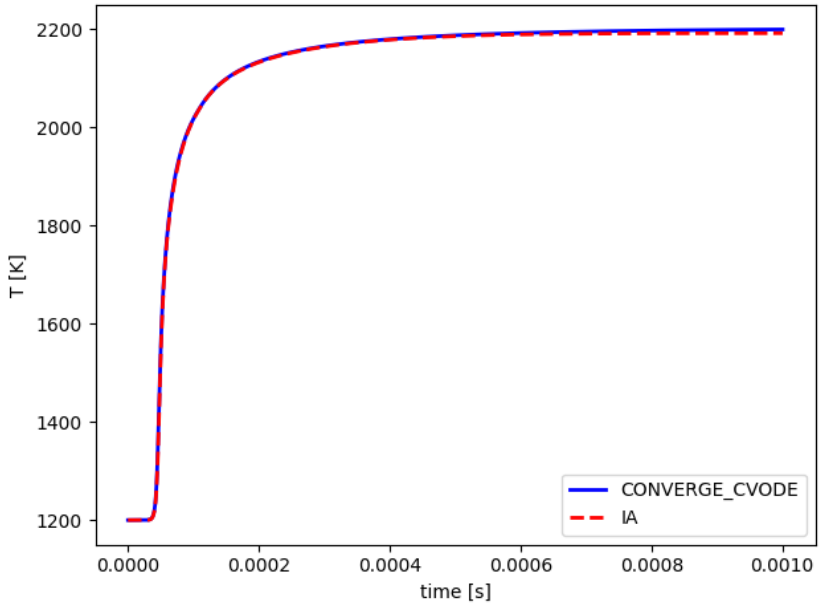


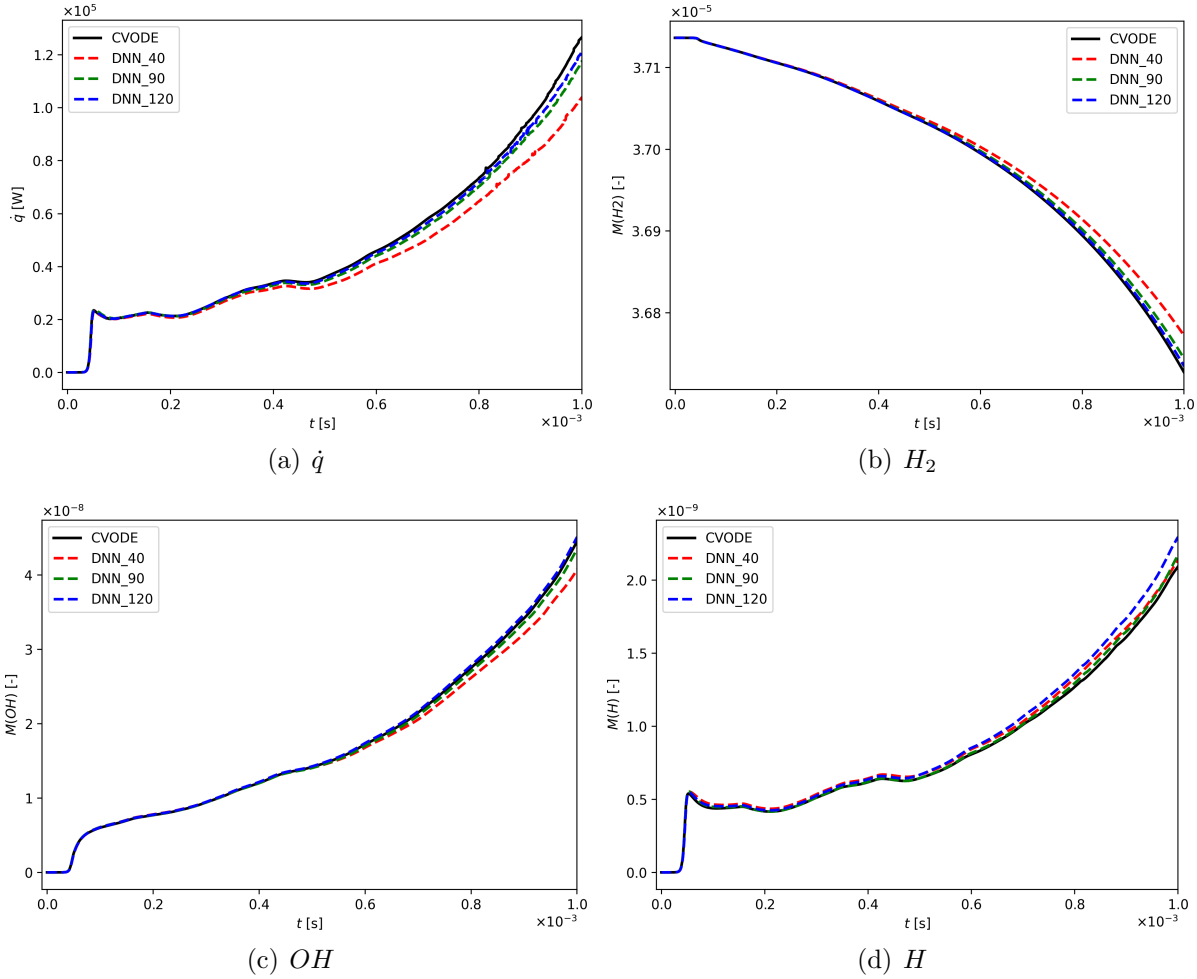
Figure 5.12: The temperature evolution of 0D inference simulation for H_2 case, the simulation is carried out by using DNN_90 model on CONVERGE solver

Figure 5.12 presents the results of the 0D inference simulations for H_2 case where the initial condition is $T = 1200.0K, \phi = 0.4$ to verify that if the model successfully learn the 0D ignition trajectory. It shows the temperature evolution curve until the system reaches the equilibrium state. The inference results, represented by the red dashed line, are compared to the CVODE solution. The inference based on the deep learning model aligns well with the CVODE resolution. This demonstrates that the deep learning model is capable of accurately predicting the system's behavior in 0D simulations. Furthermore, as shown in Figure 5.6(b), the trajectory of the 0D simulation is also included in the training dataset.

5.3.5 2D inference performance

After successfully verifying the performance for 0D simulation, we now evaluate the performance of 2D simulations comparing variables predicted by neural networks and variables numerically integrated by the CONVERGE code. To evaluate the 2D inference, we compute and compare the global quantities in 2D field, such as the heat release rate ($\dot{q}$) and the total mass fractions of field of different chemical species, with the results from CVODE numerical simulations. Figure 5.13 presents the plots of temporal evolution of global quantities. The dashed colored curves represent the results of inference simulations using each of the trained models. Overall, the inference simulation results generated by

the deep learning model align well with the results from the CVODE numerical simulations. Even in regions with rapid changes of heat release rate, around t =0.05ms, there are no significant differences between the CVODE simulation and the model predictions. The model with 90 neurons in each layer performs better than the model with 40 neurons in each layer, while the model with 120 neurons in each layer does not significantly improve the performance. For the subsequent analyses, we will primarily consider the DNN_90 model.



(a) $\dot{q}$ (b) H_2

(c) OH (d) H

Figure 5.13: Plots for 2D simulations of the heat release rate and total mass fractions of field.

Figure 5.14 displays the spatial representation of clusters in the 2D field. In the figure, cluster 2 and cluster 3 are represented in light blue and green colors, respectively, and these regions correspond to the flame front areas. cluster 4 corresponds to the final burnt gas regions with no chemical reactions. Cluster 1 represents the near-equilibrium region with high temperatures and significant heat release following the fast ignition process.

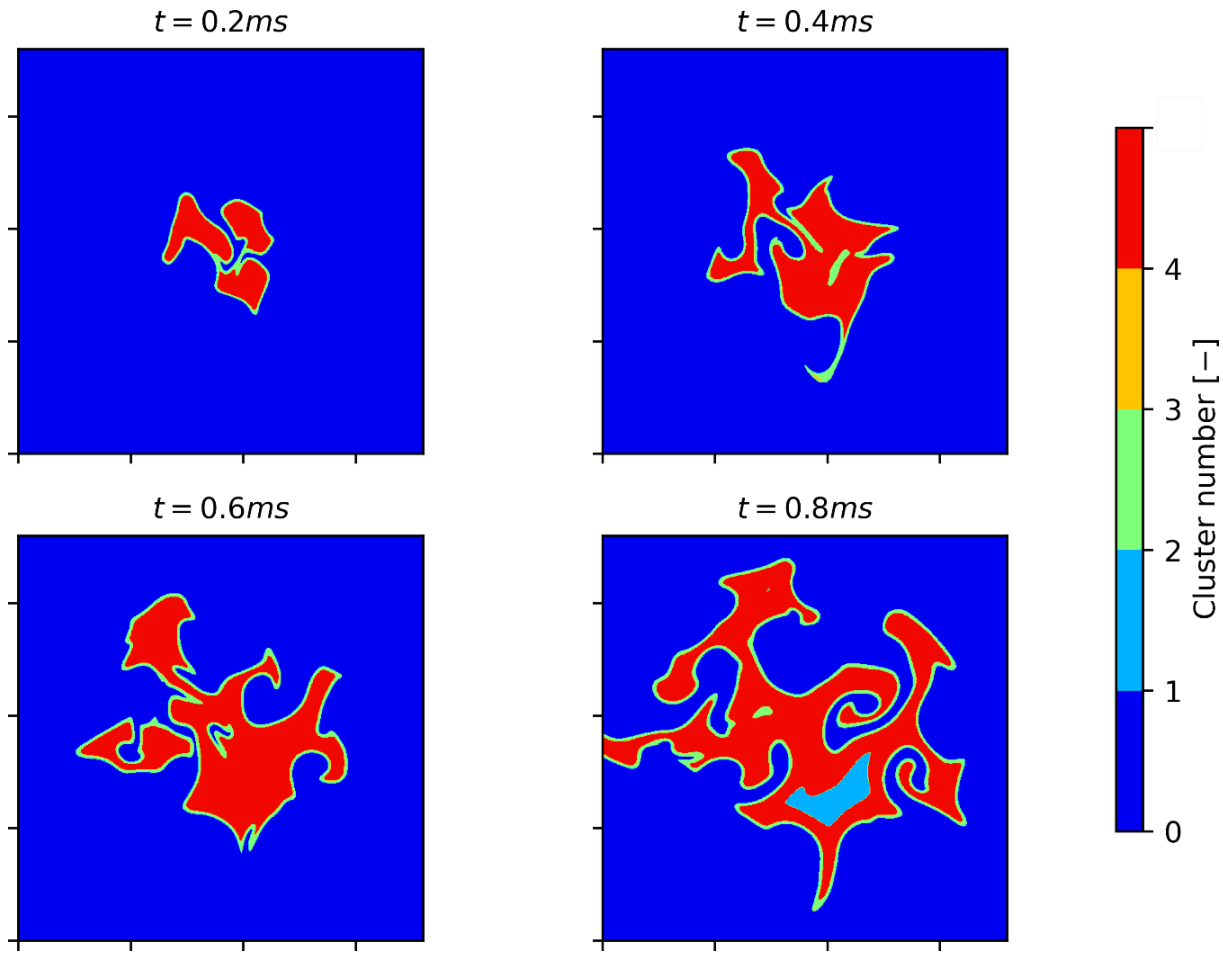


Figure 5.14: Representation of the clusters for different times in the simulation.

Another way to assess predictions is the direct comparison of the simulation fields. However, directly comparing and evaluating errors on a mesh-to-mesh basis can be challenging due to spatial differences in flame propagation between the CVODE simulation and the machine learning-based simulation. The spatial displacement of flame front can cause large instantaneous point-to-point errors. As a result, the direct mesh-to-mesh error may appear large, even though it may not accurately reflect the model's prediction performance.

To properly compare the differences between CVODE simulations and learning-based simulations, a measure based on the probabilistic density function (PDF) distribution is considered. A good prediction result should have a similar PDF distribution in the 2D field. The distribution error can be quantitatively measured using a metric known as the Hellinger distance, which is already used to evaluate the 2D field values in combustion study[125]. Compared to other distribution metric (such as Wasserstein distance and Kullback-Leibler distance), this metric normalizes the metric values between 0 and 1. The formulation to compute the Hellinger distance is as follows:

$$H^2(P(x_j), Q(x_j)) = \frac{1}{2} \int \left(\sqrt{P(x_j)} - \sqrt{Q(x_j)} \right)^2 dx \tag{5.3.2}$$

where $P(x_j)$ and $Q(x_j)$ are distribution of probabilities regarding to the 2D field representation for each dimension. For absolute simulated values in 2D field, the probability distribution $P(x_j)$ is computed as:

$$P(x_j) = \frac{x_j}{\sum x_j} \tag{5.3.3}$$

The $Q(x_j)$ on another simulated 2D field is computed by the same method. If H is equal to 1, it means $P(x_j)$ have a totally different distribution with respect to $Q(x_j)$, and vice versa.

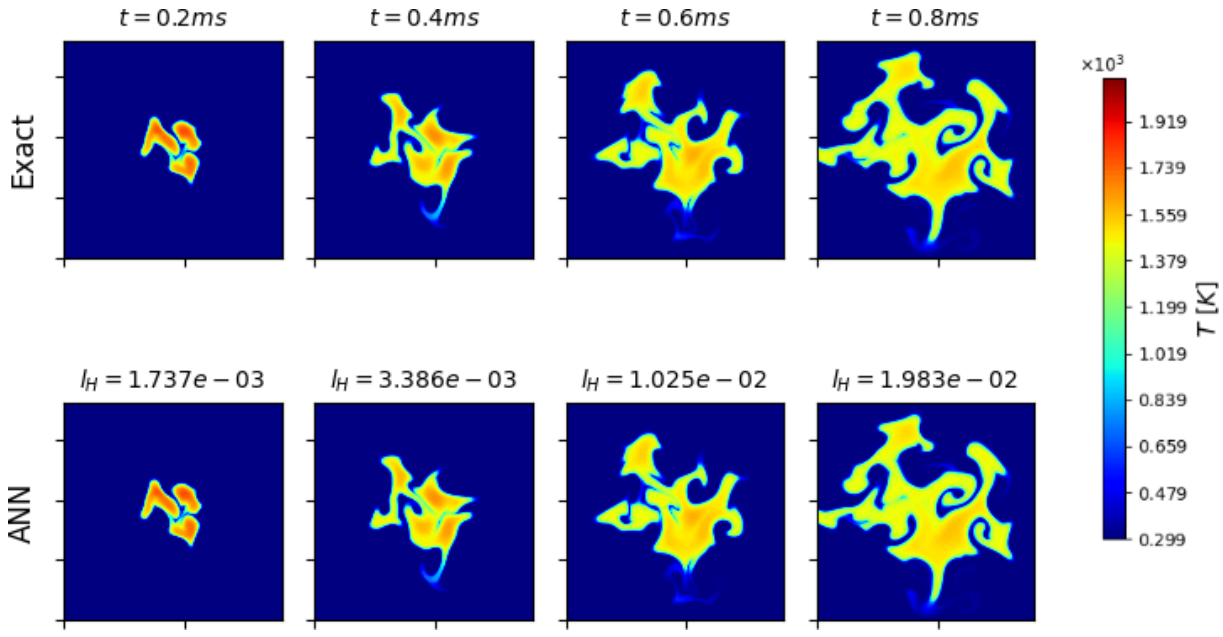


Figure 5.15: The contour plot of temperature field between the deep learning based simulation and CVODE based simulation, the simulation time is up to 10^{-3}s

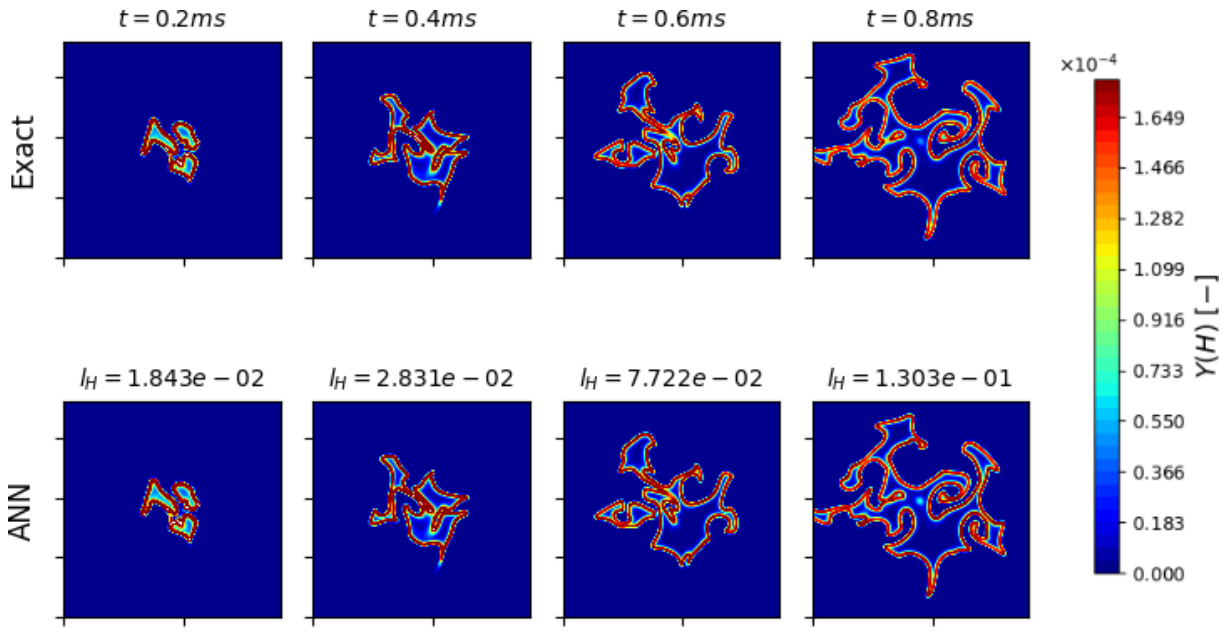


Figure 5.16: The contour plot of H field between the deep learning based simulation and CVODE based simulation, the simulation time is up to 10^{-3}s

Figure 5.15 and Figure 5.16 display 2D contour plots comparing the CVODE simulation with the learning-based simulation. The Hellinger distance values (l_H) are computed and reported on each figure. In the case of temperature 2D contours, H falls within the range of $10^{-3} - 10^{-1}$, which are near zero. This indicates that the two simulations have similar 2D field distributions, demonstrating the good performance of the deep learning model. Similar results are observed for the 2D contour plot of the mass fraction of H, where l_H is on the order of $10^{-2} - 10^{-1}$, again far from one and near zero.

In summary, the analysis of the H_2 simulation results demonstrates that the deep learning workflow works effectively for the H_2 case, with eight chemical species and temperature predicted by neural networks. The inference results globally align well with the CVODE simulations.

5.4 Results for C_2H_4 case

Since H_2 is a relatively simple fuel, this subsection will present the analysis of the C_2H_4 case, which presents more complex challenges, with more stiff reactions and more chemistry species included in the combustion process.We start by using sparse sampling DNS-based dataset to evaluate the learning model. Then, we proceed to cases using SMR-based dataset with two different approaches for transport terms. Some of the data pre-processing strategies and hyperparameters are changed to reach a better performance.

5.4.1 Training model with DNS generated dataset

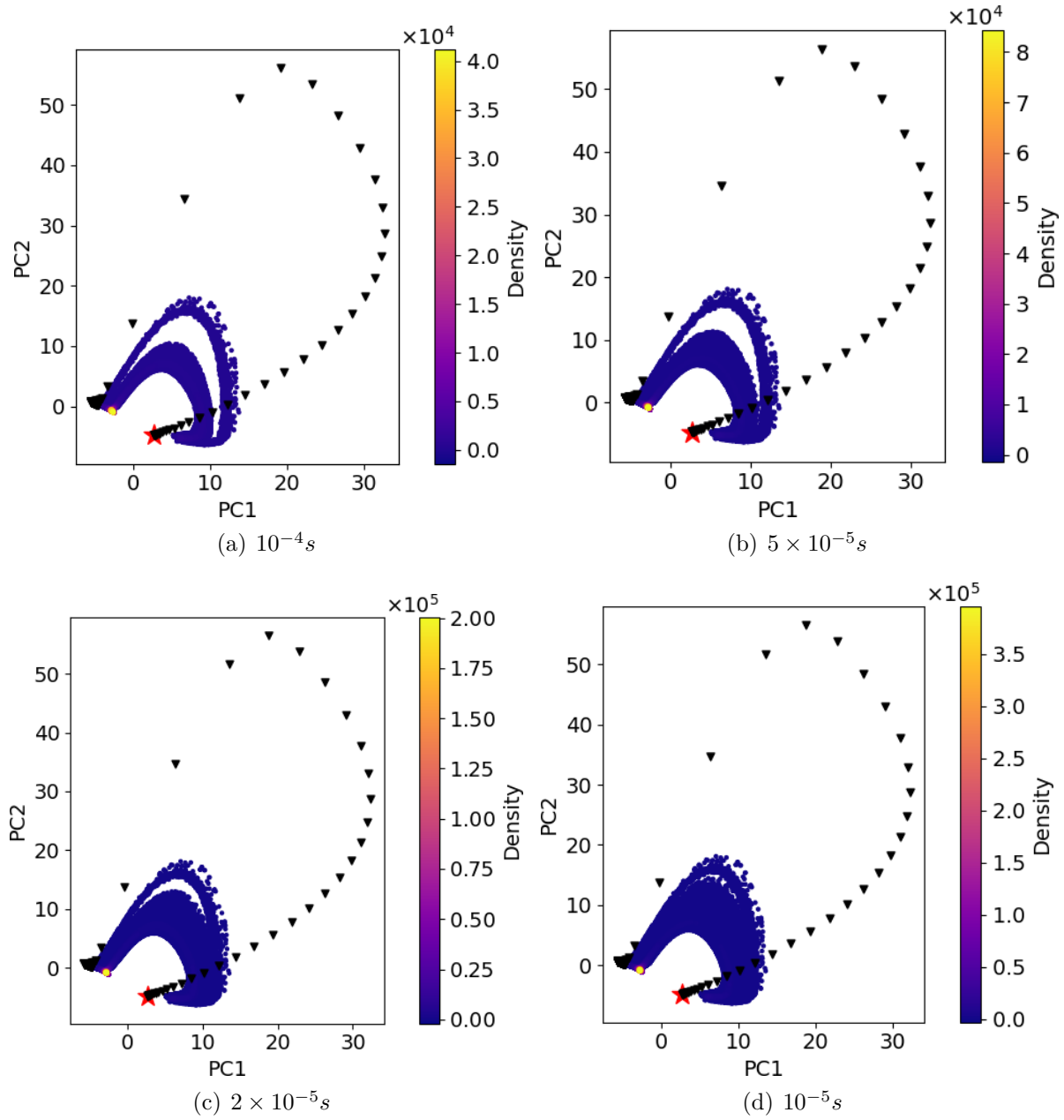
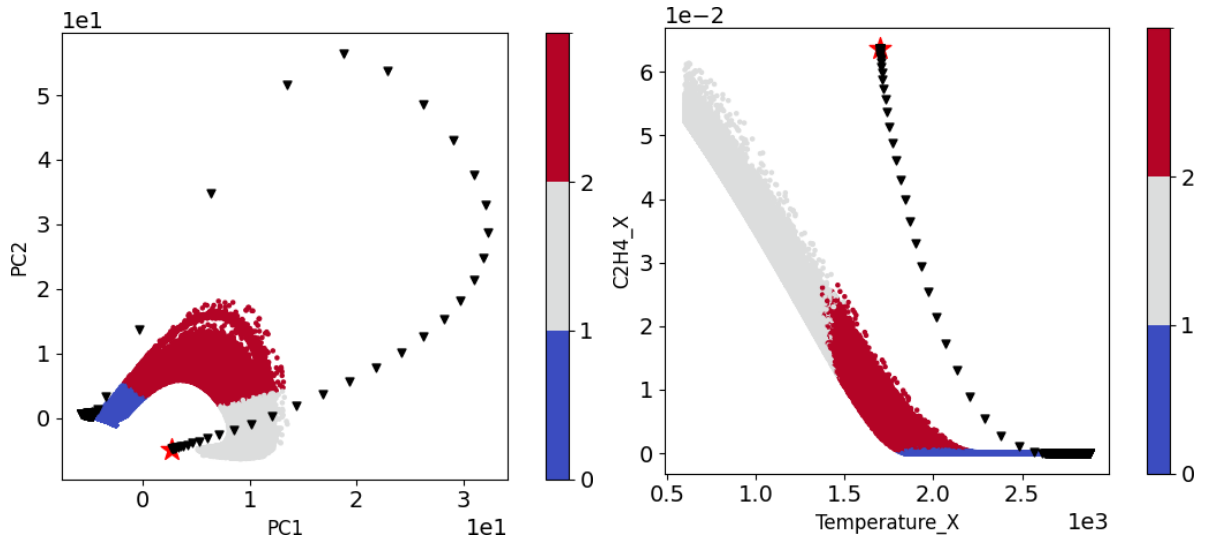


Figure 5.17: The scatter plots of samples generated for different time steps, from $10^{-4}s$ to $10^{-5}s$. The color bar represents the density of samples, and the black triangles with an initial red star point represents the 0D simulation trajectory which is no longer included into the dataset. The dataset are directly sampled from DNS simulation of C_2H_4

Sampling time step

The 2D machine learning based simulation of the C_2H_4 case is firstly carried out using dataset from the high-fidelity DNS simulation. Following the sparse sampling strategy presented previously, Figure 5.17 presents the scatter plots of samples generated with different sampling time steps, ranging from 10^{-4} seconds to 10^{-5} seconds. It can be



(a) (b)

Figure 5.18: Data points distribution in PCA reduced 2D space and in T-C_2H_4 space, where the data is from DNS snapshot samples

observed that when the time step is 10^{-5} seconds, the samples cover the entire manifold, suggesting that this time step is sufficient to create a complete dataset for training.

Data preprocessing

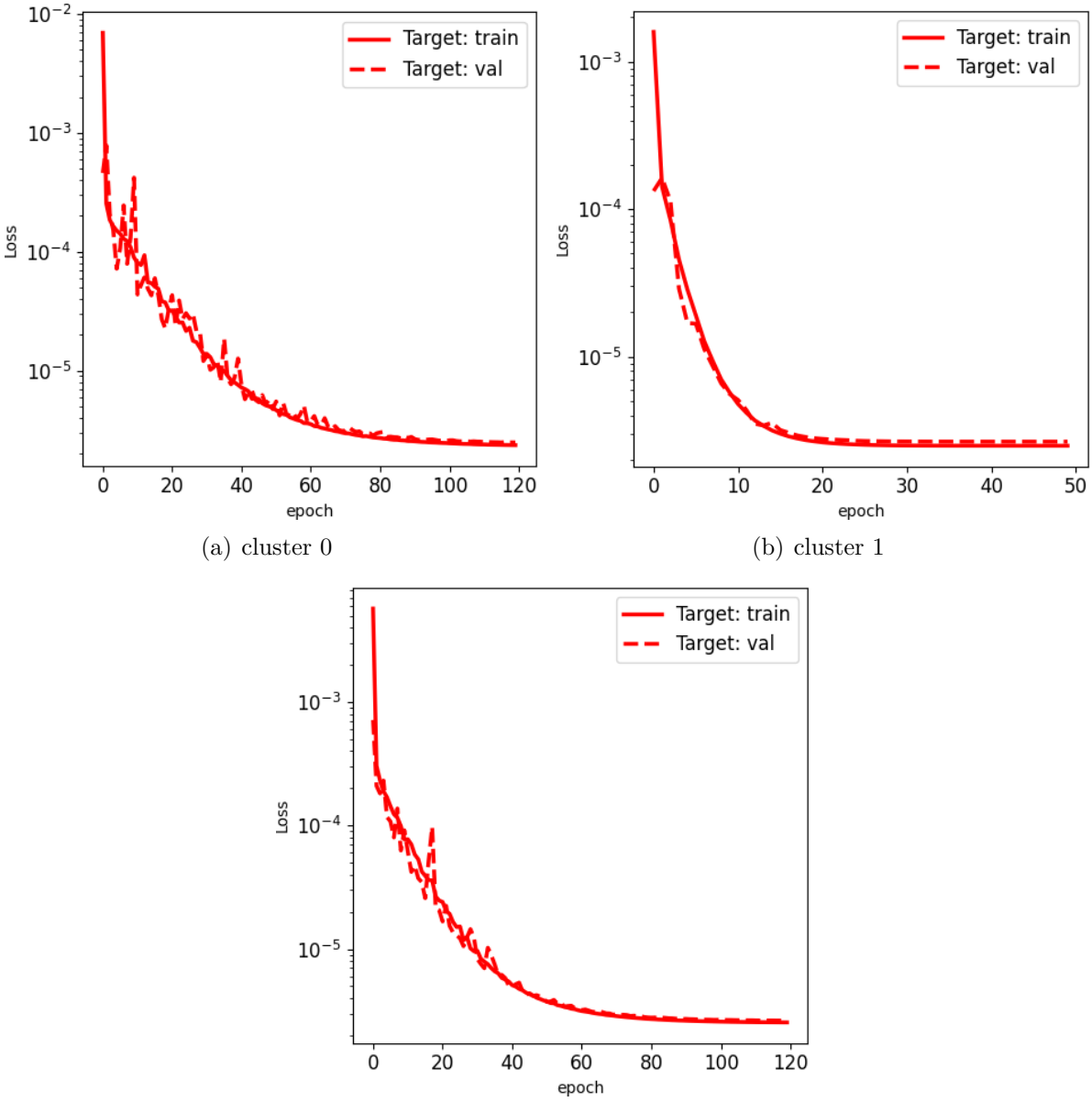
The dataset is partitioned into three clusters, similar to the previous approach. However, we no longer use logarithmic transformation during clustering, as we observed that the log transformation for this case leads to worse generalization performance, causing simulation divergence. A threshold of 10^{-12}, which is the same value than for the 0D case, is applied to clip the extreme small values of the dataset. The dataset is partitioned into three different subdomains using KMeans algorithm, containing samples with different combustion states. As for this dataset, when cluster numbers larger than 3 lead to insufficient data (less than 1000 samples) in several clusters. For deep learning models, it is supposed that a large number of data samples is needed for model training. Figure 5.18 illustrates that the clustering of the DNS data successfully identifies the reaction zones and high-temperature burnt gas zones.

Model hyperparameters

The structure of residual neural networks is similar than for 0D cases, where a backbone layer is the first hidden layer and two residual blocks are added. Each layer contains 150 neurons. The *swish* function no longer led to successful generalization, and the inference simulation using models with *swish* function diverged for C_2H_4 case. Therefore, the activation function was changed to the Rectified Linear Unit ($ReLU$) function, which can produce more robust performance during the inference process. The pre-processing

and regression model activation were adjusted compared to the initial hyperparameters setup used for the H_2 0D and 2D cases. However, the overall logic of the clustering-preprocessing-regression workflow for the learning process is retained. It was found that the *ReLU* activation function in this specific experiment is more robust than the *swish* function. At this point, it is evident that the choice of activation function is a critical hyperparameter that requires tuning, not only for training performance but also for the robustness of inference during the real implementation of the model.

Training results

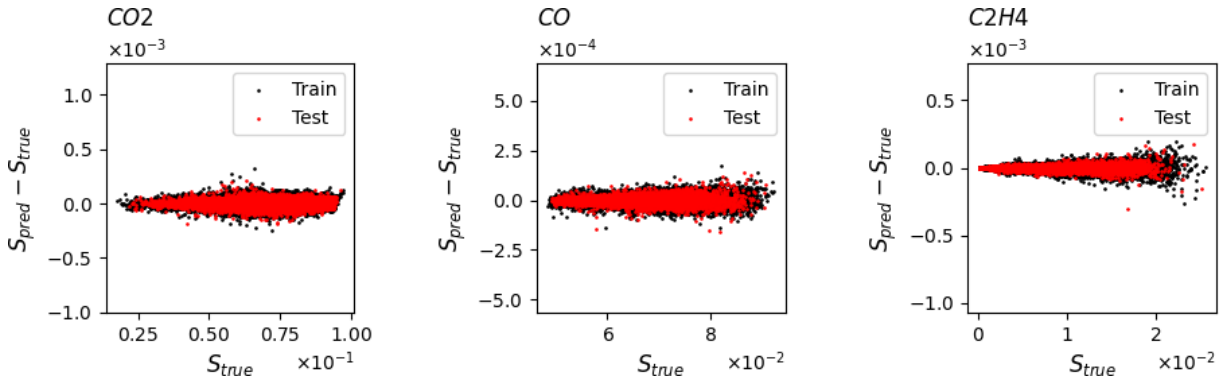


(a) cluster 0

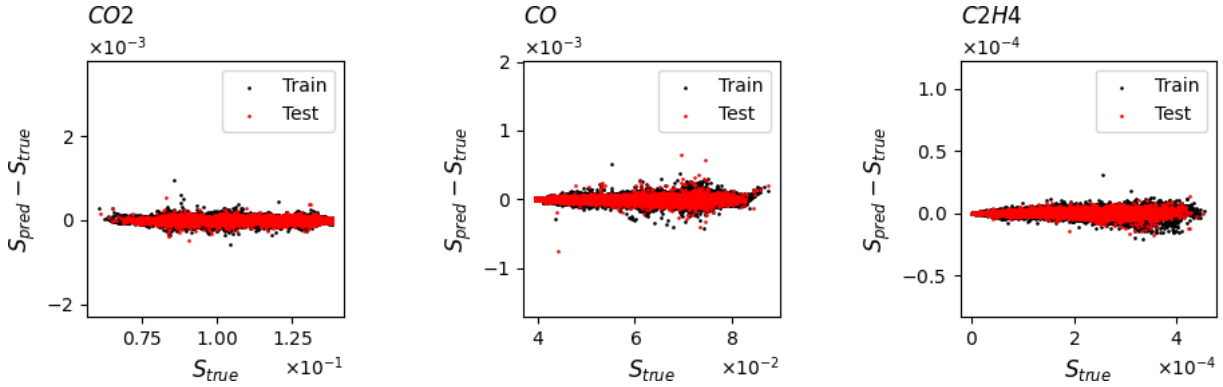
(b) cluster 1

(c) cluster 2

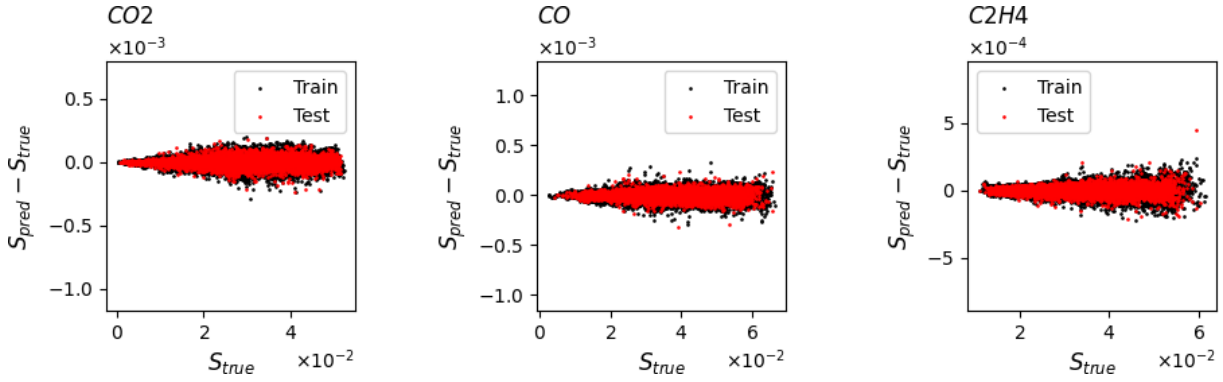
Figure 5.19: The learning curves for each cluster in C_2H_4 case.



(a) cluster 0



(b) cluster 1



(c) cluster 2

Figure 5.20: The parity plots for each cluster in C_2H_4 case, where the black points are the data from training SMR dataset, red points are the data from test SMR dataset, and the green points are directly from DNS simulation of C_2H_4

Figure 5.19 indicates that all the loss function values for training and validation converged to similar values without significant differences. The loss function values converge to a small scale of 10^{-6}. Table 5.5 provides the training loss values and logMAE errors for each cluster. The logMAE errors for each dimension are in $10^{-4} - 10^{-3}$ range, and the overall logMAE errors are all around 10^{-3}. Figure 5.20 displays the parity plots between the training and test samples. The mean absolute errors for chemical species is limited to around 1% of real output values, confirming the model training performance based on

the DNS snapshot dataset.

Cluster index	Loss	logMAE(C_2H_4)	logMAE(O_2)	logMAE(CO_2)	logMAE(C_2H_5)	logMAE(CH_3CHO)	logMAE
0	5.57×10^{-6}	9.84×10^{-3}	2.23×10^{-4}	1.83×10^{-4}	5.76×10^{-3}	4.45×10^{-3}	3.85×10^{-3}
1	4.81×10^{-6}	4.37×10^{-4}	2.36×10^{-4}	7.26×10^{-4}	1.34×10^{-3}	6.35×10^{-4}	1.52×10^{-3}
2	3.70×10^{-6}	1.23×10^{-3}	3.00×10^{-4}	3.04×10^{-4}	1.55×10^{-3}	2.58×10^{-4}	9.04×10^{-4}

Table 5.5: Loss values and logMAE metrics for each dimension, for the C_2H_4 case based on DNS generated dataset

2D inference performance

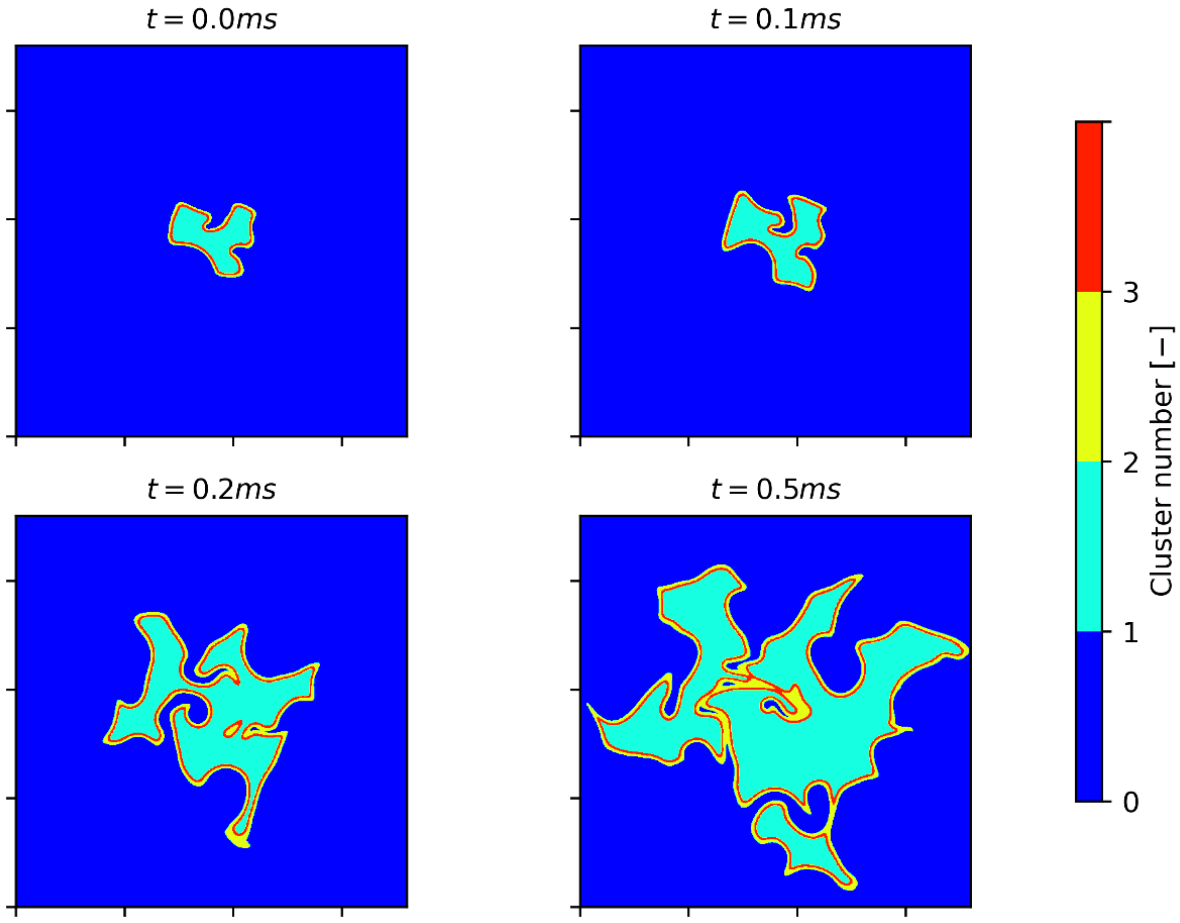


Figure 5.21: The contour plot of KMeans clustering for C_2H_4 case with DNS based dataset, the simulation time is up to 5×10^{-4}s

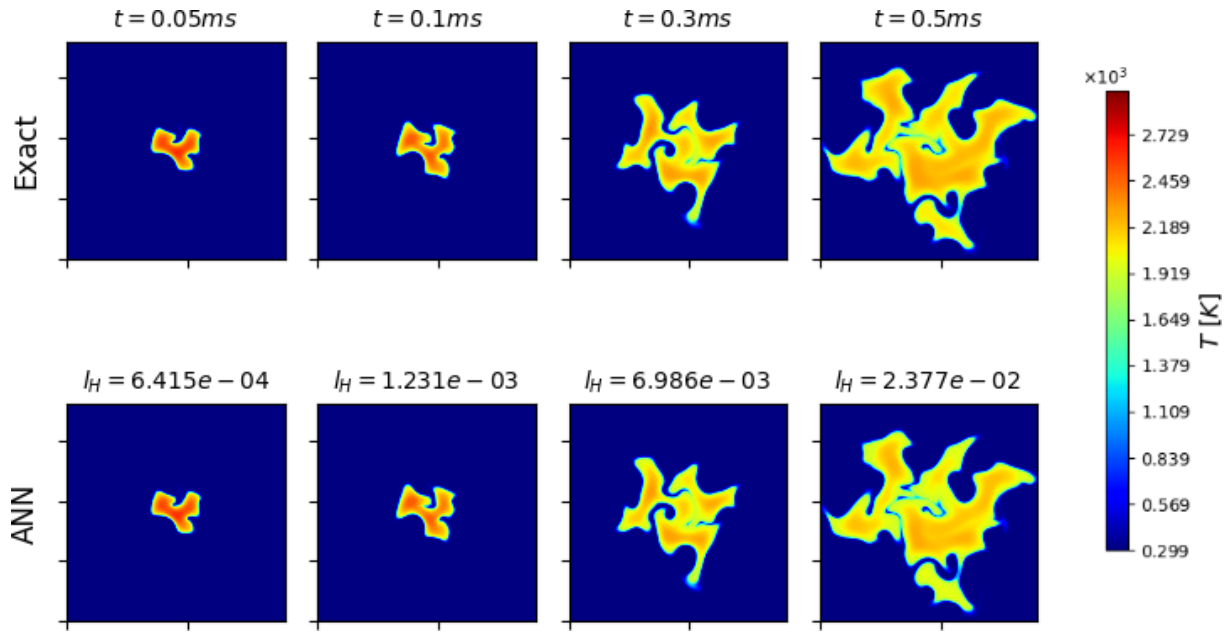


Figure 5.22: The contour plot of temperature field between the deep learning based simulation and CVODE based simulation for C_2H_4 case with DNS based dataset, the simulation time is up to 5×10^{-4}s

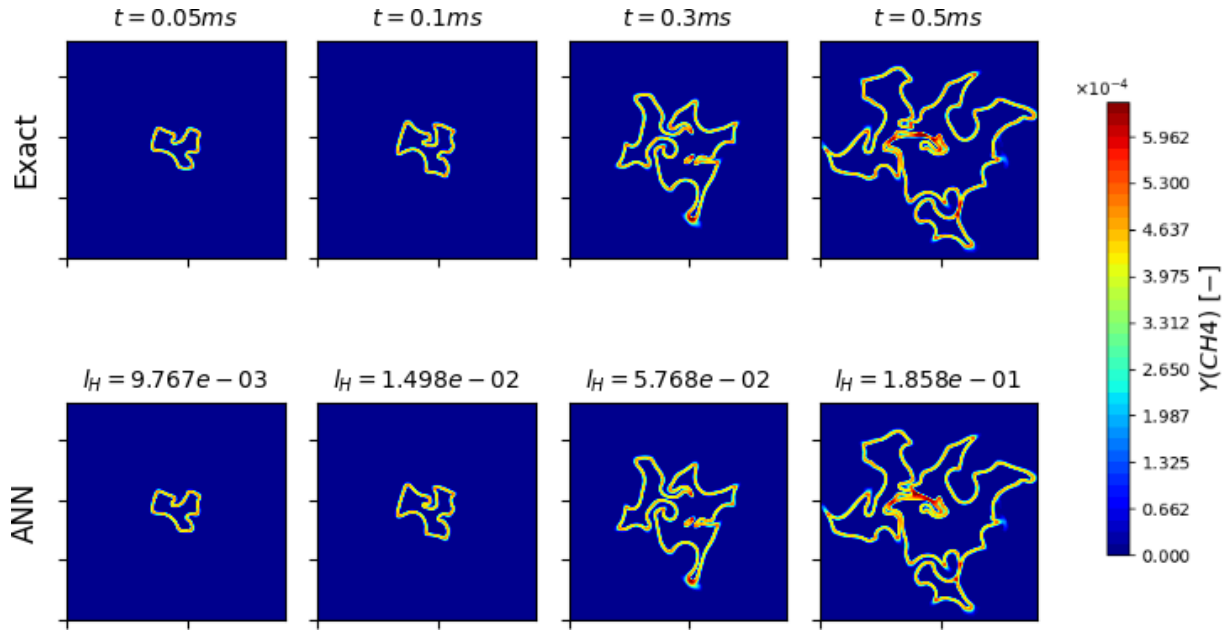


Figure 5.23: The contour plot of CH_4 field between the deep learning based simulation and CVODE based simulation for C_2H_4 case with DNS based dataset, the simulation time is up to 5×10^{-4}s

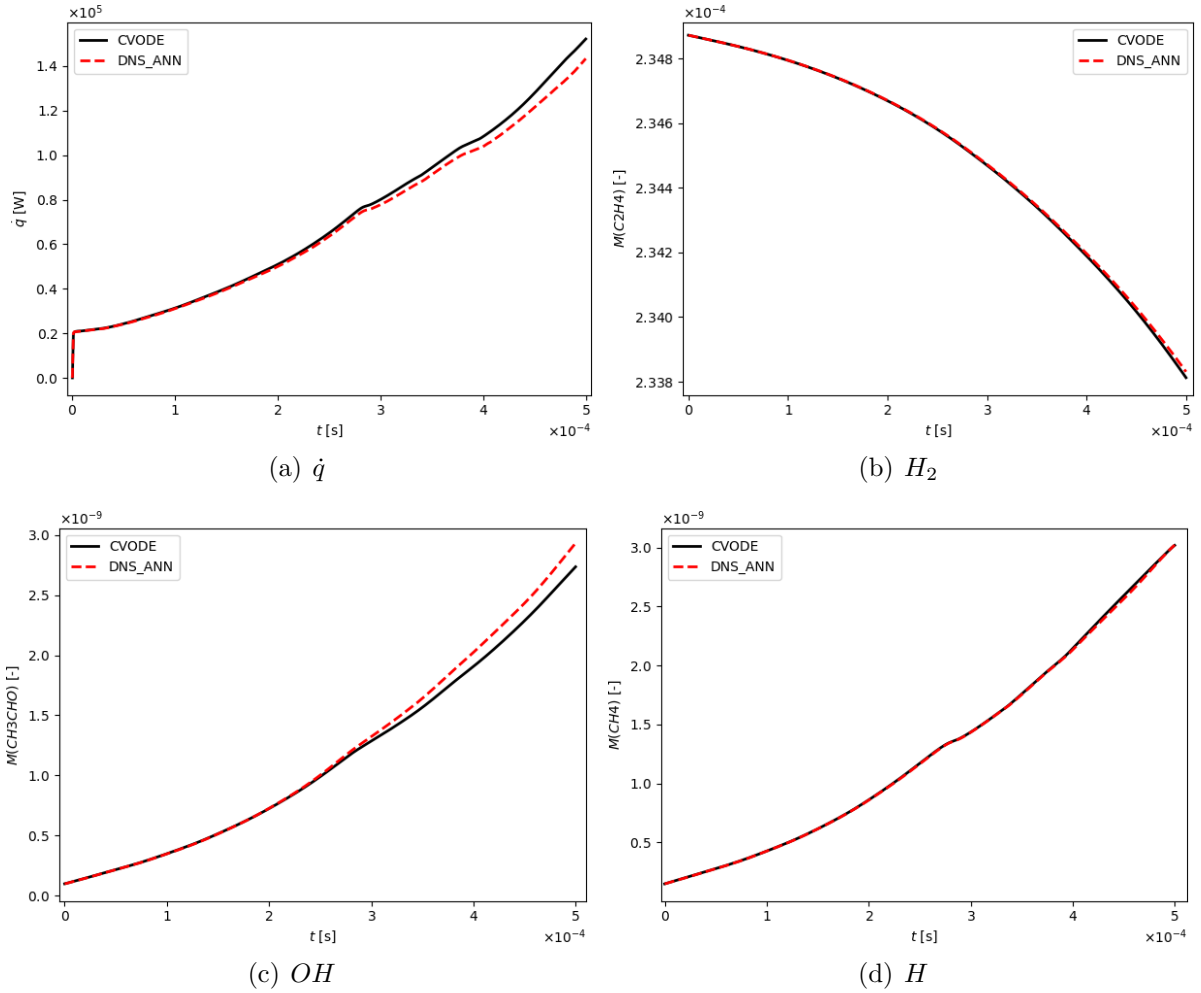


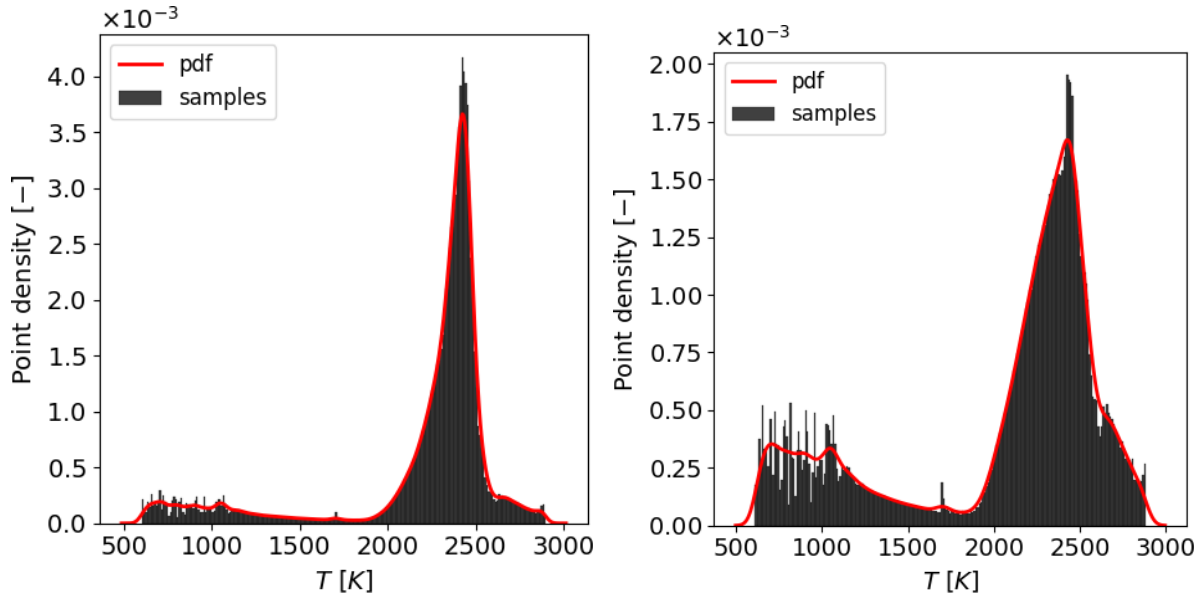
Figure 5.24: Plots for 2D simulations of the heat release rate and total mass fractions of field, simulated with DNS based dataset.

The DNS simulation for the C_2H_4 case was successfully reproduced by learning based inference, with a simulation time extending up to 5×10^{-4} seconds. Figure 5.21 displays the clustering representation plots during the simulation, revealing distributions of clusters similar to those in the H_2 case. Two flame front zones and the high-temperature burnt gas zone are identified during the simulation. Figures 5.22 and 5.23 present the 2D contour plots of temperature and CH_4 field, respectively, for both the deep learning-based simulation and the CVODE-based simulation. The predicted fields using ANN models, trained from the DNS snapshot dataset, closely correspond to the fields resolved by CVODE. The distribution and shapes of flame front with vortices are consistent to results from direct integration by CVODE. Similar to the H_2 case, the Hellinger distance metric for each predicted field remains within $(10^{-3} - 10^{-1})$ which are near zero, demonstrating a good performance of the deep learning models. Finally, Figure 5.24 shows the global quantities within the simulation field, demonstrating consistency of ANN model predictions based on DNS dataset, compared to results with CVODE direct integration.

Conclusion

In the case where models are trained on DNS-based dataset with sparse sampling, the *swish* activation function, which exhibited better training and inference performance in the 0D case, resulted in poor generalization during the inference of the 2D case, leading to diverged simulations. Hence, *ReLU* function was chosen to replace *swish* function in experiments. Moreover, KMeans clustering only produced convergent inference results when logarithmic transformation was no longer used for the clustering process. After modifying these hyperparameters, the model training performance are successfully validated, and the inference 2D simulation using pre-trained learning model also produce the consistent results with respect to simulation with direct integration by CVODE. The next section presents the results for models trained on dataset from stochastic mixing reactors model. In this case, the dataset generation is much less expensive, with no necessary of running high fidelity simulations.

5.4.2 Training model with stochastic mixing reactors dataset



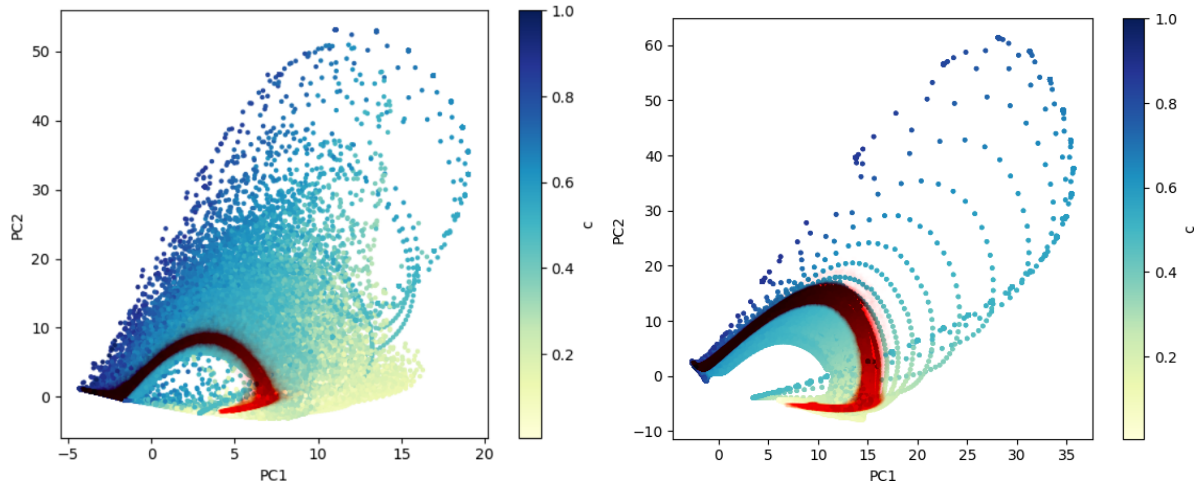
(a) (b)

Figure 5.25: (a): Data points distribution of temperature in original dataset (b) Data points distribution of temperature in resampled dataset.

Data preprocessing

The case is then carried out with models trained on stochastic mixing reactors (SMR) dataset. The C_2H_4 case presents more complexity due to the increased number of chemical species involved in the simulation. We test two models, one is trained on SMR dataset with modified Curl model and the other is trained on SMR dataset with EMST model.

The initial conditions for the C_2H_4 case are outlined in Table 5.1. Because the chemistry system is stiffer in the C_2H_4 case, to further tackle the unbalanced original data distribution with an over-generation of data points in the high-temperature zone, a data resampling method [65, 11] was applied to re-balance the total dataset. The original samples are randomly selected with probabilities which is proportional to heat release rates. By this mean, the samples distribution in regions with high reaction rates and with low reaction rates is more balanced. After resampling, the dataset size was reduced to 10^6 samples.



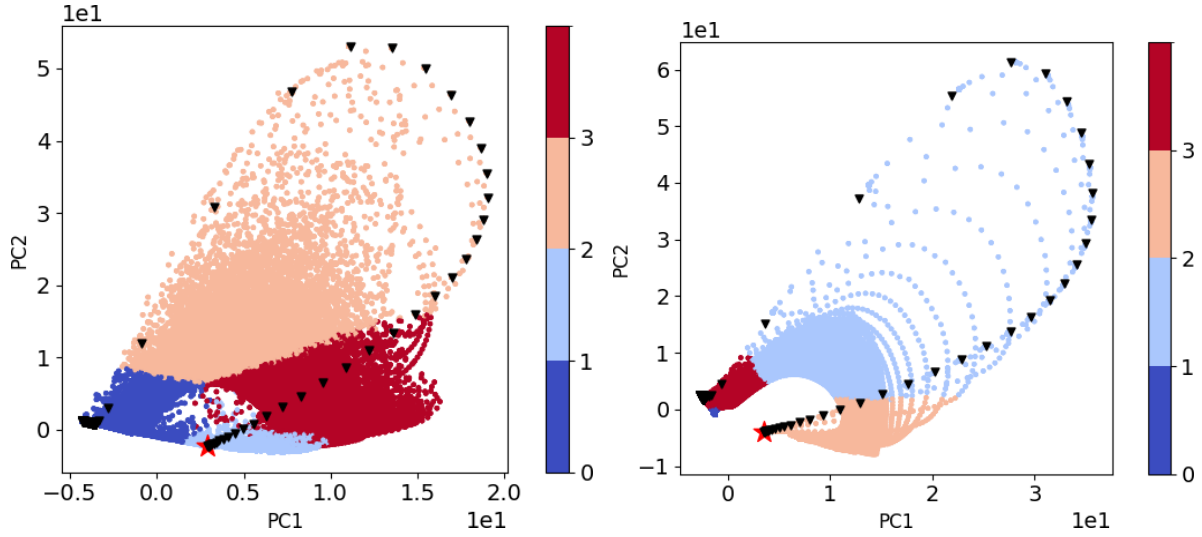
(a) (b)

Figure 5.26: The comparison between the data points from real simulation and SMR system with resampling for C_2H_4, where 5.26(a) is for modified Curl based SMR dataset, and 5.26(b) is for EMST based SMR dataset. The red color represent the points from real simulations, and the colorbar from blue to yellow represent the data from SMR simulations, the variation of colors denotes the evolution of progress variable of temperature c

The density plots of temperature data samples before and after resampling are shown in Figure 5.25. Many data samples from the high-temperature zone were removed by the resampling technique. The data distribution is slightly rebalanced for fast reaction rate regions, and the general distribution of temperature is more homogeneous. The comparison between the data points from real DNS simulation and SMR system for C_2H_4 case with data resampling technique are shown in Figure 5.26 to demonstrate the consistency of SMR dataset with respect to the real DNS simulation data. The points are projected on the reduced 2 dimensions PC_1 and PC_2 using PCA algorithm, based on the dataset of SMR particles. The red color represent the points from real simulations, and the colorbar from blue to yellow represent the data from SMR simulations, the variation of colors denotes the evolution of progress variable based on temperature $c = \frac{T-T_0}{T_f-T_0}$.

Moreover, To make the dataset more robust, the input states of original generated data samples are also perturbed as $\mathbf{S}' = \mathbf{S}(1+\epsilon)$, where ϵ is perturbed noise level. Then, the

perturbed input states are computed by Cantera solver to obtain the perturbed output states. The original data is perturbed twice with ϵ equal to 2% and 3%. It must be noted that particularly in C_2H_4 case, without the addition of noise to the dataset, models based on both approaches are extremely unrobust, resulting in the divergence of inference simulations. In total, as for model training, there are overall 3×10^6 samples in the dataset.



(a) (b)

Figure 5.27: Data points clustering in PCA reduced 2D space and in T-C_2H_4 space, where the data is from SMR models generated samples with mixing terms approximated by (a): modified Curl model (b) EMST model.

The preprocessing of data for C_2H_4 is firstly set in teh same manner than for 0D cases, where a threshold of 10^{-12} and logarithmic transformation was applied. The cluster number is set to 4 for all two cases. Figure 5.27 shows data scatter with clustering index of SMR models generated dataset, whose mixing terms are approximated by modified Curl model and EMST model respectively. Most of the data samples are located in the zone where the reactants are consumed. The samples with high reaction rates are partitioned to different clusters. As for each cluster, it is expected that local adaptive models are easier to be trained.

Hyperparameters

The neural network structure is designed identically to the models with the DNS-based dataset. The activation function is set to *ReLU*, which consistently yields better generalization performance for the 2D C_2H_4 case and ensures the convergence of calculations during inference simulations. The optimization algorithm employed is Adam, with an initial learning rate set to 0.0075. Model parameters are initialized using the Glorot Uniform method. An exponential learning rate decay is consistently applied to adaptively adjust

the optimization process.

Training results

After training the models for each cluster, the loss function values post-training are provided in Tables 5.6 and 5.7, respectively. The overall mean logMAE errors for each subdomain are on a 10^{-3} scale, which is ten times larger than the case for H_2. Even with an increase in neural network size, there is no significant improvement in training performance. Globally, models based on the SMR dataset with two different approximation models exhibit similar training performance in terms of log mean absolute error estimations. Furthermore, these errors are on a similar scale to the training results from the DNS-generated dataset. This indicates that the models on these different datasets perform similarly.

Cluster index	Loss	logMAE(C_2H_4)	logMAE(O_2)	logMAE(CO_2)	logMAE(C_2H_5)	logMAE(CH_3CHO)	logMAE
0	3.13×10^{-5}	3.54×10^{-3}	3.77×10^{-4}	3.04×10^{-4}	2.60×10^{-3}	1.08×10^{-2}	2.71×10^{-3}
1	5.92×10^{-5}	4.62×10^{-4}	2.85×10^{-4}	2.79×10^{-3}	7.50×10^{-3}	2.16×10^{-3}	4.65×10^{-3}
2	9.75×10^{-6}	2.45×10^{-3}	7.25×10^{-4}	1.97×10^{-3}	3.83×10^{-3}	2.50×10^{-3}	2.14×10^{-3}
3	1.35×10^{-5}	1.18×10^{-3}	4.96×10^{-4}	1.84×10^{-3}	1.81×10^{-3}	1.32×10^{-3}	2.23×10^{-3}

Table 5.6: Loss values and logMAE metrics for each dimension, for the C_2H_4 case based on SMR dataset with modified Curl model

Cluster index	Loss	logMAE(C_2H_4)	logMAE(O_2)	logMAE(CO_2)	logMAE(C_2H_5)	logMAE(CH_3CHO)	logMAE
0	7.23×10^{-4}	3.85×10^{-3}	3.09×10^{-4}	2.01×10^{-4}	1.86×10^{-3}	1.45×10^{-2}	3.00×10^{-3}
1	3.35×10^{-6}	1.20×10^{-3}	3.64×10^{-4}	1.03×10^{-3}	1.86×10^{-3}	3.67×10^{-4}	1.09×10^{-3}
2	6.76×10^{-5}	4.72×10^{-4}	2.39×10^{-4}	2.29×10^{-3}	3.94×10^{-3}	2.01×10^{-3}	3.24×10^{-3}
3	8.22×10^{-6}	9.09×10^{-3}	2.02×10^{-4}	2.51×10^{-4}	5.45×10^{-3}	1.31×10^{-2}	4.78×10^{-3}

Table 5.7: Loss values and logMAE metrics for each dimension, for the C_2H_4 case based on SMR dataset with EMST model

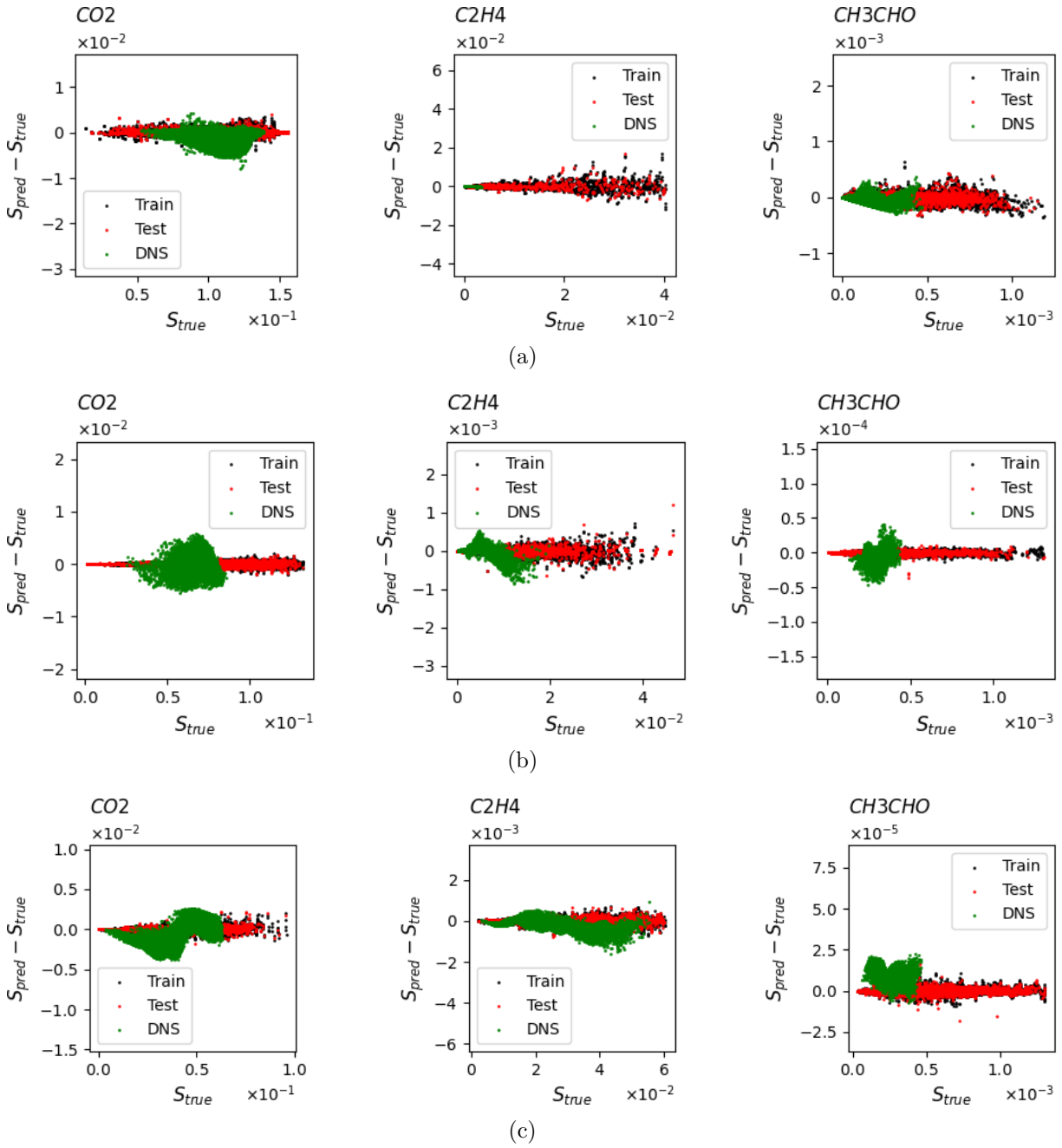


Figure 5.28: Parity plots for: (a) cluster 0, (b) cluster 2, (c) cluster 3 in modified Curl based SMR dataset case. The black points are the data from training SMR dataset, red points are the data from test SMR dataset, and the green points are directly from DNS simulation of C_2H_4

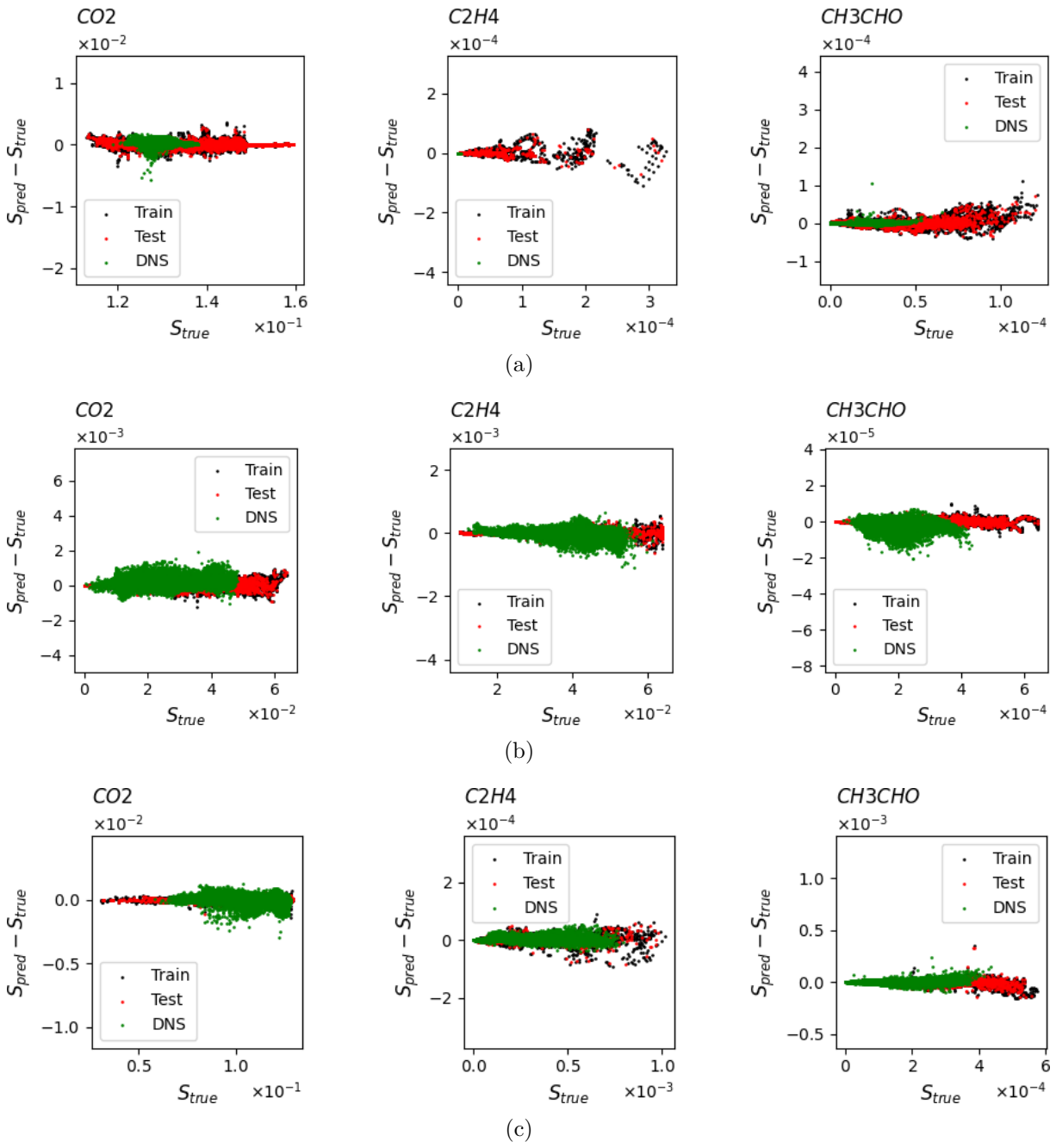
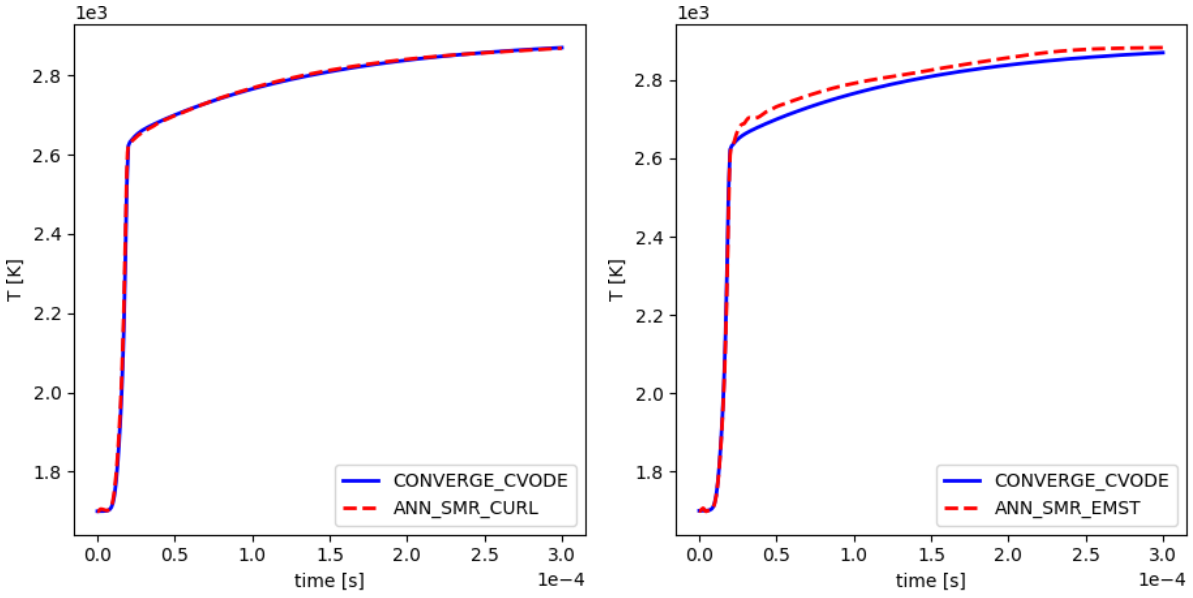


Figure 5.29: Parity plots for: (a) cluster 0, (b) cluster 2, (c) 3 in EMST based SMR dataset case. The black points are the data from training SMR dataset, red points are the data from test SMR dataset, and the green points are directly from DNS simulation of C_2H_4.

Figure 5.28 and 5.29 illustrates the parity plots for different clusters between the training data, test data from the SMR dataset, and samples from target DNS simulations with the same initial conditions, represented in green. Here the training and test data are from SMR dataset, while DNS data are from the target high fidelity simulation. As previously mentioned, the DNS simulations were conducted after the initial fast ignition process to be able to perform the simulation with a constant resolution time step. The total simu-

lation time for the C_2H_4 case is set to 5×10^{-4} seconds, allowing for a flame propagation within the 2D domain. From the parity plots, it is evident that both the training and test data from the SMR dataset align well with the trained models in each cluster, displaying small absolute errors and a lack of outliers. However, as for modified Curl based SMR dataset case, when considering the data from the real target DNS simulation, a substantial number of samples with significant prediction errors of CH_3CHO with respect to ground truth values are observed. In EMST based SMR dataset case, similar outlier is also observed in cluster 0, even the generalization on DNS samples is improved. This discrepancy in the inference of data from DNS simulations on the parity plots ultimately resulted in inconsistency of real a posteriori simulations using the trained models. As shown in training results, all models for 3 different dataset have similar training performance. It is supposed that the modified Curl based SMR dataset can not include all the necessary information of real process of C_2H_4 turbulent combustion.

0D inference performance



(a) $\dot{q}$ (b) H_2

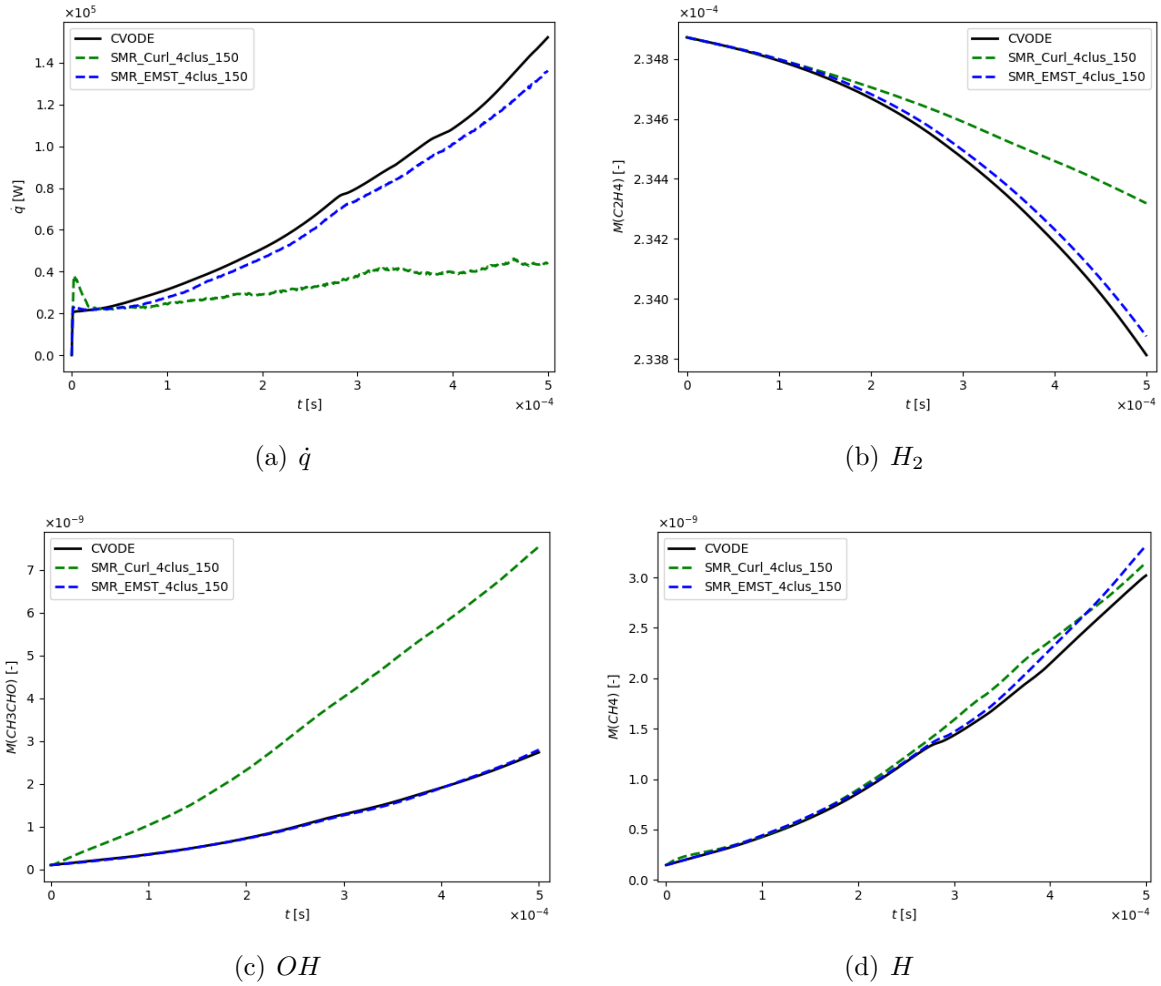
Figure 5.30: The temperature evolution of 0D inference simulation for C_2H_4 case with (a) modified Curl based SMR dataset, (b) EMST based SMR dataset. The simulations are carried out on Converge solver

Figure 5.30 illustrates the results of 0D inference simulations for the C_2H_4 case, where the initial condition is set to $T = 1700.0K$ and $\phi = 1.0$. The purpose is to verify whether the model successfully learns the 0D ignition trajectory. The graph displays the temperature evolution curve until the system reaches the burned-up equilibrium state, comparing simulations conducted with the AI-based model and the CVODE solver.

The inference results based on the deep learning model align closely with those from the CVODE solver. This alignment indicates that, irrespective of the transport approximations used in different models, the deep learning model generally predicts the system's behavior in 0D simulations. Notably, the trajectory of the 0D simulation is part of the training dataset. Nevertheless, there still exists slight discrepancy of predictions in simulation with EMST based SMR dataset.

2D inference performance

Furthermore, we evaluate learning-based simulations for the C_2H_4 case, utilizing an SMR-generated dataset with two different transport term approximations. The initial condition is set at $T_0 = 1700.0K$ and $\phi = 1.0$, and the 2D simulation is initiated after the fast hotspot ignition period to maintain a constant resolution time step.



(a) $\dot{q}$ (b) H_2

(c) OH (d) H

Figure 5.31: Plots for 2D simulations of the heat release rate and total mass fractions of field based on modified Curl based SMR dataset and EMST based dataset.

The simulation employing the modified Curl-based SMR dataset fails to maintain consistency with the simulation using direct integration by the CVODE solver after

$t = 0.1ms$. Figure 5.32 displays clustering contour plots during the simulation. Figures 5.33 and 5.34 present 2D contour plots of temperature and CH_4 fields, respectively, indicating inconsistencies with distorted flame fronts during flame propagation. The Hellinger distance metric for each predicted field is far from zero and more closed to one for CH_4 when flames are fully propagated.

In contrast, the simulation using the EMST-based SMR dataset produces more consistent results with respect to the simulation using direct integration by the CVODE solver. Figure 5.35 shows that the KMeans algorithm partitions the dataset into clusters representing different states of combustion, with flame front regions exhibiting chemical reactions and burned-up regions. Figures 5.36 and 5.37 provide 2D contour plots of temperature and CH_4 fields in this case, capturing flame structures more accurately. Though the Hellinger distance of CH_4 in this case is around 0.5 in the final simulation time step which is not sufficiently closed to zero, it is smaller than that in Curl based SMR dataset case, demonstrating improvement of prediction performance.

Moreover, global quantities are evaluated for these two cases. Figure 5.31 illustrates that, concerning global quantities, the curves of the simulation based on the EMST-based SMR dataset correspond closely to the curves of the simulation based on direct integration by the CVODE solver. All these results demonstrate that the SMR dataset with the EMST approximation for transport terms aligns more with real physical performance in the target turbulent combustion scenario, leading to more accurate predictions from the model.

5.5 Acceleration performance

2D case	CVODE	ANN
H_2	$0.145s$	$0.009s$
C_2H_4	$2.64s$	$0.95s$

Table 5.8: Comparisons of computing time for chemistry in each iteration in 2D case

Finally, we evaluate the acceleration performance for H_2 and C_2H_4 case using ANN based learning workflow for chemistry computations. Up to now, the acceleration performance is simply estimated from the Converge solver. Comparisons for two cases are given by Table 5.8, where the average computing time of chemistry resolution using CVODE and ANN is provided. It is shown that ANN based simulations are faster than CVODE based simulations. However, for C_2H_4 case, the chemistry is more complex and the neural networks are larger, thus the acceleration factor decreases. More works for acceleration performance estimation need to be carried out regarding to the implementation and optimization within the Converge code.

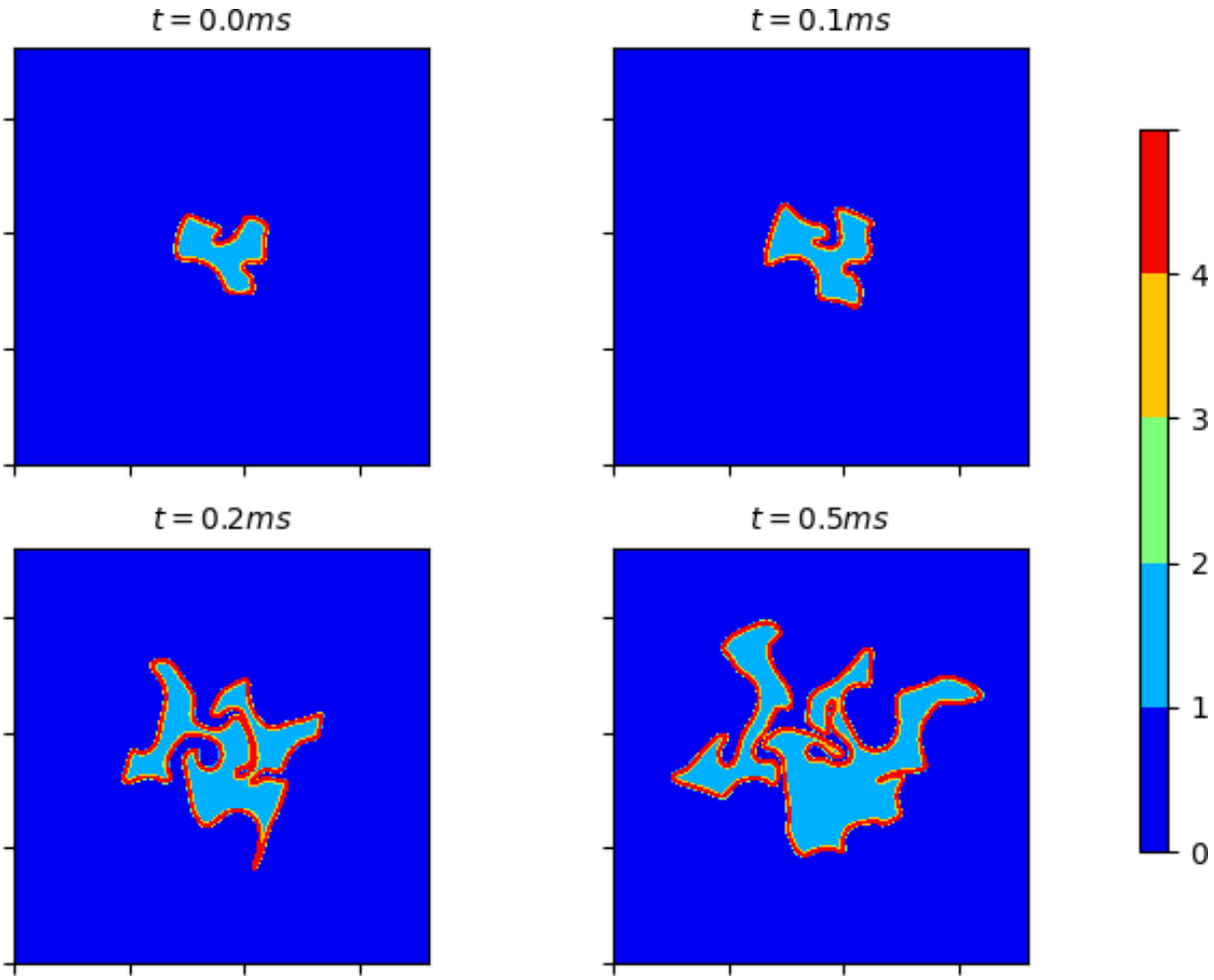


Figure 5.32: The contour plot of KMeans clustering for C_2H_4 case with modified Curl based SMR dataset, the simulation time is up to 5×10^{-4}s

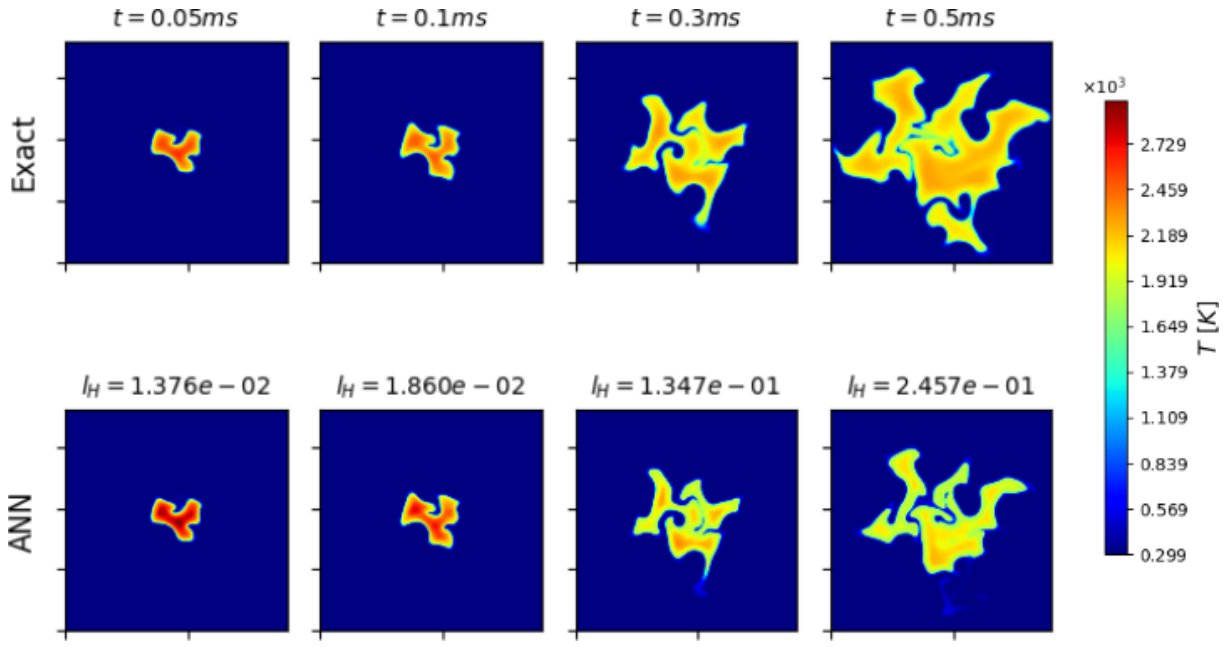


Figure 5.33: The contour plot of temperature field between the deep learning based simulation and CVODE based simulation with modified Curl based SMR dataset, the simulation time is up to 5×10^{-4}s

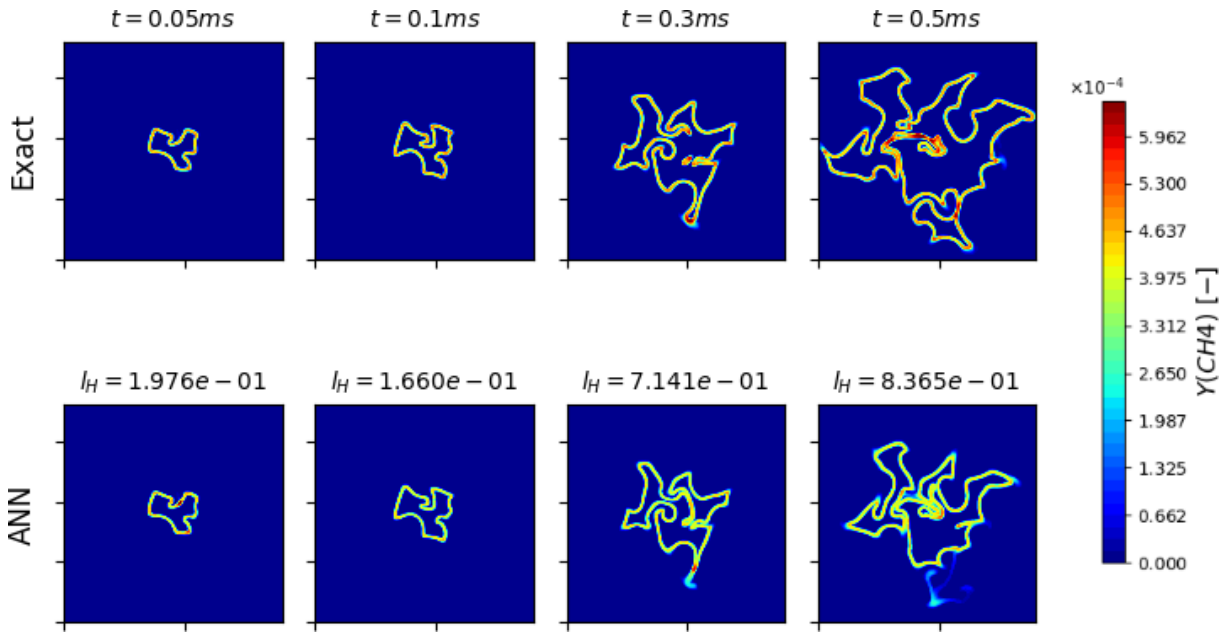


Figure 5.34: The contour plot of CH_4 field between the deep learning based simulation and CVODE based simulation for C_2H_4 case with modified Curl based SMR dataset, the simulation time is up to 5×10^{-4}s

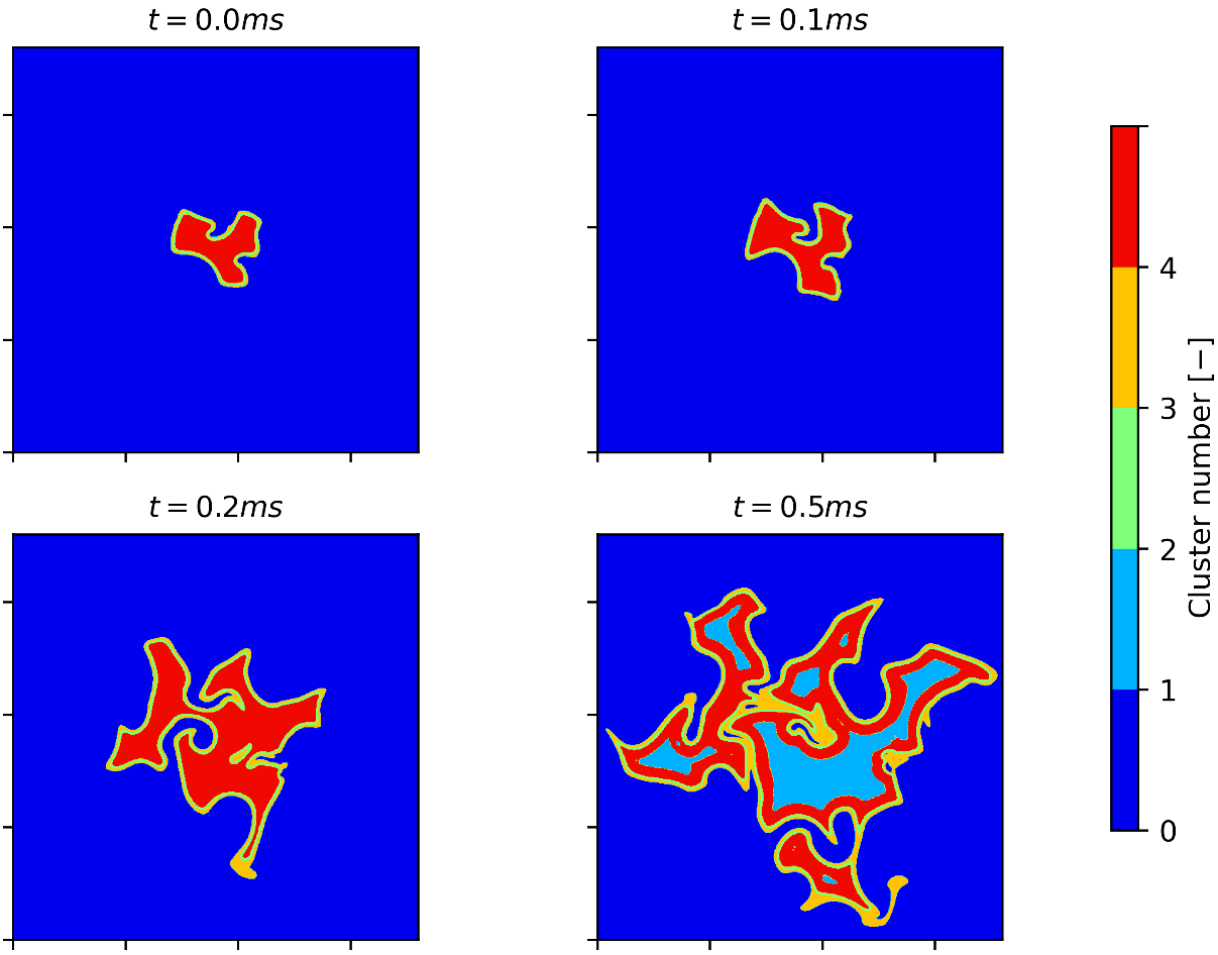


Figure 5.35: The contour plot of KMeans clustering for C_2H_4 case with EMST based SMR dataset, the simulation time is up to 5×10^{-4}s

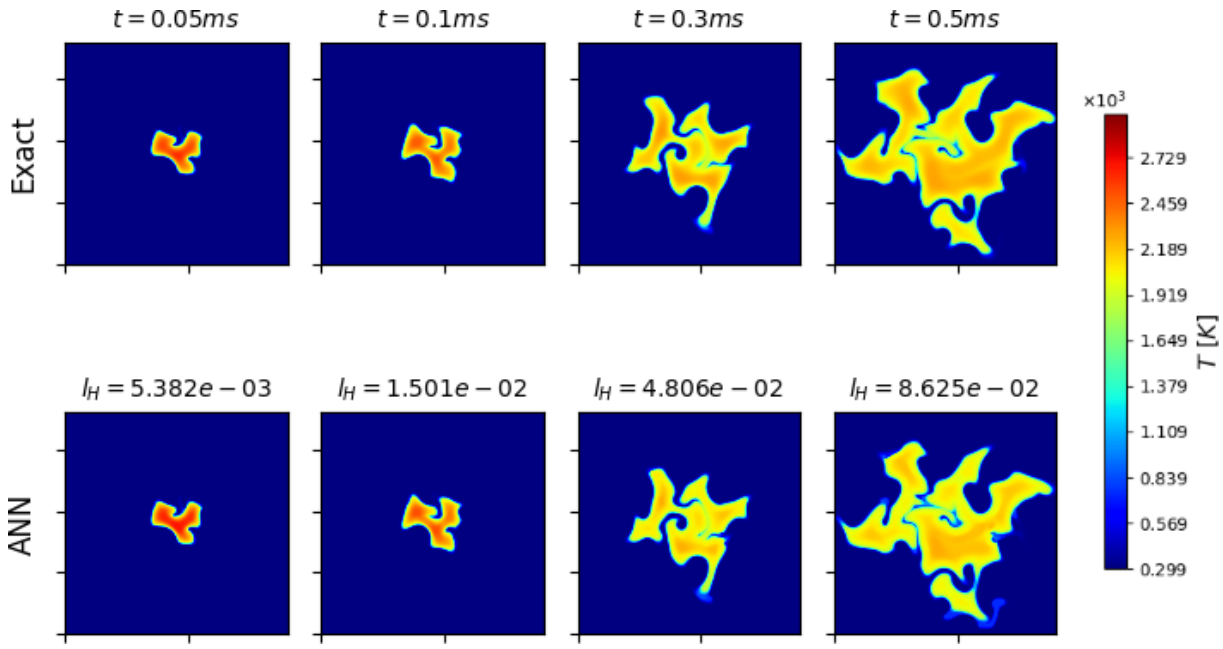


Figure 5.36: The contour plot of temperature field between the deep learning based simulation and CVODE based simulation for C_2H_4 case with EMST based SMR dataset, the simulation time is up to 5×10^{-4}s

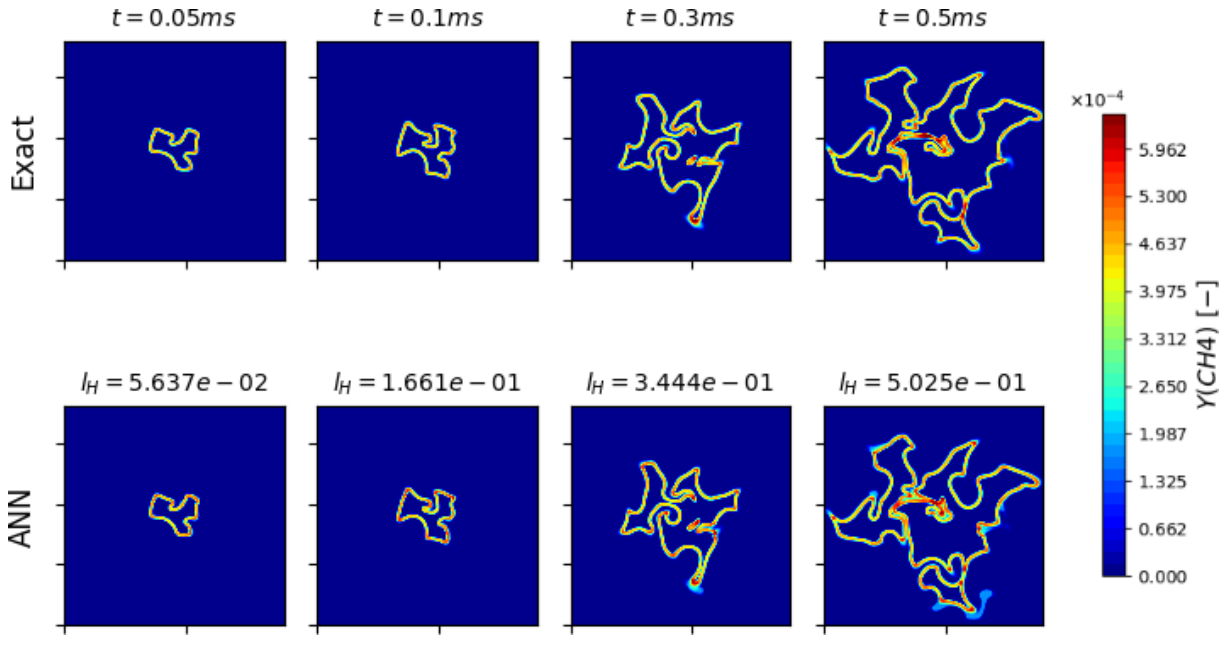


Figure 5.37: The contour plot of CH_4 field between the deep learning based simulation and CVODE based simulation for C_2H_4 case with EMST based SMR dataset, the simulation time is up to 5×10^{-4}s

5.6 Conclusion

Chapter 5 of our work explores the application of the machine learning workflow to 2D scenarios involving hydrogen (H_2) and ethylene (C_2H_4). We generated the dataset using (i) sparse sampling from DNS simulations and (ii) stochastic mixing reactors (SMR) models to simulate combustion processes. For the SMR, the mixing terms were modeled using the modified Curl model and EMST model. The DNS-based dataset serves to validate our basic learning workflow and strategy.The SMR-based dataset under the canonical problem allows to save time in data generation and improve the efficiency of the training process.

In the case of H_2, the learning workflow successfully generated 0D and 2D simulations that closely matched CVODE simulations, showing good agreement in global field quantities such as heat release rate and species mass concentrations. Applying the same learning-based simulation to the C_2H_4 case, we used the DNS-generated dataset for model training, and the resulting 2D simulation exhibited consistency with simulations using the CVODE solver. For simulations using SMR datasets, the SMR dataset based on the EMST model produced more accurate simulations compared to the case with the SMR dataset based on the modified Curl model. This observation suggests that the SMR dataset based on the EMST model contains more accurate information about turbulent-chemistry interactions, resulting in a more effective dataset for model training under the target scenario.

The limit of this study lies in the lack of robustness of the learning model when utilizing SMR datasets for the C_2H_4 case. When training various models with the same hyperparameters but different initialization seeds, the model does not consistently exhibit robustness during the inference process, resulting in simulation failures within the CFD solver. Two potential solutions can be employed to address this issue. Firstly, we may implement a more advanced learning workflow, incorporating techniques to enhance generalization performance, such as increased dataset perturbations, layer normalization, or a physics-informed learning process. Additionally, the choice of the dataset for model training is crucial. Developing a more robust dataset is essential to enhance both training and inference performance.

Overall, this chapter highlights the success and limitations of our learning workflow in more complex combustion simulations and points to areas for future research and improvement.

Chapter 6

Conclusion and future perspective

This study implemented a machine-learning-based workflow for calculating chemical source terms in simulations of reacting flows. After successfully testing the method in a hydrogen (H_2) combustion case, we extended the framework to handle more complex chemistry mechanisms, such as ethylene (C_2H_4) and methane (CH_4). The workflow relies on a combination of unsupervised learning algorithms for clustering and supervised algorithms for data regression.

Initially, this workflow was studied in a simplified 0D combustion case with no transport terms, focusing on an auto-ignition homogeneous reactor system. Various neural network architectures were tested and analyzed. Subsequently, within the same scenario, we introduced a data-driven dimension reduction technique to capture the latent dynamics of complex chemistry systems, verifying the approach with a case study of CH_4 combustion. Finally, the study was extended to multidimensional simulation cases, analyzing the performance in 2D turbulent premixed combustion for both H_2 and C_2H_4 cases.

In the 0D combustion simulation case, the machine learning workflow consistently produced results matching simulations based on the CVODE solver. To address extreme small scales in the data space, a nonlinear data transformation—logarithmic transformation—was applied before the artificial neural network (ANN) regression, enhancing learning performance for extremely small values. The acceleration performance was also assessed. The acceleration factor reached up to 30 times that of direct integration by the CVODE solver. In addition, various neural network designs, incorporating structures based on the original proposition of the residual network and numerical schemes for ordinary differential equations (ODEs), were tested with performance analysis during inference. All three different models consistently produced results that aligned with direct integration. This may provide some inspiration for the design of networks for more complex chemistry.

Subsequently, we expanded the workflow by incorporating an autoencoder, which was employed to reduce the high-dimensional chemical states to latent states with lower di-

mensions. This not only projected the dynamics of the thermochemical system but also combined the learning of the dynamics of the latent states by deep learning models in conjunction with autoencoders. Different models for learning latent dynamics were constructed, utilizing discrete ODE network models and the NeuralODE model. This extended model was applied to the CH_4 combustion case, characterized by its high complexity with 53 chemical states. The results demonstrated that this extended workflow exhibited similar prediction performance than direct integration.

Furthermore, the deep learning workflow underwent additional testing in multidimensional simulations. A 2D turbulent premixed combustion scenario was proposed, and simulations were conducted using deep learning models. These models were trained using data from high-fidelity DNS simulations directly. Additionally, training was performed using data from stochastic micro-mixing reactors (SMR)—a canonical problem with simplified physical models. In this case, mixing is approximated using both the modified Curl model and the EMST model. Among the three cases involving models from different datasets, the scenario with sparse DNS-generated data and the EMST-based SMR dataset consistently produced results in line with simulations based on the CVODE solver. However, models trained from the SMR dataset exhibited a lack of robustness, as convergent simulations could not always be reproduced using the same hyperparameters for the model.

Future perspectives for this project may focus on the following aspects:

- For multidimensional cases, enhancing robustness is a key focus. The robustness of models needs to be carefully analysed. It would be useful to establish the relationships between the validation/test performance of single time step prediction on the dataset and the inference with accumulated time steps prediction. This could be based on statistics derived from training experiments or well-defined estimation factors for learning models. In a first attempt to improve the robustness in the C_2H_4 case, efforts are made regarding model training. The logarithmic transformation for the clustering process is no longer used, leading to an improvement in the generalization performance for inference simulations. Besides, the *swish* activation function no longer produces optimal inference results in 2D simulations of the C_2H_4 case. Models with the *swish* function exhibit poor generalization during direct numerical simulation processes. The generalization of models based on this activation function requires further investigation. Indeed, in this work, it is challenging to systematically carry out the fine-tuning of hyperparameters when unrobust inference simulations are consistently present. Moreover, the current limitations in robustness may stem from the SMR-based dataset, warranting further investigation and studies on SMR models.

- Applying data-driven dimension reduction techniques in multidimensional scenarios

could be valuable. This could lead to the development of a reduced-order system incorporating transport terms. It is important to explore and refine methodologies for projecting transport terms in this context. The projection of transport terms to the reduced latent space has been carried out using linear or nonlinear PCA models in previous works, indicating potential extensions of our dimension reduction workflow.

- Advanced learning workflows can be developed to improve the performance, and to extend the application scenarios. Firstly, for the state-of-the-art models in this work, the training performance has reached saturation and is no longer able to be improved using larger network structures. Models for clusters with pre-ignition states and burned-up states pose challenges in training. In fact, as mentioned in the literature review in Chapter 1, some researchers have already integrated advanced hierarchical KMeans clustering algorithms to further partition the data samples. In future research, similar advanced methods need to be explored and investigated to improve the training performance. In addition, models with variable prediction time steps can be trained. Consequently, they can be utilized for simulations with varying time steps in complex combustion processes, such as fast ignition.

- The learning workflow developed in this work has been successfully tested on a 2D turbulent premixed flames simulation case. Additional test cases involving different combustion regimes, such as non-premixed flames or partially premixed flames, can be further explored using this framework. Beyond academic models, simulations for more complex scenarios within an industrial context, such as combustion chambers in gas turbines and internal combustion engines, may be investigated. Depending on the specific problem, the generation of more comprehensive and robust datasets will be essential.

Appendix A

Reverse-mode derivative of neural ordinary differential equation

Denoting the input and output time step t_i and $t_i+\Delta t$ for a surrogate dynamic, to optimize the continuous loss function, the gradients are computed with respect to θ, t and the input and output time steps t_i and t_{i+1}. As introduced in chapter 4, the dynamical system of adjoint is:

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^T \frac{\partial h(\mathbf{z}(t), \theta)}{\partial \mathbf{z}(t)} \tag{A.0.1}$$

Firstly, the gradient of loss function with respect to the input states $z(t)$ can be computed directly by the inverse integration of adjoint ODE:

$$\frac{\partial L}{\partial \mathbf{z}(t_i)} = \int_{t_i+\Delta t}^{t_i} \mathbf{a}(t)^T \frac{\partial h(\mathbf{z}(t), \theta)}{\partial \mathbf{z}(t)} dt \tag{A.0.2}$$

To compute the other gradients, an augmented state with additional states for θ and t can be expressed as:

$$\frac{d}{dt}\begin{bmatrix}\mathbf{z}\\ \theta \\ t\end{bmatrix}(t) = h_{aug}([\mathbf{z}, \theta, t]) := \frac{d}{dt}\begin{bmatrix}h([\mathbf{z}, \theta, t])\\ \mathbf{0} \\ 1\end{bmatrix} \tag{A.0.3}$$

This equation is based on the fact that θ and t are states with constant differential equations with no change with time, and the adjoint of augmented dynamic are noted as:

$$\mathbf{a}_{aug} := \begin{bmatrix}\mathbf{a}\\ \mathbf{a}_\theta \\ \mathbf{a}_t\end{bmatrix}, \quad \mathbf{a}_\theta(t) := \frac{dL}{d\theta(t)}, \quad \mathbf{a}_t(t) := \frac{dL}{dt(t)} \tag{A.0.4}$$

The ODE of the augmented adjoint is:

$$\frac{d\mathbf{a}_{aug}(t)}{dt} = -\begin{bmatrix}\mathbf{a}(t) & \mathbf{a}_\theta(t) & \mathbf{a}_t(t)\end{bmatrix}\mathcal{J}(t) = -\begin{bmatrix}\mathbf{a}\frac{\partial h}{\partial \mathbf{z}} & \mathbf{a}\frac{\partial h}{\partial \theta} & \mathbf{a}\frac{\partial h}{\partial t}\end{bmatrix}(t) \tag{A.0.5}$$

where $\mathcal{J}$ is the Jacobian of the augmented dynamic, which is formulated as:

$$\mathcal{J} = \frac{\partial h_{aug}}{\partial[\mathbf{z}, \theta, t]} = \begin{bmatrix}\frac{\partial h}{\partial \mathbf{z}} & \frac{\partial h}{\partial \theta} & \frac{\partial h}{\partial t} \\ 0 & 0 & 0 \\ 0 & 0 & 0\end{bmatrix} \tag{A.0.6}$$

As for the first element of augmented adjoint, it is exactly the form as denoted in A.0.1. Finally, as aforementioned methodology underlined, the backward differentiation is based on the inverse resolution of the dynamic adjoint states from $t_i + \Delta t$ to t_i. Therefore, the inverse resolution of the augmented dynamic of adjoint states gives:

$$\begin{aligned}\frac{\partial L}{\partial \theta} &= -\int_{t_i+\Delta t}^{t_i} \mathbf{a}(t)^T \frac{\partial h(\mathbf{z}(t), \theta)}{\partial \theta} dt \\ \frac{\partial L}{\partial t_i} &= -\int_{t_i+\Delta t}^{t_i} \mathbf{a}(t)^T \frac{\partial h(\mathbf{z}(t), \theta)}{\partial t} dt \\ \frac{\partial L}{\partial \mathbf{z}(t_i)} &= -\int_{t_i+\Delta t}^{t_i} \mathbf{a}(t)^T \frac{\partial h(\mathbf{z}(t), \theta)}{\partial \mathbf{z}(t)} dt\end{aligned} \tag{A.0.7}$$

And the gradient with respect to the output time $t_i + \Delta t$ is directly the initial states of the inverse resolution of the adjoint dynamic:

$$\frac{\partial L}{\partial (t_i + \Delta t)} = \mathbf{a}(t_i + \Delta t)^T h(\mathbf{z}(t_i + \Delta t), \theta) \tag{A.0.8}$$

Besides, the gradient with respect to the output states, is originally the adjoint in the output time:

$$\frac{\partial L}{\partial \mathbf{z}(t_i + \Delta t)} = \mathbf{a}(t_i + \Delta t) \tag{A.0.9}$$

which can be computed by auto-differentiation method directly. All the gradients of loss function are provided by A.0.7,A.0.8 and A.0.9.

Appendix B

Supplementary figures for latent states prediction with three different models

These figures show the latent states prediction results, where 4 latent states are picked up to plot the latent dynamic curves.

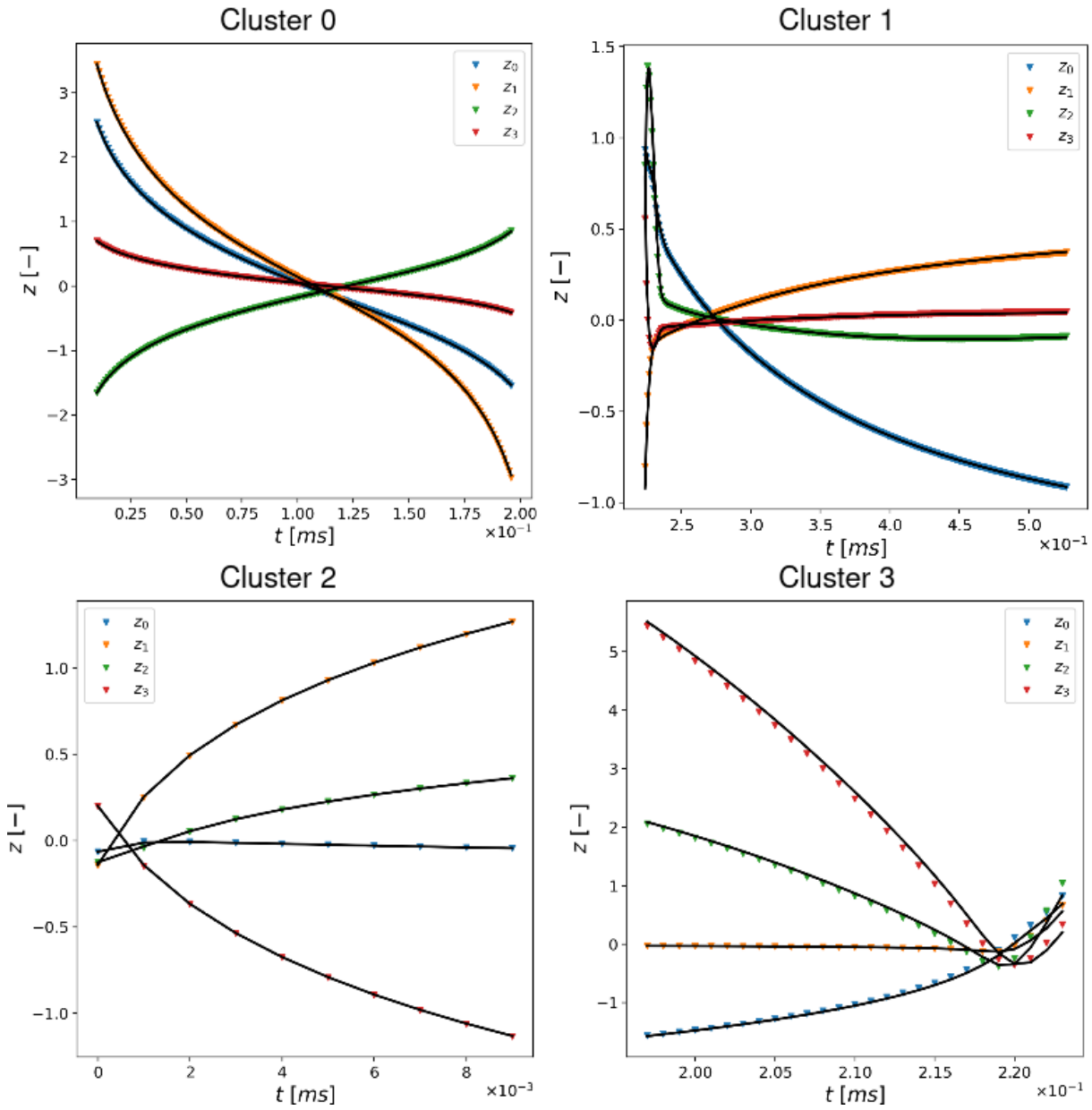


Figure 2.1: Prediction of latent space dynamics using EulerNET model, the initial condition is $T = 1700.0K$, $\phi = 1.1$.

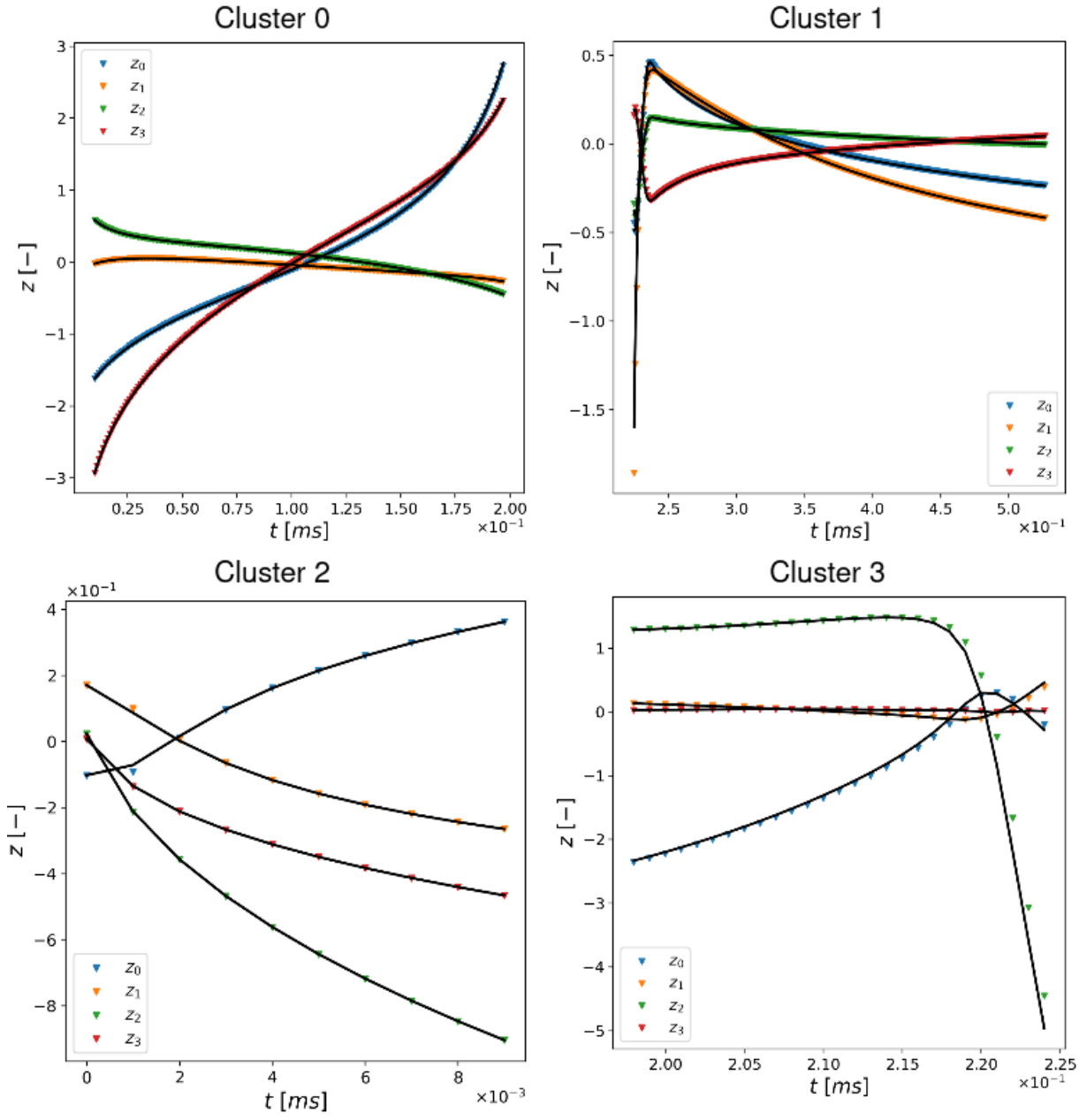


Figure 2.2: Prediction of latent space dynamics using RK4NET model, the initial condition is $T = 1700.0K$, $\phi = 1.1$.

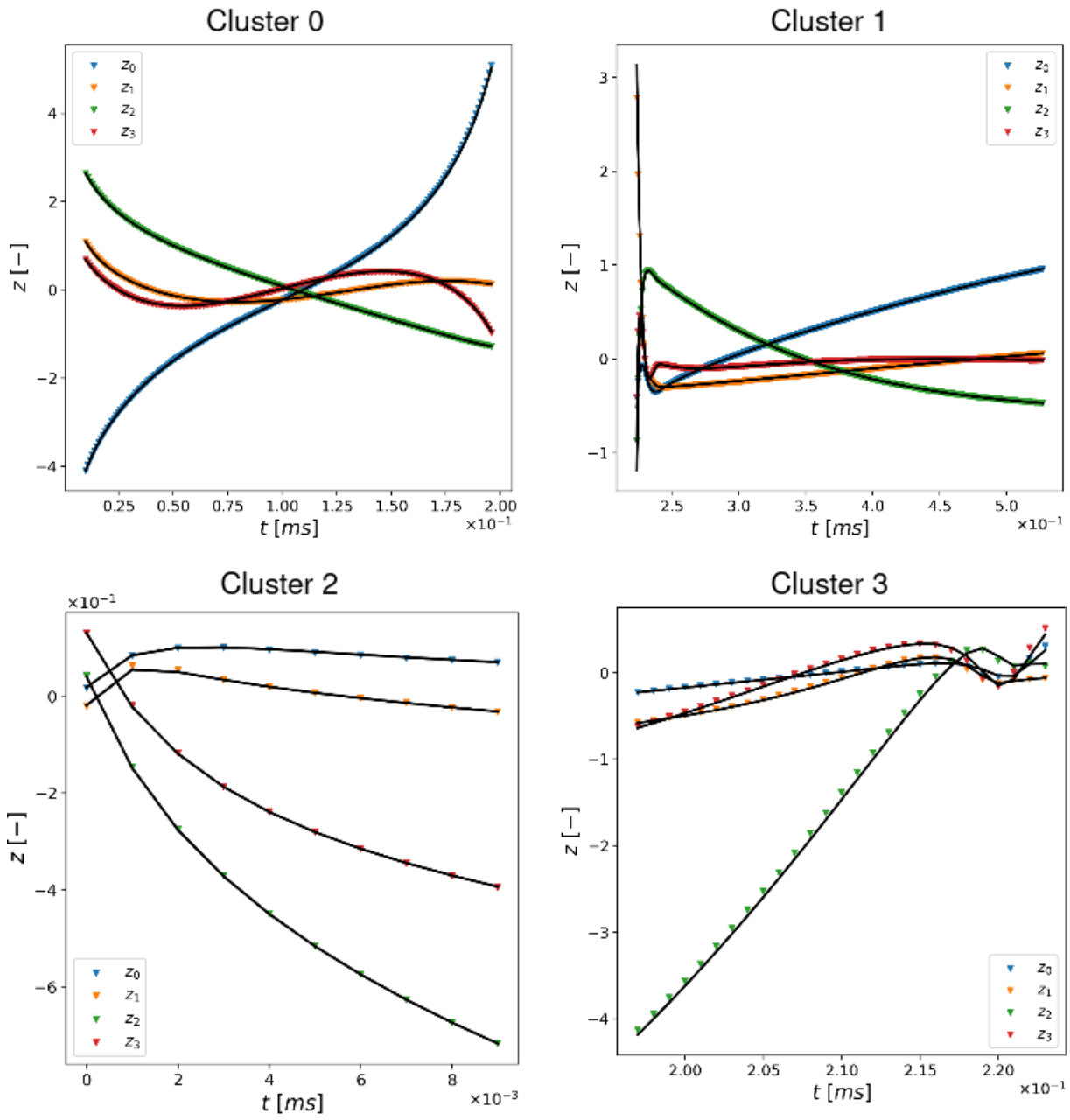


Figure 2.3: Prediction of latent space dynamics using NODE model, the initial condition is $T = 1700.0K$, $\phi = 1.1$.

Bibliography

[1] Dominik Rutz and Rainer Janssen. Biofuel technology handbook. *WIP Renewable energies*, 95, 2007.

[2] Jurgen Warnatz, Ulrich Maas, Robert W Dibble, and J Warnatz. *Combustion*. Springer, 2006.

[3] Kaidi Wan, Camille Barnaud, Luc Vervisch, and Pascale Domingo. 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, 2020.

[4] Jian An, Guoqiang He, Kaihong Luo, Fei Qin, and Bing Liu. Artificial neural network based chemical mechanisms for computationally efficient modeling of hydrogen/carbon monoxide/kerosene combustion. *International Journal of Hydrogen Energy*, 45(53):29594–29605, 2020.

[5] Julian Bissantz, Jeremy Karpowski, Matthias Steinhausen, Yujuan Luo, Federica Ferraro, Arne Scholtissek, Christian Hasse, and Luc Vervisch. Application of dense neural networks for manifold-based modeling of flame-wall interactions. *Applications in Energy and Combustion Science*, 13:100113, 2023.

[6] Lucas LC Franke, Athanasios K Chatzopoulos, and Stelios Rigopoulos. 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, 2017.

[7] Tong Qin, Kailiang Wu, and Dongbin Xiu. Data driven governing equations approximation using deep neural networks. *Journal of Computational Physics*, 395:620–635, 2019.

[8] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[9] Kamila Zdybał, James C Sutherland, and Alessandro Parente. Manifold-informed state vector subset for reduced-order modeling. *Proceedings of the Combustion Institute*, 39(4):5145–5154, 2023.

[10] Kamila Zdybał, Elizabeth Armstrong, Alessandro Parente, and James C Sutherland. Pcafold 2.0—novel tools and algorithms for low-dimensional manifold assessment and optimization. *SoftwareX*, 23:101447, 2023.

[11] Cédric Mehl and Damien Aubagnac-Karkar. On-the-fly accuracy evaluation of artificial neural networks and hybrid method to improve the robustness of neural network accelerated chemistry solving. *Physics of Fluids*, 35(6), 2023.

[12] Shev MacNamara and Gilbert Strang. Operator splitting. In *Splitting methods in communication, imaging, science, and engineering*, pages 95–114. Springer, 2016.

[13] Debby Lanser and Jan G Verwer. Analysis of operator splitting for advection–diffusion–reaction problems from air pollution modelling. *Journal of computational and applied mathematics*, 111(1-2):201–216, 1999.

[14] Raymond L Speth, William H Green, Shev MacNamara, and Gilbert Strang. Balanced splitting and rebalanced splitting. *SIAM Journal on Numerical Analysis*, 51(6):3084–3105, 2013.

[15] Zhen Lu, Hua Zhou, Shan Li, Zhuyin Ren, Tianfeng Lu, and Chung K Law. Analysis of operator splitting errors for near-limit flame simulations. *Journal of Computational Physics*, 335:578–591, 2017.

[16] Scott D Cohen, Alan C Hindmarsh, and Paul F Dubois. Cvode, a stiff/nonstiff ode solver in c. *Computers in physics*, 10(2):138–143, 1996.

[17] M Caracotsios and WE Stewart. Sensitivity analysis of initial-boundary-value problems with mixed pdes and algebraic equations: applications to chemical and biochemical systems. *Computers & chemical engineering*, 19(9):1019–1030, 1995.

[18] Benoit Fiorina, Denis Veynante, and Sébastien Candel. Modeling combustion chemistry in large eddy simulation of turbulent flames. In *Eighth International Symposium on Turbulence and Shear Flow Phenomena*. Begel House Inc., 2013.

[19] Tianfeng Lu and Chung K Law. Toward accommodating realistic fuel chemistry in large-scale computations. *Progress in Energy and Combustion Science*, 35(2):192–215, 2009.

[20] Dimitris A Goussis and Ulrich Maas. Model reduction for combustion chemistry. In *Turbulent combustion modeling*, pages 193–220. Springer, 2011.

[21] Terese Løvås. Model reduction techniques for chemical mechanisms. *Chemical Kinetics*, pages 79–114, 2012.

[22] Alison S Tomlin, Tamás Turányi, and Michael J Pilling. Mathematical tools for the construction, investigation and reduction of combustion mechanisms. In *Comprehensive chemical kinetics*, volume 35, pages 293–437. Elsevier, 1997.

[23] Tianfeng Lu and Chung K Law. A directed relation graph method for mechanism reduction. *Proceedings of the Combustion Institute*, 30(1):1333–1341, 2005.

[24] Norbert Peters and Bernd Rogg. *Reduced kinetic mechanisms for applications in combustion systems*, volume 15. Springer Science & Business Media, 2008.

[25] Anne Felden, Eleonore Riber, and Benedicte Cuenot. Impact of direct integration of analytically reduced chemistry in les of a sooting swirled non-premixed combustor. *Combustion and Flame*, 191:270–286, 2018.

[26] Hai Wang, Rui Xu, Kun Wang, Craig T Bowman, Ronald K Hanson, David F Davidson, Kenneth Brezinsky, and Fokion N Egolfopoulos. A physics-based approach to modeling real-fuel combustion chemistry-i. evidence from experiments, and thermodynamic, chemical kinetic and statistical considerations. *Combustion and Flame*, 193:502–519, 2018.

[27] Rui Xu, Kun Wang, Sayak Banerjee, Jiankun Shao, Tom Parise, Yangye Zhu, Shengkai Wang, Ashkan Movaghar, Dong Joon Lee, Runhua Zhao, et al. 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, 2018.

[28] Melody Cailler, Nasser Darabiha, Denis Veynante, and Benoit Fiorina. Building-up virtual optimized mechanism for flame modeling. *Proceedings of the Combustion Institute*, 36(1):1251–1258, 2017.

[29] Perrine Pepiot-Desjardins and Heinz Pitsch. An efficient error-propagation-based reduction method for large chemical kinetic mechanisms. *Combustion and Flame*, 154(1-2):67–81, 2008.

[30] Nicolas Jaouen, Luc Vervisch, Pascale Domingo, and Guillaume Ribert. Automatic reduction and optimisation of chemistry for turbulent combustion modelling: Impact of the canonical problem. *Combustion and Flame*, 175:60–79, 2017.

[31] Long Liang, John G Stevens, and John T Farrell. A dynamic multi-zone partitioning scheme for solving detailed chemical kinetics in reactive flow computations. *Combustion Science and Technology*, 181(11):1345–1371, 2009.

[32] A Babajimopoulos, DN Assanis, DL Flowers, SM Aceves, and RP Hessel. A fully coupled computational fluid dynamics and multi-zone model with detailed chemical kinetics for the simulation of premixed charge compression ignition engines. *International journal of engine research*, 6(5):497–512, 2005.

[33] Graham M Goldin, Zhuyin Ren, and Selma Zahirovic. A cell agglomeration algorithm for accelerating detailed chemistry in cfd. *Combustion Theory and Modelling*, 13(4):721–739, 2009.

[34] J-Y Chen, W Kollmann, and RW Dibble. Pdf modeling of turbulent nonpremixed methane jet flames. *Combustion Science and Technology*, 64(4-6):315–346, 1989.

[35] Ulrich Maas and Stephen B Pope. Simplifying chemical kinetics: intrinsic low-dimensional manifolds in composition space. *Combustion and flame*, 88(3-4):239–264, 1992.

[36] Olivier Gicquel, D Thevenin, Martin Hilka, and Nasser Darabiha. Direct numerical simulation of turbulent premixed flames using intrinsic low-dimensional manifolds. *Combustion Theory and Modelling*, 3(3):479, 1999.

[37] Olivier Gicquel, Nasser Darabiha, and Dominique Thévenin. Liminar premixed hydrogen/air counterflow flame simulations using flame prolongation of ildm with differential diffusion. *Proceedings of the Combustion Institute*, 28(2):1901–1908, 2000.

[38] Mohamed Embouazza. *Etude de l'auto-allumage par réduction des schémas cinétiques chimiques: application à la combustion homogène diesel*. PhD thesis, Châtenay-Malabry, Ecole centrale de Paris, 2005.

[39] Jéremy Galpin, Alexandre Naudin, Luc Vervisch, Christian Angelberger, Olivier Colin, and Pascale Domingo. Large-eddy simulation of a fuel-lean premixed turbulent swirl-burner. *Combustion and Flame*, 155(1-2):247–266, 2008.

[40] Vincent Fichet. *Modélisation de la combustion du gaz naturel par réseaux de réacteurs avec cinétique chimique détaillée*. PhD thesis, Ecole Centrale Paris, 2008.

[41] Norbert Peters. Turbulent combustion. *Measurement Science and Technology*, 12(11):2022–2022, 2001.

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

[43] Matthias Ihme, Chong M Cha, and Heinz Pitsch. Prediction of local extinction and re-ignition effects in non-premixed turbulent combustion using a flamelet/progress variable approach. *Proceedings of the Combustion Institute*, 30(1):793–800, 2005.

[44] Matthias Ihme and Heinz Pitsch. Prediction of extinction and reignition in non-premixed turbulent flames using a flamelet/progress variable model: 2. application in les of sandia flames d and e. *Combustion and flame*, 155(1-2):90–107, 2008.

[45] H-Y Zhang and JT McKinnon. Elementary reaction modeling of high-temperature benzene combustion. *Combustion Science and Technology*, 107(4-6):261–300, 1995.

[46] Matthias Ihme and Heinz Pitsch. Modeling of radiation and nitric oxide formation in turbulent nonpremixed flames using a flamelet/progress variable formulation. *Physics of Fluids*, 20(5), 2008.

[47] S.B. Pope. Computationally efficient implementation of combustion chemistry using in situ adaptive tabulation. *Combustion Theory and Modelling*, 1(1):41–63, 1997.

[48] Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547–555, 2018.

[49] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[50] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

[51] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[52] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[53] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[54] Matthias Ihme, Wai Tong Chung, and Aashwin Ananda Mishra. Combustion machine learning: Principles, progress and prospects. *Progress in Energy and Combustion Science*, 91:101010, 2022.

[55] Lei Zhou, Yuntong Song, Weiqi Ji, and Haiqiao Wei. Machine learning for combustion. *Energy and AI*, 7:100128, 2022.

[56] FC Christo, AR Masri, and EM Nebot. Artificial neural network implementation of chemistry with pdf simulation of h2/co2 flames. *Combustion and Flame*, 106(4):406–427, 1996.

[57] JA Blasco, N Fueyo, C Dopazo, and J Ballester. Modelling the temporal evolution of a reduced combustion chemical system with an artificial neural network. *Combustion and Flame*, 113(1-2):38–52, 1998.

[58] Baris Ali Sen, Evatt R Hawkes, and Suresh Menon. Large eddy simulation of extinction and reignition with artificial neural networks based chemical kinetics. *Combustion and Flame*, 157(3):566–578, 2010.

[59] Tianjie Ding, Thomas Readshaw, Stelios Rigopoulos, and WP Jones. Machine learning tabulation of thermochemistry in turbulent combustion: An approach based on hybrid flamelet/random data and multiple multilayer perceptrons. *Combustion and Flame*, 231:111493, 2021.

[60] Huu-Tri Nguyen, Camille Barnaud, Pascale Domingo, Phuc-Danh Nguyen, and Luc Vervisch. Large-eddy simulation of flameless combustion with neural-network driven chemistry. *Applications in Energy and Combustion Science*, 14:100126, 2023.

[61] Yan Zhang, Shijie Xu, Shenghui Zhong, Xue-Song Bai, Hu Wang, and Mingfa Yao. Large eddy simulation of spray combustion using flamelet generated manifolds combined with artificial neural networks. *Energy and AI*, 2:100021, 2020.

[62] Rene Prieler, Matthias Moser, Sven Eckart, Hartmut Krause, and Christoph Hochenauer. Machine learning techniques to predict the flame state, temperature and species concentrations in counter-flow diffusion flames operated with ch4/co/h2-air mixtures. *Fuel*, 326:124915, 2022.

[63] Maximilian Hansinger, Yipeng Ge, and Michael Pfitzner. 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, 2022.

[64] Kaimeng Li, Pourya Rahnama, Ricardo Novella, and Bart Somers. Combining flamelet-generated manifold and machine learning models in simulation of a non-premixed diffusion flame. *Energy and AI*, 14:100266, 2023.

[65] Cheng Chi, Xiaopeng Xu, and Dominique Thévenin. Efficient premixed turbulent combustion simulations using flamelet manifold neural networks: A priori and a posteriori assessment. *Combustion and Flame*, 245:112325, 2022.

[66] Matthias Ihme, Christoph Schmitt, and Heinz Pitsch. Optimal artificial neural networks and tabulation methods for chemistry representation in les of a bluff-body swirl-stabilized flame. *Proceedings of the Combustion Institute*, 32(1):1527–1535, 2009.

[67] Z Nikolaou, L Vervisch, and Pascale Domingo. Criteria to switch from tabulation to neural networks in computational combustion. *Combustion and Flame*, 246:112425, 2022.

[68] Javier A Blasco, Norberto Fueyo, JC Larroya, C Dopazo, and Y-J Chen. A single-step time-integrator of a methane–air chemical system using artificial neural networks. *Computers & Chemical Engineering*, 23(9):1127–1133, 1999.

[69] Thomas Readshaw, Tianjie Ding, Stelios Rigopoulos, and WP Jones. Modeling of turbulent flames with the large eddy simulation–probability density function (les–pdf) approach, stochastic fields, and artificial neural networks. *Physics of Fluids*, 33(3), 2021.

[70] Huu-Tri Nguyen, Pascale Domingo, Luc Vervisch, and Phuc-Danh Nguyen. Machine learning for integrating combustion chemistry in numerical simulations. *Energy and AI*, 5:100082, 2021.

[71] Xu Han, Ming Jia, Yachao Chang, and Yaopeng Li. An improved approach towards more robust deep learning models for chemical kinetics. *Combustion and Flame*, 238:111934, 2022.

[72] Tianhan Zhang, Yuxiao Yi, Yifan Xu, Zhi X Chen, Yaoyu Zhang, E Weinan, and Zhi-Qin John Xu. A multi-scale sampling method for accurate and robust deep neural network to predict combustion chemical kinetics. *Combustion and Flame*, 245:112319, 2022.

[73] Cheng Chi, Gábor Janiga, and Dominique Thévenin. On-the-fly artificial neural network for chemical kinetics in direct numerical simulations of premixed combustion. *Combustion and Flame*, 226:467–477, 2021.

[74] Javier A Blasco, Norberto Fueyo, C Dopazo, and JY Chen. A self-organizing-map approach to chemistry representation in combustion applications. *Combustion Theory and Modelling*, 4:61–76, 2000.

[75] Federico Perini. High-dimensional, unsupervised cell clustering for computationally efficient engine simulations with detailed combustion chemistry. *Fuel*, 106:344–356, 2013.

[76] Shivam Barwey, Supraj Prakash, Malik Hassanaly, and Venkat Raman. Data-driven classification and modeling of combustion regimes in detonation waves. *Flow, Turbulence and Combustion*, 106:1065–1089, 2021.

[77] Manabu Saito, Jiangkuan Xing, Jun Nagao, and Ryoichi Kurose. Data-driven simulation of ammonia combustion using neural ordinary differential equations (node). *Applications in Energy and Combustion Science*, 16:100196, 2023.

[78] Ryota Nakazawa, Yuki Minamoto, Nakamasa Inoue, and Mamoru Tanahashi. Species reaction rate modelling based on physics-guided machine learning. *Combustion and Flame*, 235:111696, 2022.

[79] VF Nikitin, IM Karandashev, M Yu Malsagov, and EV Mikhalchenko. Approach to combustion calculation using neural network. *Acta Astronautica*, 194:376–382, 2022.

[80] Weiqi Ji, Weilun Qiu, Zhiyu Shi, Shaowu Pan, and Sili Deng. Stiff-pinn: Physics-informed neural network for stiff chemical kinetics. *The Journal of Physical Chemistry A*, 125(36):8098–8106, 2021.

[81] Yuting Weng and Dezhi Zhou. Multiscale physics-informed neural networks for stiff chemical kinetics. *The Journal of Physical Chemistry A*, 126(45):8534–8543, 2022.

[82] S Yao, A Kronenburg, A Shamooni, Oliver Thomas Stein, and W Zhang. Gradient boosted decision trees for combustion chemistry integration. *Applications in Energy and Combustion Science*, 11:100077, 2022.

[83] Victor Churchill and Dongbin Xiu. Flow map learning for unknown dynamical systems: Overview, implementation, and benchmarks. *Journal of Machine Learning for Modeling and Computing*, 4(2), 2023.

[84] Yi-Jen Wang and Chin-Teng Lin. Runge-kutta neural network for identification of dynamical systems in high accuracy. *IEEE Transactions on Neural Networks*, 9(2):294–307, 1998.

[85] Byungjoo Kim, Bryce Chudomelka, Jinyoung Park, Jaewoo Kang, Youngjoon Hong, and Hyunwoo J Kim. Robust neural networks inspired by strong stability preserving runge-kutta methods. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pages 416–432. Springer, 2020.

[86] Zhengbo Luo, Zitang Sun, Weilian Zhou, Zizhang Wu, and Sei-ichiro Kamata. Rethinking resnets: improved stacking strategies with high-order schemes for image classification. *Complex & Intelligent Systems*, 8(4):3395–3407, 2022.

[87] Opeoluwa Owoyele and Pinaki Pal. Chemnode: A neural ordinary differential equations framework for efficient chemical kinetic solvers. *Energy and AI*, 7:100118, 2022.

[88] Shubhangi Bansude, Farhad Imani, and Reza Sheikhi. Performance assessment of chemical kinetics neural ordinary differential equations in pairwise mixing stirred reactor. *ASME Open Journal of Engineering*, 2, 2023.

[89] Henry E Dikeman, Hongyuan Zhang, and Suo Yang. Stiffness-reduced neural ode models for data-driven reduced-order modeling of combustion chemical kinetics. In *AIAA SCITECH 2022 Forum*, page 0226, 2022.

[90] James C Sutherland and Alessandro Parente. Combustion modeling using principal component analysis. *Proceedings of the Combustion Institute*, 32(1):1563–1570, 2009.

[91] Benjamin J Isaac, Axel Coussement, Olivier Gicquel, Philip J Smith, and Alessandro Parente. Reduced-order pca models for chemical reacting flows. *Combustion and flame*, 161(11):2785–2800, 2014.

[92] Axel Coussement, Benjamin J Isaac, Olivier Gicquel, and Alessandro Parente. Assessment of different chemistry reduction methods based on principal component analysis: Comparison of the mg-pca and score-pca approaches. *Combustion and flame*, 168:83–97, 2016.

[93] Kamila Zdybał, Elizabeth Armstrong, Alessandro Parente, and James C Sutherland. Pcafold: Python software to generate, analyze and improve pca-derived low-dimensional manifolds. *SoftwareX*, 12:100630, 2020.

[94] Alessandro Parente, James C Sutherland, Leonardo Tognotti, and Philip J Smith. Identification of low-dimensional manifolds in turbulent flames. *Proceedings of the Combustion Institute*, 32(1):1579–1586, 2009.

[95] Giuseppe D'Alessio, Alessandro Parente, Alessandro Stagni, and Alberto Cuoci. Adaptive chemistry via pre-partitioning of composition space and mechanism reduction. *Combustion and Flame*, 211:68–82, 2020.

[96] Kamila Zdybał, Giuseppe D'Alessio, Antonio Attili, Axel Coussement, James C Sutherland, and Alessandro Parente. Local manifold learning and its link to domain-based physics knowledge. *Applications in Energy and Combustion Science*, 14:100131, 2023.

[97] Mohammad Rafi Malik, Ruslan Khamedov, Francisco E Hernández Pérez, Axel Coussement, Alessandro Parente, and Hong G Im. Dimensionality reduction and unsupervised classification for high-fidelity reacting flow simulations. *Proceedings of the Combustion Institute*, 39(4):5155–5163, 2023.

[98] Mohammad Rafi Malik, Benjamin J Isaac, Axel Coussement, Philip J Smith, and Alessandro Parente. Principal component analysis coupled with nonlinear regression for chemistry reduction. *Combustion and Flame*, 187:30–41, 2018.

[99] Suliman Abdelwahid, Mohammad Rafi Malik, Hasan Abed Al Kader Hammoud, Francisco E Hernández-Pérez, Bernard Ghanem, and Hong G Im. Large eddy simulations of ammonia-hydrogen jet flames at elevated pressure using principal component analysis and deep neural networks. *Combustion and Flame*, 253:112781, 2023.

[100] Mohammad Rafi Malik, Axel Coussement, Tarek Echekki, and Alessandro Parente. Principal component analysis based combustion model in the context of a lifted methane/air flame: Sensitivity to the manifold parameters and subgrid closure. *Combustion and Flame*, 244:112134, 2022.

[101] Hessam Mirgolbabaei and Tarek Echekki. Nonlinear reduction of combustion composition space with kernel principal component analysis. *Combustion and flame*, 161(1):118–126, 2014.

[102] Pei Zhang and Ramanan Sankaran. Autoencoder neural network for chemically reacting systems. *Journal of Machine Learning for Modeling and Computing*, 3(4), 2022.

[103] Anuj Kumar, Martin Rieth, Opeoluwa Owoyele, Jacqueline H Chen, and Tarek Echekki. Acceleration of turbulent combustion dns via principal component transport. *Combustion and Flame*, 255:112903, 2023.

[104] Thierry Poinsot and Denis Veynante. *Theoretical and numerical combustion*. RT Edwards, Inc., 2005.

[105] David G Goodwin. Cantera c++ user's guide. *California Institute of Technology*, 2002.

[106] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[107] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[108] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[109] Jingang Qu, Thibault Faney, Jean-Charles de Hemptinne, Soleiman Yousef, and Patrick Gallinari. Ptflash : A vectorized and parallel deep learning framework for two-phase flash calculation. *Fuel*, 331:125603, 2023.

[110] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.

[111] David Arthur and Sergei Vassilvitskii. K-means++ the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, 2007.

[112] Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. *Mathematics for machine learning*. Cambridge University Press, 2020.

[113] Marcus Ó Conaire, Henry J Curran, John M Simmie, William J Pitz, and Charles K Westbrook. A comprehensive modeling study of hydrogen oxidation. *International journal of chemical kinetics*, 36(11):603–622, 2004.

[114] Zhaoyu Luo, Chun Sang Yoo, Edward S Richardson, Jacqueline H Chen, Chung K Law, and Tianfeng Lu. Chemical explosive mode analysis for a turbulent lifted ethylene jet flame in highly-heated coflow. *Combustion and Flame*, 159(1):265–274, 2012.

[115] G Smith. Gri-mech.-an optimized detailed chemical reaction mechanism for methane combustion. *http://www. me. berkeley. edu/gri_mech*, 1999.

[116] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[117] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[118] Kaichao You, Mingsheng Long, Jianmin Wang, and Michael I Jordan. How does learning rate decay help modern neural networks? *arXiv preprint arXiv:1908.01878*, 2019.

[119] Alisha J Sharma, Ryan F Johnson, David A Kessler, and Adam Moses. Deep learning for scalable chemical kinetics. In *AIAA scitech 2020 forum*, page 0181, 2020.

[120] Damien Aubagnac-Karkar and Cedric Mehl. Nnice: Neural network inference in c made easy, February 2023.

[121] Thierry Passot and Annick Pouquet. Numerical simulation of compressible homogeneous flows in the turbulent regime. *Journal of Fluid Mechanics*, 181:441–466, 1987.

[122] Rane L Curl. Dispersed phase mixing: I. theory and effects in simple reactors. *AIChE journal*, 9(2):175–181, 1963.

[123] S Subramaniam and Stephen B Pope. A mixing model for turbulent reactive flows based on euclidean minimum spanning trees. *Combustion and Flame*, 115(4):487–514, 1998.

[124] Mehl Cédric and Aubagnac-Karkar Damien. ai_reacting_flows (arf), March 2023.

[125] Himanshu Dave, Nedunchezhian Swaminathan, and Alessandro Parente. Interpretation and characterization of mild combustion data using unsupervised clustering informed by physics-based, domain expertise. *Combustion and Flame*, 240:111954, 2022.

RÉSUMÉ

Cette thèse porte sur l'accélération des calculs de cinétique chimique dans les simulations de CFD en s'appuyant sur des méthodes d'apprentissage automatique. Le principe consiste à remplacer la résolution de la chimie dans les calculs par un modèle d'apprentissage équivalent, dont l'évaluation est beaucoup plus rapide que la résolution du système original d'équations différentielles. Les travaux de thèse se concentrent d'abord sur l'étude de cas de combustion simplifiés (système 0-dimensionnel sans termes de transport). Nous proposons un cadre de traitement des données permettant la construction d'un modèle substitutif précis. Nous fournissons une comparaison approfondie entre la résolution de base et la résolution reposant sur le modèle appris. Ensuite, des modèles combinés à des techniques de réduction de dimension sont développés et appliqués à des cas de chimie complexe, constituant une première étape vers des approches de prédiction avec les pas de temps variables. Enfin, notre approche est appliquée à un cas multidimensionnel avec des termes de transport, incluant notamment la turbulence.

MOTS CLÉS

Combustion, cinétique chimique, apprentissage automatique, CFD

ABSTRACT

This thesis deals with the acceleration of chemical kinetics calculations in CFD simulations by relying of machine learning methods. The principle is to replace the resolution of the chemistry in the calculations by an equivalent machine learning model, whose evaluation is much faster than the resolution of the original system of differential equations. The thesis work first focuses on the study of simplified combustion cases (0-dimensional system without transport terms). We propose a data processing framework enabling the construction of an accurate surrogate model. We provide an in-depth comparison between the baseline resolution and the resolution relying on the learned model. Then, models combined with dimension reduction technique and continuous flow learning are developed and applied to complex chemistry cases, providing a first step towards variant time steps prediction approaches. Finally, our approach is applied to a multidimensional case with transport terms, notably including turbulence.

KEYWORDS

Combustion, chemical kinetics, machine learning, CFD