NATIONAL TECHNICAL UNIVERSITY OF ATHENS

SCHOOL OF NAVAL ARCHITECTURE AND MARINE ENGINEERING

DIVISION OF MARINE ENGINEERING

DIPLOMA THESIS

Analysis of Premixed Flame Data using Machine Learning Methods

Ioannis Stoupas

Supervisors:	**L. Kaiktsis, Professor NTUA**
	Dr. C. E. Frouzakis, ETH Zürich
Examination Committee:	**L. Kaiktsis, Professor NTUA**
	M. Founti, Professor NTUA
	G. Triantafyllou, Professor NTUA

Athens, November 2021

Acknowledgements

This thesis marks the end of my undergraduate studies at the School of Naval Architecture and Marine Engineering of the National Technical University of Athens. Therefore, I would like to seize the opportunity to express my gratitude to all those people who supported me during the past five years.

First of all, I would like to thank from the bottom of my heart my supervisors, Lambros Kaiktsis, Professor at the National Technical University of Athens, and Dr. Christos E. Frouzakis of ETH Zürich, for the excellent collaboration and great assistance in the course of my Diploma Thesis.

Special thanks go to Georgios Rossopoulos, PhD student at the Division of Marine Engineering, for the mentoring, tutoring and integration on the field of Machine Learning. Furthermore, I would like to thank Anastasios Kallieros, graduate of the School of Naval Architecture and Marine Engineering, for his help and offering of programming codes during the first steps of the present work.

Last but not least, I would like to thank my family, my friends and my colleagues for their support and encouragement throughout my five-year studies at the National Technical University of Athens.

Abstract

Chemistry acceleration via Machine Learning methods is a relatively recent area of research, driven by the need for computational cost reduction in Computational Fluid Dynamics (CFD) simulations. The present work deals with the reduction of computational time associated with chemistry integration in combustion simulations. In particular, the target of computational cost reduction is achieved via a neural network approach, promising to replace direct integration in the forthcoming years. Here, methane combustion is studied as an approximation to LNG combustion, which gains importance for the marine industry, in order for vessels to comply with emission regulations.

The methodology proposed in the present study includes two main objectives: (i) clustering of the state space, which is assessed by simulations of methane combustion at representative conditions, and (ii) prediction of thermochemical states. Both objectives are handled by means of Machine Learning methods. The desired subdomain of thermochemical states is covered by data generated via premixed laminar flame simulations. Premixed laminar flames are a "canonical" combustion problem, through which various points in state space can be assessed; thus, they support the applicability of the developed approach to simulate a family of similar problems. In the present study, the premixed laminar flame simulations are performed by means of the CANTERA open-source suite, while Machine Learning methods are implemented by code development in Python.

The first objective is achieved by the development of a hybrid Self-Organizing Map – K-Means neural network, which undertakes the clustering of the state space. This hybrid neural network generates two-dimensional maps for each data feature, which are temperature and species mass fractions for the present implementation, while pressure is kept constant. These maps can be used for correlation of features, quality analysis of state space, as well as for the purposes of chemistry acceleration, such as local mechanism reduction.

Regarding the second objective, Artificial Neural Networks are assigned in each cluster obtained from the hybrid neural network, in order to regress the thermochemical states and simulate the function of the ODE system describing chemistry. Two cases are studied: the first one refers to direct prediction of heat release rate and species net production rates, whereas the second one corresponds to prediction of thermochemical states after a specified time step, given the initial thermochemical state. For both cases, the present approach yields an excellent accuracy. Further, a 21x reduction in computational time is attained regarding chemistry direct integration. Overall, the present approach is very promising, and should be further studied and optimized, with the goal of being coupled with a reactive flow CFD code.

Σύνοψη

Η επιτάγχυνση των υπολογισμών που αφορούν τη χημεία ενός φαινομένου μέσω μεθόδων Μηχανικής Μάθησης αποτελεί μια σχετικά νέα ερευνητική περιοχή, καθοδηγούμενη από την ανάγκη για μείωση του υπολογιστικού κόστους σε προσομοιώσεις Υπολογιστικής Ρευστομηχανικής (CFD). Η παρούσα διπλωματική εργασία πραγματεύεται τη μείωση του υπολογιστικού χρόνου που απαιτείται για τη χρονική ολοκλήρωση της χημείας στο πλαίσιο της προσομοίωσης φαινομένων καύσης. Συγκεκριμένα, ο στόχος της μείωσης του υπολογιστικού κόστους επιτυγχάνεται μέσω ενός μηχανισμού Νευρωνικών Δικτύων, ο οποίος υπόσχεται να αντικαταστήσει την άμεση ολοκλήρωση της χημείας κατά τα επόμενα χρόνια. Στην παρούσα εργασία, μελετάται η καύση μεθανίου, ως μια προσέγγιση της καύσης Υγροποιημένου Φυσικού Αερίου (LNG), η χρήση του οποίου κερδίζει έδαφος στον τομέα της ναυτιλίας, ώστε τα πλοία να συμμορφώνονται με τους κανονισμούς για τις εκπομπές αέριων ρύπων.

Η μεθοδολογία που προτείνεται στην παρούσα εργασία περιλαμβάνει την υλοποίηση δύο κύριων στόχων: (i) ομαδοποίηση του χώρου των θερμοχημικών καταστάσεων, ο οποίος προσεγγίζεται μέσω προσομοιώσεων της καύσης μεθανίου σε αντιπροσωπευτικές συνθήκες, και (ii) πρόβλεψη των θερμοχημικών καταστάσεων. Καί οι δύο στόχοι υλοποιούνται με χρήση μεθόδων Μηχανικής Μάθησης. Ο επιθυμητός χώρος των θερμοχημικών καταστάσεων δημιουργείται από δεδομένα που παράγονται μέσω προσομοιώσεων στρωτής φλόγας προανάμιξης. Η στρωτή φλόγα προανάμιξης είναι ένα πρότυπο πρόβλημα καύσης, μέσω του οποίου προσεγγίζονται διάφορες περιοχές του χώρου των θερμοχημικών καταστάσεων. Κατά αυτόν τον τρόπο, υποστηρίζεται η δυνατότητα της παρούσας μεθοδολογίας να εφαρμοστεί επιτυχώς στην προσομοίωση μιας οικογένειας παρόμοιων προβλημάτων καύσης. Στην παρούσα εργασία, οι προσομοιώσεις στρωτής φλόγας προανάμιξης υλοποιούνται με χρήση της βιβλιοθήκης ανοιχτού λογισμικού CANTERA, ενώ η υλοποίηση των μεθόδων Μηχανικής Μάθησης γίνεται με ανάπτυξη κώδικα σε γλώσσα Python.

Ο πρώτος στόχος επιτυγχάνεται με την ανάπτυξη ενός υβριδικού νευρωνικού δικτύου, με συνδυασμό των μεθόδων Self-Organizing Map και K-Means. Το δίκτυο αυτό είναι υπεύθυνο για την ομαδοποίηση του χώρου των θερμοχημικών καταστάσεων. Δημιουργεί δισδιάστατους χάρτες κατανομής των εξαρτημένων μεταβλητών του προβλήματος καύσης (δεδομένα του δικτύου), οι οποίες είναι η θερμοκρασία και τα κλάσματα μάζας των ενώσεων, ενώ η πίεση διατηρείται σταθερή. Οι χάρτες αυτοί μπορούν να χρησιμοποιηθούν για τη συσχέτιση ιδιοτήτων, για την ποιοτική μελέτη του χώρου των θερμοχημικών καταστάσεων, καθώς επίσης και στο πλαίσιο της επιτάγχυνσης της χημείας, όπως για παράδειγμα στην τοπική μείωση μηχανισμών χημικής κινητικής.

Όσον αφορά στην υλοποίηση του δεύτερου στόχου, Νευρωνικά Δίκτυα συνδέονται σε κάθε ομάδα-σύνολο που προέρχεται από το υβριδικό δίκτυο ομαδοποίησης, με σκοπό να προσεγγίσουν τις θερμοχημικές καταστάσεις και να προσομοιώσουν τη λειτουργία του συστήματος διαφορικών εξισώσεων που περιγράφει τη χημεία. Μελετώνται δύο περιπτώσεις: η πρώτη αφορά στην απευθείας πρόβλεψη του ρυθμού έκλυσης θερμότητας και του ρυθμού παραγωγής ή κατανάλωσης των χημικών ενώσεων που συμμετέχουν στον μηχανισμό, ενώ η δεύτερη πραγματεύεται την πρόβλεψη της θερμοχημικής κατάστασης μετά από ένα δεδομένο χρονικό βήμα ολοκλήρωσης, έχοντας ως δεδομένο εισόδου την αρχική θερμοχημική κατάσταση. Καί για τις δύο περιπτώσεις, η παρούσα μεθοδολογία επιτυγχάνει εξαιρετική ακρίβεια. Επιπροσθέτως, επιτυγχάνεται μείωση του υπολογιστικού χρόνου εκτέλεσης της άμεσης ολοκλήρωσης της χημείας κατά 21 φορές. Συνολικά, η παρούσα μεθοδολογία εμφανίζει σημαντικά πλεονεκτήματα, και ενδείκνυται η περαιτέρω μελέτη και η βελτιστοποίησή της, με στόχο της σύζευξή της με κώδικα Υπολογιστικής Ρευστομηχανικής για την προσομοίωση αντιδρώσας ροής.

CONTENTS

CHAPTER 1: Introduction

1.1 Ship Emissions

The air pollution caused by the shipping industry is a small fraction of the total man-made emissions. According to the International Maritime Organization (IMO), maritime contributes to 2-3% of total anthropogenic CO_2 emissions, 4-10% of total SO_x emissions and 14-31% of total NO_x emissions. Emissions are projected to increase from about 90% to 130% of 2008 emissions by 2050 for a range of plausible long-term economic, energy and maritime development scenarios. Therefore, measures have to be taken in order to limit ship emissions. The main contributors to air pollution due to the shipping industry are sulphur oxides (SO_x) and nitrogen oxides (NO_x) that relate to bunker fuel types and engine operation conditions. SO_x pollutants originate from the low-quality fuel typically used, as the contained sulphur entering the combustion chamber is oxidized forming sulphur oxides. NO_x pollutants are also produced in the combustion chamber from the nitrogen (N_2) and oxygen (O_2) contained in scavenge air by an extremely complex formation process, mainly at high temperature and long residence time.

The International Maritime Organization (IMO) and local authorities have adopted several regulations in order to protect environmentally sensitive sea areas, named as Emission Control Areas (ECAs). Therefore, there must be interventions both for the exhaust gas treatment and regarding the fuel used in the engine in order to comply with the IMO's regulations and regulations within ECAs. Even if a vessel is not calling at any port in an ECA, the vessel must still comply with its requirements when passing through an ECA. There are currently four established ECAs and some others into consideration for future establishment, as seen in Figure 1-1.

According to IMO's MARPOL convention, the four currently established ECAs are:

- Baltic Sea area – as defined in Annex I of MARPOL (SO_x only)
- North Sea area – as defined in Annex V of MARPOL (SO_x only)
- North American area (entered into effect 1 August 2012) – as defined in Appendix VII of Annex VI of MARPOL (SO_x, NO_x and Particular Matter) and
- United States Caribbean Sea area (entered into effect 1 January 2014) – as defined in Appendix VII of Annex VI of MARPOL (SO_x, NO_x and Particular Matter).

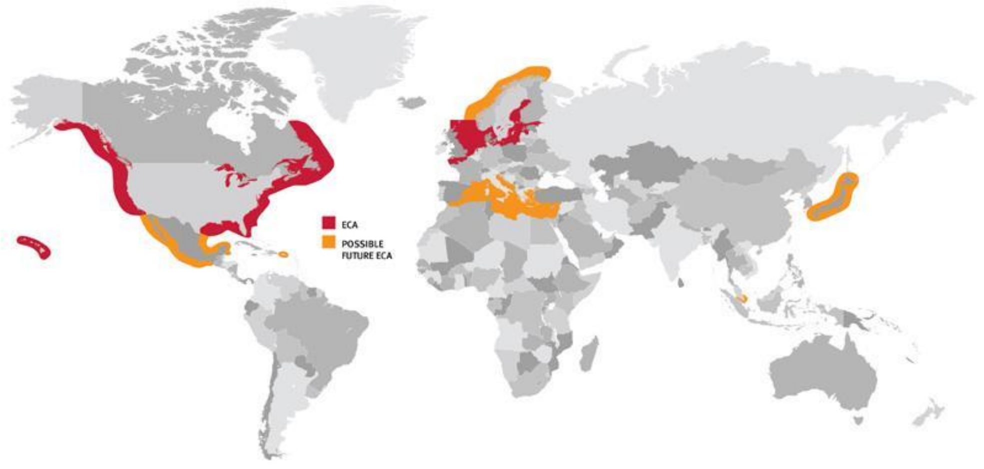


Figure 1-1: Established and possible future Emission Control Areas.

Regarding SO_x emissions reduction, the most realistic methods are the switch to higher-quality fuels, low in sulphur content and generally referred to as lower-sulphur distillates, the usage of exhaust gas cleaning systems, referred to as scrubbers, and the use of LNG. Other solutions, such as use of alternative fuels (ammonia, biofuels, etc.), are also investigated for the near future. Regarding the NO_x emissions reduction, the use of exhaust gas cleaning systems such as selective catalytic reduction (SCR) or exhaust gas recirculation (EGR) are considered to be a necessity for seagoing ships.

Focusing on the SO_x emissions reduction, the switch to lower-sulphur distillates is partially proved as the measure that generally carries the lowest capital costs, as the modifications to the vessel are smaller than in the other cases and lower-sulphur distillate fuels are now in principle available worldwide. The major issue with lower-sulphur distillates has to do with the price at which they are available and the price spread between low-sulphur fuels and heavy fuel oil. Exhaust gas cleaning systems make use of chemical processes to clean the ship exhausts of sulphur up to the regulatory limits. While scrubbers are a well-tested technology on land, their use at sea is not well established yet. Maritime scrubbers are large pieces of equipment and their use often requires detailed technical studies and carries relatively high installation cost, as well as operational and maintenance costs. Exhaust gas cleaning systems for NO_x emissions reduction can be combined with each technology or method used for SO_x emissions reduction, and the choice between them is declared by the designer and the owner.

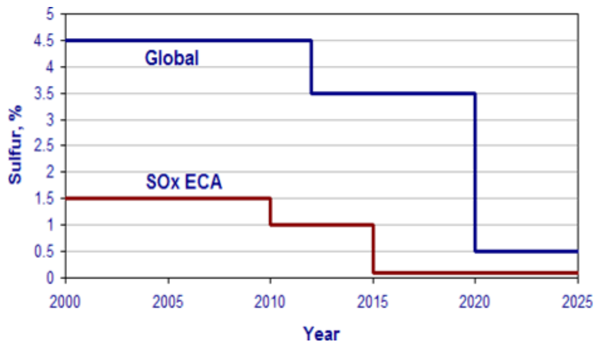


Figure 1-2: IMO limits for reduction of SO_x emissions [1].

Concerning NO_x emissions, IMO has set three Tiers [1], based on the ship construction date and referring to engine certification according to specific NO_x production. The Tier III controls are applied only to ships operating in ECAs established to limit NO_x emissions, while outside such areas the Tier II controls are applied. IMO's target of introducing Tier III is to reduce NO_x emissions inside ECAs to a total rate of 75%, as shown in Figure 1-3.

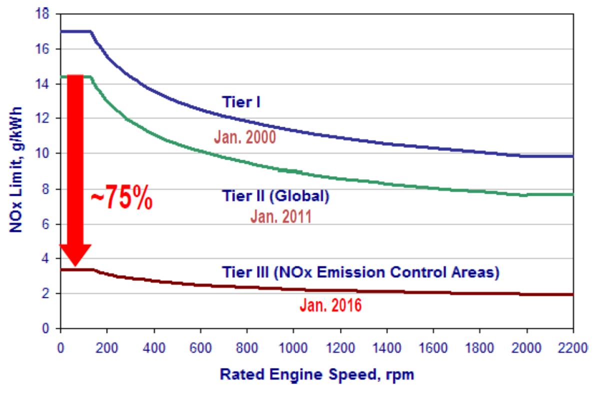


Figure 1-3: IMO Tiers for reduction of NO_x emissions [1].

Carbon dioxide (CO_2) is a main product of perfect hydrocarbon combustion process. However, it is a gas responsible for the greenhouse effect (Greenhouse Gas or GHG). Carbon monoxide is a product of incomplete hydrocarbon combustion and is also considered as GHG. The production of these substances is intensified for fuels with large number of carbon atoms in their molecules, such as marine fuels, which are hydrocarbon mixtures. Reduction of GHG emission can be achieved by optimization of the combustion process or by consumption of hydrocarbons with smaller carbon chains, such as LNG, or alternative fuels, such as ammonia, which is carbon free.

In conclusion, the solutions suggested for the reduction of ship emissions are several and it is an owner's or designer's choice which path to follow. The present work focuses on LNG, a fuel that can provide significant advantages by meeting requirements, offering competitiveness and reducing GHG emissions.

1.2 Liquified Natural Gas

Liquified Natural Gas (LNG) is considered a superior marine fuel providing the best option to improve air quality and is the only marine fossil fuel that advances the greenhouse gas reduction objectives of the shipping industry [2]. According to DNV-GL [3], LNG reduces NO_x emissions by up to 80% and almost eliminates SO_x and Particulate Matter emissions. Furthermore, GHG emissions can be reduced by up to 23% with modern engine technology optimizing the combustion process. Although LNG has several advantages regarding the reduction of ship's emissions, its use as a marine fuel faces difficulties associated with the storage and transportation at low temperatures.

LNG is basically a hydrocarbon mixture of small size carbon chains: methane, ethane, propane and butane. The mixture components depend on its origin and way of extraction. Commonly, the LNG mixture consists up to 98% v/v by methane, thus the present study deals only with the methane combustion process. The global chemical reaction of methane burned in air is:

$$\mathrm{CH_4} + 2\lambda \cdot \left(\mathrm{O_2} + \frac{79}{21}\mathrm{N_2}\right) \rightarrow \mathrm{CO_2} + 2\mathrm{H_2O} + 2\lambda \cdot \frac{79}{21}\mathrm{N_2} + 2(\lambda - 1)\mathrm{O_2}\,, \quad \text{for } \lambda \geq 1$$

where $\lambda=1/\varphi$ is the air-fuel ratio. If $\lambda<1$, corresponding to rich methane-air mixtures, the oxygen is not sufficient to oxidize all available methane into carbon dioxide, thus resulting in a complex reaction mechanism containing carbon monoxide in the products.

1.3 Motivation and Objectives of Present Study

Towards a wide adoption of LNG combustion technology in the marine industry, computational tools for simulating methane combustion, or supporting and optimizing methane combustion simulations should be developed and validated. Methane-air mixture combustion is not described by a one-step global reaction as mentioned above, but by a detailed kinetics mechanism containing a vast number of species and reactions. Therefore, numerical simulations of methane combustion require the solution of numerous differential equations illustrating the reactive flow. In the context of the present study, a neural network mechanism is developed, in order to reduce computational cost of chemistry integration for CFD simulations of methane combustion process, as chemistry term integration accounts for the largest portion of execution time.

CHAPTER 2: Chemistry Acceleration

2.1 Chemistry Acceleration in Computational Fluid Dynamics

Numerical simulations are essential in the study of reactive flows, such as combustion. Reactive flows refer to fluid flows with chemical reactions occurring either within the fluid phase, at the interface between different fluid phases (e.g. spray combustion) or at interphases to solids (e.g. catalytic combustion). To simulate a reactive flow, a Strang-based splitting scheme is often applied to deal with the governing equations, supposing a loose coupling between flow field and chemistry. The decoupling of governing equations [4] can be expressed as:

$$\frac{\mathrm{d}\Phi}{\mathrm{dt}} = \mathrm{M}(\Phi) + \mathrm{S}(\Phi)$$

where **Φ** is the vector of thermochemical composition, specifically containing pressure, temperature and species concentrations, and M and S represent the transport and the chemistry terms, respectively.

The transport term, including convection and diffusion, is described by second-order partial differential equations (PDEs). The chemistry term is described by first-order ordinary differential equations (ODEs) and can be expressed as:

$$\frac{\mathrm{d}\Phi}{\mathrm{dt}} = \mathrm{S}(\Phi)$$

With such a splitting scheme, the integration of the chemical term is separated from transport and at each time step the CFD solver first deals with the transport equations for all computational cells and subsequently integrates the chemical kinetics equations. The calculated transport and thermochemical properties are then evaluated before proceeding to the next time step. An overview of the aforementioned methodology is presented in Figure 2-1, taken from [4]. The bottleneck of reactive flow simulations seems to be the direct integration of chemistry, as it occupies a large portion of total CPU time, approaching rates even of 90%, depending on the complexity of the kinetic mechanism.

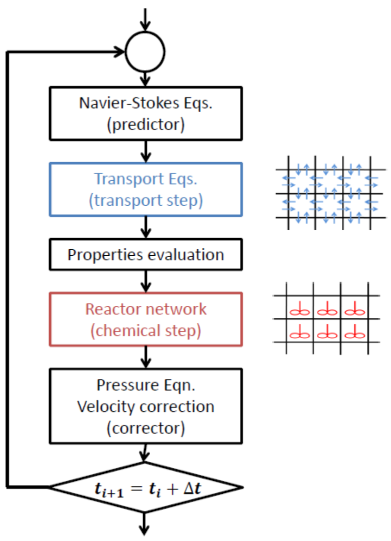

Navier-Stokes Eqs. (predictor)
Transport Eqs. (transport step)
Properties evaluation
Reactor network (chemical step)
Pressure Eqn. Velocity correction (corrector)
$t_{i+1} = t_i + \Delta t$

Figure 2-1: Flowchart during a time step of a reactive flow simulation [4].

Detailed chemical kinetics are necessary for reliable predictions of flames and emissions. However, accurate simulations with detailed chemical mechanisms describing the combustion process typically involve hundreds of species and hundreds to thousands of reactions, hence inducing major challenges even for the most powerful computational facilities. These challenges include the large number of coupled transport equations resulting from coupling a detailed kinetics mechanism with a CFD code, the intrinsic non-linearity of equations because of reaction rates expressions described by power-law and exponential terms, and the stiffness of these ordinary differential equations due to the wide spectrum of characteristic evolution time-scales of the species involved in detailed chemistry. Due to the non-linearity and stiffness of chemistry ODE systems, implicit ODE solvers are mandatory, which however are expensive and their computational cost increases following a power-law with the number of species. Furthermore, the ODE system has to be solved in each computational cell at every time step, hence adding CPU cost of direct integration. Nevertheless, direct integration can be performed for relatively simple fuels. Another category of ODE solvers is stiff integrators, which are more expensive, but allow for longer time steps.

Several strategies for chemistry integration have been developed to overcome these challenges. One should be able to efficiently use large detailed kinetic mechanisms and easily integrate them in existing numerical codes in order to access the desirable rate of accuracy, while retaining the lowest possible computational cost. Some methods dealing with the aforementioned impediments include reduction of chemical complexity, reduction of number of reactive environments and storage/retrieval methodologies for acceleration of ODE integration [4].

Reduction of chemical complexity refers to reduction of chemical mechanisms in order to generate skeletal or reduced mechanisms. A general mechanism reduction can be achieved by removing species and reactions from the initial detailed mechanism, which are considered inconsequential for the description of the phenomenon, and their absence will not significantly decrease the accuracy of results. Other techniques classify the thermochemical states met in a simulation, group the species into clusters and apply a local reduction of chemistry in each respective subdomain.

Dynamic Adaptive Chemistry (DAC) and Sample-Partitioning Adaptive Reduced Chemistry (SPARC) are some prominent techniques used for local reduction of chemistry. The word "adaptive" refers to the fact that no a priori knowledge regarding simulation conditions is required. DAC reduces the detailed mechanisms into sub-mechanisms at each computational cell and at each hydrodynamic step of the calculation, during the simulation (on-the-fly) [5],[6]. On the other hand, SPARC creates a library of reduced mechanisms during a pre-processing step (off-line), covering the thermochemical states expected to be visited. During the CFD simulation, each cell is classified and a proper reduced mechanism from the library is applied in order to construct the describing kinetics equations and subsequently execute the chemistry integration.

Reduction of the number of reactive environments is basically the grouping of cells modeling based on similar thermochemical conditions. This leads to reduction of the total number of ODE systems that need to be integrated, as they are solved for each group of thermochemically similar cells, thus resulting in a lower computational cost. The chemical kinetics equations based on cluster averaged state variables are solved and the cluster averaged solution is then mapped to each individual cell, preserving properties such as temperature and species stratification.

Mechanism reduction where the number of species and reactions is reduced systematically is an important research area, and is usually the first choice regarding chemistry acceleration. Some other interesting techniques aim directly at acceleration of ODE solution, by improving or replacing the solvers. In this category belong storage and retrieval methodologies, which essentially reduce the number of direct ODE integrations through tabulating and re-using solutions. Thermochemistry is precalculated, and the values of thermochemical variables at the end of a specific time are recorded and stored as a function of the thermochemical state at the beginning of the time step. During the simulation, the table is scanned and interpolated in order to calculate and retrieve the desired thermochemical state.

A practical and often used tabulation method is the In Situ Adaptive Tabulation (ISAT). Its main characteristics are the storage of information in a binary tree structure, the restriction of tabulation into only the region accessed by the specific problem ("in situ") and the compromise between accuracy and number of tabulated points ("adaptive") in order to minimize the storage requirements of the high-dimensional state space visited by conventional tabulation methods [6],[7].

Chemistry Agglomeration is also a method used for reduction of time for solving the single ODE systems. Specifically, the chemistry integration is not executed in a single step, but is split in a sequence of ODE integrations of a cluster of species. A clustering algorithm is assigned in each cell of the simulation, grouping together species with strong interactions. Chemical kinetics ODEs are solved in each cluster and afterwards the solutions are coupled together to evaluate the final thermochemical state after the specified time step in each cell [4].

Chemistry acceleration via Machine Learning methods is a relatively new field (see, for example, [8] for an introduction in this technique and the recent work of S. Rigopoulos [10], [11], [17], [18]), thus not yet mature, with significant potential benefits to be further explored. Machine Learning methods, such as unsupervised clustering and deep learning networks essentially replace the standard ODE solver for the execution of chemistry integration. The available variety of Machine Learning methods and their intrinsic flexibility to sufficiently deal with non-linear, multi-dimensional characteristics makes them a notably useful tool for reactive flow simulations.

Artificial Neural Networks (ANNs) are an evolution of Machine Learning, most of them falling into the category of Deep Learning. Further examination of this terminology is presented in *Chapter 3: Machine Learning* of the present study. The application of Artificial Neural Networks for chemistry acceleration and the substitution of conventional ODE solvers is a procedure consisting of four steps:

1. The first step is data generation, which will be used for training the ANN. The training dataset is obtained either from previous simulations of the same combustion problem or via a simpler "canonical" problem.
2. The next step is data preprocessing in order to filter and remove outliers. Usually, the ANN which will be applied is sensitive to feature scaling. Thus, pre-scaled dataset may also result in more robust predictions.
3. Subsequently, the ANN type and architecture is defined. ANN type refers to how the information flows in the network, while ANN architecture is a hyperparameter to be defined and is selected during a trial-and-error process targeting a compromise between accuracy and computational cost.
4. Finally, the ANN is trained, meaning that its weights are adjusted in such a way that it sufficiently learns and predicts patterns of the input data. The trained ANN is then stored and called during the simulation.

In order to achieve satisfactory accuracy, multiple or more complex ANNs can be exploited, at the expense of higher computational cost. A solution to this problem can be found by clustering the state space. In combustion simulations, it is possible to identify several zones where the rate of change of the relevant variables may differ widely. By applying proper unsupervised clustering algorithms, the state space can be divided into subdomains that contain thermochemical conditions close to each other. An ANN-regressor is then assigned to each cluster, reducing the complexity and the computational cost of the previous proportion. Consequently, in each case, the selection of ANN configuration is based on the problem which will be modeled, as well as its complexity, and the researcher decides on the compromise between accuracy and computational cost.

An overview of the aforementioned methods for chemistry acceleration is presented in the following diagram (Figure 2-2), also taken from [4].

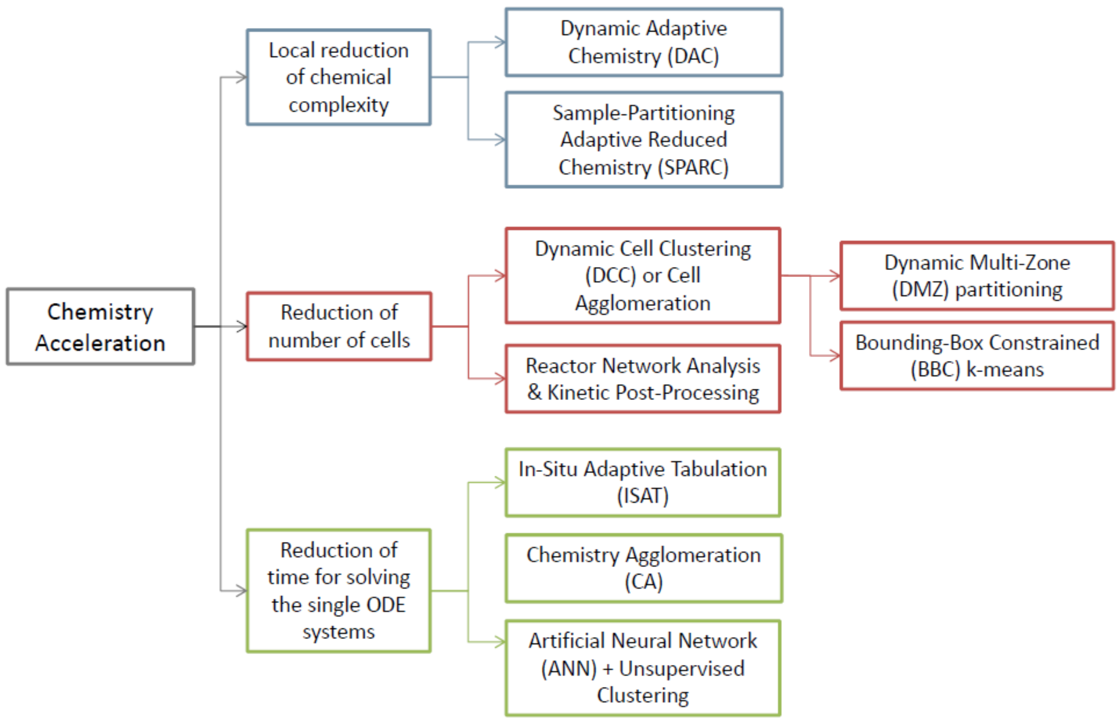

Chemistry Acceleration
Local reduction of chemical complexity
Dynamic Adaptive Chemistry (DAC)
Sample-Partitioning Adaptive Reduced Chemistry (SPARC)
Reduction of number of cells
Dynamic Cell Clustering (DCC) or Cell Agglomeration
Reactor Network Analysis & Kinetic Post-Processing
Dynamic Multi-Zone (DMZ) partitioning
Bounding-Box Constrained (BBC) k-means
Reduction of time for solving the single ODE systems
In-Situ Adaptive Tabulation (ISAT)
Chemistry Agglomeration (CA)
Artificial Neural Network (ANN) + Unsupervised Clustering

Figure 2-2: An overview of chemistry acceleration techniques in CFD simulations [4].

The present study focuses on chemistry acceleration via Machine Learning methods. Both unsupervised clustering and deep learning neural networks are studied and evaluated in terms of ease of use, accuracy and computational cost reduction for chemistry integration.

2.2 Literature Review

Numerous studies have been performed on the topic of adaptive chemistry and tabulation of combustion via Artificial Neural Networks. Several alternative techniques can be found in the literature associated with avoiding the numerical integration of stiff thermochemical equations during the simulation of combustion processes, recently by applying neural network mechanisms. The present study focuses on the Self-Organizing Map – Multi-Layer Perceptron (SOM-MLP) concept, firstly introduced by *Blasco et al.* [8] (to the author's knowledge). According to this implementation, the SOM is employed for the unsupervised partitioning of the state space into subdomains, while a specialized MLP is assigned to fit the thermochemical states belonging to each subdomain. The concept was evaluated on a constant pressure partially stirred reactor (PaSR) with a reduced methane-air mechanism derived from GRI 2.11 [56].

A recent work was conducted by *An et al.* [9] on the SOM-MLP concept for acceleration of chemistry calculation. Compared to previous studies, the chemical reaction mechanism in this study is a complex though skeletal mechanism, containing 41 species and 132 reactions for hydrogen / carbon monoxide / kerosene supersonic combustion. The training data are generated by RANS simulations of a rocket-based combined cycle (RBCC) combustion chamber, where each cell is modeled as a PaSR. A splitting scheme is applied in order to solve separately the chemical and the transport term. The proposed ANN-based mechanisms are validated through LES simulations of the RBCC combustor.

Franke et al. [10] studied tabulation of methane turbulent combustion by ANNs trained via a canonical problem. The state space is clustered into subdomains using the SOM and an MLP is assigned to each subdomain for regression and time integration of the kinetics. The training data are generated through simulations of an ensemble of igniting/extinguishing laminar flamelets to anticipate the compositions visited in a typical simulation of a non-premixed turbulent flame. The approach is employed to tabulate a Rate Controlled Constrained Equilibrium (RCCE) - reduced mechanism based on GRI 1.2 [56]. The methodology is then verified through LES-PDF simulations of Sydney flame L.

A previous study in this context was performed by *Chatzopoulos* and *Rigopoulos* [11], who proposed a similar SOM-MLP approach for the chemistry tabulation and prediction of temporal evolution of two non-premixed and non-piloted $CH_4/H_2/N_2$ turbulent flames (DLR-A and DLR-B). The training data are also generated through simulations of a canonical problem, specifically non-premixed laminar flames, by recording the compositions of RCCE leading species. The methodology is then evaluated via RANS-PDF simulations.

A work with many similarities to the current study is presented in *Chi et al.* [12]. They developed ANNs for the tabulation of chemical reaction terms in DNS of both igniting and established premixed methane/air flames. The flame structure is divided into four clusters based on the chemistry evolving locally in each of them, after executing trial-and-error tests. The fact that the whole training process takes place on-the-fly, so that the obtained ANN is properly adapted to the specific configuration considered, is a point to be emphasized that differentiates this approach from the current study. An ANN is assigned to each cluster and it is fitted properly during the simulations in order to predict the mass production rate of each species. The detailed kinetic mechanism GRI 3.0 is used for the simulations [56], which are performed by a DNS solver coupled with CANTERA.

All aforementioned studies focus on the SOM-MLP concept. Specifically, the state space is a priori clustered with an unsupervised machine learning method and an MLP regressor is assigned to each one of the obtained clusters in order to predict either the thermochemical state after a time step or species' production rates. Other related studies deal with the partitioning of state space for adaptive chemistry applications, such as generation of reduced mechanisms, while others consider the direct assignment of ANNs without previous clustering of input space.

D' Alessio et al. [13], [14] deal with the so-called Sample-Partitioning Adaptive Reduced Chemistry (SPARC) methodology, which basically is state space partitioning for generation of locally reduced mechanisms. A 2D unsteady co-flow laminar diffusion flame of methane – nitrogen mix is studied and a respective detailed mechanism is used. The clustering methods proposed are Local Principal Analysis (LPCA) and a hybrid SOM – K-Means which are compared in terms of accuracy and speed up of adaptive chemistry simulations. An interesting point drawn from this implementation is the efficient application of K-Means method on SOM feature maps in order to group the closest neurons into clusters.

Zhang et al. [15] developed a binary ANN structure in order to consider the drastic changes of the species vector gradient in low-temperature chemistry and high-temperature chemistry. The network architecture is vast and complex, including a large number of neurons per hidden layer, in order to model the homogenous autoignition of dimethyl-ether / air mixture with only two ANNs. Each ANN executes the time integration for a state vector, containing temperature, pressure and species mass fractions. A recently developed neural network architecture consists of sub-networks, each responsible for the prediction of specific features, without pre-partitioning of state space. For each case, the relative thermochemical state vector is used as network input.

In this framework, *Sharma et al.* [16] attempted with success to evolve the chemical source term, after a splitting operation. They developed 12 MLPs in total, one for temperature, one for pressure and the remaining 10 for each species mass fraction. Their model is demonstrated with a hydrogen-oxygen reaction, while the simulations are performed in a perfectly stirred reactor (PSR). An important notation from this work shows that data generated from CANTERA simulations are sparse at low temperatures and, hence, the trained ANNs may lead to lower prediction accuracy.

Departing from his previous works, the research team of Rigopoulos has been involved in various recent studies proposing separate ANNs for each species over the entire training dataset, without previous partitioning of state space. *Readshaw et al.* [17] focused on the construction of ANNs aiming at generalization rather than specialization, in order to improve performance by obtaining training data from a canonical combustion problem of one-dimensional laminar flamelets [11] and then augmenting the training dataset with randomly generated data to better cover the state space. Several ANN architectures combining SOM and MLPs were considered and compared. The ANNs are applied to the tabulation of the GRI 1.2 mechanism [56], containing 32 species and 177 elementary reactions. The MLP error analysis shows that the assignment of one MLP per species outperforms the SOM-MLP concept of pre-clustering the state space in terms of generalization. Furthermore, it is shown that the most meaningful clusters are biased to the major variables, while the minor species play a less important role. The most efficient architecture seems to be the one with separate MLPs for the prediction of concentration change of each considered species and no previous partitioning of the state space, but it requires a slightly higher computation time than the SOM-MLP concept (one MLP attached to each SOM subdomain and integrating all the features at once). The latter is validated with LES-PDF simulations on the Sydney Flame L.

An interesting note extracted from this study is that proper clustering of the Self-Organizing Map seems favorable for the ANN precision and performance, in the case of separate MLPs dealing with each considered species. By decreasing the number of SOM subdomains, the mean MLP error is generally improved, even though poorer performance is observed for minor species.

Ding et al. [18] continued the above implementation [17] on tabulation of the GRI 1.2 mechanism [56], in order to enhance the capacity for generalization and to improve the accuracy of ANN prediction. Another method for random data generation is proposed, endowing the generalization ability of ANN, while multiple MLPs are applied to states featuring different dynamics, retaining the concentration change of each species as the target output. MLP error analysis emphasizes that when using only a single MLP responsible for a specific species over the whole dataset may result in similar absolute errors for all predictions, but this leads to large relative errors for outputs with small magnitude. The only sufficient solution seems to be the training of separate MLPs depending on the output range of concentration change. In other words, the multiple MLP approach divides the domain based on the magnitude of concentration change, while the SOM-MLP concept divides the domain via unsupervised clustering. The HFRD – MMP (hybrid flamelet/random data – multiple multilayer perceptron) approach can be efficiently applied in a variety of chemical kinetics simulations.

Finally, a recent different approach by *Owoyele* and *Pal* [19] presents the application of Neural Ordinary Differential Equations (NODEs) for chemical kinetics solvers. Instead of training the neural networks on thermochemical states of the system, their weights are adjusted accordingly to minimize the difference between the predicted and the actual ODE solutions, in order to reduce the propagating error during time integration.

2.3 Thesis Structure

In the present thesis, an artificial neural network approach is developed for reducing the computational time required for the integration of chemistry ODEs in CFD simulations. The thesis structure is outlined below.

Chapter 3 introduces the Machine Learning methods, and focuses on the ones applied in the present study. The theoretical background of each method is examined, and techniques regarding the network hyperparameters selection are presented. Additionally, Chapter 3 describes the way of coupling the selected methods in order to develop the final hybrid SOM – K-Means neural network, which undertakes the state space clustering.

Chapter 4 serves as an introduction to basic Chemical Kinetics theory. The ODE system describing the chemistry term in CFD simulations is generated via the Law of Mass Action and Principle of Energy Conservation. The canonical combustion problems of premixed laminar flame propagation and homogeneous reactor time evolution are analyzed. Finally, the simulations performed in order to generate training data for the neural networks are described in detail.

Chapter 5 examines the development of the hybrid SOM – K-Means neural network responsible for clustering the state space. The sample generation process is presented, and both training processes of SOM and K-Means neural networks are described. The final results are physically evaluated through observations on distribution maps, created by the neural network.

Chapter 6 refers to the first ANN developed in the present study for the purposes of state space regression. It receives a composition state vector as an input and the target outputs are the heat release rate and the species net production rates. The sample generation process is described, and the selection of the network hyperparameters and the training process are presented. Finally, the prediction results for each ANN developed are assessed.

Chapter 7 refers to the second ANN developed in the present study for the purposes of state space regression. It receives a composition state vector as an input and the target output is the composition state vector after a specified time step. In other words, it executes the time integration of chemistry. Similarly, the sample generation process is described, and the selection of network hyperparameters and the training process are presented. Finally, the prediction results for each developed ANN are assessed.

Chapter 8 concludes the present work by summarizing the observations and results of this study. Finally, suggestions for future work, continuing the current thesis, are provided.

CHAPTER 3: Machine Learning

3.1 Introduction to Machine Learning

Artificial Intelligence (AI) was inspired by characteristics of brain function and refers to the simulation of human intelligence processed by computer systems. The core characteristic of AI is its ability to rationalize and take actions that have the best chance of achieving a specific goal.

Machine Learning (ML) is a subset of AI and refers to computer algorithms that execute tasks without given specific instructions, but by learning and improving automatically through experience. An algorithm is considered to learn through experience regarding a specific task when its performance on the task is continuously improving. The three main categories of ML are:

1. Supervised Learning: The algorithm receives input and target data in order to learn the rule that best matches the inputs with the results. Requires the human to define the labels first. It includes classification and regression methods.
2. Unsupervised Learning: The algorithm must itself find patterns in input in order to create a structure predicting the outputs from the inputs. It includes clustering and dimensionality reduction methods.
3. Reinforcement Learning: The algorithm interacts with a dynamic environment, receiving rewards for good “responses” and “punishments” for bad ones.

An evolution of ML is Deep Learning (DL), inspired by information processing and distributed communication nodes in biological systems. The adjective “deep” refers to the use of multiple layers in the network, connected together like a web. Its main intrinsic abilities are handling of large data and enabling machines to process data in a non-linear approach.

Artificial Neural Networks (ANNs) is a DL method. They are computational systems inspired by the function of biological neural networks of the human brain. They do not require prior fundamental understanding of processes or phenomena, as they are able to identify and learn correlated patterns between sets of inputs and targets. They are also able to predict outcomes from new unseen input data, if properly trained [23]. Their architecture consists of synapsed neurons in a set of layers. Each connection is assigned a weight that is adjusted during the training process. The two main categories of ANNs are:

- Feed–Forward Neural Networks: information flows only forward.
- Recurrent Neural Networks: information flows both forward and backward, due to loops created by connections between neurons.

According to *Cybenko’s* universal approximation theorem, any linear or non-linear function can be approximated by a sufficiently deep ANN with an adequate number of hidden layers and neurons per layer [24]. The generality of universal approximators, however, comes with the price that each input affects the output in some complex and intransparemt way, rendering interpretability of the fitted model difficult [25]. In general, for a network to be able to generalize, it should have fewer parameters than the data points in the training set. In neural networks, one wants to use the simplest network that can adequately represent the training set [26].

The Machine Learning methods applied in this study are Principal Component Analysis for dimensionality reduction, Self-Organizing Maps and K-Means for clustering the input dataset, and Multi-Layer Perceptrons for regression and prediction, which are a class of Feed–Forward Neural Networks, composed of multiple layers of perceptrons with threshold activation. Further description of these methods is presented in the following paragraphs.

3.2 Neural Network for Clustering

A hybrid neural network is developed for the purposes of clustering the input data space for the current implementation. First, Principal Component Analysis (PCA) is applied for dimensionality reduction and selection of initial values for the neurons' weights. The main clustering method employed is Self-Organizing Maps (SOMs), which are able to deal with high dimensional data and create distribution maps for each feature. For certain applications, the number of clusters obtained by SOMs may be higher than the desirable. Therefore, subsequently, the K-Means clustering method is coupled and executed on SOM neurons in order to group them. All of these ML techniques belong to unsupervised methods, because the algorithm itself finds patterns and develops a proper structure.

3.2.1 Principal Component Analysis

Principal Component Analysis (PCA) is a statistical dimensionality reduction technique, in which a number of related variables are transformed to a smaller set of uncorrelated variables. It is used to decompose a multivariate dataset in a set of successive orthogonal components, that explain a maximum amount of the variance. It is noted that reducing the number of variables of a dataset naturally comes at the expense of accuracy, but a trade between accuracy and simplicity is finally achieved [27],[28].

A standardization method usually precedes the Principal Component Analysis, so that each of the input variables contributes equally to the analysis, and possible outliers are detected and removed. Let X be a matrix containing the observations, where each observation is described by *n* variables. The PCA algorithm for matrix X consists of 3 steps:

Step 1: Compute the dataset covariance matrix. The covariance matrix is a *nxn* symmetric matrix that has as entries the covariances associated with all possible pairs of the input variables. By definition, the main diagonal of the covariance matrix includes the variances of the *n* input variables. Furthermore, since the covariance formula is commutative (Cov(a,b) = Cov(b,a)), the upper and the lower triangular portions of the covariance matrix are symmetric. The calculation formula is:

$$S = \frac{1}{n-1} \cdot X^T \cdot X$$

Step 2: Decompose the covariance matrix into:

$$S = A \cdot L \cdot A^T$$

where:

- L is a diagonal matrix containing the eigenvalues of the covariance matrix S.
- A is a *nxn* matrix containing the eigenvectors of the covariance matrix S. These eigenvectors are called Principal Components (PCs), they are uncorrelated and constructed as linear combinations of the input variables. PCA assigns the largest possible variance (maximum possible information) as the first component, the maximum remaining variance in the second and so on (Figure 3-1). The variance is quantified by the eigenvalues λ_i contained in the diagonal matrix L, and the percentage of variance contained in each PC is $\lambda_i / \sum_i \lambda_i$.

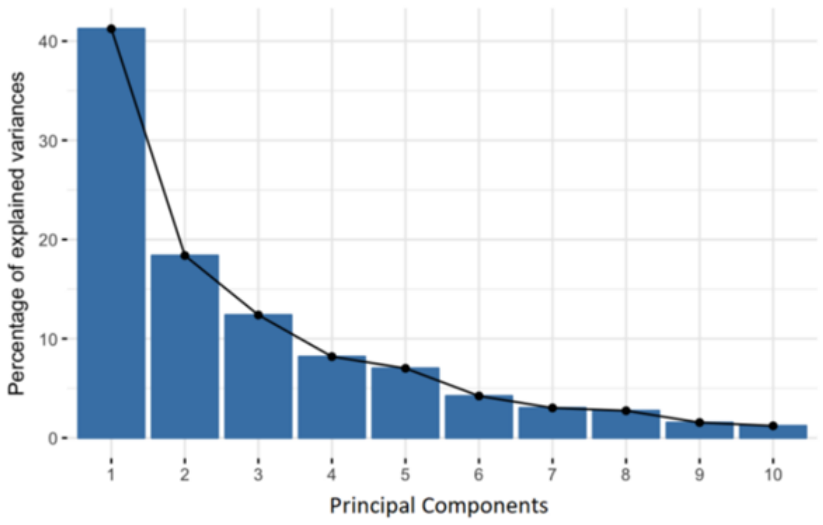


Figure 3-1: Example of percentage of variance (information) contained by each Principal Component [29].

Step 3: Project the dataset matrix X onto a lower-dimension space, on the basis of principal components. Dimensionality reduction comes by considering only a subset of PCs, depending on the amount of variance accounted for. The most energetic eigenvectors are assigned as columns of a non-square matrix A_q, and the projection Z_q of the dataset matrix X follows the formula:

$$Z_q = X \cdot A_q$$

Subsequently, the reconstruction of the original dataset in a q-dimensional space can be conducted by inverting the above equation:

$$X_q = Z_q \cdot A_q^T$$

It is important to note that the PCs are less interpretable and many times do not have any real meaning. Geometrically speaking, they represent the directions of the data that explain a maximum amount of variance (information). The projection of dataset by PCA is depicted in Figure 3-2.

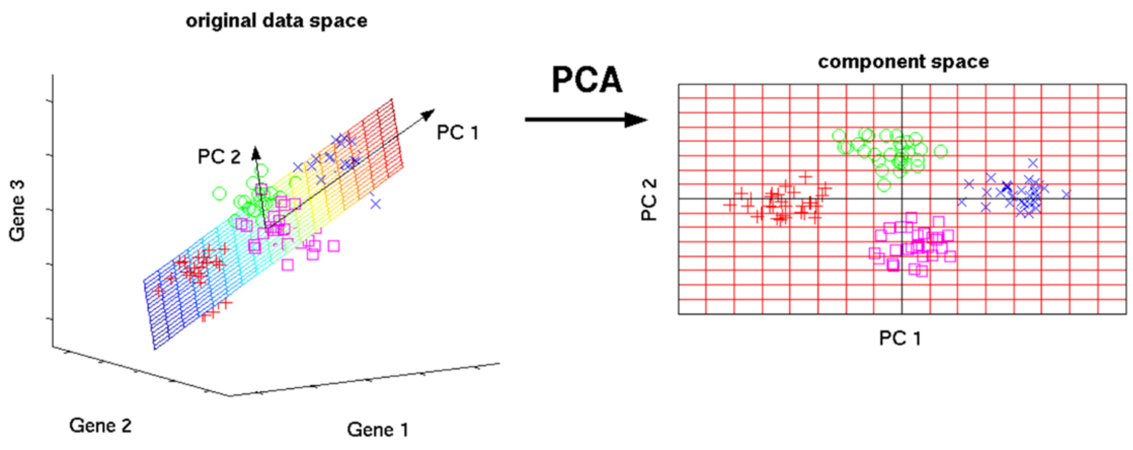


Figure 3-2: Dataset projection of a 3-dimensional space into two PCs by means of Principal Component Analysis.

Each PC is associated with a certain percentage of variance by the magnitude of its associated eigenvalue. Therefore, it is useful to measure the global variance explained by a selected subset of PCs by defining the ratio:

$$t_q = \frac{\sum_{i=1}^{q} \lambda_i}{\sum_{i=1}^{p} \lambda_i}$$

where q is the chosen number of PCs, p is the total number of PCs and λ_i is the ith eigenvalue obtained from the decomposition of covariance matrix S. The largest the ratio, the highest the amount of information described by the selected number of PCs [30].

Principal Component Analysis is a linear transformation, while combustion is a highly non-linear phenomenon. Its intrinsic linearity represents a major problem of the technique to deal with non-linear processes in a proper way. Previous studies have proposed Local Principal Component Analysis (LPCA), which essentially is performing PCA in each cluster of an already partitioned dataset space [13],[14]. In the present study, PCA is used to initialize the neurons' weights of the Self-Organizing Map neural network, by decomposing the dataset into the first two Principal Components.

3.2.2 Self-Organizing Map

Among the architectures and algorithms suggested for artificial neural networks for clustering, the Self-Organizing Map (SOM) has the special property of effectively creating spatially organized internal representations of various features of input observations and their abstractions [31]. Based on brain behavior under visual and memory stimulation, SOMs are a type of neural network which use competitive unsupervised training, as proposed by *Kohonen* [31],[32]. Since their introduction, they have been applied in many research fields, mainly because of their intrinsic ability to learn from high-dimensional input data, resulting in a low-dimensional (usually bi-dimensional) output layer.

Most applications of SOMs are based on regular two-dimensional grids. The map structure consists of neurons, capable of presenting high-dimensional data received as low-dimensional data on a two-dimensional array. It is important to notice that each map neuron is connected to each input manifold, and map neurons are not connected to each other [33]. The neurons are basically vectors, having weights of the same dimensions as input data and acquired during the training progress (Figure 3-3).

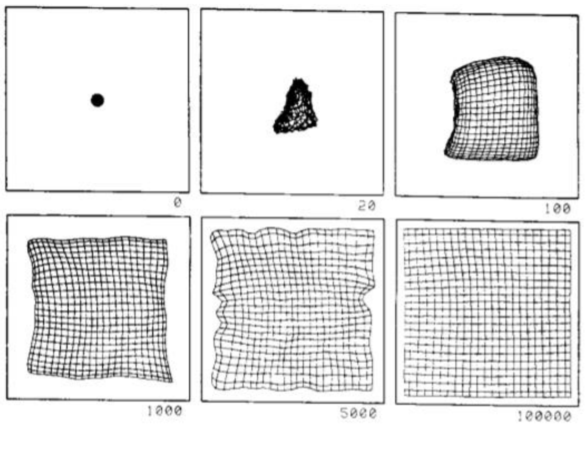


Figure 3-3: SOM grid adaptation through training process, on a two-dimensional array [31].

The learning progress of SOM consists of three stages; competition, cooperation and synaptic update. The competition phase is responsible for identifying the map region which is most activated by a specific input data. Such a region can be defined as the neighborhood of the most activated neuron, the so-called winning neuron. In the cooperation phase, the influence of the winning neuron on its neighborhood is defined. Finally, in the synaptic phase, the winning neuron updates its weight value, leading to new and similar input data to activate the neuron again. At the same time, neurons belonging to the neighborhood also change their weight values to a rate prescribed from SOM hyperparameters. This iterative progress is called mesh adjustment, and results in the representation of the feature space on low-dimensional grids. The learning progress algorithm is iterative for each input data vector *X* and consists of the following steps:

<u>Step 1</u>: Initialize the weight values for each neuron. The most common method is random initialization, but in this case, the training progress may not converge as fast as desired. A more robust technique is drawing a random fold of input data and setting the initial weight values accordingly.

<u>Step 2</u>: Determine the winning neuron, which is the neuron with the minimum Euclidean distance from the input data vector *X* in the high-dimensional space. Let m_i be the position of neuron i and m_c the winning neuron position, which is basically a vector containing the weights for each dimension. The Euclidean distance between a n-dimensional input data vector *X* and a random n-dimensional neuron m_i is defined as:

$$\mathrm{d} = \sqrt{||\mathrm{X}_1 - \mathrm{m}_{\mathrm{i},1}||^2 + ||\mathrm{X}_2 - \mathrm{m}_{\mathrm{i},2}||^2 + \cdots + ||\mathrm{X}_\mathrm{n} - \mathrm{m}_{\mathrm{i},\mathrm{n}}||^2}$$

SOM seeks the smallest possible distance in order to define the winning neuron m_c:

$$\mathrm{d} = \mathrm{argmin}\{||\mathrm{X} - \mathrm{m}_\mathrm{i}||\}\ ,\ \ \forall \mathrm{i}$$

<u>Step 3</u>: Update the weight values (or position respectively) of the winning neuron, as well as neurons included in its neighborhood by:

$$\mathrm{m}_\mathrm{i}(\mathrm{t}+1) = \mathrm{m}_\mathrm{i}(\mathrm{t}) + \mathrm{h}_{\mathrm{ci}}(\mathrm{t}) \cdot [\mathrm{X}(\mathrm{t}) - \mathrm{m}_\mathrm{i}(\mathrm{t})]$$

where:

- t is the iteration step,
- m_i(t) is the position of any of the SOM's neurons (i is the spatial index of the grid) at the iteration step t, and
- h_{ci}(t) is the neighborhood function. This function resembles the kernel that is applied in usual smoothing processes and its use aims to the determination of which and to what degree neurons will update their weight values. The subscript c refers to the winning neuron in the grid, namely the one with the position m_c, implying that the function is defined around this neuron. As a general approach, it is recommended that the closest neurons to the winning neuron update their weight values to a high degree, while the furthest neurons should be affected the minimum. *Kohonen* [32] suggested the so-called bell curve, multiplied with a monotonically decreasing scalar function, as a neighborhood function. A commonly used bell curve, also applied in the current study, is the Gaussian distribution, given by the formula:

$$\mathrm{h}_{\mathrm{ci}}(\mathrm{t}) = \mathrm{a}(\mathrm{t}) \cdot \frac{1}{\sqrt{2\pi \cdot \sigma^2(\mathrm{t})}} \cdot \exp\left\{-\frac{||\mathrm{m}_\mathrm{c}(\mathrm{t}) - \mathrm{m}_\mathrm{i}(\mathrm{t})||^2}{2\sigma^2(\mathrm{t})}\right\}$$

where σ(t) is a monotonically decreasing standard deviation function, the so-called neighborhood radius, and a(t) is also a monotonically decreasing function, named learning rate. The justification on the choice of monotonically decreasing functions lies on the robust convergence of the training process and the reduction of weight modification as the iteration index increases. An illustration of the above features is represented on Figure 3-4.

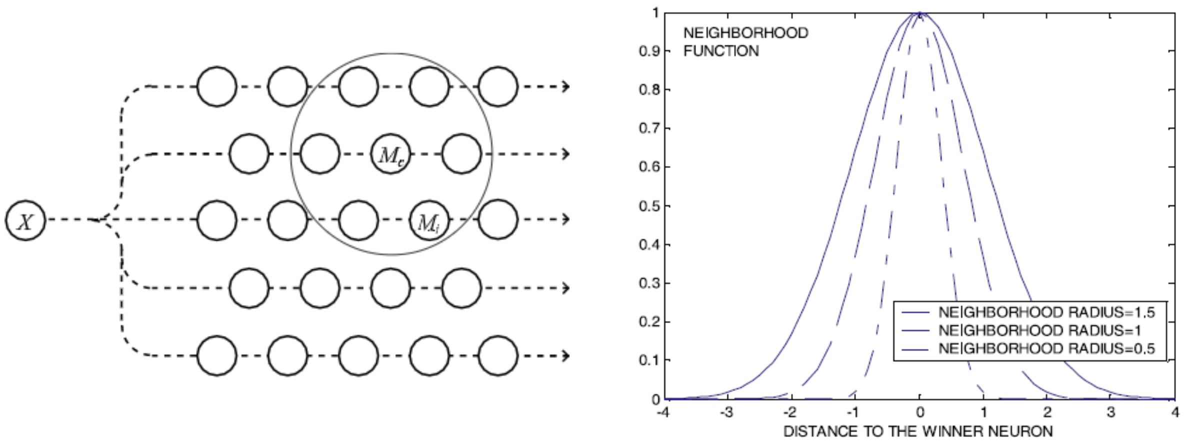


Figure 3-4: Determination of winning neuron's neighborhood (left) and definition of neighborhood radius (right) [34].

The SOM algorithm assumes that the aforementioned process converges and produces the desired ordered values for the models after a specified number of iterations. During the training, the SOM forms an elastic net that enfolds the cluster of the input data (Figure 3-5). The learning process finishes when either the mesh update is smaller than a pre-defined threshold, or the maximum number of iterations is reached. At this point, convergence has been achieved and the mesh can represent the feature space in low-dimensional grids. Each SOM neuron's location is fixed and each neuron represents the center of a cluster. Therefore, new input data can be classified by the nearest neuron (winning neuron), forming a set of clusters on the low-dimensional grid.

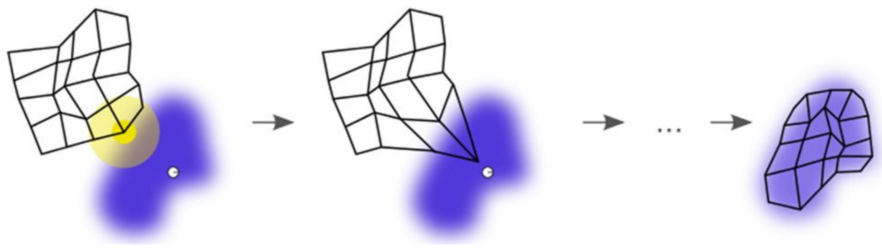


Figure 3-5: Training process of SOM (black grid) on a dataset cluster (blue color). At any iteration, an input data (white point) excites a specific neighborhood (yellow color) and the SOM adapts properly its nodes, until convergence is achieved (right illustration).

From the above description, an important and useful property of SOMs derives: neurons are connected to adjacent ones by a neighborhood relation. Therefore, data points lying near each other in the input space are mapped onto nearby winning neurons. In other words, SOM preserves the input space topology [34]. The SOM quality evaluation is presented in a following paragraph, but first an effective algorithm for speeding up the training process and the convergence of the network is discussed.

Batch Training Process

A way of speeding up the training process is by using the batch computation algorithm. Let *{X}* be a set of input data vectors and N_i be a neighborhood set of nodes. At the first stage of this algorithm, each input data vector *X* is compared and assigned to a sublist, which is associated with its winning neuron. Afterwards, the mean of *X* over neighborhood N_i is computed, which is basically the mean of all vectors *X* that have been assigned into the union of all sublists in the neighborhood N_i. A similar computation is executed above all winning neurons and their neighborhoods. The final step consists of updating each neuron weights m_i by their respective means, in one concurrent computing operation over all nodes of the grid. This concludes an updating cycle, instead of using only one input data vector *X* at each iteration t.

Quality Measures

It is essential to evaluate the quality of the resulting maps in order to ensure that the model is indeed representative of the underlying data. Various measures have been proposed that quantify the map performance on preservation of topology and neighborhood [35], as well as indicating the training process convergence. *Kohonen* [31] proposes the quantization error (Q_{error}) as a measure of the average distance between data points and their respectively winning neurons. Another common quality measure is topographic error (T_{error}), which is a measure of how well the structure of the high-dimensional input space is modeled by the low-dimensional grids of SOM.

Let *X* be a specific input data and m_c be the respectively winning neuron. For *N* total observations, the mathematical formula of quantization error is:

$$Q_{error} = \frac{1}{N}\sum_{i=1}^{N} ||X - m_c||$$

It is important to note that quantization error is on the same scale as the input data, and hence shall only be used to compare and evaluate maps to each other rather than as a standalone assessment of quality. In addition, it accounts for the relationship between the input data and the winning neurons and not for the topographical organization of neurons. In other words, quantization error only evaluates the local data structure.

For a more proper evaluation of SOM topography, topographical error shall be used. Let m_c be the winning neuron (best matching unit – BMU) of an input data and m_c' be the second best matching unit in the map. If the neurons are next to each other, then topology is supposed to have been preserved, else this is counted as an error. The formula to compute the topographical error is:

$$T_{error} = \frac{1}{N}\sum_{i=1}^{N} f(t)$$

$$f(t) = \begin{Bmatrix} 0, & \text{if } m_c \text{ is next to } m_c' \\ 1, & \text{otherwise} \end{Bmatrix}$$

where t is the iteration index. By definition, smaller values of Q_{error} and T_{error} indicate a better fit and a satisfactory preservation of input space topology.

In this study, Self-Organizing Map neural network is applied as the basic clustering algorithm for the input data. The input component space is high dimensional and a random initialization of

SOM may not lead to convergence of the training process. Therefore, Principal Component Analysis precedes, in order to span the high-dimensional input data into its first two Principal Components, and then randomly select bi-dimensional data weight values for the nodes of SOM. Before the start of the SOM training process, the inverse transformation is applied in order to compound the initial weight values back to the input high-dimensional space. This method is called linear initialization and is expected to speed up the algorithm by orders of magnitude [32].

The generation of SOM is performed so that the reference map can subdivide the high-dimensional input component space according to similarities in the data. Therefore, it surely results in a large number of clusters spanned on two-dimensional maps. For certain applications, the number of clusters obtained through SOM may be much higher. To reduce the number of clusters, the K-Means algorithm offers an ideal method in order to group the SOM neurons into a desirable number of clusters.

3.2.3 K-Means

The K-Means algorithm is an unsupervised Machine Learning method used to identify clusters in a dataset. It belongs to partitional clustering methods, as it divides data in non-overlapping groups. In other words, no object can be a member of more than one cluster, and every cluster must have at least one object [37].

The core idea of the K-Means algorithm is to update the cluster center which is represented by the center of data points, by iterative computation and the iterative progress will be continued until a specified convergence criterion is met [38]. More specifically, this algorithm consists of the following steps [39]:

Step 1: Select the number of clusters (groups) and randomly initialize their respective center points. The center points are vectors of the same length as each data point vector. Let *k* be the predefined number of clusters.

Step 2: Compute the distance between each data point and each cluster center and then assign each data point to the cluster whose center is closest to it.

Step 3: Recompute the center of each cluster as the mean of all the vectors assigned in the cluster. As a result, the cluster center is shifted by minimizing an error function, defined as the sum of squared Euclidean distances between the set of observations belonging to the cluster and its centroid:

$$\mathrm{E} = \sum_{\mathrm{j}=1}^{\mathrm{k}} \sum_{\mathrm{x_i} \in \mathrm{C_j}} \left|\left|\mathrm{x_i} - \mathrm{m_j}\right|\right|^2$$

where x_i is the ith component of the multi-dimensional data vector, C_j is the specific cluster and m_j is the cluster centroid, for j=1,…,*k*, where *k* is the predefined number of clusters. The minimization is accomplished over all *k* clusters.

Step 4: Repeat the above steps for a fixed number of iterations or until convergence is achieved. Convergence is defined when the positions of cluster centers do not change between two consecutive iterations more than a user-defined threshold.

Both the initialization of K-Means algorithm during the first iteration and the clustering evaluation at final iteration are represented on Figure 3-6. Initial and final cluster centers are noted with red circles to make them more visible.

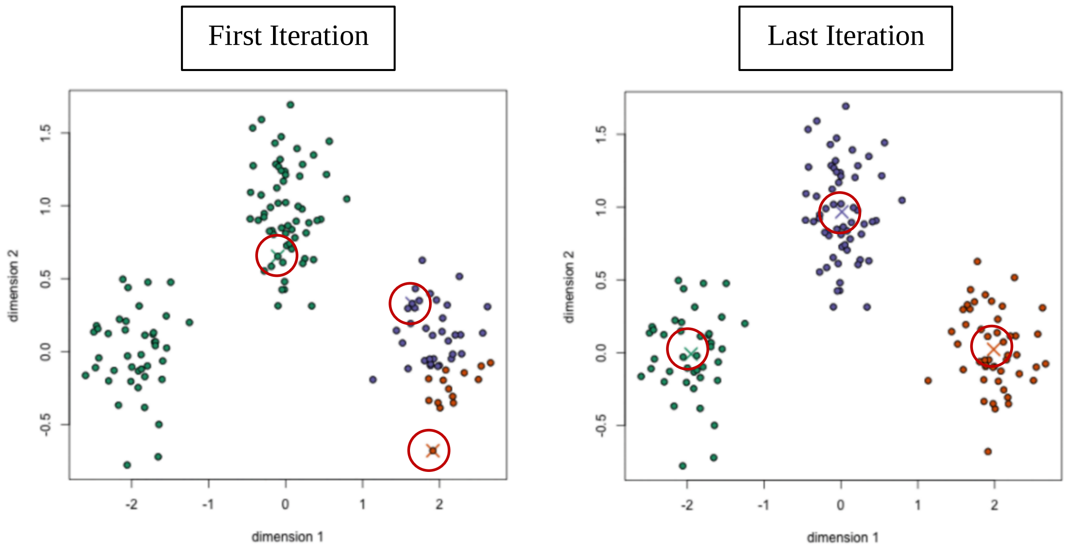


Figure 3-6: K-Means algorithm evaluation on discrete data objects [39].

The K-Means algorithm requires two user-specified parameters: the number of clusters and the initial weights of cluster centroids, while Euclidean distance is usually used as distance metric between dataset vectors and cluster centroids. As a result, K-Means finds hyper-spherical clusters [40].

The most common technique for the initialization of cluster centroids is random initialization, which causes the algorithm to be non-deterministic, meaning that cluster assignments will vary for different executions, leading to local, non-optimal minima of error function. This depends on the seed of the random algorithm. If the same seed is used, the resulting random sequence will be the same. This intrinsic characteristic of K-Means method is treated to a specific degree by running the algorithm multiple times and finally selecting the best clustering option, i.e. the one characterized by the lowest error.

The main drawback of the K-Means clustering method is that the algorithm requires the number of clusters to be a priori specified. Typically, K-Means is run independently for different values of *k* (number of clusters), and the partition that appears the most meaningful to the dataset dealing with is finally selected [40]. Generally, it is not possible to find any prescribed mathematical criterion to efficiently set a priori the number of clusters, *k*. However, different indices have been proposed in the literature to assist the choice of the optimal number of clusters. These indices usually consider the capacity to separate data objects with maximum distance (external evaluation) and the capacity of the data to gather together around the centroids, generating maximal compact groups (internal evaluation) [36]. The most common indexes found in the literature for the a priori determination of the number of clusters are the Silhouette Coefficient and the Davies-Bouldin index.

3.2.3.1 Silhouette Coefficient

The Silhouette Coefficient allows for an appreciation of the relative quality of the clusters, as a method of interpretation and validation of consistency within the clusters [43]. For a given data sample, the Silhouette Coefficient is defined as:

$$s = \frac{b - a}{\max\{a, b\}}$$

where:

- a is the average distance between the sample and all other points of the same cluster, and
- b is the average distance between the sample and all other points of the nearest cluster.

The Silhouette Coefficient for a set of samples is given as the mean of the Silhouette Coefficient of each sample. By definition, this coefficient can achieve values in the range [-1,1]. The minimum value (-1) appears when a>>b, meaning that the samples lie on the average closer to the corresponding nearest cluster than their labeled cluster. This implies incorrect and overlapping clustering. On the other hand, the maximum value (1) appears when a<<b, meaning that the samples lie on average closer to their labeled cluster than the regarding nearest cluster. This situation corresponds to a high dense clustering. An intermediate case occurs when Silhouette Coefficient lies near zero. In this case, distance a≈b and hence it is not clear if each sample should have been assigned to a specific cluster or to its nearest one.

In conclusion, the Silhouette Coefficient measures how well a sample has been classified. The score is higher when clusters are dense and well separated from each other. Figure 3-7 is a representation of intra-cluster and inter-cluster distances used for the calculation of the Silhouette Coefficient.

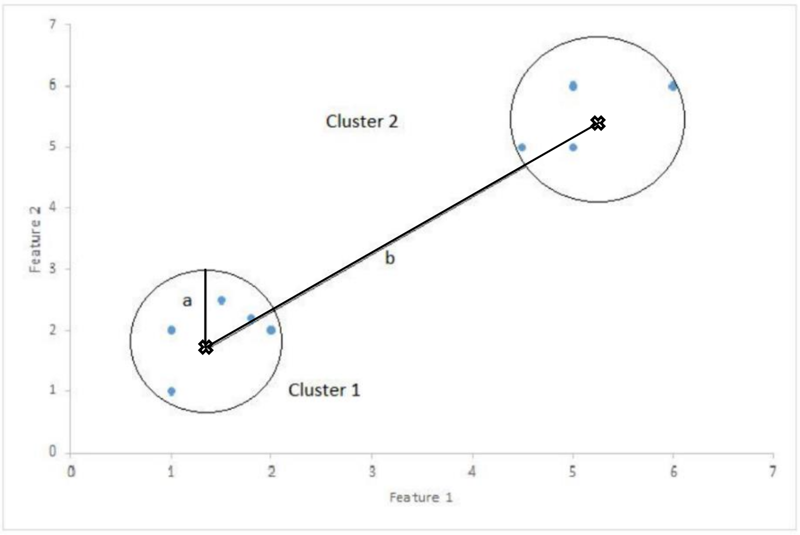


Figure 3-7: Intra-cluster and inter-cluster distances for the calculation of Silhouette Coefficient [44].

3.2.3.2 Davies-Bouldin index

The Davies-Bouldin index is a similarity measure between two clusters, defined on a measure of dispersion of one cluster and of dissimilarity between these two clusters. Similarity, or equally, dissimilarity works as a measure that compares the distance between clusters with the size of the clusters themselves (Figure 3-8).

Let C_i, C_j be two as similar as possible clusters, for i,j = 1,…,*k*, where *k* is the total number of clusters. The mathematical formulation of the Davies-Bouldin index is:

$$\mathrm{DB} = \frac{1}{\mathrm{k}}\sum_{\mathrm{i}=1}^{\mathrm{k}} \max \mathrm{R}_{\mathrm{ij}}\,, \text{for } \mathrm{i} \neq \mathrm{j}$$

where R_{ij} represents the average similarity between each cluster C_i and its most similar one C_j. R_{ij} is defined as:

$$\mathrm{R}_{\mathrm{ij}} = \frac{\mathrm{s}_{\mathrm{i}} + \mathrm{s}_{\mathrm{j}}}{\mathrm{d}_{\mathrm{ij}}}$$

with:

- s_i the average distance between each point of cluster C_i and the centroid of this cluster, commonly defined as cluster diameter and
- d_{ij} is the Euclidean distance between cluster C_i, C_j centroids.

By definition, the Davies-Bouldin index is symmetric and non-negative. The lowest possible value is zero (0), which corresponds to a better partitioning, as the clusters are characterized by the minimum possible similarity to each other.

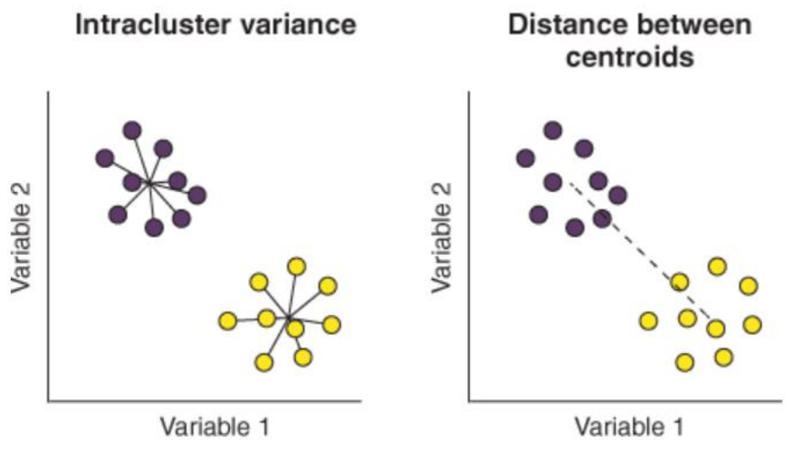


Figure 3-8: Discovery of two similar clusters (left) with approximately equal cluster diameter and distance between their centroids (right) [45].

It is important to be noted that the Davies-Bouldin index may result to an optimal partitioning different from the one resulting from the Silhouette Coefficient, due to their different definitions. In this study, both Silhouette Coefficient and Davies-Bouldin index are used for the clustering evaluation analysis, and their results are evaluated by also taking into account the physics that describe the phenomenon.

3.2.3.3 Clustering of Self-Organizing Map

The K-Means clustering method works properly for low complexity data, but seems not to be ideal for high-dimensional data. The computational issue becomes critical when the application involves large scale data, such as in the case of combustion. An immediate and tangible solution is to pre-cluster the high-dimensional data and use the K-Means algorithm to group these sub-clusters. In this study, the Self-Organizing Map neural network has already been introduced to deal with combustion data clustering, and will be used to generate clustering maps for each feature. Each cell on the 2D array-clustering map represents a sub-cluster.

Previous studies proposed clustering of the Self-Organizing Map (SOM). *J. Vesanto* and *E. Alhoniemi* [41] firstly introduced SOM clustering by applying agglomerative and partitioning algorithms. A predefinition of the number of clusters results by visual inspection of the unified distance map (U-Matrix), which shows the distances between node vectors (neurons). Some details of U-Matrix can be observed in Figure 3-9, for a SOM trained on the Iris dataset, a well-known dataset for Machine Learning applications [42]. To select the best clustering among partitions with different number of clusters, the Davies-Bouldin index was used.

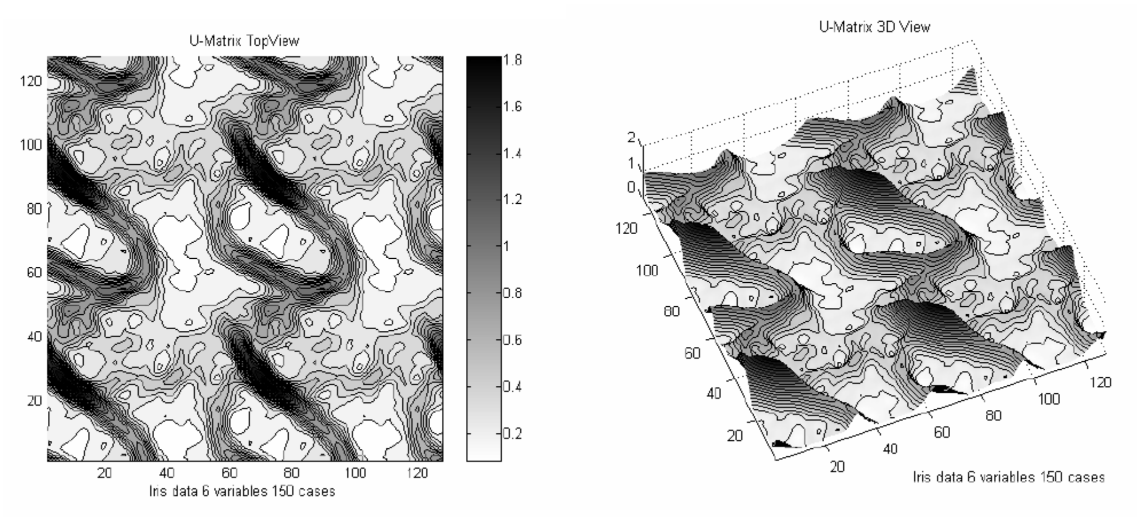


Figure 3-9: U-Matrix of a trained Self-Organizing Map neural network on the Iris dataset. Light colors depict closely spaced node vectors (neurons), and darker colors indicate more widely separated node vectors [42].

Garcia et al. [34] and *Brentan et al.* [36] developed a Self-Organizing Map (SOM) coupled with K-Means for implementation on water distribution and treatment monitoring. The data clustering process consists of a two-stage procedure. The first step is SOM training, and next the acquired node vectors are grouped to form the actual clusters. An initial idea regarding the number of clusters in the SOM is also acquired by visual inspection of the U-Matrix, while Davies-Bouldin index is used to evaluate the several structures obtained by K-Means. A major difference from previous studies (e.g. [41]) is the application of K-Means algorithm on the dissimilarity matrix rather than the SOM 2D array, in order to consider and emphasize the topography of the dataset.

In conclusion, by applying SOM as a clustering method, it is expected that the number of neurons may be very high regarding the number of observations. Although SOM has the ability to efficiently map any high-dimensional space, it could be undesirable for clustering purposes, as the number of clusters may be overestimated and the number of observations in each cluster may be too low to have any physical or statistical meaning. For this reason, SOM is coupled with the K-Means algorithm in the current study, in order to group the closest neurons into clusters, obtaining the prescribed or desired number of clusters required for the high-dimensional combustion process.

The primary benefits of this two-level approach, in lieu of directly clustering the data, are the reduction of the computational cost and the rough visual presentation and interpretation of clusters on a 2D grid. Furthermore, clustering the SOM manually or grouping cells of the 2D array into rectangular clusters may decrease the quality of clustering.

3.3 Neural Network for Regression

A Multi-Layer Perceptron (MLP) is a class of feedforward artificial neural network (ANN), and it is used in the present study for regression. It consists of at least three layers of neurons: an input layer, an output layer and at least one intermediate hidden layer. The information propagates through the network in a forward direction, on a layer-by-layer basis. In other words, if a random neuron *b* takes an input from another neuron *a*, its output will not affect neuron *a*. Furthermore, neurons belonging to a given layer are not connected with each other [46], [47]. The fundamental structural element of MLP is the perceptron.

The perceptron can be considered as an element executing an algorithm for supervised learning, utilizing linear binary classification tasks. Let *x* be a n-dimensional input vector. The perceptron can learn a binary classifier called threshold function: a function that maps the input vector *x* to a single binary output value f(x). For the simplest case, the threshold function is defined as:

$$f(x) = \left\{ \begin{array}{ll} 1, & \text{if } \sum_{i=1}^{n} w_i x_i + w_0 > 0 \\ 0, & \text{otherwise} \end{array} \right\}$$

where w is a vector with real-valued weights, w·x is the inner product and w_0 is a weight constant called bias. Depending on the sign of threshold function, the input data is assigned to a specific class.

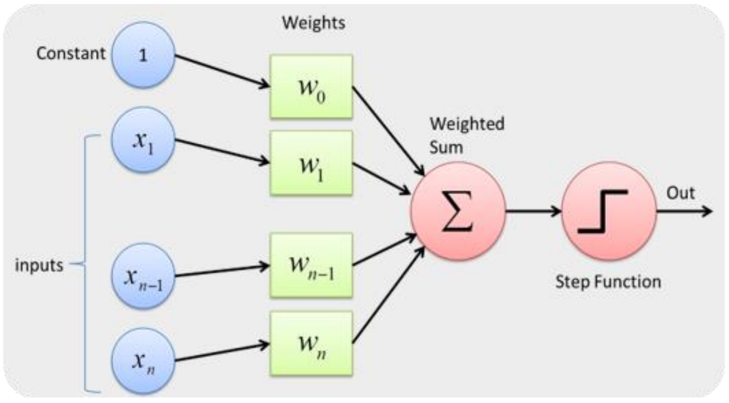


Figure 3-10: Structure of Perceptron [23].

A Multi-Layer Perceptron is a generalization of single layer Perceptrons. Its architecture consists of multiple neurons represented as Perceptrons, organized into hidden layers and connected with each other. Connections exist only between neurons from adjacent layers and not between neurons belonging to the same layer. The first hidden layer is fed the information from the input data, and the results are then applied as an input to the next hidden layer; and so on for the rest of the network. Each neuron is designed to perform the computation of the function or discrete value appearing at its output, which is expressed as the application of a continuous non-linear function on the input signal and the synaptic weights associated with that neuron.

The final target is to assemble a MLP neural network to approximate a non-linear representation from the input space R^N to the output space R^o:

$$f(\cdot) : R^N \rightarrow R^o$$

where N is the dimensionality of the input and o is that of the output. Given a set of features X and a target Y, the MLP is assumed to learn and create a non-linear function approximator from the input space R^N to the output space R^o.

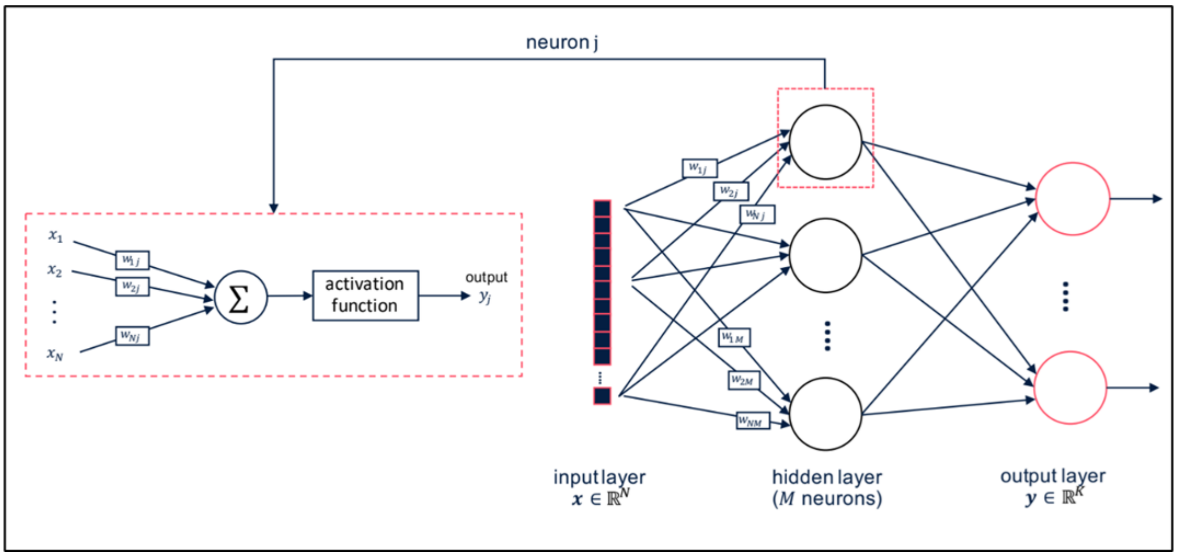


Figure 3-11: Structure of Multi-Layer Perceptron [23].

Let x be the N-dimensional input data vector and design a MLP neural network supplied k hidden layers, each one having a specific number of neurons. The MLP computation algorithm of a neuron consists of the following steps:

<u>Step 1</u>: Each feature of input data vector is multiplied with a weight w_{ij}, where i represents the previous and j the current neuron index. Thus, i and j acquire values depending on the number of neurons at each hidden layer.

<u>Step 2</u>: A bias term θ is added, to shift the activation function $\varphi(\cdot)$, as it is defined in a following paragraph. This term determines whether and how much the neuron will be activated.

<u>Step 3</u>: All products from the previous steps are summed on the respective neuron.

<u>Step 4</u>: An activation function is applied, producing the final neuron's output.

The general mathematical formula of a neuron's output computation for each MLP's hidden layer can be expressed by the following equations. Index j corresponds to jth neuron at input or k hidden layer, respectively.

Neuron's output for first hidden layer:

$$y_j^1 = \varphi^1\left(\sum_{i=1}^{N} w_{ij}^1 \cdot x_i + \theta_j^1\right)$$

Neuron's output for *k* hidden layer:

$$y_j^k = \varphi^k\left(\sum_{i=1}^{M_{k-1}} w_{ij}^k \cdot y_i^{k-1} + \theta_j^k\right)$$

The superscript indicates the layer and the subscript indicates the respective neuron within the specific layer. The weight w_{ij} is associated with the connection of the *i*th neuron of the *(k-1)th* layer (previous layer) to the *j*th neuron of the *k* layer (present layer). The number of input features to the MLP is denoted as *N* and the number of neurons within the *k*th hidden layer is denoted by M_k. The input data follows the forward flow direction through each layer using the above equations until the final output is calculated.

The function φ is the so-called activation function, a non-linear function which is the source of the MLP's ability to perform non-linear function approximations. It shall be continuous, typically in the range [0,1] or [-1,1] and activate a neuron whenever its value is sufficiently high. In this study, the hyperbolic tangent is employed as the activation function for every hidden layer, due to its property of being a zero-centered, smooth differentiable function, a property considered important for the execution of the back propagation algorithm (see Section 3.3.2).

$$\varphi(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

The weights and biases are the unknown coefficients for the model and the ones adjusted during the training procedure. This fitting process is an optimization problem, where one seeks to find the optimal values of weights and biases in order to minimize an objective function, expressing the error between predictions and target output. For the current implementation, the back propagation algorithm is employed to deal with training and optimization, and it is presented in Section 3.3.2. But first of all, the architecture of MLP must be chosen by applying the cross validation technique.

3.3.1 Cross Validation

An important issue of neural network design is the network architecture. The choice involves setting the number of hidden layers and the neurons per layer. One should select the simplest possible network that can adequately represent the training dataset [26], since this choice has a significant impact on the network's computation cost, complexity and generalization ability. Since this type of neural networks have a large number of tunable parameters, they tend to overfit.

The ideal network architecture for a task must be found via experimentation guided by monitoring the validation set error. In the present study, the training procedure employs the cross validation technique. The training data is randomly split into a training set and a validation set that is used for testing the generalization ability. Apart from offering a preliminary evaluation of the MLP performance and trend for overfitting, the cross validation technique can also be used as a termination criterion of the training process.

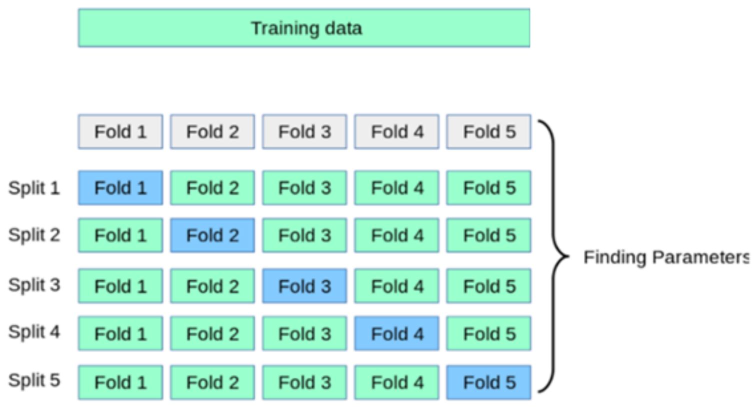


Figure 3-12: Illustration of 5-Folds Cross Validation technique [61].

In the basic approach, called k-Folds cross validation, the initial training dataset is split into *k* smaller groups of pairs of input-target data and the following procedure is executed:

- A clone model of regressor (MLP in this case) is trained using the *k*-1 folds.
- The resulting model is validated on the remaining fold, which is used as a test set for performance accuracy, overfitting avoidance and generalization ability.

The performance measure reported by k-Folds cross validation is the average of the values computed in the loop.

By applying k-Folds cross validation, one can decide on the so-called Bias-Variance Tradeoff. The bias of an estimator is an indication of its average error, while its variance indicates how sensitive it is to varying input datasets. High bias can cause an algorithm to miss relevant relations between input features and target outputs (underfitting). On the other hand, high variance may result from an algorithm modelling the random noise in the training data (overfitting). A compromise between bias and variance is strived for in order to predict optimally the training set while executing sufficiently on the basis of generalization.

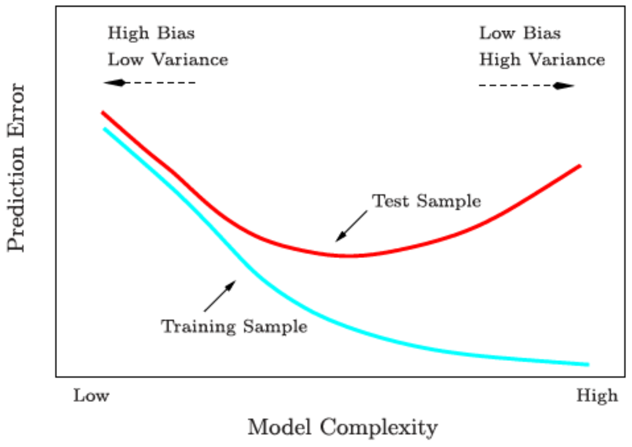


Figure 3-13: Bias – Variance Tradeoff. Training and testing error as a function of model complexity [50].

For the current implementation, 3, 5 and 8-Folds cross validation is applied in order to compare the results and evaluate the behavior of the neural network. Regarding the performance measure, the Root Mean Square Error (RMSE) is used:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_{i,\mathrm{target}} - y_{i,\mathrm{pred}})^2}$$

where *N* is the total number of input data (observations).

After defining the network's architecture and selecting a priori some basic hyperparameters, such as the activation function, the training process is ready to start. The back propagation algorithm is applied in this case.

3.3.2 Back Propagation

In Machine Learning, back propagation is a widely used algorithm for training feedforward neural networks. The fitting process is based on efficiently computing the gradient of the loss function with respect to the weights of the neural network. This efficiency justifies the use of gradient descent methods for training the network, by updating the weights in order to minimize the loss function.

The loss function is an objective function appearing in every optimization problem. In ANN training, the optimization problem aims at assigning the optimal values of weights and biases. Essentially, it measures the inconsistency between real and predicted output of the network, and its output is a real number, representing a cost. For regression analysis, the most common loss function is Mean Squared Error Loss Function (the so-called L2 Loss Function):

$$\epsilon(y_{\mathrm{real}}, y_{\mathrm{pred}}) = \frac{1}{N}\sum_{i=1}^{N}\left(y_{i,\mathrm{real}} - y_{i,\mathrm{pred}}\right)^2$$

The loss is the mean overseen data of the squared differences between true and predicted values. A main property of Mean Squared Error Loss Function is that large errors are significantly (quadratically) more penalized than small ones.

The back propagation algorithm computes the gradient of the loss function ($\partial\epsilon/\partial w_{ij}$) for each layer *k*, by applying the chain rule and evaluating the expression for the derivative of the loss function as a product of the derivatives between each layer. The algorithm yields the gradients of the activation functions φ^k for each layer, *k*, starting from the output layer and flowing backwards to the first hidden layer. These gradients can be interpreted as an indication of how each layer's output will change in order to reduce the error [48]. A schematic of the back propagation algorithm is presented in Figure 3-14. During the forward pass, the weights and biases of neurons are all fixed, while, during the backward pass, their values are all adjusted in accordance to the error-correction rule.

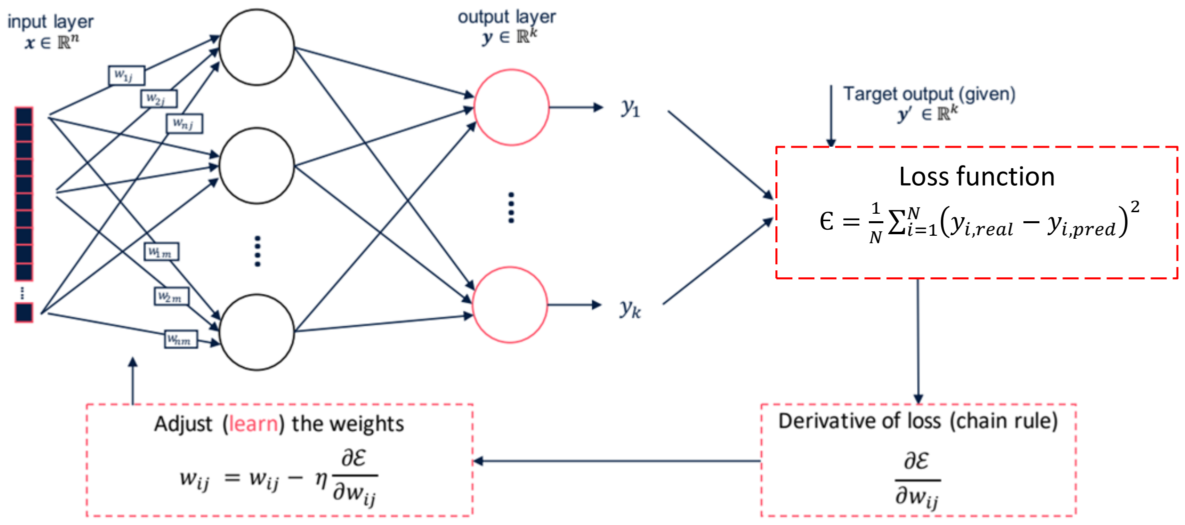

input layer
$x \in \mathbb{R}^n$
w_{1j}
w_{2j}
w_{nj}
w_{1m}
w_{2m}
w_{nm}
output layer
$y \in \mathbb{R}^k$
y_1
y_k
Target output (given)
$y' \in \mathbb{R}^k$
Loss function
$\epsilon = \frac{1}{N}\sum_{i=1}^{N}(y_{i,real} - y_{i,pred})^2$
Adjust (learn) the weights
$w_{ij} = w_{ij} - \eta\frac{\partial \mathcal{E}}{\partial w_{ij}}$
Derivative of loss (chain rule)
$\frac{\partial \mathcal{E}}{\partial w_{ij}}$

Figure 3-14: Back propagation algorithm [23].

The initial values of weights and biases are randomly selected. It is important to note that they are usually assigned small values [48]. During the backward bass at an epoch, i.e. at an iteration *t*, the back propagation algorithm consists of the below steps:

Step 1: Compute the predicted outputs from the network, y_{pred}, and calculate the loss function.

Step 2: Calculate the derivative of loss function at layer *k* by applying the chain rule:

$$\frac{\partial \epsilon}{\partial w_{ij}^k} = \frac{\partial \epsilon}{\partial y_j^k} \cdot \frac{\partial y_j^k}{\partial w_{ij}^k} = \frac{\partial \epsilon}{\partial y_j^k} \cdot \frac{\partial y_j^k}{\partial u_j^k} \cdot \frac{\partial u_j^k}{\partial w_{ij}^k}$$

where u_j^k is the net output of neuron j at layer *k*, before applying the activation function. As already defined, the net output of a neuron is given by the expression:

$$u_j^k = \sum_{i=1}^{M_{k-1}} w_{ij}^k \cdot y_i^{k-1} + \theta_j^k$$

Therefore:

- The last term on the above chain rule simplifies to $\frac{\partial u_j^k}{\partial w_{ij}^k} = y_i^k$.
- The middle term of the chain rule can be expressed as:

$$\frac{\partial y_j^k}{\partial u_j^k} = \frac{\partial \varphi^k(u_j^k)}{\partial u_j^k}$$

  where the partial derivative of the activation function φ^k with respect to the net neuron output u_j^k can be computed with knowledge of the activation function itself. Here lies the justification for the back propagation algorithm requiring the activation function to be differentiable.

- The last term on the above chain rule is directly computed for the output neuron. However, it takes a more complicated formula when calculating this derivative for hidden layers. It can be shown that [48]:

$$\frac{\partial \epsilon}{\partial \mathrm{y}_{\mathrm{j}}^{\mathrm{k}}} = \sum_{\mathrm{l} \in \mathrm{L}} \frac{\partial \epsilon}{\partial \mathrm{u}_{\mathrm{l}}^{\mathrm{k}}} \cdot \mathrm{w}_{\mathrm{lj}}^{\mathrm{k}}$$

where L is the set containing all neurons receiving input from the jth neuron. This term is easily computed when the derivatives of the loss function are known at the next layer, the one nearest to the network output.

Step 3: Update the weights, based on the formula:

$$\mathrm{w}_{\mathrm{ij}}^{\mathrm{k}}(\mathrm{t}+1) = \mathrm{w}_{\mathrm{ij}}^{\mathrm{k}}(\mathrm{t}) - \eta \frac{\partial \epsilon}{\partial \mathrm{w}_{\mathrm{ij}}^{\mathrm{k}}}$$

where t is the training epoch (or iteration) index and η is a positive decreasing factor for the convergence of the process, the so-called learning rate. It is important to note that each update changes the weights in such a way that the loss function decreases, regardless of the sign of its derivative.

Convergence of the above algorithm is achieved when the maximum number of epochs is reached or when a desired threshold of loss function decrease is achieved. Additional and useful information on the back propagation algorithm can be found in the literature (e.g. [26],[46],[48]). It is important to note that the optimization problem is not convex, so there is no guarantee of converging to the global minimum.

The computational cost of the algorithm is not negligible, and its application may lead to prohibitively high execution times. A commonly used method to address this problem is stochastic gradient descent. Instead of calculating the gradient for each neuron and each layer at each epoch, an estimation is used by sampling a batch of input data drawn uniformly from the input dataset.

An extension of the stochastic gradient descent method is the Adam (Adaptive Moment Estimation) algorithm, suitable for deep learning applications and essential for large datasets. The main difference from classical stochastic gradient descent is the computation of individual adaptive learning rates for each network weight from estimations of first and second moments of the gradients (mean and uncentered variance respectively). Due to this fact, the Adam optimizer can properly avoid local minima. In the present study, the Adam optimizer is used to train each MLP with constants, as they are proposed in the literature [51],[52].

CHAPTER 4: Chemical Kinetics

4.1 Introduction to Chemical Kinetics

Chemical Kinetics is the branch of Chemistry dealing with the progress of chemical reactions and investigating the way through which a reaction reaches its final condition. It focuses on qualitative and quantitative study of the rates of progress of chemical reactions as well as on the evaluation of factors affecting this process. Better understanding of processes happening during the conversion of reactants to products and development of mechanisms modelling properly these phenomena are some of the major assets offered by Chemical Kinetics [53].

Each chemical reaction evolves with a finite rate depending on the conditions of the system, such as concentrations of reactants, temperature, radiation effects and the presence of a catalyst. The reaction rate can be expressed in terms of concentration change of any reactant or product. In other words, the rate of a chemical reaction may be expressed as the rate of decrease of a reactant concentration or the rate of increase of a product concentration [54]. The rate of a chemical reaction and its dependencies are mathematically expressed by the Law of Mass Action.

The temporal evolution of a chemical reaction is fully defined by integrating an ODE system, consisting of equations describing the changes in temperature and species concentrations [53]. In the case of an isobaric process, the temperature change is calculated from the Energy Conservation equation, while species concentrations evolution is described via the Law of Mass Action. The number of ODEs is N_s+1, where N_s is the number of species participating in the reaction mechanism. As shown, this ODE system is non-linear and stiff, due to the different time scales of the species observed for each reaction.

4.1.1 Law of Mass Action

A stoichiometric relation describing an arbitrary one-way elementary chemical reaction of N_s species can be written in the form:

$$\sum_{i=1}^{N_s} \nu_i' M_i \rightarrow \sum_{i=1}^{N_s} \nu_i'' M_i$$

where M_i is the species participating in the reaction, ν_i' the stoichiometric coefficient of the reactants and ν_i'' the stoichiometric coefficient of the products. If a species represented by M_i does not participate as a reactant or product, $\nu_i = 0$.

The Law of Mass Action states that the rate of consumption or production of a chemical species, defined as reaction rate (RR), is proportional to the product of concentrations of the chemical species, where each concentration is raised to a power equal to the corresponding stoichiometric coefficient [54]. The mathematical formula of Law of Mass Action is:

$$RR \sim \prod_{i=1}^{N_s} [M_i]^{\nu_i'} \quad \text{or} \quad RR = k \cdot \prod_{i=1}^{N_s} [M_i]^{\nu_i'}$$

The rate of change of the concentration of a chemical species M_i, defined as ω_i, is then given by:

$$\dot{\omega}_\iota = \frac{d[M_i]}{dt} = (\nu_i'' - \nu_i') \cdot RR = (\nu_i'' - \nu_i') \cdot k \cdot \prod_{i=1}^{N_s} [M_i]^{\nu_i'}$$

since ν_i'' moles of M_i are created for every ν_i' moles of M_i consumed. The convention used in this report is that parentheses around a chemical symbol signify the concentration of that species, i.e. $[M_i]$ signifies the concentration of species M_i.

The proportionality constant k is called reaction rate constant and introduces the temperature dependence for reaction rates. It refers to a specific reaction and direction and it is calculated through the empirical Law of Arrhenius, modified in the form:

$$k = A \cdot T^b \cdot \exp\left\{-\frac{E_A}{R \cdot T}\right\}$$

where:

- A, b: constants depending on the nature of the reaction, stereochemistry and molecular characteristics, such as equivalent diameter and equivalent mass, and independent of temperature,
- T: the absolute temperature (in Kelvin),
- E_A: the activation energy, which is the energy barrier that must be surpassed for chemical bonds to break and a reaction to proceed, and
- R = 8.314462 J/(mol·K): the universal gas constant.

The pre-exponential term $A \cdot T^b$ accounts for the frequency and the orientation of molecular collisions, while the exponential term accounts for the probability of a collision to be successful, in other words to have energy greater than activation energy. Consequently, the specific reaction rate coefficient essentially represents the frequency of successful collisions. A notable observation is that, at increasing temperature, the specific reaction rate coefficient also increases, meaning that the number of successful collisions between molecules increases. As a result, the reaction rate of the chemical reaction increases, following the proportional relation of the Law of Mass Action.

The units of measurement of specific reaction rate coefficient k are determined by the order of reaction. The order of a chemical reaction as to a specific species is equal to the stoichiometric coefficient of this species. The overall order of a chemical reaction is the sum of the individual orders.

Large kinetic mechanisms usually contain several independent, simultaneous, bidimensional, elementary reactions. In this case, the rate of change of concentration of each specific species is calculated via the principle of superposition, by adding the individual rates of change of concentration for each reaction the species is participating in.

4.1.2 Energy Conservation

The first Law of Thermodynamics for a closed and adiabatic system subjected to an isobaric chemical process of N_s species is expressed as:

$$\frac{dh}{dt} = \sum_{i=1}^{N_s}\left(Y_i \cdot \frac{dh_i}{dt} + h_i \cdot \frac{dY_i}{dt}\right) = 0 => \sum_{i=1}^{N_s}\left(Y_i \cdot \frac{dh_i}{dT} \cdot \frac{dT}{dt}\right) + \sum_{i=1}^{N_s}\left(h_i \cdot \frac{dY_i}{dt}\right) = 0 =>$$

$$=> \sum_{i=1}^{N_s}\left(Y_i \cdot c_{p,i} \cdot \frac{dT}{dt}\right) + \sum_{i=1}^{N_s}\left(h_i \cdot \frac{dY_i}{dt}\right) = 0 => c_p \cdot \frac{dT}{dt} + \sum_{i=1}^{N_s}\left(h_i \cdot \frac{dY_i}{dt}\right) = 0 =>$$

$$=> \frac{dT}{dt} = -\frac{\sum_{i=1}^{N_s}\left(h_i \cdot \frac{dY_i}{dt}\right)}{c_p}$$

where:

- h_i , h: the specific enthalpy of species *i* and the specific enthalpy of system respectively.
- T: the absolute temperature (in Kelvin).
- Y_i: the mass fraction of species *i*, defined as:

  $$Y_i = \frac{m_i}{m} = \frac{n_i \cdot MW_i}{\sum_{i=1}^{N_s} n_i \cdot MW_i}$$

  where m_i is mass of species *i*, m the total mass of the mixture, n_i the moles of species *i* and MW_i the molecular weight of species *i*.
- $c_{p,i}$, c_p: the specific heat capacity of species *i* and the specific heat capacity of the mixture respectively.

The above equation obtained from the first Law of Thermodynamics is used to calculate the temporal evolution of temperature during a reaction process. In combination with the equations obtained from the Law of Mass Action and referring to species concentrations changes, a system of N_s+1 ODEs is formulated, which models the evolution of the thermochemical state of the system [53].

4.1.3 Kinetic Mechanisms

Chemical reactions are described by detailed mechanisms, usually containing hundreds of species and thousands of reactions. A detailed mechanism essentially consists of elementary chemical reactions, which are reactions taking place at molecular level exactly in the way described by their chemical reactions. A group of elementary chemical reactions may describe a specific chemical process happening during combustion. The total set of these reactions form the detailed mechanism and can model sufficiently the global chemical reaction.

Figure 4-1 summarizes the complexity of the available hydrocarbon combustion mechanisms. The higher the hydrocarbon molecules contained in the fuel mixture, the more complex a mechanism becomes, as it requires a large number of species and reactions to model the combustion process.

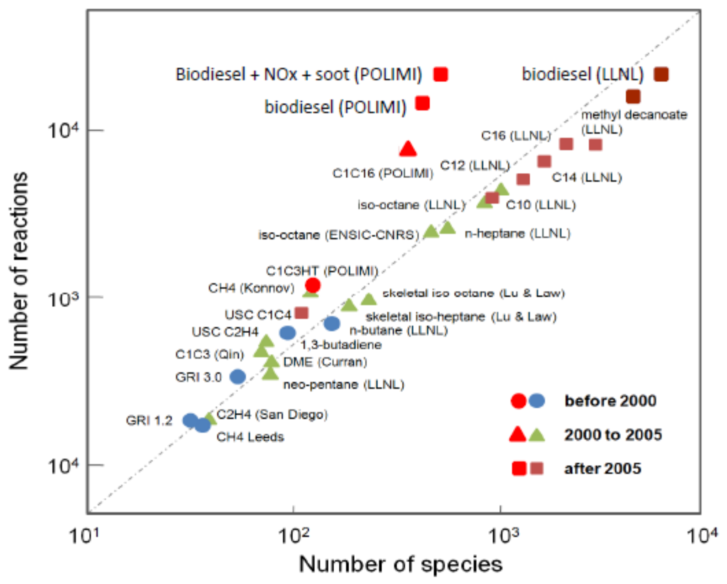


Figure 4-1: Detailed kinetic mechanisms for hydrocarbons combustion [4].

For modeling sufficiently hydrocarbon combustion, a kinetic mechanism shall account at least for the following features [53]:

1) Hydrocarbon oxidation, representing the consumption of the fuel, the production of water and carbon monoxide (CO) and its further oxidation to carbon dioxide (CO_2), a process that releases the highest amount of heat during the combustion process,
2) NO_x formation, an extremely complex process, mainly occurring at high temperature and long residence time,
3) SO_x formation, due to the sulphur contained into the combustible mixture, and
4) Soot and unburned hydrocarbons formation.

An optimized detailed mechanism to model natural gas combustion is the GRI Mech 3.0, developed at Berkeley University, and used in the present study. In total, it contains 53 species and 325 reactions. It includes NO_x formation and reburn chemistry, and contains higher hydrocarbon kinetics, up to propane.

The optimization refers to adjustment of the constants A, b, E_A of specific reaction rates for each reaction, in order for the mechanism to properly fit different available experimental data. More specifically, the rate parameters are optimized against laminar flame speed experimental data, shock-tube ignition delay and species profile measurements, prompt NO, HCN oxidation and reburning measurements. The mechanism performance is validated via laminar flame speed and ignition delay calculations, laminar flame and shock tube species profiles, flow and stirred reactor experiments. For more details about construction and evaluation of the mechanism, the reader is referred to the GRI Mech 3.0 official forum [56].

Kallieros [57] in his diploma thesis also validated the results of GRI Mech 3.0 with experimental data, for premixed laminar flame propagation, ignition delay of homogenous mixture and species profiles in perfectly stirred reactor (PSR). The simulations were performed in CANTERA, an open-source suite of tools for problems involving chemical kinetics, thermodynamics and transport processes [58]. The results show very good agreement with the experimental data for the premixed laminar flame speed and ignition delay of homogenous mixture. As for the species profiles in PSRs, the results show a small deviation from experimental data for specific inlet conditions, probably due to non-target fitting of the present mechanism to these experimental data or its relatively small size comparing to other available mechanisms, such as NUIG-NGM, containing 293 species and 1593 reactions. Nevertheless, GRI Mech 3.0 is used extensively in studies, and can sufficiently model natural gas combustion.

The present study deals with methane combustion, and employs GRI Mech 3.0 for premixed laminar flame propagation and homogeneous reactors. These standard problems and their theoretical background are discussed in the following section.

4.2 Premixed Laminar Flame Propagation

When a quiescent combustible premixed gas mixture within flammability limits contained in an open tube is ignited by a spark, a combustion wave spreads through the gas, the so-called deflagration wave. In the case of an open tube, this wave is subsonic, often observed to be nearly planar and to travel at approximately a constant speed. This constant speed is an empirical laminar burning velocity or flame speed, which is characteristic of the combustible mixture [55]. Thus, premixed laminar flame speed is defined as the speed at which an unstretched laminar flame will propagate through a quiescent, homogenous premixed mixture of unburned reactants.

The flame can be considered as an interface, separating the mixture into the unburned and the burned gas region. The flame is thin and characterized by convective flow of unburned gas mixture throughout the flame and diffusion of radicals from the high-temperature reaction zone against the convective flow and towards the preheat region. A flame can hence be considered as a rapid, self-sustaining chemical reaction occurring in a discrete reaction zone. Reactants may be introduced into this reaction zone, or the reaction zone may move into the reactants, depending on whether the unburned gas velocity is greater than or less than the flame deflagration velocity [54]. *Glassman et al.* [54] define the laminar flame speed as the velocity at which unburned gases move through the combustion (deflagration) wave in the direction normal to the wave surface.

The deflagration wave propagation is not steady for a still observer out of the system. However, it is steady in time for an observer moving with the wave. The moving observer sees the burned mixture fending off with velocity V_b and the unburned mixture approaching with velocity V_u. Another definition for the premixed laminar flame speed is thus the relative speed of unburned gas mixture for an observer moving with the flame (Figure 4-2).

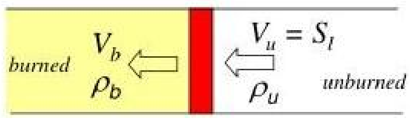


Figure 4-2: Premixed laminar flame propagation in a tube. S_L is defined as laminar flame speed.

It is proven from Hugoniot curve analysis that the process can be considered isobaric, as the burned gas pressure is slightly lower than the unburned gas pressure. The parameters from which the laminar flame speed depends can arise by Mallard and Le Chatelier theory analysis.

The convention used in this thesis is that subscript *u* or *unb* refers to unburned gas mixture and subscript *b* or *bur* refers to burned gas mixture.

4.2.1 Mallard and Le Chatelier Theory

Mallard and *Le Chatelier* developed a thermal theory based on energy equation to define the laminar flame speed. They postulated that the flame consists of two zones, preheat zone and reaction zone, separated at the point where the next layer ignites (Figure 4-3). The heat conducted from the reaction zone is equal to that necessary to raise the unburned gas mixture to the ignition temperature (the boundary between preheat zone and reaction zone) [54]. Assuming a linear slope of temperature curve and according to Fourier's Law of thermal conduction, the enthalpy balance for an adiabatic flame gives:

$$\dot{m} \cdot c_p \cdot (T_{ign} - T_0) = \lambda \cdot \frac{T_f - T_{ign}}{\delta} \cdot A$$

where $\dot{m}$ is the mass rate of the unburned gas mixture into the combustion wave, c_P is the specific heat capacity of the unburned gas mixture, λ is the thermal conductivity, A is the cross-sectional area and δ is the reaction zone thickness. Additionally, T_0 refers to inlet mixture temperature, T_f refers to the flame temperature, approximately equal to burned gas temperature in the case of adiabatic flame, and T_{ign} is the mixture's ignition temperature.

The mass rate of unburned gas mixture can be expressed as:

$$\dot{m} = \rho_u \cdot V_u \cdot A = \rho_u \cdot S_L \cdot A$$

where ρ_u is the unburned gas density and $V_u = S_L$ is the laminar flame speed by definition. By substituting in the enthalpy balance equation, the following expression for laminar flame speed is obtained:

$$S_L = \frac{\lambda}{\rho_u \cdot c_p} \cdot \frac{T_f - T_{ign}}{T_{ign} - T_0} \cdot \frac{1}{\delta} \quad \text{or} \quad S_L = \alpha \cdot \frac{T_f - T_{ign}}{T_{ign} - T_0} \cdot \frac{1}{\delta}$$

where $\alpha = \lambda/\rho_u \cdot c_p$ is the thermal diffusivity coefficient.

Unfortunately, the reaction zone thickness, δ, is not known nor can it be easily defined. However, it is possible to relate δ to the laminar flame speed S_L. The total mass rate per unit area entering the reaction zone must be equal to the consumption rate in that zone, given by chemistry for the steady flow problem. The mathematical expression of the aforementioned is:

$$\frac{\dot{m}}{A} = \rho_u \cdot S_L = \dot{\omega} \cdot \delta$$

where $\dot{\omega}$ is the reaction rate in terms of concentration per unit of time. Therefore, the expression for laminar flame speed becomes:

$$S_L = \left(\alpha \cdot \frac{T_f - T_{ign}}{T_{ign} - T_0} \cdot \frac{\dot{\omega}}{\rho_u}\right)^{1/2} \quad \text{or} \quad S_L \sim \left(\alpha \cdot \frac{\dot{\omega}}{\rho_u}\right)^{1/2}$$

From the above analysis, it is concluded that laminar flame speed is a thermophysical property of the unburned gas mixture, independent of the flow field. In other words, it depends on the thermophysical properties of unburned gas mixture, namely the thermal diffusivity (α) and the density (ρ_u), as well as on the combustion kinetics through consumption rate in reaction zone ($\dot{\omega}$).

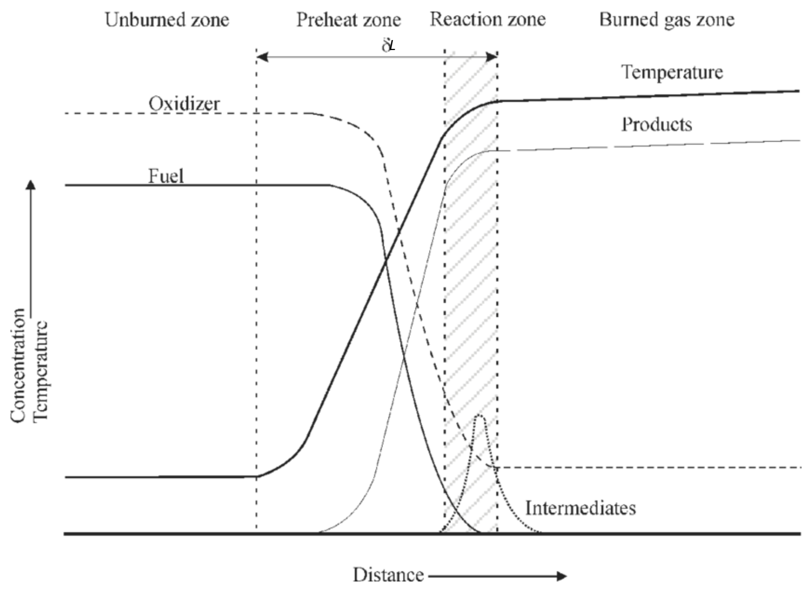


Figure 4-3: Schematic representation of premixed flame structure and characteristic profiles for temperature and concentrations [59].

The laminar flame speed mainly depends on the temperature of the inlet gas mixture T_u. An increase of T_u will strongly affect the total reaction rate by increasing it, while thermal diffusivity coefficient will also increase. Consequently, the laminar flame speed will increase..

4.3 Perfectly Stirred Reactor

Reactors are a standard layout for reactive flow simulations and validation of kinetic mechanisms. In combustion simulations, Perfectly Stirred Reactor (PSR) and Partially Stirred Reactor (PaSR) are extensively used. PSR is a constant volume system, in which happens an instantaneous and complete mixing of reactants, offering spatial homogeneity. As a result, the system thermochemical properties are defined exclusively by chemical kinetics. On the other hand, PaSR considers a large number of regions with different thermochemical states which are mixing.

The convention used in this report is that homogeneous reactors are basically perfectly stirred reactors, where complete mixing of fuel-oxidizer is achieved and, hence, transport properties are neglected and the system behavior is affected only by chemical kinetics. In other words, properties such as temperature, pressure and species concentrations can be considered constant throughout the reactor volume. Subsequently, the reactor can be mathematically modeled as a zero-dimensional configuration.

Homogeneous reactors can be modeled by open or closed volumes. Open reactors have inlet and outlet valves ensuring the continuous mixture flow, while closed reactors do not exhibit mass exchange with the environment. In the case of equal mass inlet and outlet, open reactors achieve steady state operating conditions and, except of no spatial distribution, neither time change of properties is observed. Closed reactors are usually applied for the evaluation of mixture's time change of temperature and species concentrations [53]. Fuel-oxidizer premixed mixture enters the reactor at time t=0, and time progress of temperature and species are calculated through the species and energy conservation equations.

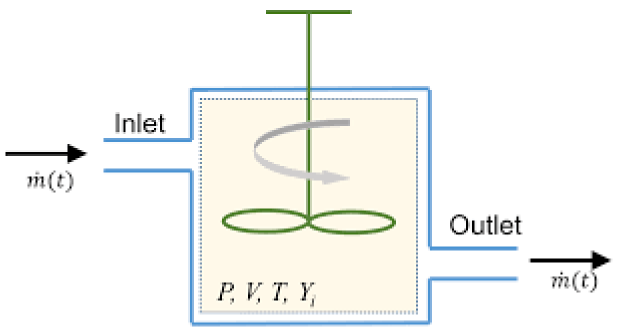


Figure 4-4: Homogeneous open reactor (open Perfectly Stirred Reactor), exchanging mass with the environment and ensuring continuous mixture flow [58].

A widespread method of computationally modeling combustion phenomena is the application of reactor networks. As the method's name suggests, several reactors are grouped into subsets and properly connected, generating a reactor network. Each subset is responsible for modeling a specific combustion zone, appearing similar flow and thermochemical properties. The reactors' combination into a network is thus claimed to sufficiently simulate the total combustion process.

4.4 CANTERA Simulations

The structure of steady 1-D premixed laminar flames can be computed numerically by solving the steady-state comprehensive mass, species and energy conservation equations coupled with a detailed kinetic mechanism, dealing with the reaction term. The momentum conservation equation is usually not necessary to be solved, as pressure remains approximately unchanged for the whole system. More precisely, burned gas pressure is slightly lower than unburned gas pressure, according to the Hugoniot analysis.

CANTERA is an open-source suite of tools for problems involving chemical kinetics, thermodynamics and transport processes. It can be used as a third-party library interpreted in external reactive flow simulation codes, compiled into Matlab, Python or C++ programming language. Among other things, it can be used to evaluate thermodynamic and transport properties of mixtures, compute chemical equilibrium, evaluate species chemical production rates, conduct kinetics simulations with large reaction mechanisms, as well as simulate one-dimensional flames and networks of stirred reactors [58].

The latter features are useful for the current implementation. More specifically, CANTERA includes a set of models for representing steady-state one-dimensional flows, such as freely propagating premixed laminar flames, burner stabilized premixed flames, counterflow premixed flames and counterflow diffusion flames. Furthermore, CANTERA can model zero-dimensional reactors and their interactions with the surroundings, as well as interconnect reactors into networks. CANTERA reactor configuration corresponds to an extensive thermodynamic control volume, in which all state variables are homogeneously distributed but at the same time are functions of time. All of these configurations are simulated using a set of governing equations within a zero or one-dimensional, respectively, flow domain, namely mass, species and energy conservation equations.

4.4.1 Premixed Laminar Flame Simulations

Premixed laminar flame simulations are performed to the flame structure, flame speed, flame thickness and temperature and species spatial distributions of freely propagating, one-dimensional, adiabatic premixed flames into isobaric, steady-state ideal gas flow.

The code for premixed laminar flame simulations is developed in Python language, using extensively the open-source CANTERA suite. The developed code firstly generates an ideal gas mixture set to user-defined upstream initial conditions, reading the desired kinetic mechanism. Next, it creates an one-dimensional flame object and solves it with a mixture averaged transport model. The solver is applied to an adaptive grid, meaning that a first estimation of solution is calculated using a minimum number of grid points and then continuous refinements follow until a sufficient convergence of calculated laminar flame speed is achieved.

The developed code is tested for methane-air premixed flame propagation. The chemical kinetics are described by GRI Mech 3.0, a detailed kinetic mechanism containing 53 species and 325 reactions. The initial state of the upstream mixture is set to p=1 atm and T=300 K. Fuel-air equivalence ratio (φ) ranges from 0.6 to 1.6 with step equal to 0.2. The grid for the numerical calculations is set to 4 cm.

First, flame speed and flame thickness are calculated and plotted in proper diagrams (Figure 4-5) as a function of fuel-air equivalence ratio. An approximation of laminar flame thickness can be calculated by the equation:

$$\delta_L \approx \frac{T_f - T_0}{\left(\frac{dT}{dx}\right)_{max}}$$

where T_f is adiabatic flame temperature and T_0 is initial mixture temperature.

It is observed that laminar flame speed for methane is maximum at slightly rich conditions (φ slightly larger than 1), for which the adiabatic flame temperature is also maximum. This observation verifies the strong effect of temperature. On the other hand, laminar flame thickness values are inversely proportional to laminar flame speed, and therefore are inversely proportional to temperature. The minimum laminar flame thickness is observed for stoichiometric conditions, where laminar flame speed is maximized.

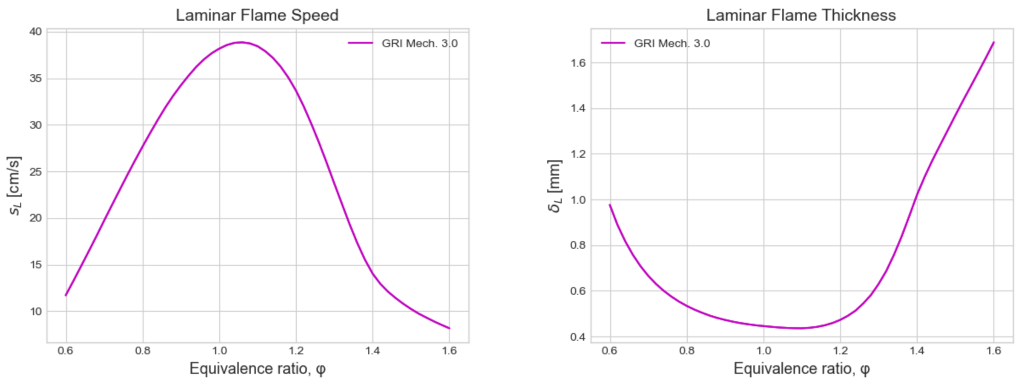


Figure 4-5: Laminar flame speed S_L [cm/s] (left) and laminar flame thickness δ_L [mm] (right) of methane-air premixed mixture at atmospheric conditions, as a function of fuel-air equivalence ratio.

Furthermore, the spatial distribution is calculated for species and for characteristic properties of flow field, such as temperature and velocity. The flame seems to be embedded in a specific grid region, meaning that the problem has been converted to a steady state. Unburned gas region, governed by low temperatures and velocities, is located to the left part of diagrams, while burned gas region, governed by high temperatures and velocities, is located to the right part of diagrams. This separation of regions can also be verified by observing the species spatial distribution diagrams. Unburned gas region refers to high mass fractions of fuel (CH_4) and oxidizer (air), while burned gas region refers to high mass fractions of main combustion products, namely carbon dioxide (CO_2) and water (H_2O).

The species selected to be displayed are methane (CH_4) and oxygen (O_2) as the main reactants (Figure 4-7) and carbon dioxide (CO_2) and water (H_2O) as the main products (Figure 4-8) of methane combustion. Additionally, carbon monoxide (CO) and hydroxyl radical (OH) are selected to be represented as species participating in main heat release processes (Figure 4-9). The species spatial distribution diagrams are partially enlarged around the region containing the thin flame layer.

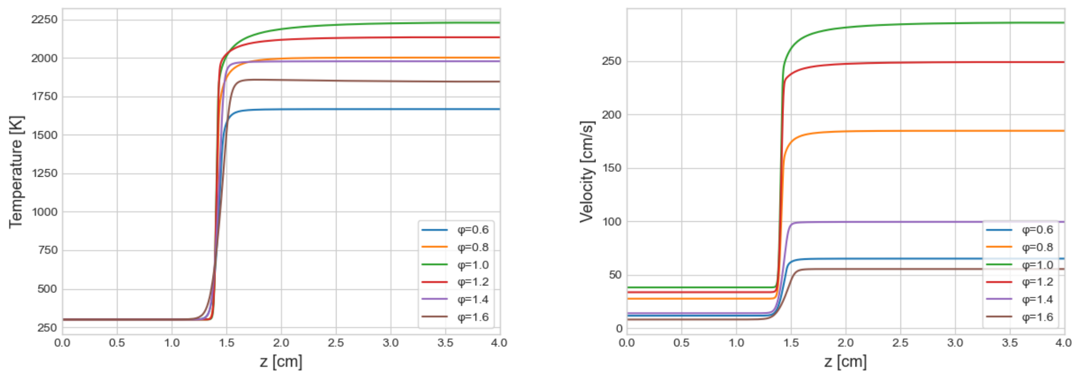


Figure 4-6: Temperature (left) and velocity (right) spatial distribution for methane-air flame. The upstream mixture is set to atmospheric conditions, while fuel-air equivalence ratio is considered as a parameter.

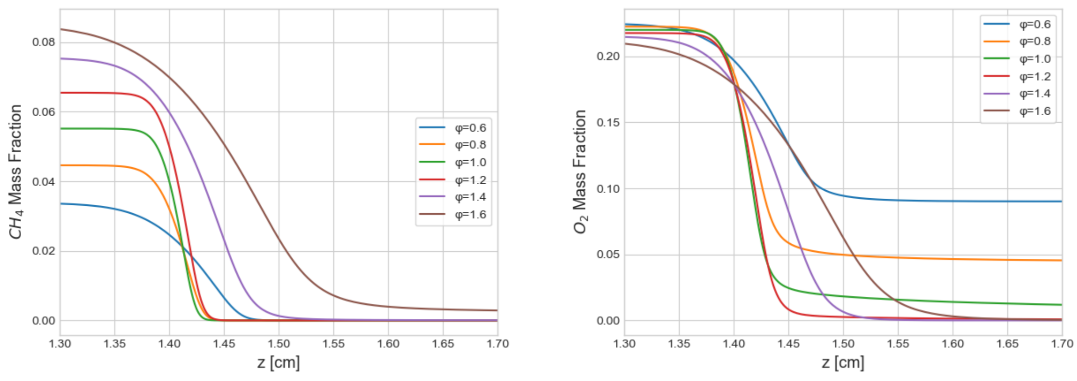


Figure 4-7: Methane (CH_4) and oxygen (O_2) mass fraction spatial distribution for methane-air flame. The upstream mixture is set to atmospheric conditions, while fuel-air equivalence ratio is considered as a parameter.

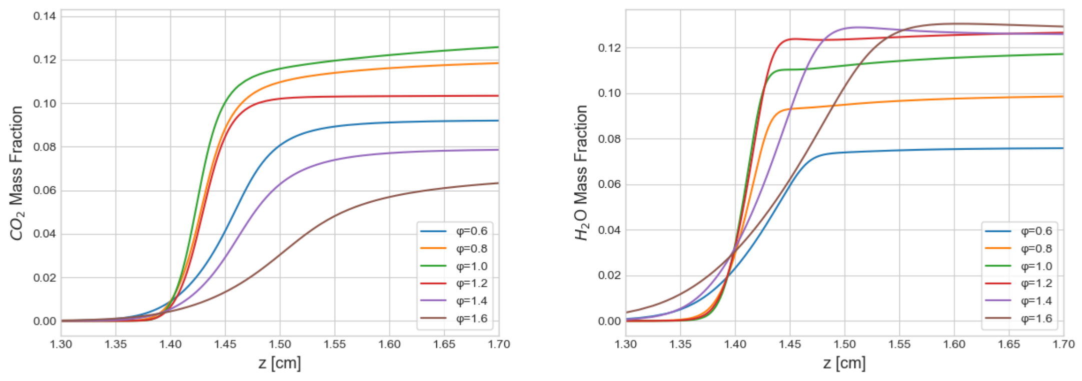


Figure 4-8: Carbon dioxide (CO_2) and water (H_2O) mass fraction spatial distribution for methane-air flame. The upstream mixture is set to atmospheric conditions, while fuel-air equivalence ratio is considered as a parameter.

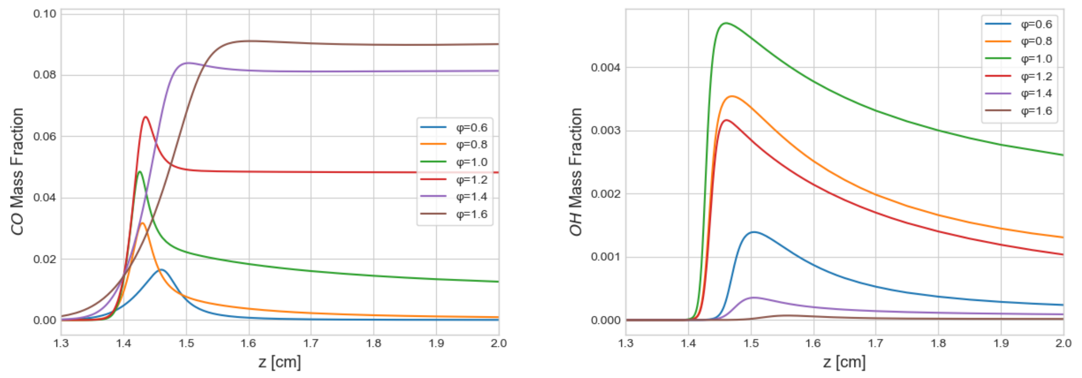


Figure 4-9: Carbon monoxide (CO) and hydroxyl radical (OH) mass fraction spatial distribution for methane-air flame. The upstream mixture is set to atmospheric conditions, while fuel-air equivalence ratio is considered as a parameter.

The above results are compared and validated with study material from the respective course of Combustion at the Department of Naval Architecture and Marine Engineering of the National Technical University of Athens, and also verified in the diploma thesis of *Kallieros* [57].

4.4.2 Homogeneous Reactor Simulations

A characteristic problem simulated by a simple batch reactor is how a homogenous chemical composition changes in time when it is left to its own devices. Simulations are performed for a homogenous, zero-dimensional reactor with adiabatic, chemically inert walls. For the specific implementation, a constant pressure reactor is considered, meaning that reactor volume changes as a function of time in order to keep the system pressure constant. A reactor network is required in order to perform time integration. Temperature and species time evaluation are calculated, while mass flow remains constant through the network.

The code for homogeneous reactor simulations is developed in Python language, using extensively the open-source CANTERA suite. The developed code firstly generates an ideal gas mixture set to user-defined initial conditions, reading the desired kinetic mechanism. The initial state is specified by composition and a set of thermodynamic parameters, like pressure and temperature. Upon this base, an ideal gas constant pressure reactor is created. The behavior of the solution in time can be simulated as a reactor network containing only the formerly created reactor.

The pre-defined premixed mixture enters the reactor at time t=0 sec. A final time shall also be determined before the simulation starts, while the time step can be defined sufficiently low or use instead an adaptive time step. Afterwards, the reactor network advances in time from the current time for the specified time step. This procedure is repeated until final time is reached and time integration is completed.

A problem simulated by the developed code is ignition delay for methane-air mixtures, for a range of initial conditions considered by *Kallieros* in his diploma thesis [57]. The chemical kinetics are described by GRI Mech 3.0, a detailed kinetic mechanism containing 53 species and 325 reactions. Ignition delay can be defined as the time when the rate of increase of temperature, pressure or a radical concentration is maximized. The criterium applied by the developed code is maximization of temperature's rate of increase. A characteristic diagram of temperature time evolution is presented below (Figure 4-10). In the same diagram, ignition delay time is also marked following the above definition.

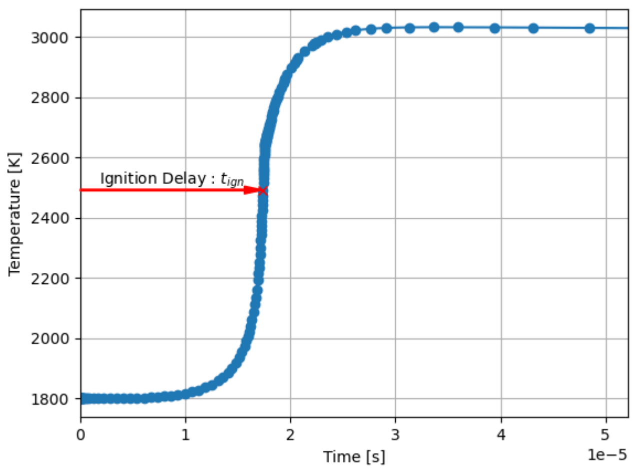


Figure 4-10: Time evolution of temperature in a constant pressure reactor, for a methane-air mixture. Initial mixture properties are p=9.22 atm, T=1800 K and φ=1. Ignition delay time is defined as the time when maximization of temperature rate of increase occurs. For the specific mixture properties, $t_{ign}=1.74\cdot10^{-5}$ s.

CHAPTER 5: Clustering of State Space

For the purposes of the current implementation, a hybrid neural network is developed in order to subdivide the state space into subdomains. The term “hybrid” is used because in total three Machine Learning methods are combined to construct the neural network: Principal Component Analysis (PCA), Self-Organizing Map (SOM) and K-Means. The basic neural network of this hybrid configuration is the Self-Organizing Map (SOM), which is used to divide the state space into several subdomains. It consists of a two-dimensional grid of neurons, the so-called map. Each neuron is represented by a vector and accounts for a cluster center in state space.

At the start of the training, the SOM neurons’ weights are randomly initialized. Due to the high dimensionality of the state space, depending on the number of thermochemical features describing the phenomenon, this random procedure may not lead to convergence, or at least fast convergence, of the training process. Therefore, Principal Component Analysis (PCA) precedes, in order to reduce the dimensionality. Initial neurons’ weights are then selected in this lower-dimensional space and, afterwards, are inversely transformed back to the input high-dimensional space, in order for the training process to start. It is highlighted that PCA is only applied during the first step of the SOM training process and finds no further usage during the simulations.

The SOM training procedure refers to the adjustment of neurons’ weights, so that each neuron represents an apparent center of a cluster in state space. The trained SOM is considered to be able to subdivide the multi-dimensional state space according to similarities, preserving the input space topology. In other words, data points lying near each other in the state space are mapped onto nearby winning neurons in the two-dimensional SOM grid. After the training process is completed, new input data can be classified into a cluster by the nearest neuron in terms of Euclidean distance. It is important to note here that SOM dimensions do not correspond to physical variables, but they construct a grid of cells which represent parts of state space.

One question appearing here is how to choose the number of areas in which to split the state space, or equally the number of cells in the SOM grid. Extreme partitioning has its drawbacks, even though it will probably offer a more refined regression, as a large number of cells will result in areas with few samples and uneven regression quality. In other words, the number of observations in each cluster obtained from SOM clustering may be too low to have physical or statistical significance. Furthermore, the predictive power of regression neural networks is expected to deteriorate at points lying at the boundaries between two subdomains. On the other hand, the use of a small number of regression neural networks will lead to overfitting problems, due to the complexity of the functions to fit. A good balance shall be found between these trends, accounting for complexity and user-defined accuracy at the same time.

For certain applications, the number of clusters obtained through SOM may be much higher than practical. To reduce the number of clusters, the K-Means algorithm is a useful method to apply in order to group the SOM’s neurons into a desirable number of clusters, achieving a balance between network’s accuracy and complexity. The number of clusters can be user- or a priori defined through index analysis, such as Silhouette Coefficient and Davies-Bouldin index.

The primary benefits of this two-level approach, in lieu of only clustering the data by means of SOM, are the reduction of computational cost and the rough visual presentation and interpretation of clusters on a 2D grid. Furthermore, clustering the SOM manually or grouping cells of the 2D array into rectangular clusters may decrease the quality of clustering.

Finally, a regression neural network is assigned to each subdomain for the purposes of prediction. In the current implementation, Multi-Layer Perceptrons (MLPs) are used as a regression model. The input of each one MLP is a state vector, containing the temperature and the species mass fractions, and assigned to a specific state space subdomain by means of SOM – K-Means concept. Regarding the output of MLPs, direct prediction of heat release rate and species net production rates, as well as the integrated value of the state over a fixed time step are analyzed in the present study. The whole neural network concept is illustrated schematically in Figure 5-1.

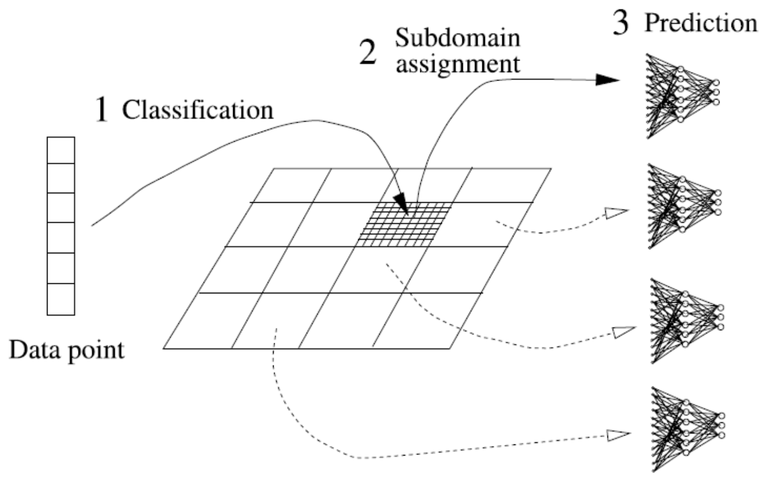


Figure 5-1: Schematic representation of the whole neural network implementation, illustrating the overall procedure: (1) Classification of an arbitrary data point by SOM – K-Means, (2) assignment of an MLP on a subdomain and (3) prediction using MLP [8].

The state space partitioning must be done on data obtained from simulations of the same phenomenon or, alternatively, from simulations of an abstract problem. Regarding the latter case, it is important to ensure that the generated training dataset covers the state space accessed during the multi-dimensional CFD simulation. An ensemble of premixed laminar flames is used to generate the training data for the current implementation.

5.1 Training Data Generation

The generation of training data for the Self-Organizing Map is carried out by means of one-dimensional premixed laminar flames. Premixed laminar flame is considered a canonical problem, such that the training data can span the state space of a family of combustion problems. The current study deals with methane-air premixed combustion. The simulations were executed using the CANTERA open-source suite in a Python code, as described in a previous section of the present report (Section 4.4).

In the simulations, the pressure is held constant with a value of 1 atm. The inlet temperature is between 300 K and 400 K. Fuel-air equivalence ratio ranges from 0.75 up to 1.25. The selection of fuel-air equivalence ratio ranging around the value of 1 is justified by the fact that natural gas combustion engines mainly operate in Otto cycle, which performs effectively near stoichiometric conditions. With regard to chemical kinetics, the GRI Mech 3.0 mechanism is used, containing 53 species and 325 reactions. Therefore, a single point in state space is described by a vector of N_s+1 = 54 variables; the temperature and the 53 species mass fractions.

Table 5-1: Initial conditions for 1D premixed laminar flame simulations.

	Initial Conditions Range	**Step**
Pressure	1 [atm]	-
Temperature	300 – 400 [K]	10 [K]
Fuel-air equivalence ratio	0.75 – 1.25	0.05

Through each premixed flame simulation, several properties are obtained in terms of spatial distribution, such as velocity, density, temperature and species mass fractions. However, the present concept is based on state space partitioning, meaning that only temperature and species mass fractions are needed to generate a state vector, while all other features are discarded. Spatial variable is unambiguously connected with temperature, meaning that the spatial distribution of species mass fractions can be converted to temperature distribution, because each spatial point in flame is characterized by a specific temperature. Indicative figures are presented, and correspond to stoichiometric methane-air premixed laminar flame simulation, with pressure and initial temperature set to atmospheric conditions.

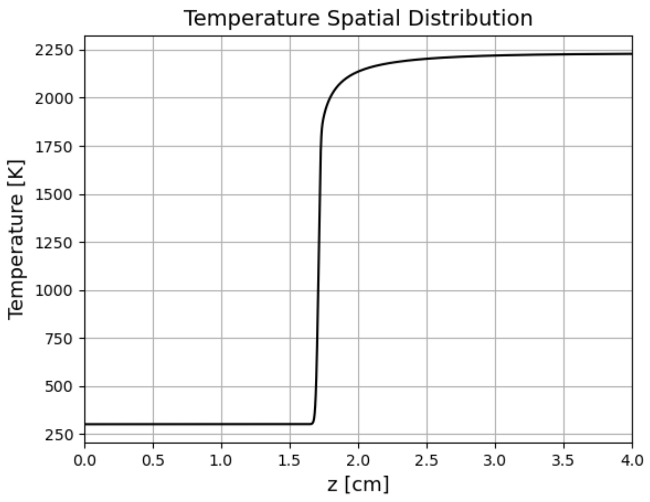


Figure 5-2: Temperature spatial distribution for a stoichiometric methane-air premixed laminar flame at atmospheric conditions.

It must be emphasized that premixed laminar flames are employed here only for sampling a set of compositions. These compositions are thoroughly shuffled before starting the training process. In other words, the spatial variable is discarded, as the current concept requires only combinations of temperature and species mass fractions in order to create composition state vectors. The conversion from spatial to temperature distribution of species mass fractions is an unambiguous and well-defined procedure. Figures presented next depict the acquired main species mass fractions indicatively through a stoichiometric methane-air premixed laminar flame (Figures 5-3, 5-4). As it can be easily observed, CANTERA uses an adaptive grid that is denser in the burned gas region and particularly in the reaction region.

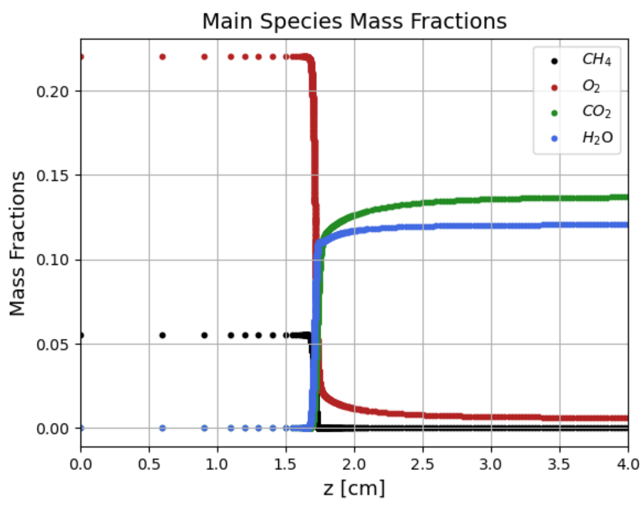


Figure 5-3: Main species mass fractions acquired data points based on spatial distribution for a stoichiometric methane-air premixed laminar flame at atmospheric condition.

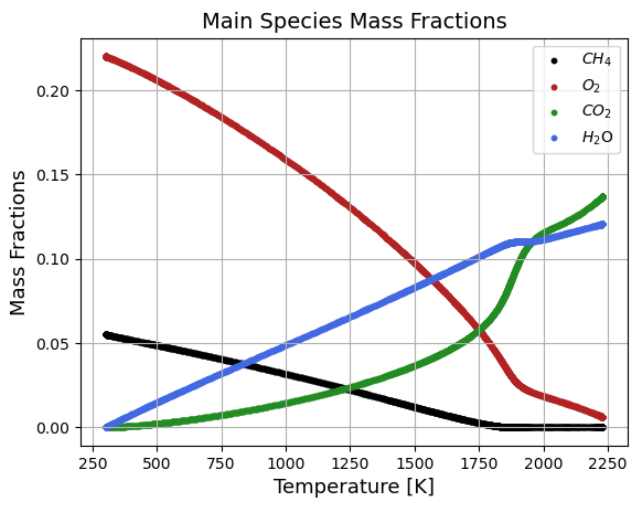


Figure 5-4: Main species mass fractions plotted with respect to temperature for a stoichiometric methane-air premixed laminar flame at atmospheric condition.

The aforementioned simulation conditions correspond to 121 methane-air premixed laminar flames, generating in total 127099 data samples. The data samples are then randomly split into 60% training set and 40% test set. A notable observation is that the produced dataset has no outliers, as it is generated through robust direct numerical simulations. Therefore, there is no need for application of data cleaning techniques before proceeding to neural network training.

Figure 5-5 presents the state space coverage from the full dataset obtained through the laminar flame simulations. The diagrams presented are indicative of the temperature distribution of main reactants (CH_4, O_2), main products (CO_2, H_2O) and intermediate species/ radicals (CO, OH) supporting the main heat release of methane-air mixture combustion process.

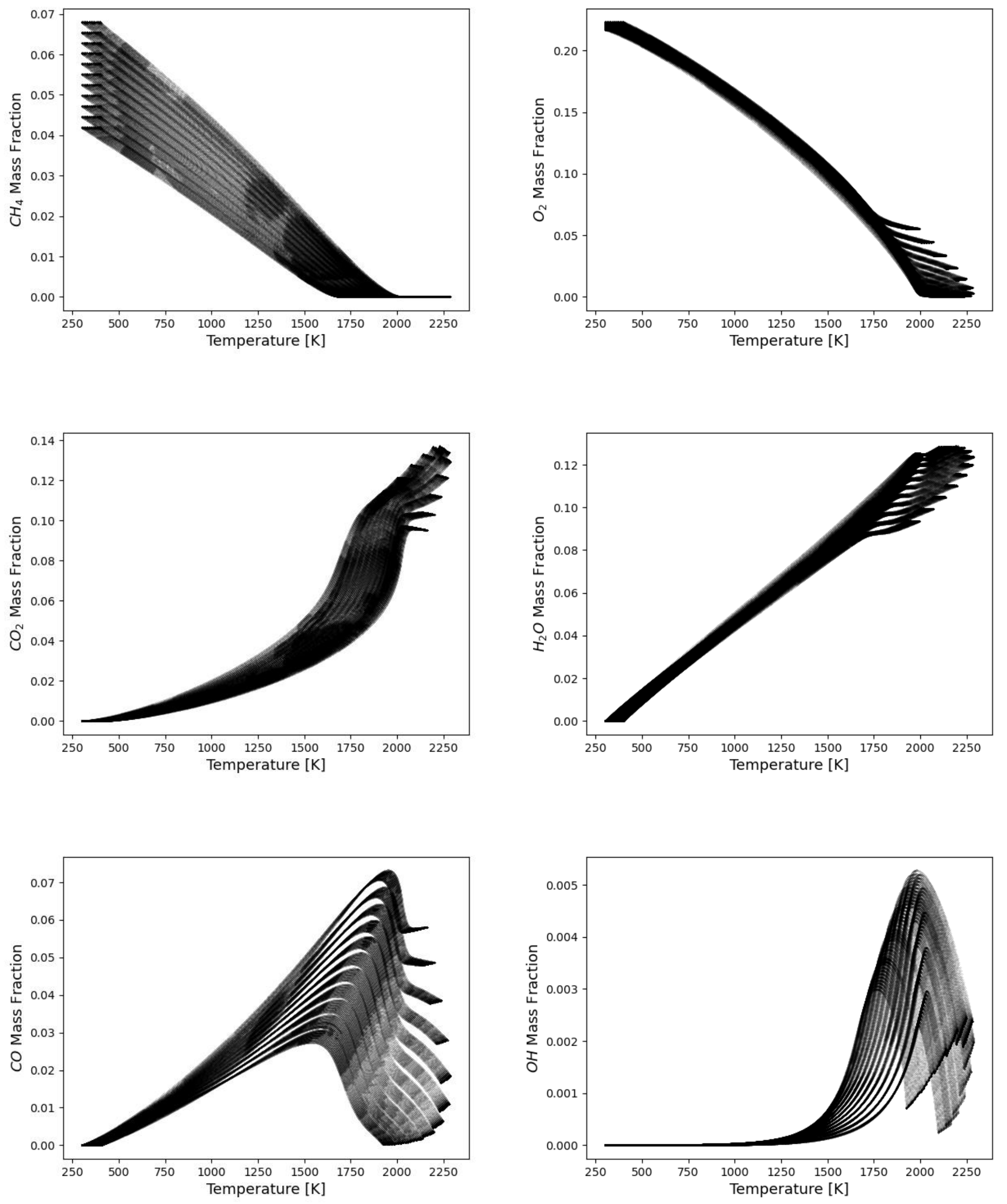


Figure 5-5: Scatter plots representing the state space coverage of the data collected from methane-air premixed laminar flame simulations.

5.2 Self-Organizing Map Training Process

The hybrid neural network applied for partitioning of state space in the current study consists of three different Machine Learning methods; Principal Component Analysis (PCA), Self-Organizing Map (SOM) and K-Means. The SOM and K-Means neural networks must be separately trained in a serial procedure, using the training data generated via premixed laminar flame simulations.

A common technique followed by machine learning researchers to split the full dataset into a training set, used for the neural network training, and the test dataset, used for evaluating the performance of the trained network. Regarding the current implementation, trial-and-error showed that random splitting of the dataset obtained from premixed laminar flame simulations to a rate higher than 50% has little effect on the final adjusted neurons. In other words, the training procedure converges in approximately the same neurons weight values for any or no a priori splitting of the dataset. Therefore, the initial dataset is randomly split in 60% training set and 40% test set, in order to sufficiently reduce the time and the computational cost for training.

As aforementioned, a single point in state space is described by a vector of $N_s+1 = 54$ variables with respect to GRI Mech 3.0; the temperature and the 53 species mass fractions. The species mass fractions vary within the range between 0 and 1, whereas the temperature variable obtains values from 300 K up to higher than 2000 K. In order to result in a robust clustering neural network and avoid neglection of characteristics with low values, all features are rescaled to the same order of magnitude.

First of all, temperature is normalized into the [0,1] range. The progress variable *c* is used for the normalization:

$$c = \frac{T - T_{unb}}{T_{max} - T_{unb}}$$

where T is the temperature desired to be normalized, T_{unb} is the unburned mixture temperature and T_{max} is the maximum temperature along the flame.

It is notable to highlight that adiabatic flame temperature is typically used instead of maximum temperature (T_{max}) for the calculation of progress variable, *c*. Consequently, the maximum value of progress variable, *c*, may be not be equal to 1, but slightly lower or larger than unity, when using adiabatic flame temperature instead of maximum temperature, as super-adiabatic temperature can exist in the flame. For this reason, maximum temperature is used instead of adiabatic flame temperature.

Two cases of input data normalization are examined in this study:

- Case 1: Temperature is normalized using progress variable *c*, while species mass fractions remain as computed. Major species have relatively high values of mass fractions, while minor species and radicals acquire values near zero.
- Case 2: Temperature is normalized using progress variable, *c*, and species' mass fractions are scaled into the [0,1] range.

A major characteristic of Case 1 is that the training process will incline to features with large values. The trained neural network will better approximate these features, while variables with values near zero are not considered important. On the other hand, a neural network trained with input data normalized according to Case 2 will consider all features with equal importance. The training process will adjust properly the neuron weights in order for the network to sufficiently approximate all the features. In each case, the respective inverse scaling transformation can then follow to restore data values into composition space.

In the following paragraphs, the training procedure is described for each of the applied Machine Learning methods. The results for each examined case of training data normalization, namely Case 1 and Case 2, are also presented in each respective paragraph. Finally, two clustering neural networks will be developed, corresponding to each one examined case of training data normalization.

5.2.1 Principal Component Analysis

Principal Component Analysis is used as a dimensionality reduction method only during the training process. Specifically, it reduces the high-dimensional state space into a lower-dimensional one, in order to properly select the initial values of SOM's neuron weights. After the selection, the weight values are transformed back to the high-dimensional state space and the SOM training process is ready to start.

For the needs of the present work, the 54-dimensional input state space is decomposed into the first two principal components by means of PCA method. The consideration of 2 principal components for the dimensionality reduction is justified by the amount of variance they account for. By default, PCA tries to put the largest possible variance (maximum possible information) in the first component, then maximum remaining variance in the second, and so on. Figures 5-6 and 5-7 present the percentage of explained information per principal component and per cumulative number of principal components for both examined cases of input data normalization. As mentioned, Case 1 corresponds to temperature normalized into [0,1] range and the rest 53 mass species fractions remaining unchanged, while Case 2 corresponds to temperature and species mass fractions scaled into the [0,1] range.

Regarding Case 1 of training data normalization, the first principal component contains 99.21% of the total information, while the second one accounts for 0.48%. The cumulative variance explained by the first two principal components is 99.69%, and it is considered very high for the purposes of a neuron weights initialization technique (Figure 5-6). The ability to describe such a high percentage of information by means of only one principal component is justified by the fact that only the temperature variable is distributed in the [0,1] range, while all other features have the values as such acquired through premixed laminar flame simulations, which are certainly smaller than 1 and many of them nearly equal to zero.

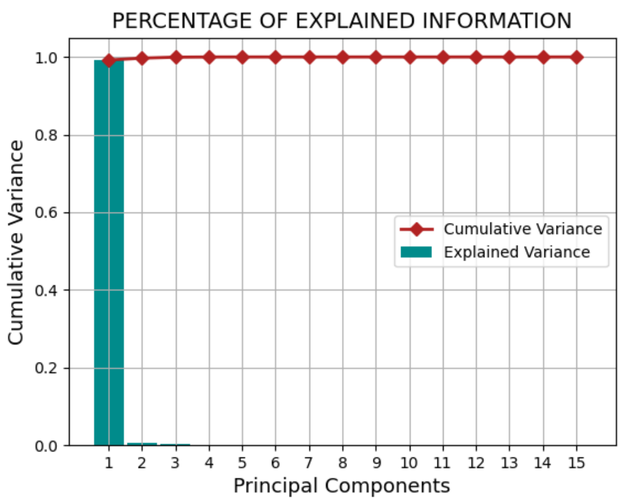


Figure 5-6: Percentage of explained information per number of principal components (explained variance) and percentage of information per cumulative number of principal components (cumulative variance), for **Case 1** of data normalization.

Regarding Case 2 of training data normalization, the first principal component contains 35.62% of total information, while the second one contains 34.09% of total information. The cumulative variance contained by the first two principal components is 69.71%, and it is considered sufficient for the purposes of a neuron weights' initialization technique (Figure 5-7).

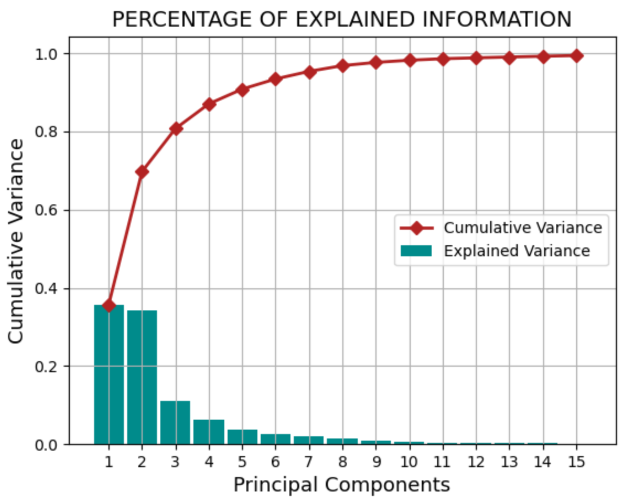


Figure 5-7: Percentage of contained information per number of principal components (contained variance) and percentage of information per cumulative number of principal components (cumulative variance), for **Case 2** of data normalization.

PCA is applied to span the 54-dimensional data generated through premixed laminar flame simulations into a 2-dimensional space defined by the first two principal components, as they are considered sufficient to describe the information contained in the training dataset. Afterwards, a nested function of the MINISOM library is applied to initialize the neurons' weights by creating a 2-dimensional grid relative to the space defined by the first two principal components. MINISOM [60] is an open-source library implementing Self Organizing Maps, and it is further analyzed in the Section 5.2.2. The results of dimensionality reduction and neurons weights initialization are plotted in Figures 5-8 and 5-9, for Case 1 and Case 2.

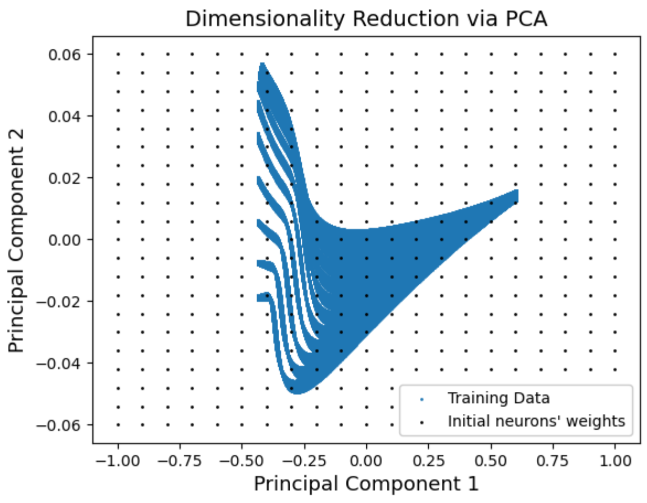


Figure 5-8: Training data and initial neuron weights in the 2-dimensional space defined by the first two principal components for **Case 1** of data normalization.

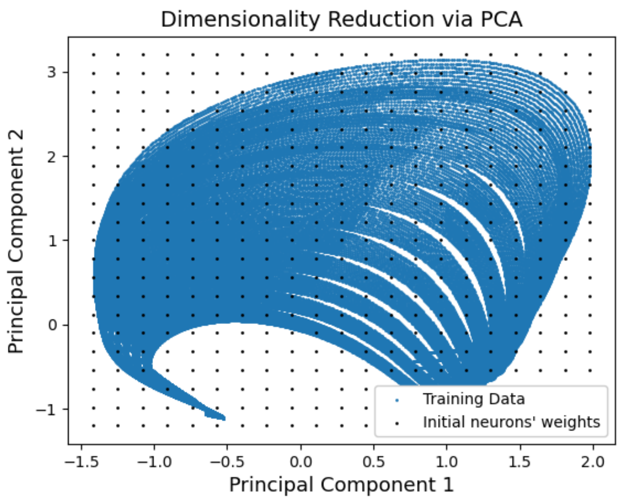


Figure 5-9: Training data and initial neuron weights in the 2-dimensional space defined by the first two principal components for **Case 2** of data normalization.

5.2.2 Self-Organizing Map

After the neuron weights initialization via PCA and inverse transformation of the weights back to the 54-dimensional state space, the training process of SOM is ready to start. In the present study, the MINISOM library is used for the SOM training process [60]. The code is developed in Python language.

There is not much literature and research on network architecture selection. A common configuration are two-dimensional rectangular grids, extensively used for state space partitioning. Each cell of the rectangular grid represents a neuron. *An et al.* [9] used 30x30, 40x40 and 50x50 SOM neurons to model turbulent hydrogen/carbon monoxide/kerosene combustion. *Blasco et al.* [8] and *Franke et al.* [10] employed a 20x20 grid for methane-air combustion data tabulation. *Chatzopoulos and Rigopoulos* [11], who conducted a study in the same context, used a 40x40 SOM configuration.

According to available literature and taking into account the size of the GRI Mech. 3.0 and the fact that the training data are generated through premixed laminar flame simulations and no turbulence exists, a 20x20 SOM configuration seems to be sufficient for the current implementation. Larger configurations were also tried (30x30 and 40x40 grids), but the network did not use all the available neurons for the clustering process. This was considered as an indication that a 20x20 configuration is appropriate for partitioning the state space of this study.

For a Self-Organizing Map neural network, the hyperparameters that need to be a priori defined are the standard deviation or neighborhood radius (σ) and the learning rate (lr). In fact, these hyperparameters are monotonically decreasing functions of iteration number, a choice resulting in robust convergence of the training process and reduction of weight values modification as the iteration index increases. They are defined as:

$$\sigma(t) = \frac{\sigma}{1+\frac{t}{T}} \quad \text{and} \quad \mathrm{lr}(t) = \frac{\mathrm{lr}}{1+\frac{t}{T}}$$

where t is the iteration number and T is half of the total number of iterations. Therefore, the initial values of the standard deviation, σ, and learning rate, lr, are only to be pre-defined. By definition, learning rate ranges between 0 and 1, whereas there is no limit on the standard deviation values.

The hyperparameters are selected iteratively. The SOM is trained for a set of hyperparameters combinations, and then it is evaluated by studying the quantization error, which is a measure of the average distance between data points and their respectively winning neurons (Section 3.2.2). A "golden section" is expected in terms of quantization error minimization, as large initial values of σ and lr may cause high modifications in neuron weights even when the iteration index is large, whereas lower values may not be sufficient for a proper neuron adjustment.

Regarding Case 1 of training data normalization, the procedure resulted in a minimum quantization error for standard deviation σ = 0.7 and learning rate lr = 0.1 (Figure 5-10). For smaller values of standard deviation, the quantization error diverges and acquires values too high to be depicted in the range shown in Figure 5-10. For larger values of standard deviation, the quantization error tends to converge in a specific value but at the same time is increased. Furthermore, a general observation on learning rate is that if high initial values are selected, the quantization error also increases.

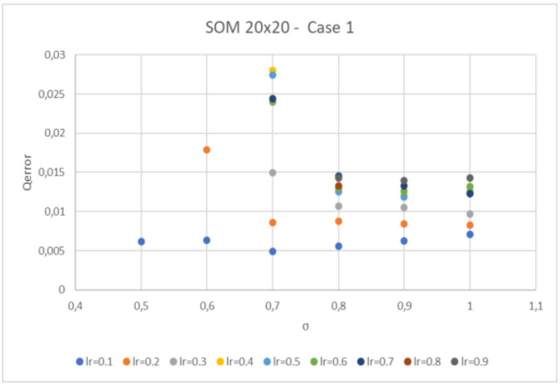


Figure 5-10: Quantization error as a function of network's hyperparameters, namely standard deviation (σ) and learning rate (lr), for **Case 1** of data normalization.

Apart from the network's hyperparameters, the total number of iterations for the training process shall be defined. The selection is based on quantization error and execution time. Evaluating the following diagrams (Figure 5-11), the training process is selected to be executed for 10^6 iterations, balancing quantization error minimization and execution time reduction.

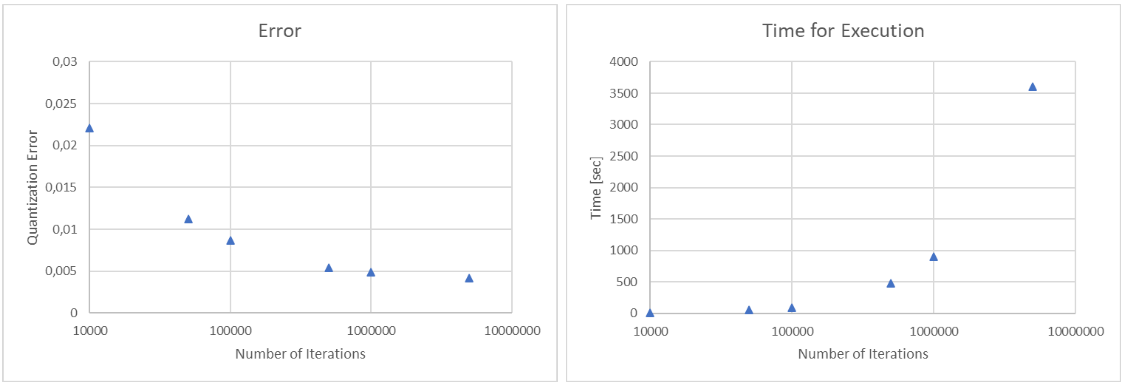


Figure 5-11: Quantization error (left) and execution time of training process (right), as a function of number of iterations, for **Case 1** of data normalization.

Regarding Case 2 of training data normalization, the iterative procedure resulted in a minimum quantization error for standard deviation σ = 0.8 and learning rate lr = 0.15 (Figure 5-12). The trends of quantization error observed are the same as for Case 1 of training data normalization. For smaller values of standard deviation, the quantization error diverges and acquires higher values. For larger values of standard deviation, the quantization error tends to converge but at the same time is increased. Furthermore, a general observation on learning rate is that if high initial values are selected, the quantization error also increases.

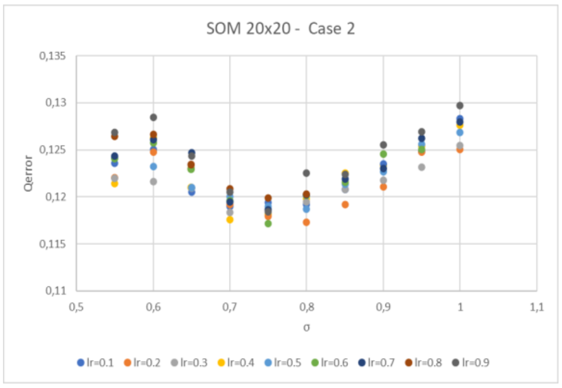


Figure 5-12: Quantization error as a function of network's hyperparameters, namely standard deviation (σ) and learning rate (lr), for **Case 2** of data normalization.

As for the total number of iterations for the training process, the selection is similarly based on quantization error and execution time. Evaluating the following diagrams (Figure 5-13), the training process is selected to be executed for 10^6 iterations, balancing quantization error minimization and execution time reduction.

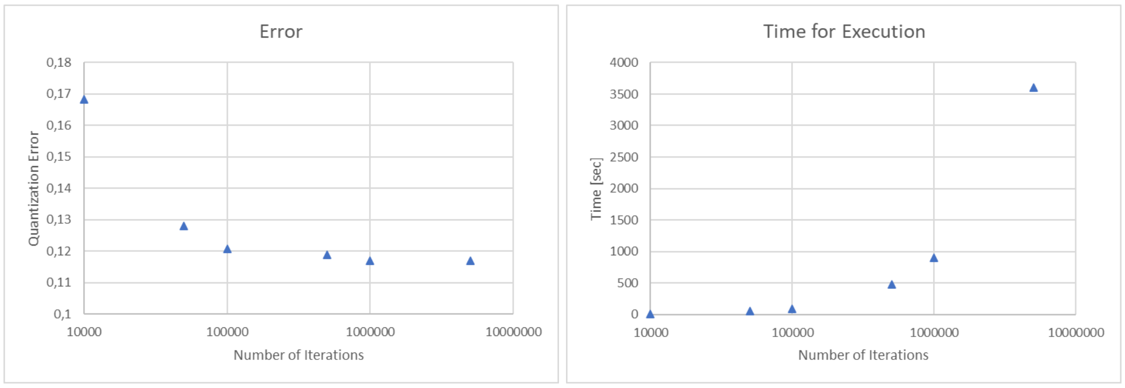


Figure 5-13: Quantization error (left) and execution time of training process (right), as a function of number of iterations, for **Case 2** of data normalization.

For the purposes of current implementation, the SOM training process was conducted in a Linux environment on a personal desktop, running on an i3 processor.

Consequently, two Self-Organizing Maps are developed in total, each one regarding a case of training data normalization. The basic characteristics as well as the selected network's hyperparameters are summarized in Table 5-2.

Table 5-2: Basic characteristics of developed SOMs for each case of data normalization.

	Case 1	**Case 2**
Characteristics	Normalized temperature	Normalized Temperature, Mass fractions scaled into [0,1] range
SOM Architecture	20x20	20x20
Standard Deviation, σ	0.7	0.8
Learning Rate, lr	0.1	0.15
Number of Iterations	10^6	10^6
Time for Training Process Execution	app. 900 [s]	app. 900 [s]
Quantization Error, Q_{error}	$4.9 \cdot 10^{-3}$	0.117

5.3 Self-Organizing Map Partitioning of State Space

Self-Organizing Map was selected for clustering the state space. After the training process is finished, the trained SOM can assign any new sample to a specific subdomain, defined by the neuron with the smallest Euclidean distance from the input sample.

SOM neurons are basically vectors of equal dimension such as the state space dimension. In the present study, GRI Mech. 3.0 is used for the purposes of chemical kinetics modeling, containing 53 species. Pressure is held constant (atmospheric), while temperature is considered as a variable. Therefore, the state space is defined by 54 dimensions, and so are the dimensions of SOM neurons.

Figures 5-14 and 5-15 present the state space coverage from the full dataset obtained through premixed laminar flame simulations, before splitting into training and test set, as well as the adjusted SOM neuron weights, accounting for the cluster centers, for both cases of training data normalization. The plots are indicative for the progress variable distribution of the main reactants (CH_4, O_2), main products (CO_2, H_2O) and intermediate species/radicals (CO, OH) supporting the main heat release of the methane-air mixture combustion process.

Each neuron represents the center of a cluster, in which the assigned samples have similar thermochemical properties, specifically temperature and species mass fractions. For some values of the temperature progress variable *c* and CO mass fraction, some neurons seem to diverge from the state space accessed by dataset, since the SOM does not take into account equally all features, but emphasizes on the ones with the higher numerical values. This is not considered a failure of the training process, because the importance of this concept lies in the cluster defined by each neuron. More specifically, the multi-dimensional center of a cluster may be outside in some dimensions of the state space accessed by the dataset generated through premixed laminar flame simulations, but the interest lies on the cluster which is defined by the specific neuron and in which a new input sample will be assigned, a procedure executed successively and sufficiently by the current SOM neural network. In addition, the state space will be further clustered by applying K-Means method on SOM grids, in order to achieve a reduced and manageable number of clusters, so the outbound neurons play a less important role.

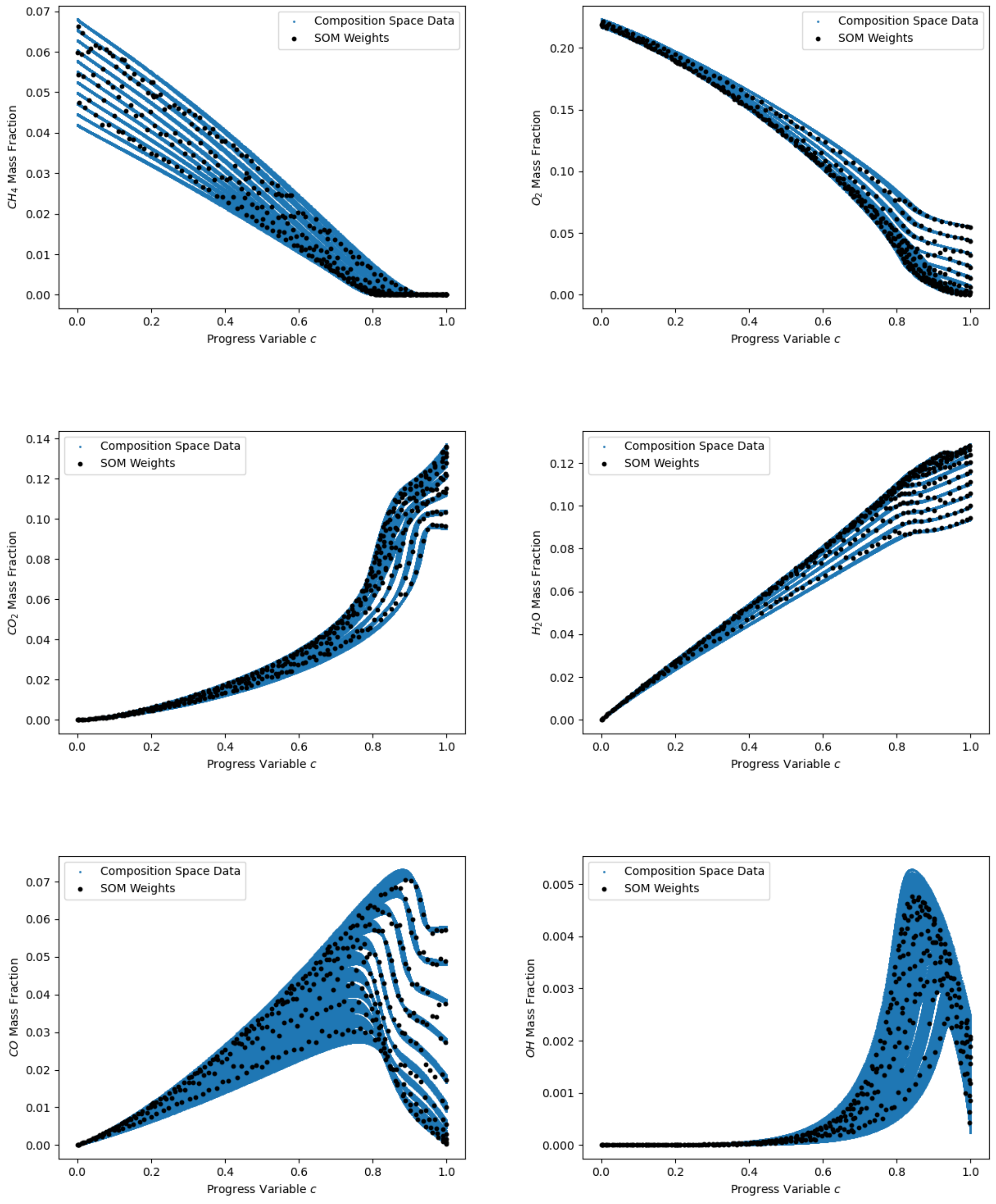


Figure 5-14: Composition points accessed by dataset and SOM weight vectors, for **Case 1** of data normalization.

As already mentioned, a neuron weight vector has as many components as the SOM input variables, or else, as the input data features, i.e. 54 in the present case. Therefore, the values of the vector components are a useful indicator of how the multi-dimensional state space is projected onto the two-dimensional SOM grid. In this context, distribution maps are used, which are basically contour plots of the values of selected components plotted onto the two-dimensional SOM grid. Each cell of the grid represents a neuron, and it is colored according to the weight value of the variable under consideration.

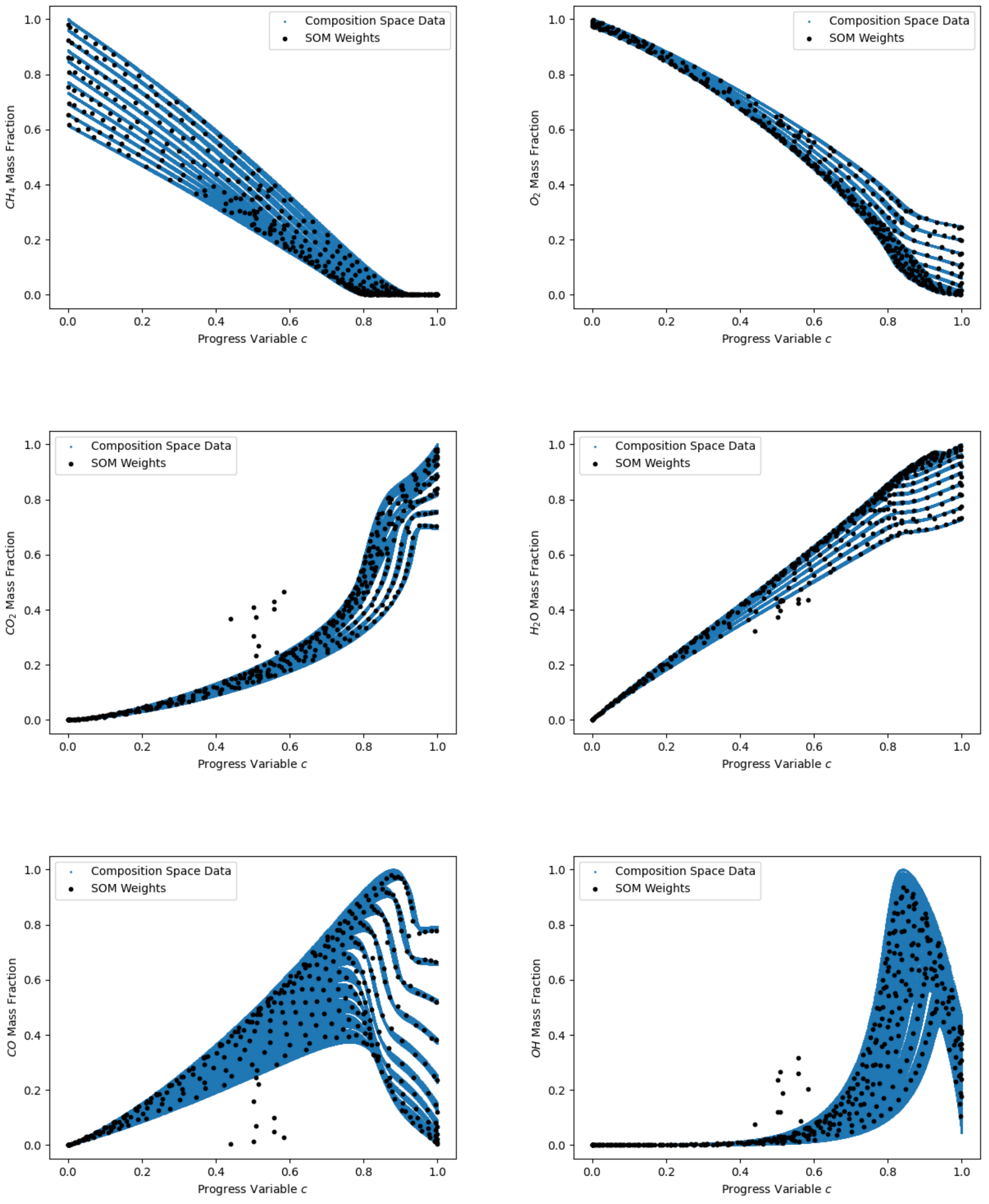
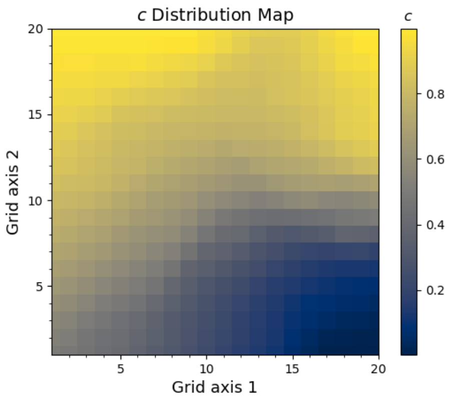
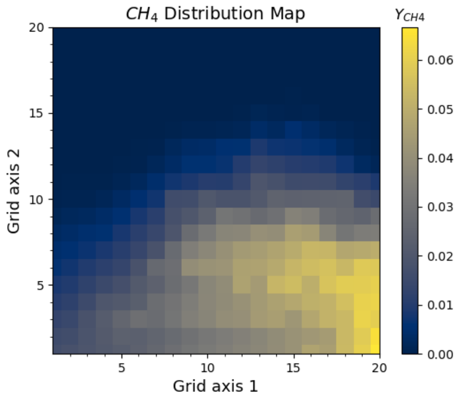
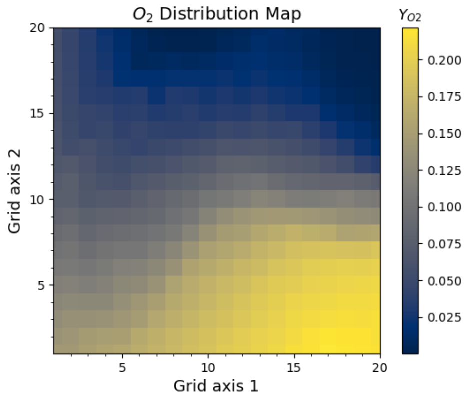


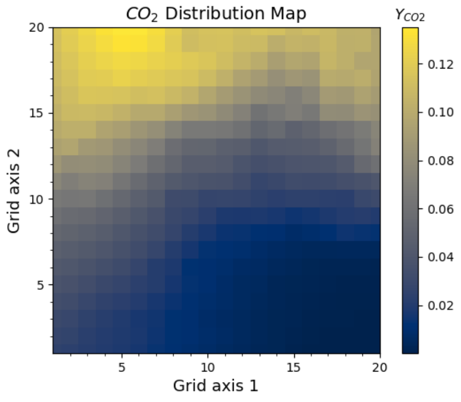
Figure 5-15: Composition points accessed by dataset and SOM weight vectors, for **Case 2** of data normalization.

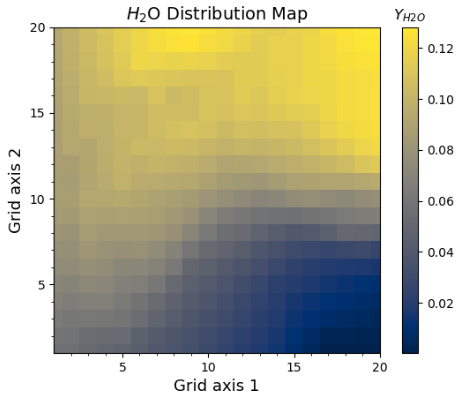
Distribution maps constitute a tool of optical observation, correlation and evaluation of features after the clustering procedure is finished by the SOM. The diagrams selected to be presented in the current report are indicative and account for the temperature distribution, expressed by progress variable, *c*, distribution of main reactants (CH_4, O_2), main products (CO_2, H_2O) and intermediate species/radicals (CO, OH) supporting the main heat release of methane-air mixture combustion process. Figure 5-16 corresponds to Case 1 of training data normalization, where temperature is normalized via progress variable *c* but all other features remain unchanged, whereas Figure 5-17 corresponds to Case 2 of training data normalization, where temperature is also normalized via progress variable *c* and each species mass fraction is scaled into the [0,1] range.

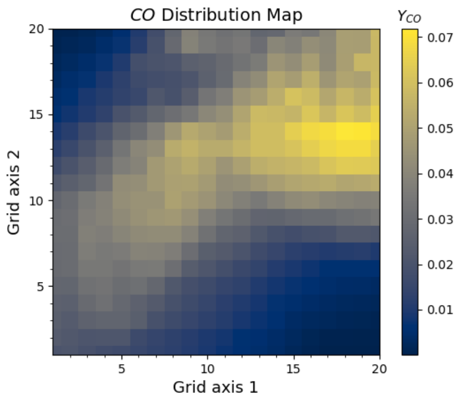












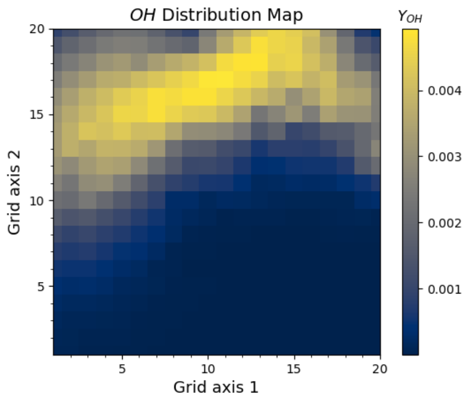


Figure 5-16: SOM distribution maps for **Case 1** of data normalization.

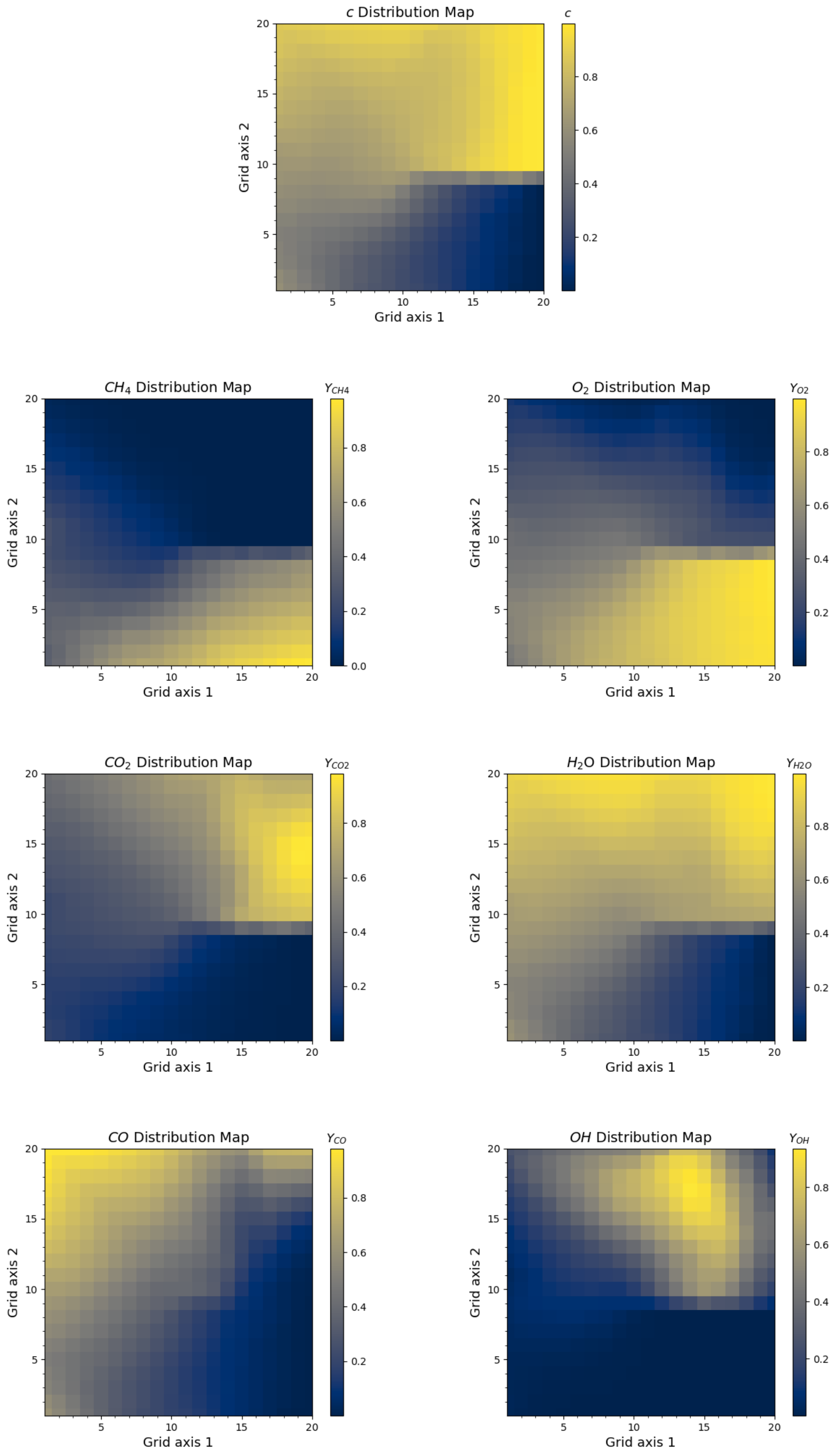


Figure 5-17: SOM distribution maps for **Case 2** of data normalization.

Some general observations can be made in both figures. Lower temperatures (lower values of progress variable, *c*) correspond to high CH_4 mass fraction and O_2 mass fraction, since the fuel is not yet consumed, and low CO_2 mass fraction and H_2O mass fraction, as the main combustion process has not yet taken place. On the other hand, higher temperatures (higher progress variable *c*) correspond to low CH_4 mass fraction and O_2 mass fraction, because the fuel is in a great extent oxidized, and high CO_2 mass fraction and H_2O mass fraction, which are the main products of the chemical reaction. CO and OH mass fractions acquire significant values for intermediate to high temperature, implying that these species appear mainly in the reaction zone and are either further oxidized or recombined to a great extent towards the burned gas region.

Analyzing the distribution maps corresponding to Case 1 of training data normalization (Figure 5-16), the unburned gas region can be seen to be located on the lower right part of the grid, while the burned gas region spreads all over the upper part of the grid. This is an expected result of clustering the state space generated through premixed laminar flame simulations, as the burned gas region is separated from the unburned gas region by the reaction regime.

Regarding the distribution maps corresponding to Case 2 of training data normalization (Figure 5-17), the unburned gas region is also observed in the lower right part of the grid, while the burned gas region lies exactly above, on the upper right part of the grid. In fact, the state space is folded and the burned gas region seems to approach the unburned gas region. This observation can be justified by the kind of normalization applied in the SOM's input data. Case 2 corresponds to all data normalized into the [0,1] range, meaning that the network perceives all features as having equal importance. Minor species and radicals exist in low concentrations both in the unburned and burned gas regions, resulting in mass fractions approximately equal to zero in these regions after the scaling is applied, while in the reaction regime they acquire values near unity. Therefore, SOM perceives this phenomenon as a similarity between unburned and burned gas regions, thus projecting and placing them near each other on the two-dimensional grid. This observation can also be justified by the dimensionality reduction diagrams of Section 5.2.1 (Figures 5-8, 5-9), where indeed the unburned gas region seems to be near the burned gas region at two dimensions defined by the first two principal components for Case 2 of training data normalization.

In conclusion, both SOMs are able to cluster properly, accurately and sufficiently the state space considered in the present work. Each network can be found useful for different implementations, such as dynamic adaptive chemistry, local mechanism reduction or regression via Artificial Neural Networks.

5.4 Hybrid Neural Network Training Process

Self-Organizing Map maintains the topology of the multi-dimensional dataset. This means that neurons close to one another on the two-dimensional map are also close in the multi-dimensional space. For several applications, the clusters created by SOM may be more than necessary and overestimated, and can be grouped to clusters with similar properties into subdomains. An increasing number of subdomains results in higher computational cost and, in the respective applications, an increasing number of specialized Artificial Neural Networks, which will result in more refined regression. On the contrary, partitioning into a large number of clusters has its drawbacks, as too many ANNs need longer to train and it may result in areas with too few samples, having no essential importance and thus uneven regression quality.

An et al. [9], *Blasco et al.* [8], *Franke et al.* [10], and *Chatzopoulos and Rigopoulos* [11] cluster the SOM a posteriori, by splitting it manually into larger rectangular subdomains, by joining adjacent cells. The configurations are then tested for the combination of SOM-MLP neural network and their final choice of SOM configuration is based on the testing error. A more innovative and interesting approach is proposed by *D' Alesssio et al.* [13], who couple the SOM with a K-Means clustering algorithm to group the closest neurons, obtaining a prescribed number of clusters required for the specific application. This concept is also applied in the present work, and the K-Means algorithm is used to cluster the SOM grids developed for both cases of training data normalization. This two-level approach has been discussed in Section 3.2.3.3.

The K-Means method used is the one offered by *scikit-learn*, an open-source Python library that is extensively used in Machine Learning for predictive data analysis [61]. The algorithm reads the Self-Organizing Map, previously stored as an object, and classifies the neurons into an a priori defined number of clusters. The number of clusters can be selected either by having a priori knowledge of regions expected to be found in the dataset or by calculating clustering performance evaluation indices, i.e., the Silhouette Coefficient and the Davies-Bouldin index. It should be noted that the K-Means algorithm performs random initialization of cluster centers. The developed code runs the K-Means algorithm ten times and selects the network with the lowest error.

5.5 Hybrid Neural Network Partitioning of State Space

The K-Means algorithm is applied on both Self-Organizing Map neural networks developed, in order to cluster the closest neurons of the two-dimensional grid. The final number of clusters obtained through the SOM – K-Means hybrid neural network is a user-defined parameter. According to *Vesanto and Alhoniemi* [41], the predefined number of clusters can be extracted through visual inspection of the unified distance map (U-Matrix), which shows the distances between neurons. Light colors in the U-Matrix depict closely spaced neurons, while darker colors indicate more widely separated neurons. Consequently, a first estimation of the number of clusters can be assessed by the number of different colors observed in the U-Matrix.

A more accurate prediction of ideal number of clusters can be extracted through the clustering performance evaluation indexes, i.e. Silhouette Coefficient and Davies-Bouldin index, which consider the capacity to separate data objects with maximum distance (external evaluation) and the capacity of the data to gather together around the centroids, generating maximal compact groups (internal evaluation), as discussed in Section 3.2.3.

5.5.1 Case 1: Final Partitioning of State Space

The dissimilar standardization followed in Case 1 and the different orders of magnitude between all data features result in clustering performance evaluation indices not functioning as expected for the purpose they were computed. Therefore, they cannot be trusted for selecting the optimal number of clusters of the SOM two-dimensional grid. The unified distance matrix (U-Matrix) is used, in lieu of an early assessment of the number of clusters, based on the number of colors observed on the two-dimensional projection.

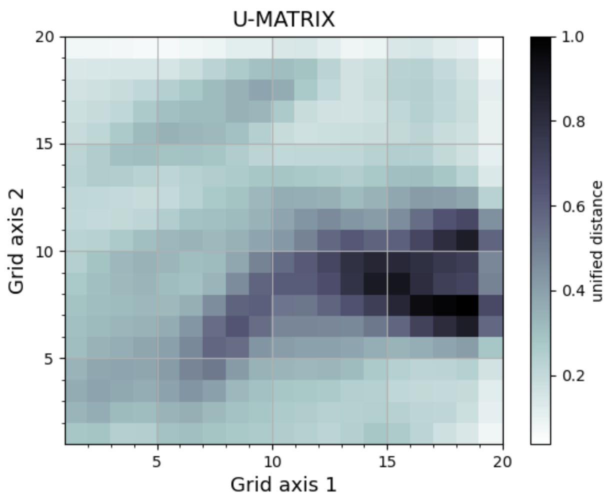


Figure 5-18: U-Matrix illustrating the unified distances between neurons of SOM, for **Case 1** of data normalization.

As can be seen in Figure 5-18, five colors can be discerned, from light blue to black. Figure 5-19 represents the selected clusters on the SOM two-dimensional grid next to progress variable, *c*, distribution map for comparison reasons.

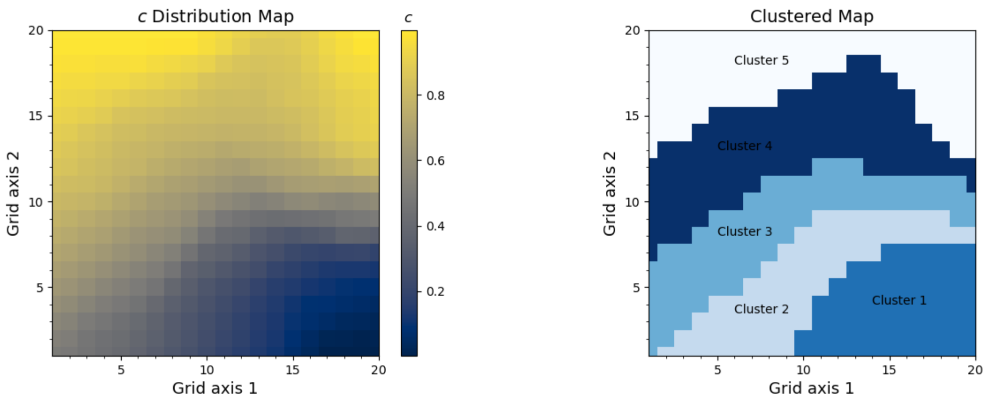


Figure 5-19: Progress variable, *c*, distribution map (left) and clustered SOM grid (right), by grouping the closest neurons into subdomains via the K-Means method, for **Case 1** of data normalization.

The selection of five clusters to group the state space data does not deviate much from reality. In each premixed laminar flame, at least three regions can be observed; unburned gas region, burned gas region and reaction region. The flame regime can be further separated into preheating zone, main reaction zone and recombination zone.

Figure 5-20 displays the state space coverage from the full dataset obtained through premixed laminar flame simulations, and its partitioning into the five clusters, indicatively for specific species mass fractions. The same colormap is used as in the clustered SOM's grid (Figure 5-19).

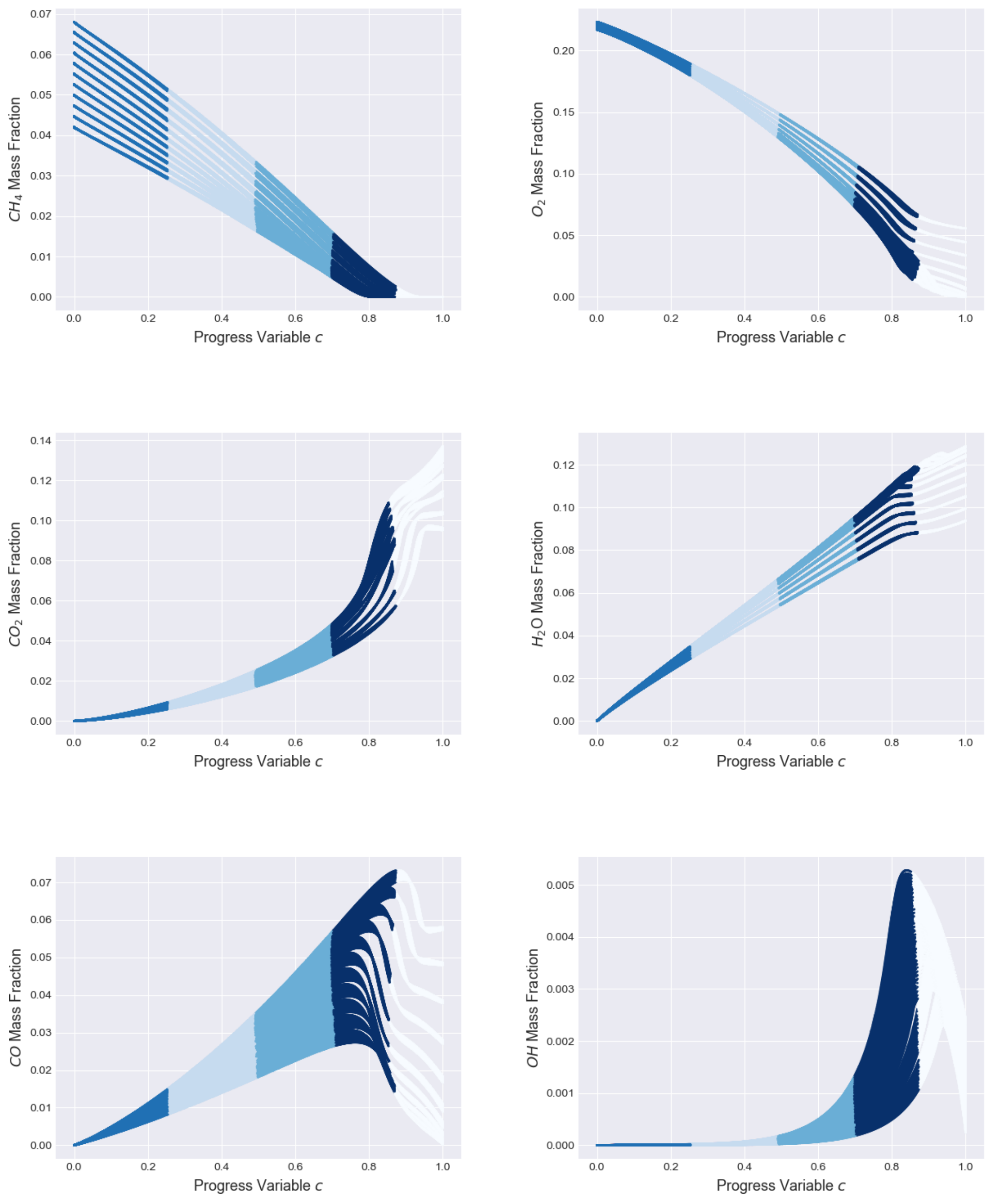


Figure 5-20: Composition points accessed by the dataset and its final partitioning, by applying the K-Means method on the SOM two-dimensional grid, for **Case 1** of data normalization.

5.5.2 Case 2: Final Partitioning of State Space

Case 2 of data normalization refers to all data features being standardized into the [0,1] range. Clustering performance evaluation indexes perform well in this case, and they are used to evaluate the number of clusters. The Silhouette Coefficient shows a higher score for four clusters, meaning that clusters are dense and well separated from each other. The Davies-Bouldin index also indicates an optimal number of clusters equal to four, suggesting that this configuration offers a better partitioning of state space, as the clusters are characterized by the minimum possible similarity to each other. Since both clustering performance evaluation indexes give the same result, the K-Means algorithm is ordered to group the SOM neurons, and so the state space as an aftermath, into four clusters.

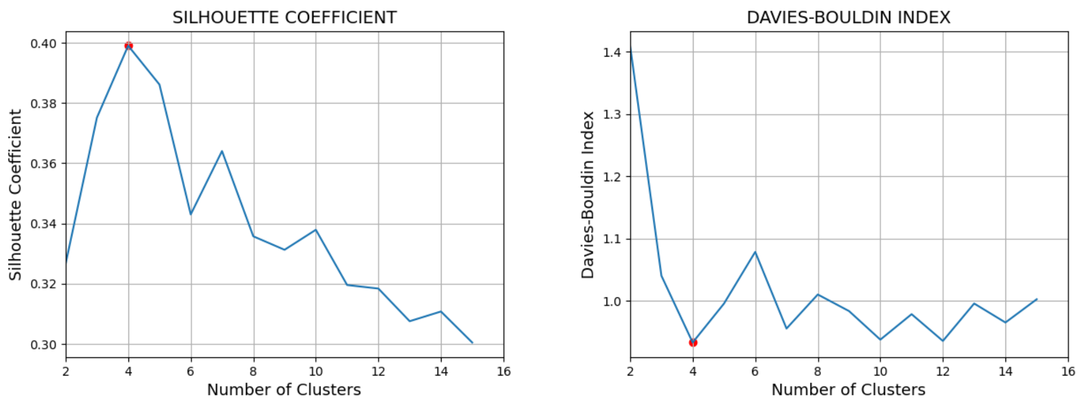


Figure 5-21: Silhouette Coefficient (left) and Davies-Bouldin index (right) versus number of clusters. Both indexes suggest an optimal number of clusters equal to 4 (marked with red point).

Figure 5-22 represents the clusters created on the SOM two-dimensional grid and the progress variable, *c*, distribution map for comparison. By definition, the cluster distribution is the same for each feature distribution map.

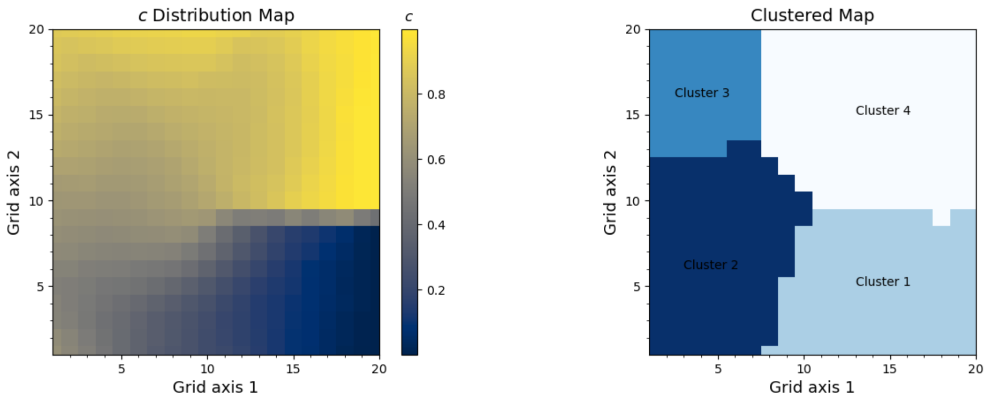


Figure 5-22: Progress variable *c* distribution map (left) and clustered SOM grid (right), by grouping closest neurons into subdomains via K-Means method, for **Case 2** of data normalization.

In this case, four clusters have a physical interpretation. Cluster 1 with low values of the progress variable *c* contains the unburned gas region, while cluster 4 with high values of the progress variable *c* contains the burned gas region. Clusters 2 and 3 correspond to intermediate values of the progress variable *c*, meaning that they contain the reaction regime.

Figure 5-23 displays the state space coverage from the full dataset obtained through premixed laminar flame simulations and its grouping into the five clusters, indicatively for specific species mass fractions. The same colormap is used as in clustered SOM grid figure (Figure 5-19).

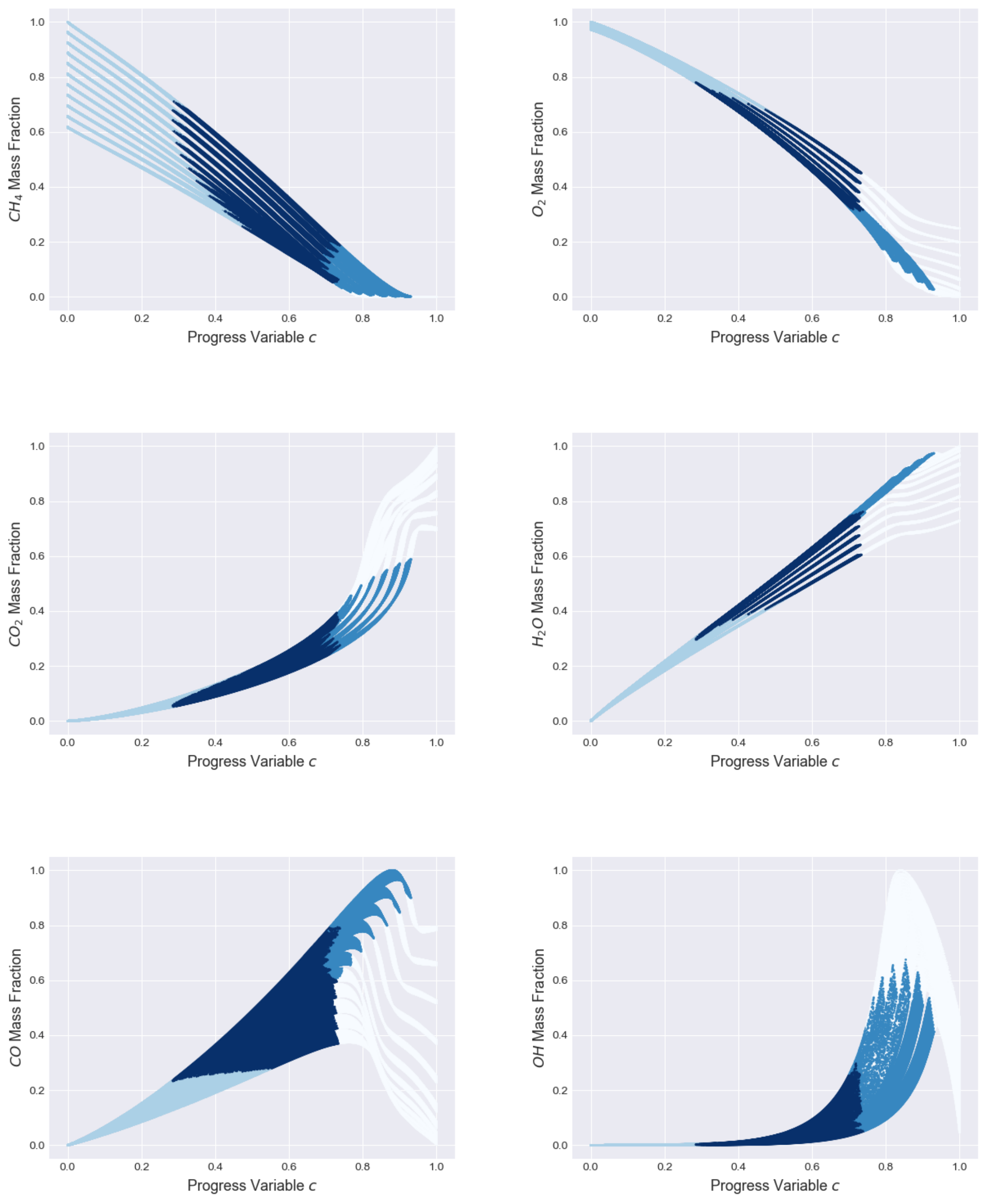


Figure 5-23: Composition points accessed by dataset and state space final partitioning, by applying the K-Means method on the SOM two-dimensional grid, for **Case 2** of data normalization.

5.6 Discussion

The partitioning of the state space into clusters (Figures 5-20, 5-23) can be related to the flame structure, for both Case 1 and 2 of data normalization. Regarding Case 1, the final partitioning of state space is in five clusters, according to the visual inspection of the SOM grid unified distance map. The clusters define specific temperature zones, which can be transformed to spatial zones.

For Case 2 of data normalization, the optimal number of clusters defined through evaluation indices is four. The generated clusters are not clearly separated in specific temperature zones, but can be considered to correspond to specific flame regions. Cluster 3, corresponding to a region with high alterations in species concentrations, has a smaller size compared to all other clusters, and contains a large number of data points.

Evidently, the physical definition of premixed laminar flame regions does not necessarily go hand in hand with the clusters defined by the neural network. Nonetheless, the state space partitioning by the hybrid neural network mechanism is appropriate for the purposes of chemistry computational cost reduction. Each of the two hybrid neural networks, as well as the SOMs without applying further grouping via the K-Means method, can be considered as appropriate for several applications for chemistry acceleration.

The hybrid neural network corresponding to Case 2 of data normalization is further studied for the purposes of chemistry tabulation via Artificial Neural Networks. More specifically, an ANN will be assigned to each one of the four clusters created by the hybrid clustering neural network, in order to predict the state evolution, as discussed in the following two chapters.

CHAPTER 6: Prediction of the Heat Release Rate and the Species Rates of Change

The next step of the present work is the development of an Artificial Neural Network for chemistry tabulation and prediction of the state space. As discussed in Chapter 4, if a chemical system is homogenous, the evolution of a chemical state is fully defined by integrating an ODE system, consisting of equations describing the changes of temperature and species concentrations.

In this context, the ANN is developed to predict heat release rate (HRR) and the species production or consumption rates, given a state vector as input. The state vector includes the temperature and species mass fractions as features. The present study deals with methane-air mixture combustion using the GRI Mech. 3.0, a detailed kinetics mechanism containing 53 species and 325 reactions. Therefore, the state vector contains N_s+1 = 54 features (temperature and 53 species mass fractions).

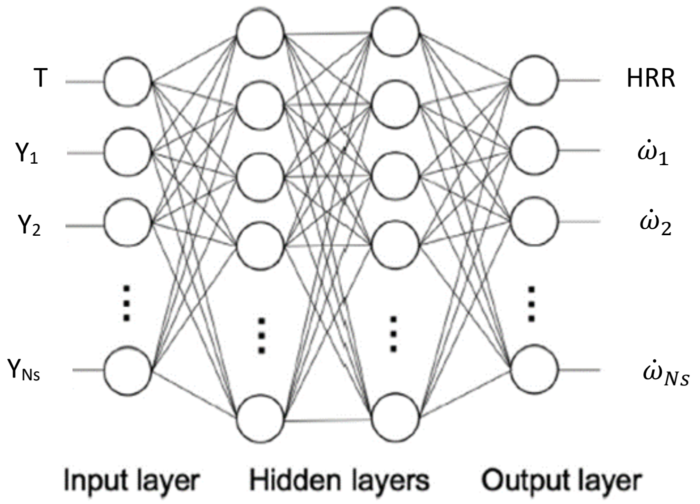


Figure 6-1: An illustration of the ANN function.

Given a mechanism, the production or consumption rate of each chemical species can be expressed by the following system of ODEs:

$$\dot{\omega}_\imath = \frac{d[M_i]}{dt} = (\nu_i'' - \nu_i') \cdot k \cdot \prod_{i=1}^{N_s} [M_i]^{\nu_i'}$$

since ν_i'' moles of M_i are created for every ν_i' moles of M_i consumed. The convention used in the present work is that parentheses around a chemical symbol signify the concentration of that species, i.e., $[M_i]$ signifies the concentration of species M_i. Additionally, $\dot{\omega}_\imath$ will be called net production rate of species *i*, for the sake of brevity. The proportionality constant *k* is the reaction rate constant, and introduces the temperature dependence for reaction rates. It is calculated through Arrhenius Law (Section 4.1.1).

If the pressure is fixed, as considered in the present work, the net production rate of each species is expressed by a function:

$$\dot{\omega}_\imath = f(T, Y_i)$$

Thus, for a given chemical mechanism and a fixed pressure, the net production rates at a specific reaction step are functions of temperature and the species mass fractions.

Furthermore, the heat release rate can be calculated as the sum over all species of net production rate multiplied with species specific enthalpy:

$$\mathrm{HRR} = \sum_{i=1}^{N_s} \dot{\omega}_i h_i$$

The developed ANNs shall be able to learn and regress the above mathematical relationships, for a user-defined level of accuracy. These equations are non-linear and stiff, thus adding a level of difficulty in the concept.

The state space is divided into four clusters using the hybrid SOM – K-Means neural network, for Case 2 of data normalization. Multi-Layer Perceptrons are used for regression purposes, as they are a class of feedforward ANNs. An MLP is then assigned to each one of these clusters, applying the concept illustrated in Figure 6-1. It receives a 54-dimension composition state vector as input, and predicts the heat release rate and species net production rates.

The Artificial Neural Network training must be done on data obtained from the simulations of the same phenomenon or, alternatively, from simulations of an abstract problem. Regarding the latter case, it is important to ensure that the generated training dataset covers the state space accessed during the multi-dimensional CFD simulation. An ensemble of premixed laminar flames is used to generate the training data for the current implementation.

6.1 Sample Generation

The approach followed in the present work implies that the developed ANNs shall be able to generalize well, instead of focusing on specialized state space domains and operating sufficiently only under specific thermochemical conditions. Therefore, the training data are obtained through an abstract problem, while covering the state space expected to be encountered in future simulations.

The generation of training data for Multi-Layer Perceptron training process is carried out by means of one-dimensional premixed laminar flame simulations, similarly with the procedure followed for SOM training data generation.

The simulation conditions are the same as those for the SOM training sample generation (Section 5.1): the pressure is held constant, at $p = 1$ atm, the inlet temperature ranges from 300 K to 400 K and the fuel-air equivalence ratio ranges from 0.75 to 1.25.

The present concept is based on state space partitioning and thermochemical states regression, meaning that only temperature and species mass fractions are needed to generate a state vector, similarly to the procedure followed for training data generation for SOM training (Section 5.1). Furthermore, heat release rate as well as species net production rates are stored as a function of composition states. It is highlighted that the spatial variable is unambiguously connected with temperature, meaning that the spatial distribution of heat release rate, species mass fractions and net production rates can be converted to temperature distribution.

The simulation conditions correspond to 121 methane-air premixed laminar flames, generating in a total of 127099 data samples. The data samples are combinations of input state vectors and output vectors containing heat release rate and species' net production rates. For the purpose of MLP training, the data samples are a priori split into 60% training set and 40% test set, similarly to the case of SOM training process. The full dataset is used only in the frame of the cross validation technique, a method which is applied in order to tune the network hyperparameters. A notable observation is that the produced dataset has no outliers, as it is generated through robust simulations. Therefore, there is no need for application of data cleaning techniques before proceeding to neural network training responsible for regression, similarly to the neural network training responsible for state space clustering.

6.1.1 Data Augmentation

Through several early trials of MLP training, it was observed that the developed neural network focused at specialized states, and could not perform sufficiently on data significantly different to the training data. In addition, during cross validation applied in order to define the network architecture, the metrics error displayed oscillations when the network architecture was slightly modified, for example, in terms of addition of one more neuron per layer.

A common technique used extensively in Machine Learning to deal with network generalization and stabilization is data augmentation. Basically, data augmentation refers to the enlargement of the dataset by adding new synthetic data from existing data. This method is expected to reduce "hollows" created during cross validation, which destabilize the neural network. Additionally, it will improve the network performance in terms of generalization, as more available data points will approximate and cover the state space. MLPs have the capacity to interpolate, but, if the regions of state space are sparsely populated by the training data, the accuracy decreases.

The so-called augmented data can be generated by performing additional simulations, or randomly, considering several restrictions and relationships between data features. For example, a restriction relative to combustion simulations is that species mass fractions shall lie within the solution tolerance at each thermochemical state. In the present work, additional data are obtained through premixed laminar flame simulations for conditions lying between the initial ones (Table 6-1), using the already developed and validated code.

Table 6-1: Initial conditions for 1D premixed laminar flame simulations, for generation of augmented data.

	Initial Conditions Range	**Step**
Pressure	1 [atm]	-
Temperature	300 – 400 [K]	20 [K]
Fuel-air equivalence ratio	0.76 – 1.21	0.05
	0.78 – 1.23	0.05

The aforementioned simulation conditions correspond to 120 methane-air premixed laminar flames, generating in total 125846 data samples. In addition to the initial acquired dataset containing 127099 data samples and obtained through 121 flames, the final dataset contains 252945 data samples.

Data augmentation increases the density of the data coverage of state space mainly in the reaction zone, characterized by intermediate to high values of temperature or progress variable, c. Regarding low temperatures, the acquired data are not denser; this is not a problem, because in the unburned gas region no changes in the heat release rate or species concentrations are observed.

Thus, this will probably not lead to reduced MLP ability to learn and generalize in this region. The difference between initial and final (initial and augmented) dataset coverage of the state space is illustrated indicatively in terms of methane mass fraction in Figure 6-2.

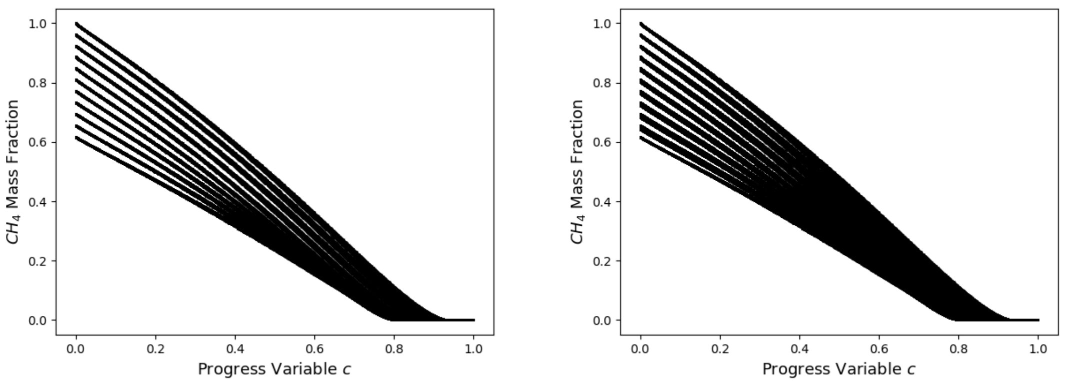
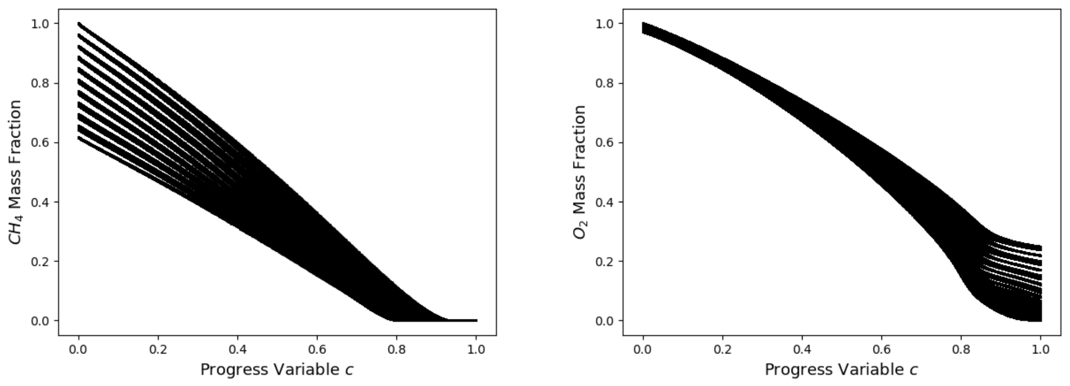


Figure 6-2: Scatter plots representing the state space coverage of the initial dataset (left) and full dataset (right) collected from methane-air premixed laminar flame simulations.

6.1.2 Final Dataset

Figure 6-3 presents the state space coverage from the full dataset, containing both initial and augmented data, obtained through laminar flame simulations. The diagrams presented are indicative for the temperature distribution of main reactants (CH_4, O_2), main products (CO_2, H_2O) and intermediate species/radicals (CO, OH), supporting the main heat release of methane-air mixture combustion process. Temperature is normalized in terms of the progress variable, *c*.



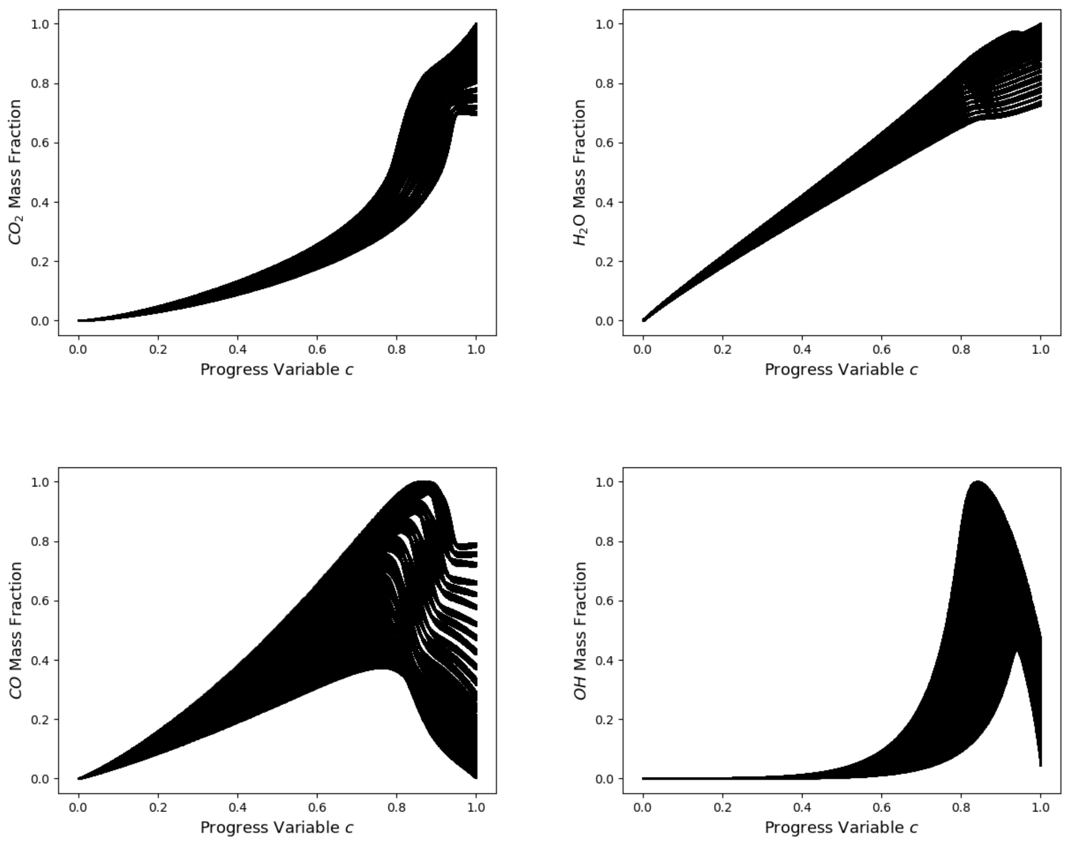
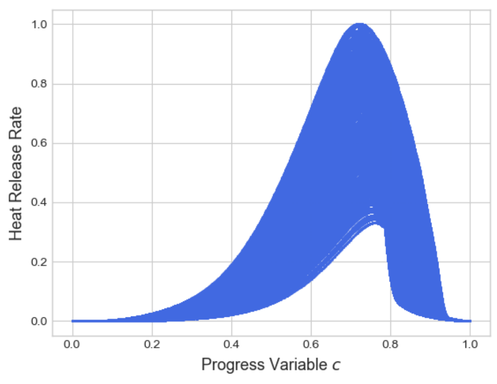


Figure 6-3: Scatter plots presenting the state space coverage of the full dataset generated by methane-air premixed laminar flame simulations.

The desired output of MLPs is the heat release rate and species net production rates. These features are also obtained via the premixed laminar flame simulations and are functions of state. Due to the differences in orders of magnitude between the heat release rate and species net production rates, normalization of all features into the [0,1] range is considered mandatory. Consequently, both input and output vectors of MLPs are standardized into the [0,1] range, a technique offering better behavior and robustness to the neural network, while the adjusted weights acquire values in the same order of magnitude.

Figure 6-4 presents the heat release rate and selected species net production rates obtained through laminar flame simulations.



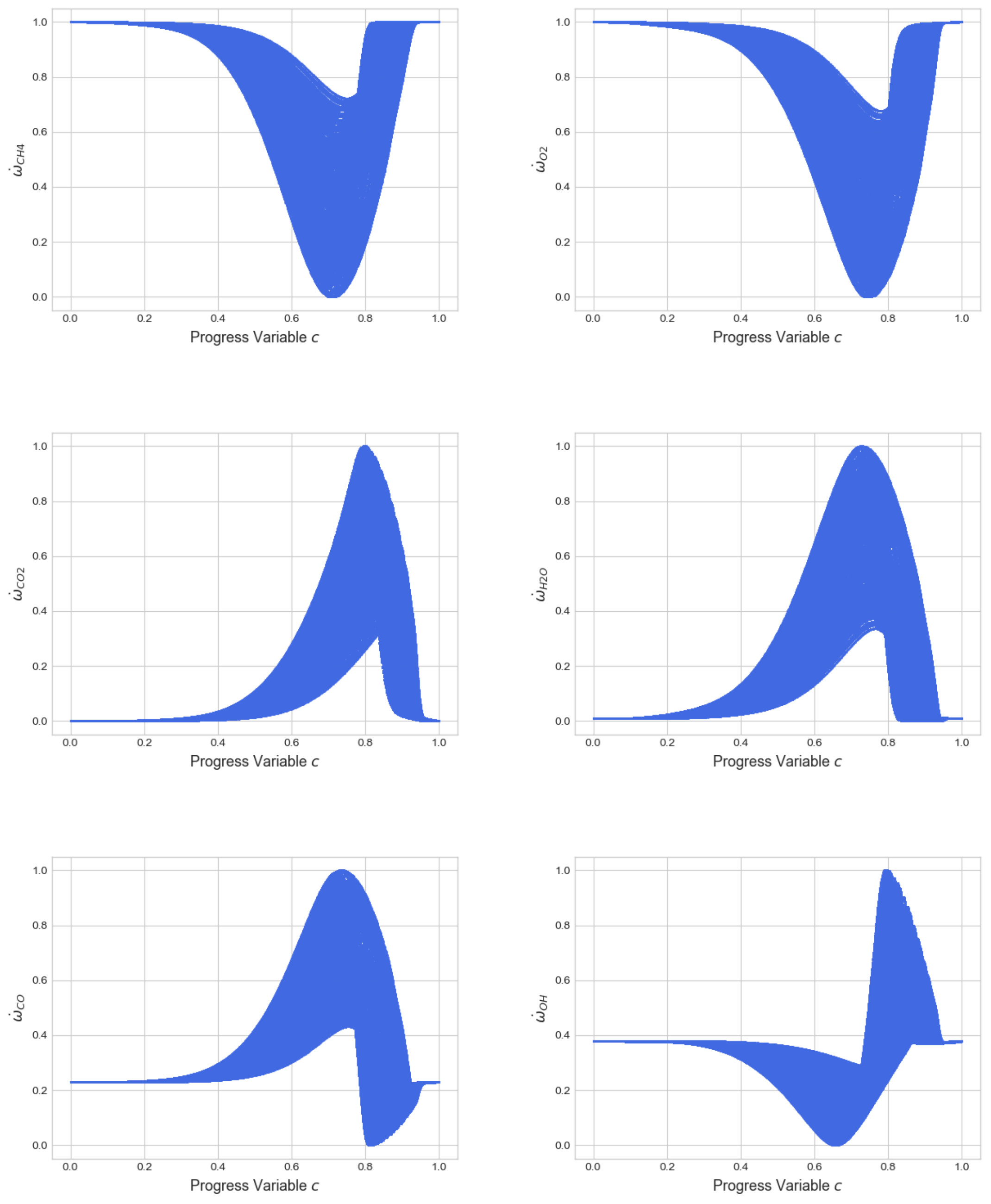


Figure 6-4: Scatter plots presenting the heat release rate (top) and indicative species net production rates as a function of temperature progress variable, *c*, collected from methane-air premixed laminar flame simulations and normalized into the [0,1] range.

Several important observations can be extracted from Figure 6-4. Regarding the top diagram, the higher values of heat release rate are observed at intermediate to high temperatures, implying that the main heat release occurs in the reaction regime.

Methane (CH_4) and oxygen (O_2) net production rates are negative, as they are consumed during combustion. In the preheating zone, these species are consumed with an increasing rate, while in the burned gas region, the rates of consumption decrease, until they finally reach a value equal to zero. The dataset normalization into the [0,1] range leads this behavior to be represented by a convex distribution, with the maximum scaled consumption rate equal to zero.

On the other hand, carbon dioxide (CO_2) and water (H_2O) net production rates are positive, as they are the main products of the methane combustion process. In the preheating zone, these species are gradually produced with an increasing rate, while in the burned gas region, the production rates decrease, while remaining positive, until they finally reach a value equal to zero. The dataset normalization into the [0,1] range leads this behavior to be represented by a concave distribution, with the absolute maximum production rate scaled equal to unity.

Carbon monoxide (CO) and hydroxyl radical (OH) are produced in intermediate temperatures and further oxidized or recombined in the burned gas region. This behavior can be observed in the respective diagrams of Figure 6-4.

6.2 Multi-Layer Perceptron Training Process

The neural network applied for regression in the present work is the Multi-Layer Perceptron, a type of feedforward Artificial Neural Network. As discussed in Section 3.3, the network is trained through back-propagation method with the Adam optimization algorithm.

Before the training process starts, the network hyperparameters must be defined, similarly to the Self-Organizing Map case (Section 5.2). The hyperparameters of an MLP neural network are the following:

- Number of hidden layers.
- Number of neurons per layer.
- Activation function. Hyperbolic tangent is employed due to its property of being a zero-centered, smooth differentiable function.
- Maximum number of epochs. An epoch is defined as a cycle of iterations, where all training data are used exactly one time. Iterations terminate based on the loss function convergence.

The number of hidden layers and the number of neurons per layer defines the neural network architecture. Commonly, the architecture, as well as other network hyperparameters, are selected by means of trial-and-error, resulting in a configuration that balances accuracy and computational cost. A more robust method used extensively to define neural network hyperparameters is the cross validation technique, as discussed in Section 3.3.1.

6.2.1 Neural Network Architecture

A common technique followed in Machine Learning is to split the full dataset into a training set, responsible for the definition of network hyperparameters and training, and a test dataset, used for evaluating the trained network. This method is applied in terms of training and testing processes in the present work, by splitting randomly the dataset into 60% training set and 40% test set, similarly to the SOM training and testing process. The cross validation technique makes use of the full dataset, which is split into *k* smaller groups, the so-called folds. A clone model of the regressor is trained using the *k-1* folds as training data and the resulting model is validated on the remaining fold of the data. In other words, it is used as a test set for performance accuracy and generalization capability. The performance measure reported by the *k*-fold cross validation method is then the average of the values computed in the loop. The metric applied for cross validation in the present work is the root mean squared error (RMSE).

The concept of the current study consists of two neural networks. Firstly, a hybrid SOM – K-Means neural network is applied to cluster the state space. Next, an MLP is assigned in each cluster in order to learn and predict the desired patterns of input data. Regarding Case 2 of data normalization, where all data features are scaled into the [0,1] range, four clusters are identified by the hybrid neural network responsible for state space clustering. Therefore, four MLPs will be developed, each assigned to a specific cluster.

Figure 6-5 depicts the results of the 3-folds cross validation technique applied on each of the four MLPs, as a function of the number of neurons per layer. 5- folds and 8-folds cross validations were also performed, and resulted to similar behavior with respect to the RMSE. A configuration of two hidden layers is selected. Three hidden layers were also examined, but the error reduction did not compensate the additional complexity of the neural network and the resulting increase in computational cost.

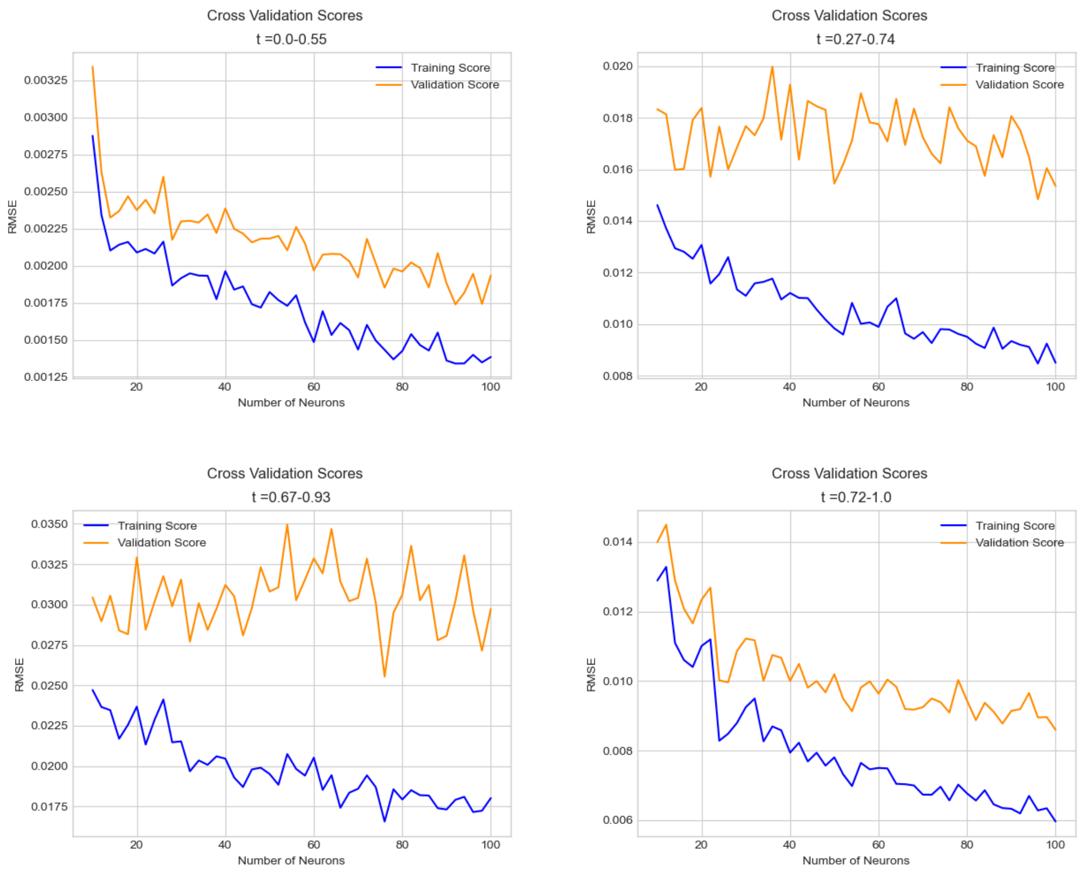


Figure 6-5: 3-folds cross validation for each cluster generated by the SOM – K-Means neural network. Progress variable, *c* (here noted as *t*), is used to define each cluster's temperature range.

One should select the simplest possible network that can adequately represent the training dataset, since this choice has a significant impact on network complexity and computational cost. Observing Figure 6-5, a compromise between accuracy and complexity can be achieved for approximately 40 up to 60 neurons per layer, for each of the four MLPs. Furthermore, the selection of a lower number of neurons per layer may result in a biased neural network, leading to missing relevant relations between input features and target outputs (underfitting). Taking into account similar applications of combustion data regression (e.g. [8],[9]), a 54-54 neural network configuration is selected, meaning that there are two hidden layers with 54 neurons each.

Network complexity can be expressed in terms of degrees of freedom (DoF), i.e., the total number of synapses between neurons. For example, a 100-100 hidden layer configuration with 54 features as input and 54 features as target output has in total:

$$\mathrm{DoF} = 54 \cdot 100 + 100 \cdot 100 + 100 \cdot 54 = 20800$$

The degrees of freedom are plotted in Figure 6-6 as a function of the number of neurons per layer. The network architecture consists of two hidden layers with equal number of neurons per layer. The degrees of freedom follow a power-law increase. The selected 54-54 configuration seems to offer a compromise regarding network complexity, as it can be justified by the power-law relationship in Figure 6-6.

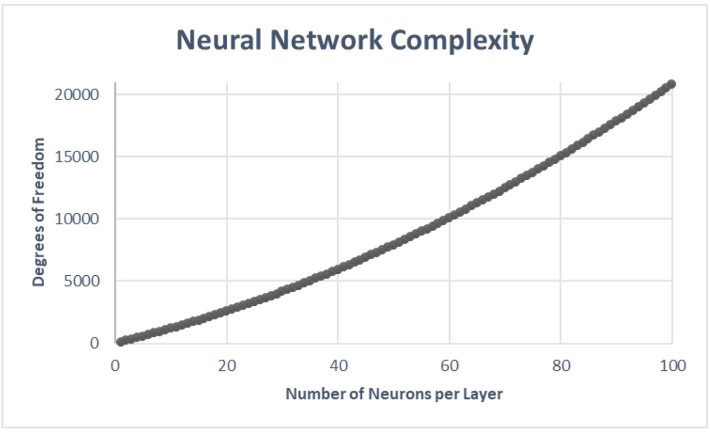


Figure 6-6: Neural network degrees of freedom as a function of the number of neurons per hidden layer. The network architecture consists of 2 hidden layers having the same number of neurons.

6.3 Prediction Results

Multi-Layer Perceptrons serve the role of the basic regression algorithm of state space data. After training is finished, the trained MLPs can predict the target output receiving a state vector as an input. The outputs that each MLP attempts to predict are the heat release rate and the species net production rates. In total, four MLPs are developed, each one assigned on each cluster generated by the hybrid SOM – K-Means neural network.

Both MLP input and output data are vectors of equal dimension as the state space dimension, which for the considered case is equal to 54. The output vector contains 54 features: the heat release rate and the 53 species net production rates.

The following paragraphs discuss the results obtained by the trained MLPs, for each cluster generated by the hybrid neural network responsible for clustering of state space. For convenience, the clustering of state space shown in Figure 6-7 is taken from Section 5.5.

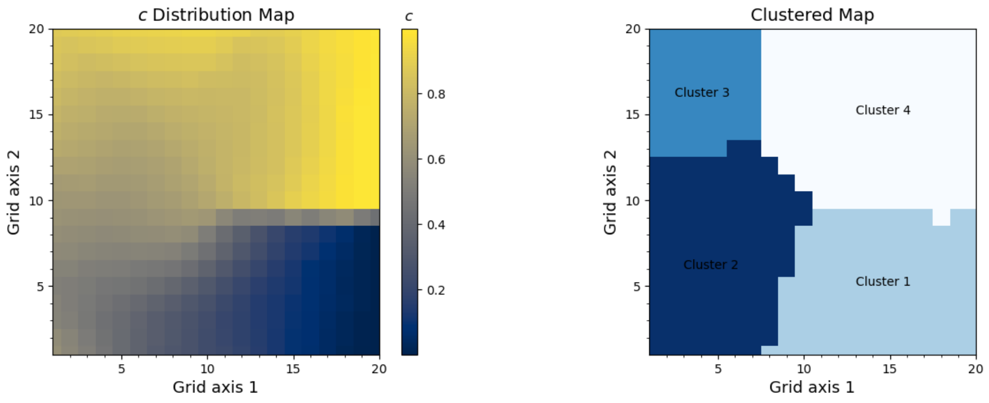


Figure 6-7: Progress variable, *c*, distribution map (left) and clustered SOM grid (right), by grouping the closest neurons into subdomains via the K-Means method, for **Case 2** of data normalization.

Table 6-2 summarizes the temperature region spanned by each cluster in terms of the progress variable, *c*.

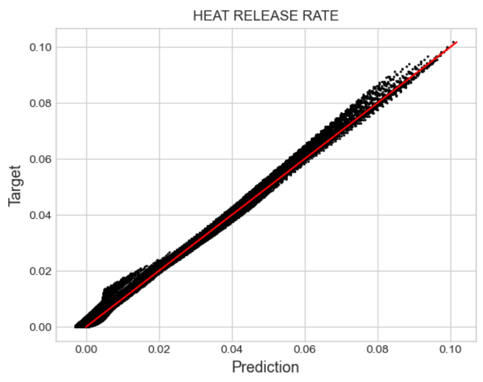
Table 6-2: Clusters of state space generated by the hybrid SOM – K-Means neural network.

	Range of Progress Variable, *c*
Cluster 1	0 – 0.55
Cluster 2	0.27 – 0.74
Cluster 3	0.67 – 0.93
Cluster 4	0.72 – 1

6.3.1 Results for Cluster 1

Cluster 1 ($0 \leq c \leq 0.55$) corresponds to the lower temperatures appearing in the state space considered in the present study. It spans the unburned gas region and a part of the reaction zone, where changes in temperature and species compositions start to occur.

Figure 6-8 shows the predicted values of the heat release rate and indicative species net production rates versus the ones obtained from premixed laminar flame simulations. The species selected to be plotted are the main reactants (CH_4, O_2), the main products (CO_2, H_2O) and the intermediate species/radicals (CO, OH) supporting the main heat release of methane-air mixture combustion process.



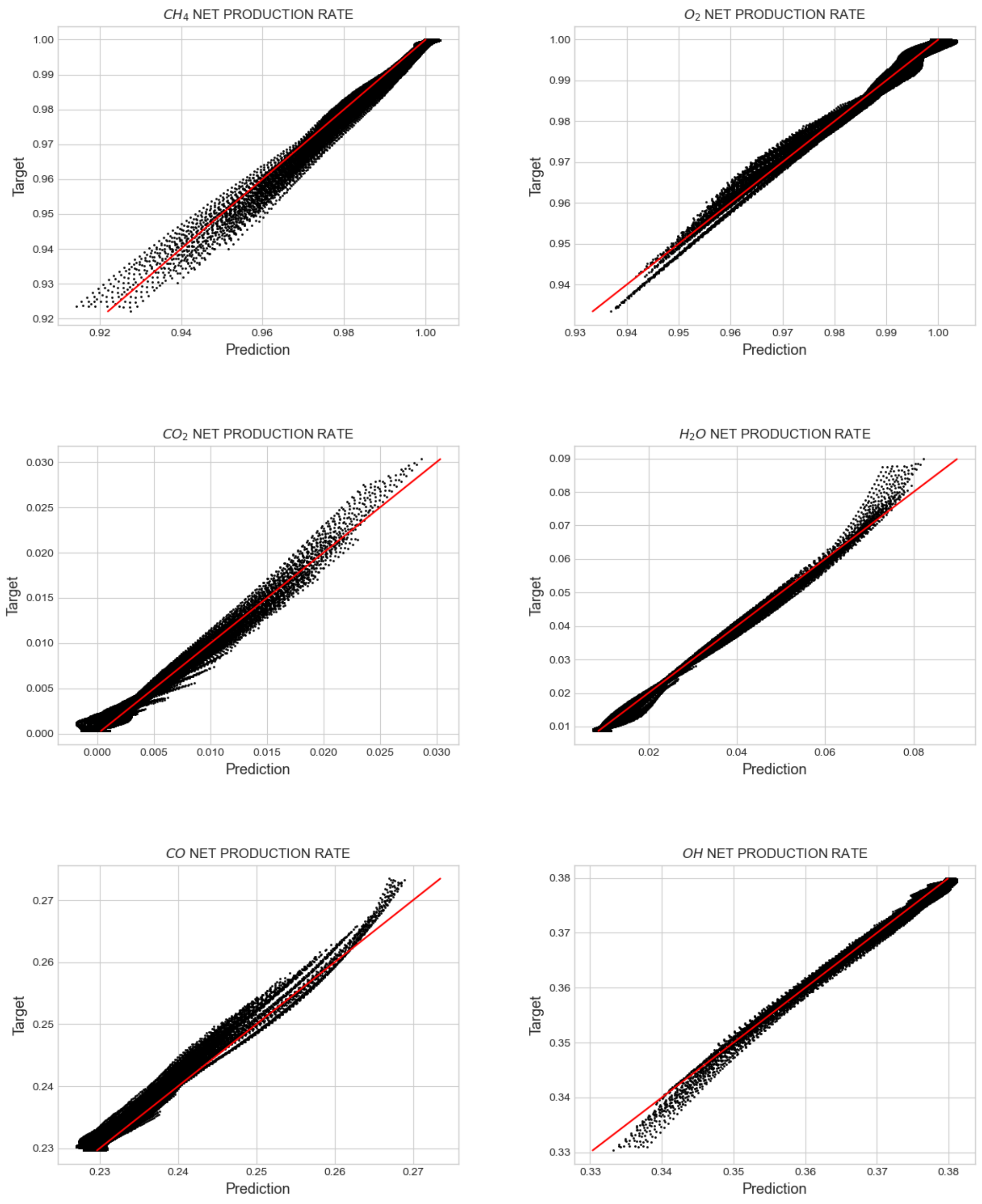


Figure 6-8: Prediction vs target output values of MLP assigned to Cluster 1, corresponding to temperature progress variable $c = 0 – 0.55$.

The diagrams of Figure 6-8 are basically a representation of a confusion matrix. If the points lie on the straight line $y = x$, then the prediction is exactly equal to the target output, and the fit is considered excellent. Studying the above diagrams, a divergence on the higher temperatures corresponding to the specific cluster is observed for all represented features. This divergence is caused due to the fact that this cluster represents both the unburned gas region and a part of the reaction regime. In the unburned gas region, the heat release rate and species net production rates are actually equal to zero, whereas they acquire values different than zero while moving into the reaction zone. The main reactants' net production rates equal to zero are scaled equal to 1 following the dataset normalization, while the main products' net production rates being equal to zero are also scaled equal to zero, following the dataset normalization.

Thus, the prediction is sufficient for the low temperatures, corresponding to the unburned gas region, where all rates are equal to zero, while the ANN output deviates from the target output for the higher temperatures of the current cluster. In all cases, the ANN performance can be considered satisfactory, taking into account the order of magnitude of the data belonging to the current cluster.

Figure 6-9 presents the RMSE of each feature contained in the MLP output vector, regarding Cluster 1. On average, the main reactants and the main products net production rates are predicted with excellent accuracy, while several minor species, such as CH3O, exhibit higher error values, probably due to the actual order of magnitude of these features. The mean RMSE for the specific MLP is equal to $2.2 \cdot 10^{-3}$.

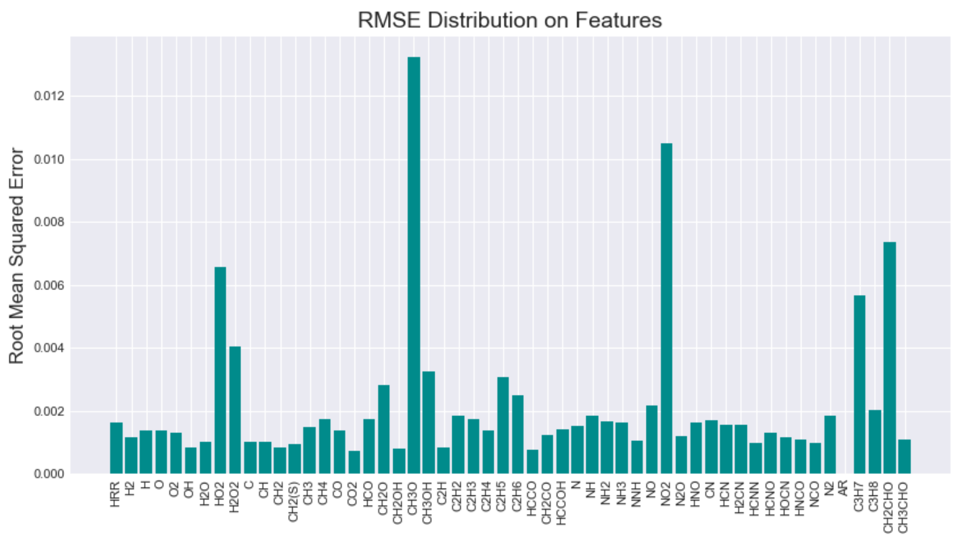


Figure 6-9: Root Mean Squared Error (RMSE) of each feature contained in the MLP output vector, regarding Cluster 1.

6.3.2 Results for Cluster 2

Cluster 2 ($0.27 \leq c \leq 0.74$) corresponds to intermediate temperatures appearing in the state space covered in the present study. It spans the preheating zone and a part of the main reaction zone, where high changes in temperature and species composition occur in both of them. A significant overlap between Cluster 1 and Cluster 2 is observed in terms of the corresponding values of progress variable, c. This can be justified by the intrinsic property of the clustering algorithm to create groups of approximately equal number of data, while at the same time dealing with high-dimensional data, resulting in overlapping of features in several projections of the high-dimensional space.

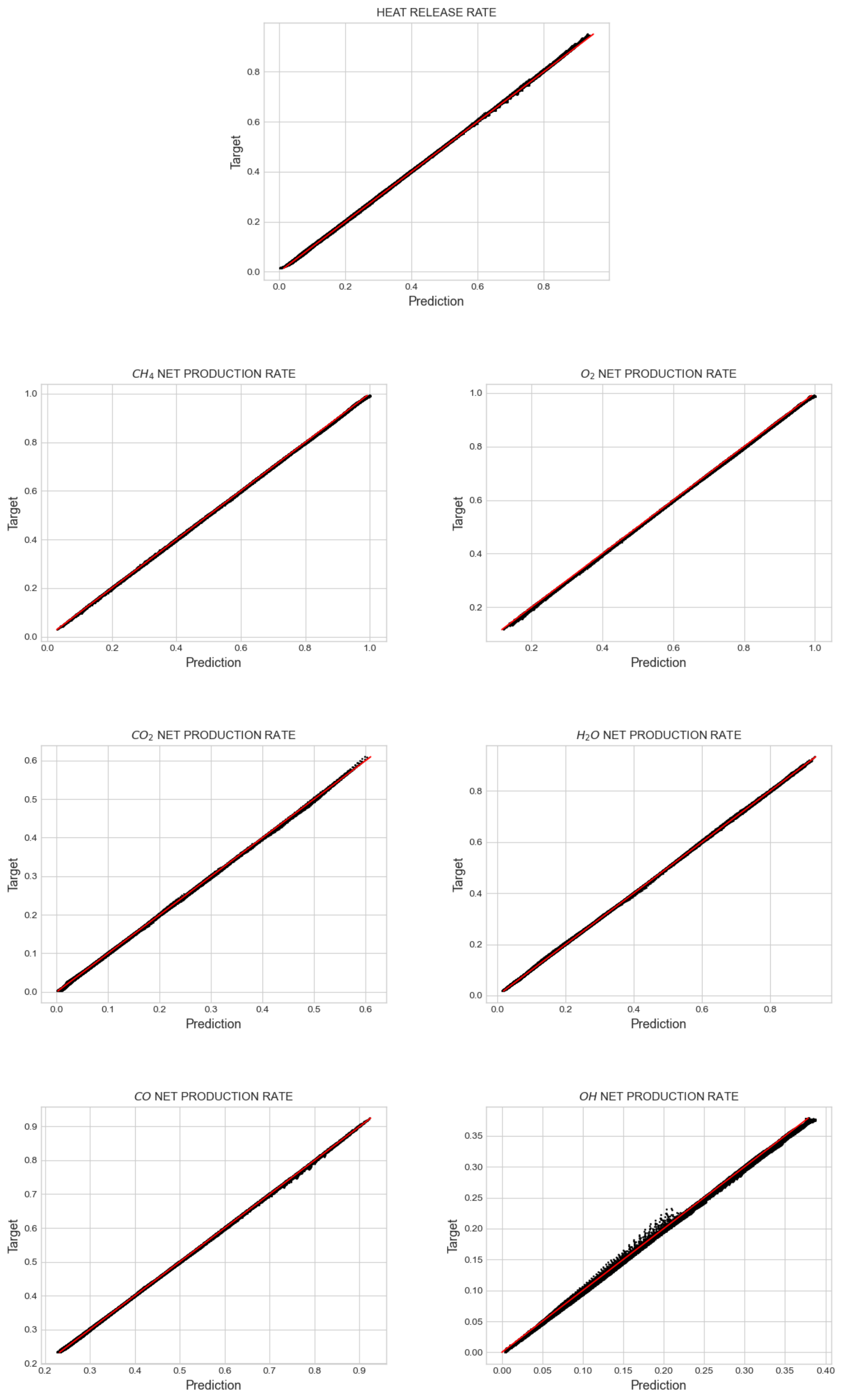


Figure 6-10: Prediction vs target output values of MLP assigned to Cluster 2, corresponding to temperature progress variable c = 0.27 – 0.74.

Figure 6-11 represents the RMSE of each feature contained in the MLP output vector, regarding Cluster 2. On average, the main reactants and the main products net production rates are predicted with excellent accuracy, while several minor species, such as CH_2CHO, exhibit higher error values, probably due to the actual order of magnitude of these features. Similar behavior was observed for the MLP assigned to Cluster 1. The mean RMSE for the specific MLP is equal to $4.0 \cdot 10^{-3}$.

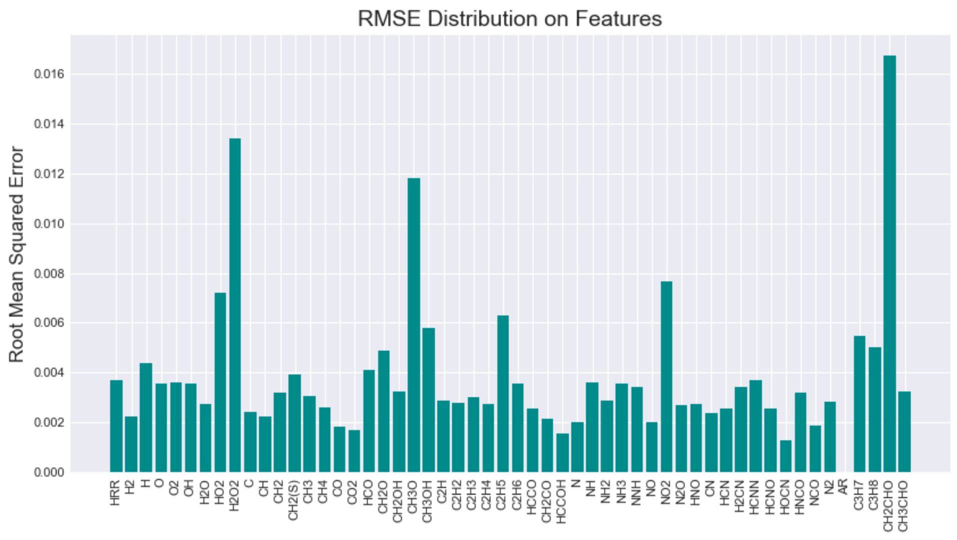
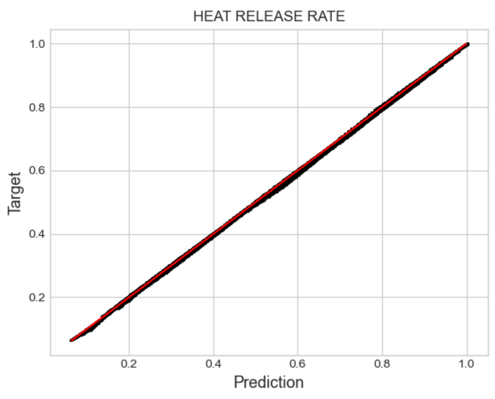


Figure 6-11: Root Mean Squared Error (RMSE) of each feature contained in the MLP output vector, regarding Cluster 2.

6.3.3 Results for Cluster 3

Cluster 3 ($0.67 \leq c \leq 0.93$) corresponds to intermediate to high temperatures appearing in the state space covered in the present study. It spans the main reaction regime, where the highest changes in temperature and species compositions occur. An overlap between Cluster 2 and Cluster 3 is also observed in terms of the corresponding values of progress variable, c.



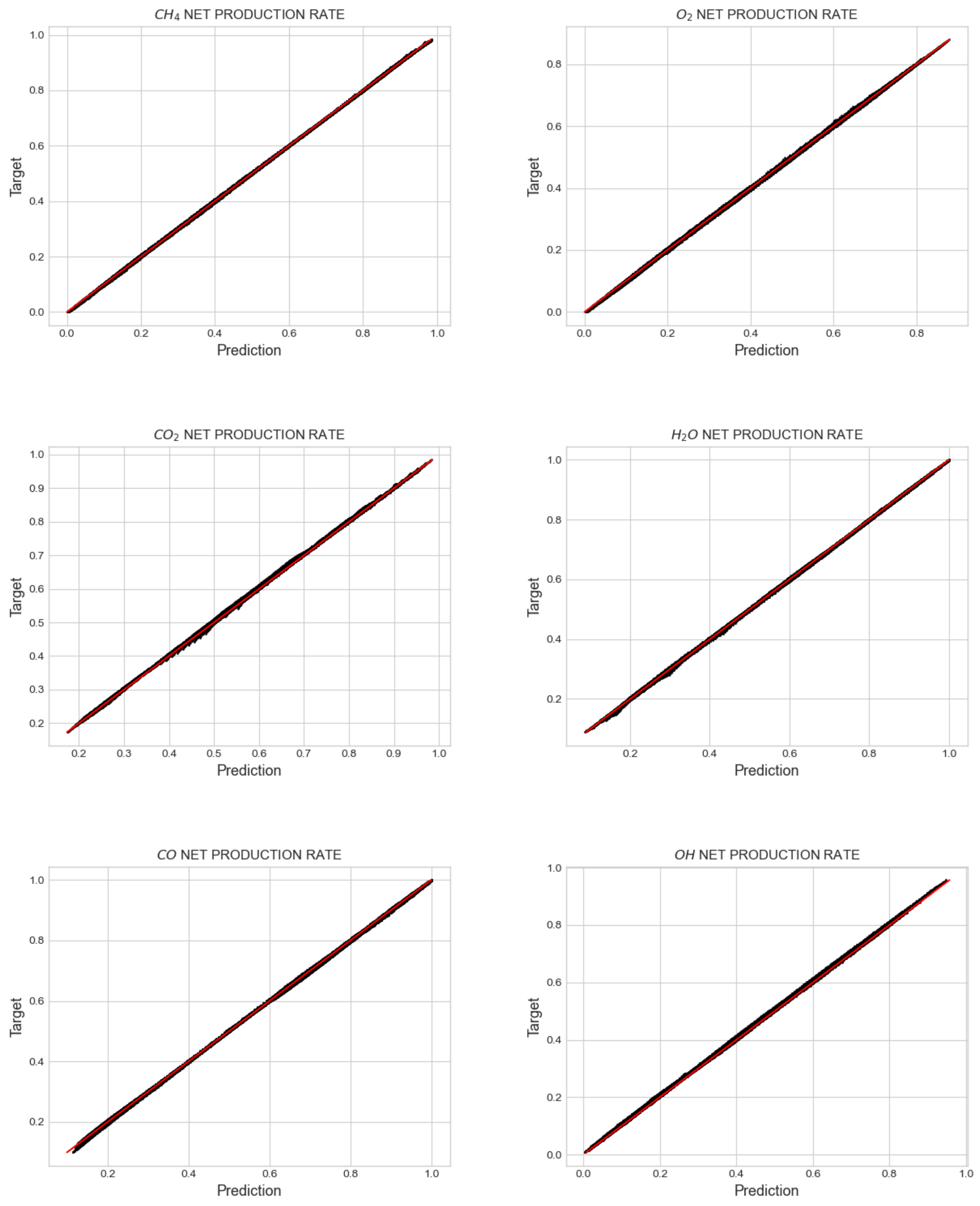


Figure 6-12: Prediction vs target output values of MLP assigned to Cluster 3, corresponding to temperature progress variable c = 0.67 – 0.93.

Figure 6-13 presents the RMSE of each feature contained in the MLP output vector, regarding Cluster 3. There is not much deviation in error between the main reactants and the main products net production rates, as well as between several minor species ones. This is probably due to the fact that radicals and minor species are activated for a small spatial window, contained in the specific cluster, and acquire values of a higher order of magnitude. The mean RMSE for the specific MLP is equal to $5.4 \cdot 10^{-3}$.

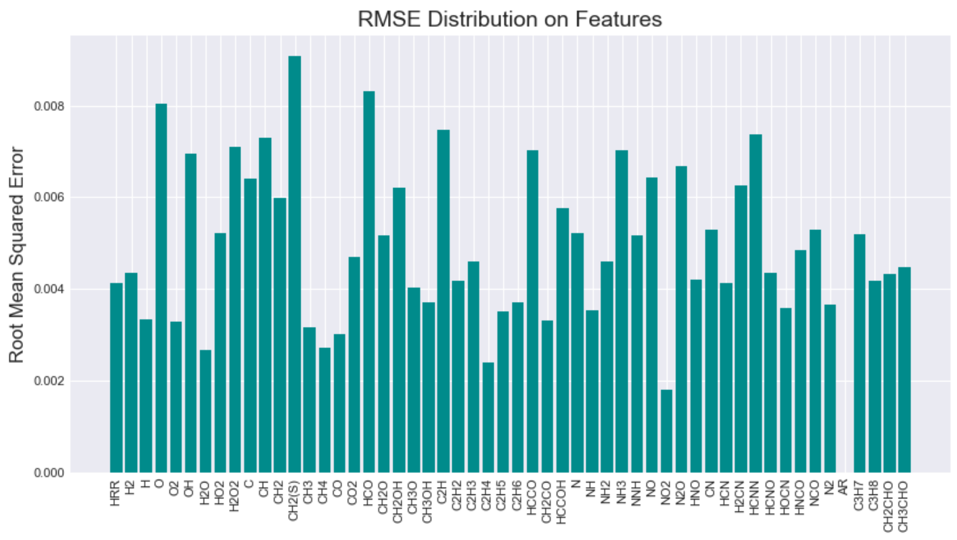
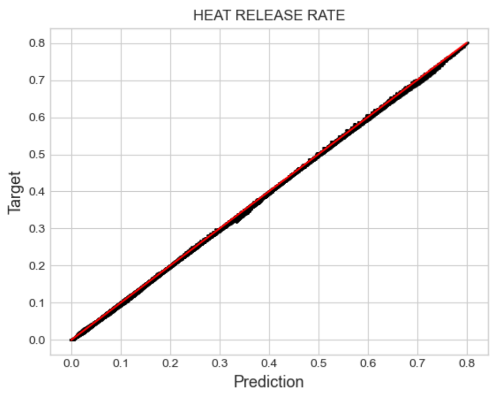


Figure 6-13: Root Mean Squared Error (RMSE) of each feature contained in the MLP output vector, regarding Cluster 3.

6.3.4 Results for Cluster 4

Cluster 4 ($0.72 \leq c \leq 1$) corresponds to the higher temperatures appearing in the state space covered in the present study. It spans the burned gas region and a part of the reaction regime, where several species are oxidized or recombined, and the changes occurring in temperature and species concentrations are not as high as in Cluster 3. A significant overlap between Cluster 3 and Cluster 4 is also observed in terms of the corresponding values of progress variable, *c*.



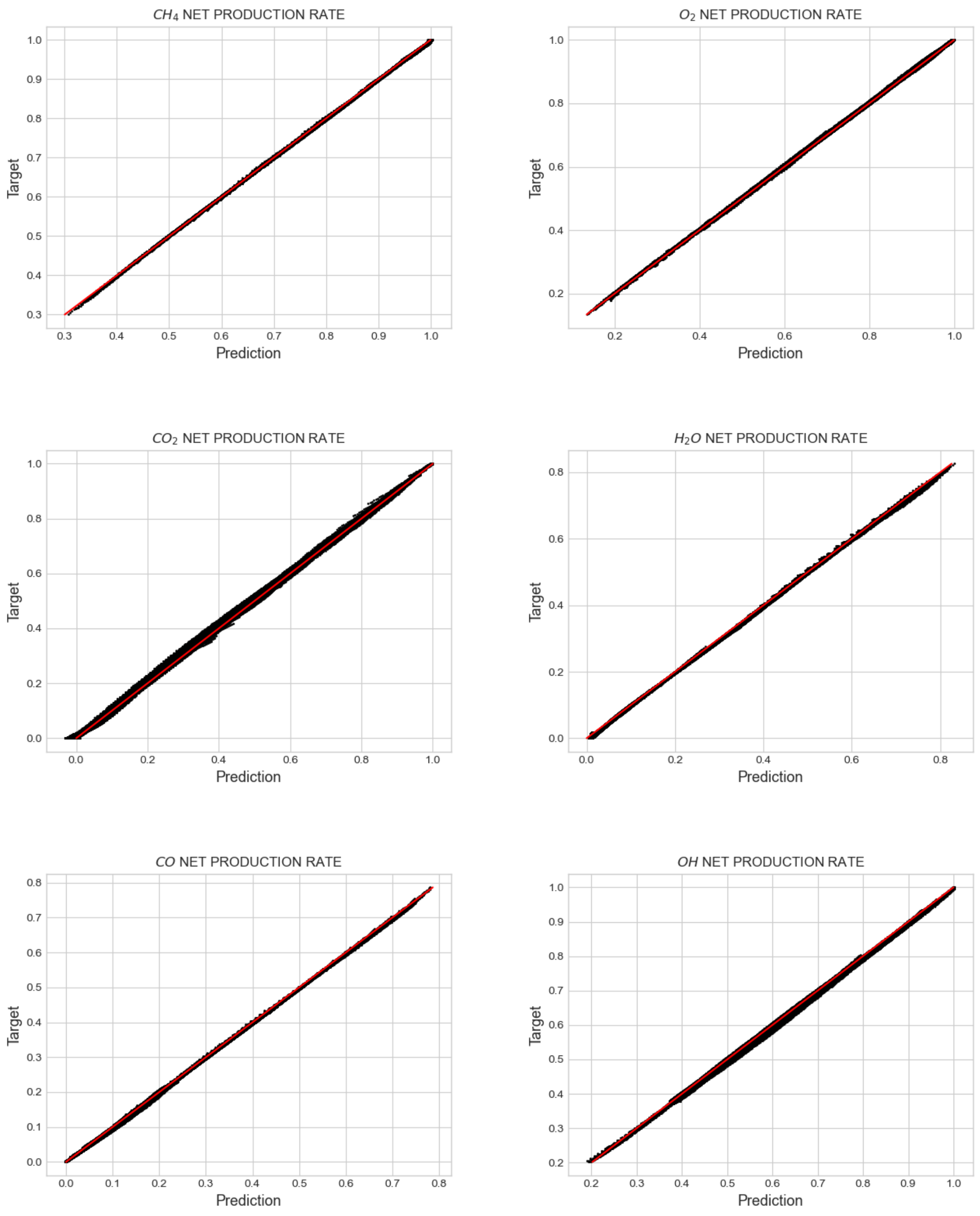


Figure 6-14: Prediction vs target output values of MLP assigned to Cluster 4, corresponding to temperature progress variable c = 0.72 – 1.

Figure 6-15 presents the root mean squared error (RMSE) of each feature contained in the MLP output vector, regarding Cluster 4. On average, the main reactants and the main products net production rates are predicted with excellent accuracy, while several minor species, such as N_2O, exhibit higher error values, probably due to the actual order of magnitude of these features. Similar behavior was observed for MLPs assigned to Cluster 1 and Cluster 2. The mean RMSE for the specific MLP is equal to $3.8 \cdot 10^{-3}$.

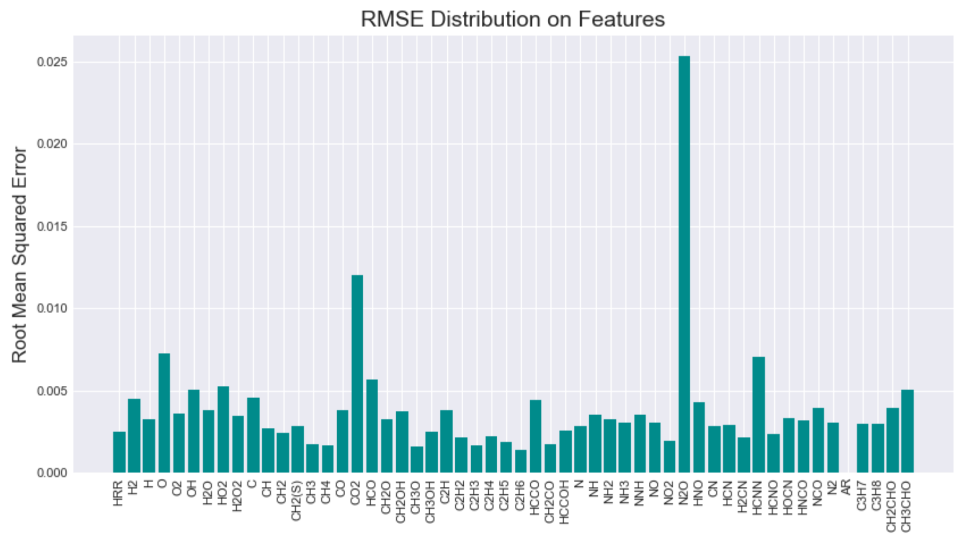


Figure 6-15: Root Mean Squared Error (RMSE) of each feature contained in the MLP output vector, regarding Cluster 4.

6.4 Discussion

As aforementioned, the state space is partitioned into four clusters and a 54-54 MLP is assigned in each cluster. The MLP is responsible for prediction of the heat release rate and the species net production rates, receiving a state vector as an input. Figures 6-16 and 6-17 depict both target and prediction values for the heat release rate and the main species net production rates. The target data are plotted with blue color, whereas the prediction values are plotted with red color.

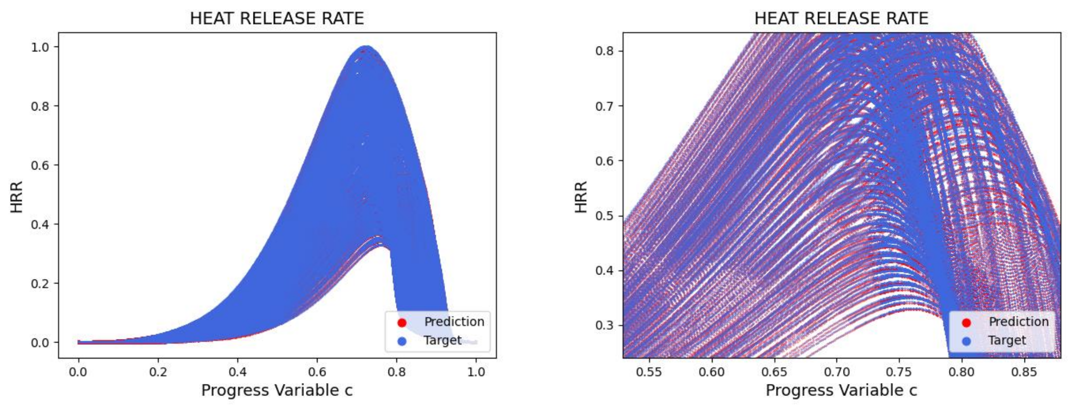


Figure 6-16: Heat release rate target and prediction values, for all four clusters of the state space. The diagram on the right is a partial enlargement of the left one.

According to Figure 6-16, the target values overlap with the prediction values, meaning that the prediction is identical with the target output. In the partial enlargement diagram, it is observed better that the curves representing the prediction outputs (red color) lie on top of the curves representing the target outputs (blue color). This observation indicates that the MLP regression of the state space is satisfactory. Similar observations can be noted on the subsequent diagrams (Figure 6-17), representing the net production rates of indicatively selected species. The species selected are the main reactants (CH_4, O_2), the main products (CO_2, H_2O) and the intermediate species/radicals (CO, OH) supporting the main heat release of methane-air mixture combustion process.

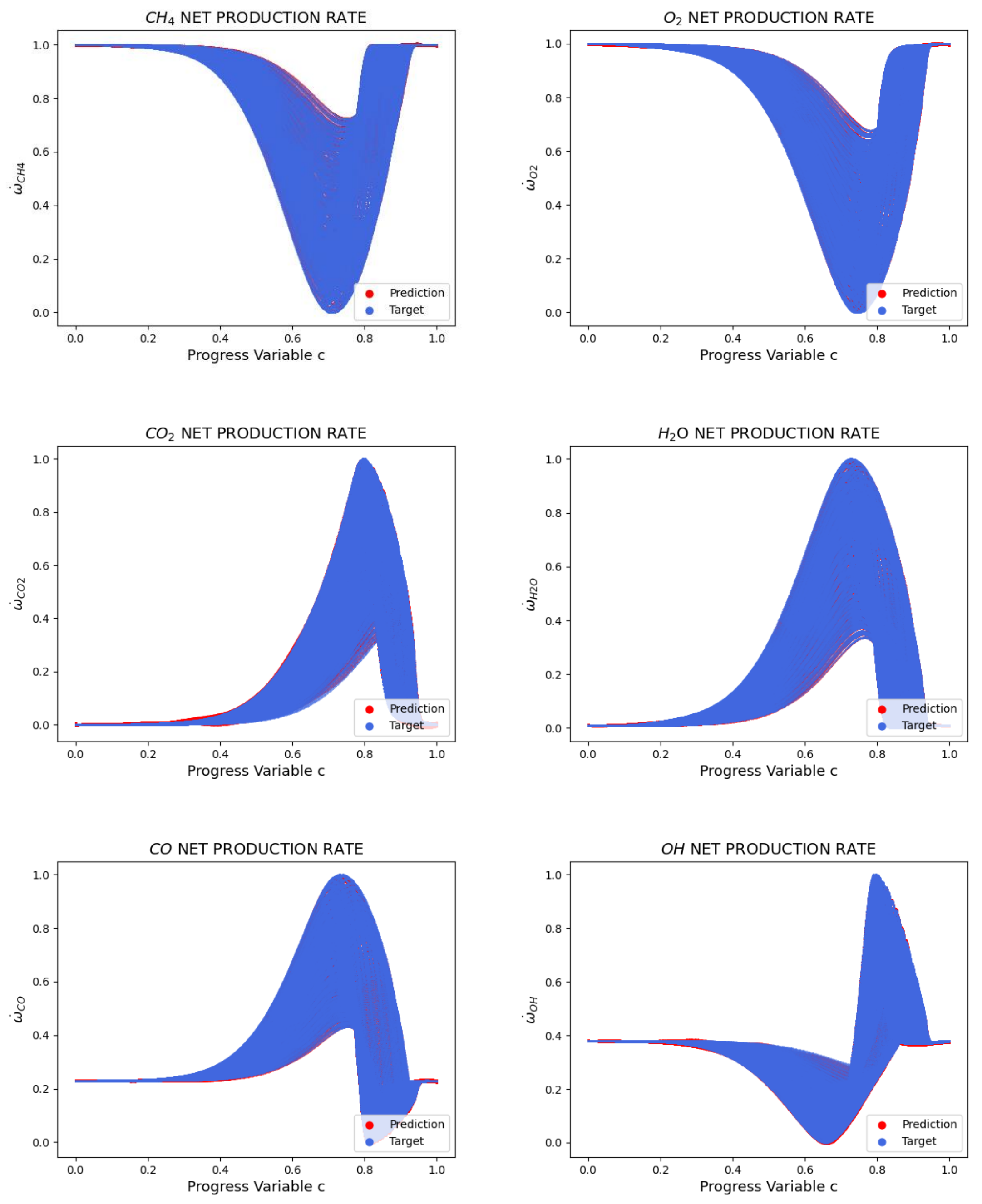


Figure 5-17: Species net production rates target and prediction values, for all four clusters of the state space.

Both Figures 6-16 and 6-17 demonstrate the prediction accuracy of all four developed MLPs, combined into a common figure. The prediction performance can be considered excellent for the state space covered in the current implementation. Small deviations can be observed mainly in the lower temperature region, where the red color becomes more obvious, an observation also denoted in the discussion of Cluster 1 results (Section 6.3.1). For a first trail on the concept, this is not considered as a loss in network accuracy, due to the order of magnitude of the error. However, this weakness of MLP assigned to Cluster 1 can probably be confronted through a filter, responsible to set the heat release rate and the net production rates equal to their constant values, appearing in the unburned reaction region, during the simulations.

Table 6-3 summarizes the average root mean squared error (RMSE) calculated for each one of the four developed MLPs. The accuracy can be considered sufficient, taking into account that the errors' order of magnitude is 10^{-3}, whereas the data are normalized into the [0,1] range. Higher errors appear in intermediate to high temperature zones, where the main proportion of chemical reaction occurs.

Table 6-3: Root mean squared error (RMSE) of each developed MLP.

MLP assigned to:	**RMSE**
Cluster 1	$2.2\cdot10^{-3}$
Cluster 2	$4.0\cdot10^{-3}$
Cluster 3	$5.4\cdot10^{-3}$
Cluster 4	$3.8\cdot10^{-3}$

CHAPTER 7: Prediction of Thermochemical State Time Evolution

The final step of the present work includes the development of an Artificial Neural Network for the chemistry tabulation and prediction of the state space temporal evolution, expected to be visited in combustion simulations. According to Chapter 4 discussion, the evolution of a chemical system is fully defined by integrating an ODE system, consisting of equations describing the changes of temperature and species concentrations.

In this context, the ANN is developed in order to predict the temporal evolution of a thermochemical state, given a thermochemical state vector as an input. In other words, it executes the time integration of chemistry, from a specific time t to t+δt, where δt is the selected integration time step. The thermochemical state is described by 54 features regarding the application of GRI Mech. 3.0.

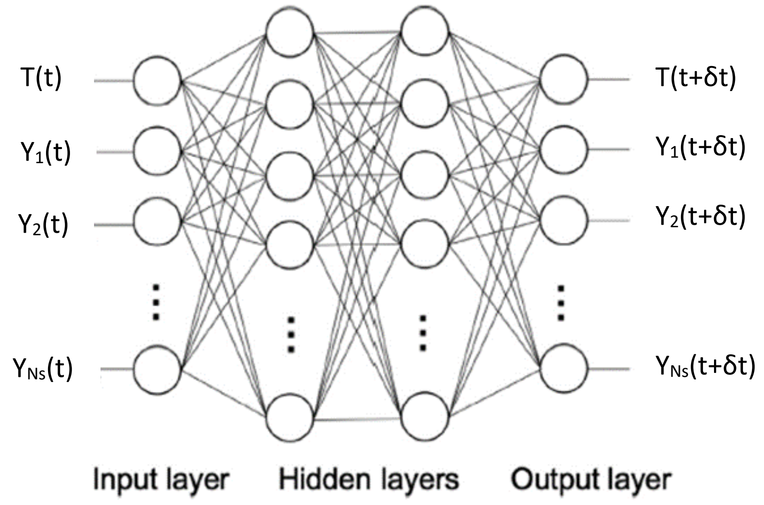


Figure 7-1: An illustration of the ANN function.

Given a mechanism, the production or consumption rate of each chemical species can be expressed by the following system of ODEs:

$$\frac{d[M_i]}{dt} = \dot{\omega}_i = (\nu_i'' - \nu_i') \cdot k \cdot \prod_{i=1}^{N_s} [M_i]^{\nu_i'}$$

since ν_i'' moles of M_i are created for every ν_i' moles of M_i consumed and $[M_i]$ denotes the molar concentration of species *i*. Additionally, $\dot{\omega}_i$ will be called net production rate of species *i* for the sake of brevity. The proportionality constant, *k*, is the reaction rate constant, and introduces the temperature dependence for reaction rates. It is calculated through Arrhenius Law (Section 4.1.1).

Species concentration can be converted to mass fraction through the formula:

$$[M_i] = \frac{n_i}{V} = \frac{m_i/MW_i}{V} = \frac{m_i}{m} \cdot \frac{m}{V} \cdot \frac{1}{MW_i} = Y_i \cdot \frac{\rho}{MW_i}$$

where n_i are the moles of species *i*, V is the system volume, m_i is the mass of species *i*, MW_i is the molecular weight of species *i* and m is the total mass contained in the system volume. Correspondingly, ρ is the density of gas contained in the system volume and Y_i is the mass fraction of species *i*. Therefore, the temporal evolution of the species mass fraction can be calculated by integrating the equation:

$$\frac{dY_i}{dt} = \frac{\dot{\omega}_i \cdot MW_i}{\rho}$$

The time evolution of temperature can be obtained from Energy Conservation (Section 4.1.2):

$$\frac{dT}{dt} = -\frac{\sum_{i=1}^{N_s}\left(h_i \cdot \frac{dY_i}{dt}\right)}{c_p}$$

In conclusion, the ANNs shall be able to learn and regress the above mathematical relationships for a user-defined level of accuracy. These equations are non-linear and stiff.

The state space is divided into four clusters using the hybrid SOM – K-Means neural network for Case 2 of data normalization, which corresponds to all features standardized into the [0,1] range. Multi-Layer Perceptrons are used for regression, applying the concept illustrated in Figure 7-1. The ANN receives a 54-dimension state vector as an input and predicts the temporal evolution of this state vector.

The Artificial Neural Network training must be done on data obtained from simulations of the same phenomenon or, alternatively, from simulations of an abstract problem. Regarding the latter case, it is important to ensure that the generated training dataset covers the state space accessed during the multi-dimensional CFD simulation. An ensemble of premixed laminar flames was previously used to generate a dataset covering a specific part of state space. These thermochemical states are loaded as input data in homogeneous reactor simulations, where time integration is executed and the evolution of thermochemical states after a specific time step is calculated.

7.1 Sample Generation

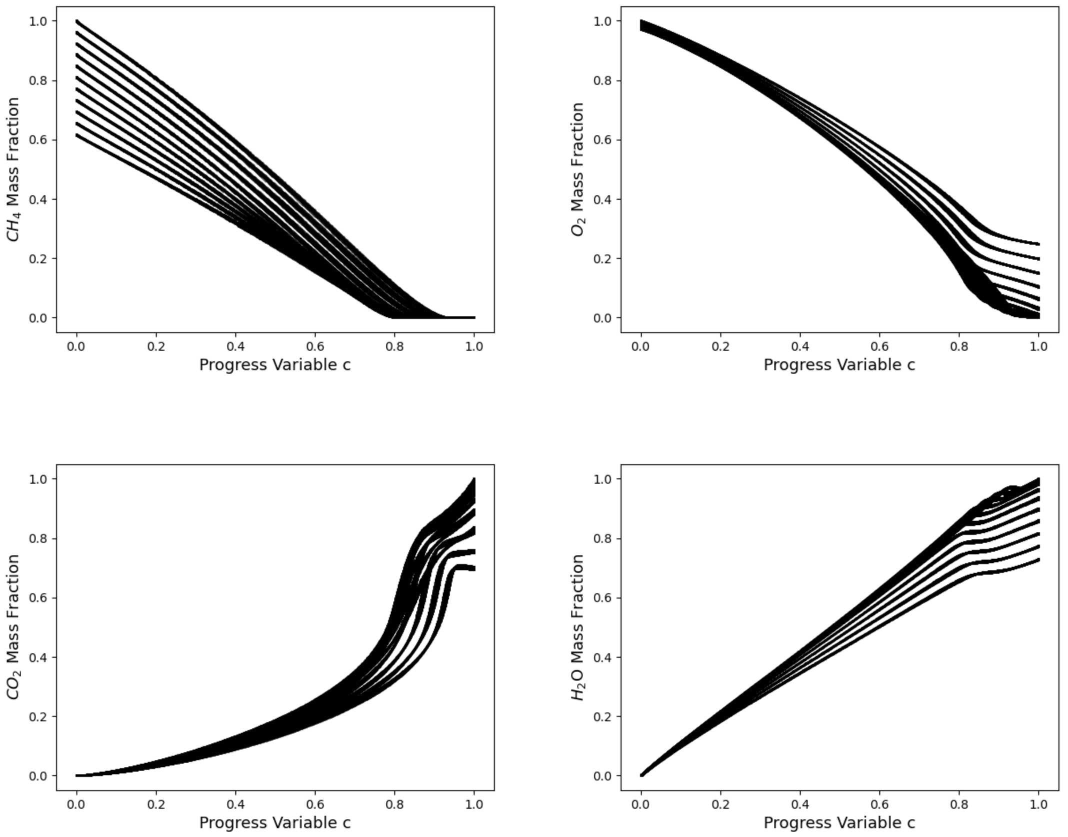
The approach followed in the present work implies that the developed ANNs shall be able to generalize well, instead of focusing on specialized state space domains and operating sufficiently only under specific thermochemical conditions. Therefore, the training data are decided to be obtained through a canonical problem, while covering the state space expected to be encountered in future simulations.

The generation of training data for the Multi-Layer Perceptron training process is carried out by integrating homogeneous, constant-pressure, zero-dimensional reactors with adiabatic, chemically inert walls. Commonly, a time step $\delta t = 10^{-6}$ s is sufficient for chemistry integration, however depending on the integration method and the state. The current study deals with methane-air premixed mixture combustion. The simulations are executed using the CANTERA open-source suite in a Python code, as described in Section 4.4.

The simulation conditions are not limited in a range of temperature and fuel-air equivalence ratio, as they were in premixed laminar flame simulations (Section 5.1). Each state vector generated by the premixed laminar flame simulations is introduced as an input condition in the homogeneous reactor. Therefore, the simulation conditions of the homogeneous reactor basically refer to the state space assessed by the premixed laminar flame simulations. Afterwards, a specific time step is defined and then time integration is executed in order to calculate the new state vector. The present concept is based on the state space partitioning and thermochemical states regression, meaning that only temperature and species mass fractions are needed to generate a state vector.

The concept developed in the present Chapter is to test the ability of the 54-54 configuration of MLPs to predict the value of temperature and species mass fractions after a time step δt, instead of direct calculation of the time derivative, a concept studied in the previous chapter (Chapter 6). For reasons of simplification of the procedure, the state vectors only from the initial 121 methane-air premixed laminar flames are used, which correspond in total 127099 data samples. The data samples are combinations of input state vectors and output state vectors. For the purpose of the MLP training, the data samples are a priori split into 60% training set and 40% test set, similarly to the case of the SOM training process. The cross validation, a method which is applied in order to tune the network hyperparameters, is not introduced in the present case study, as the interest focuses on evaluation of the already developed 54-54 MLP configuration. A notable observation is that the produced dataset has no outliers, as it is generated through robust time integration. Therefore, there is no need for application of data cleaning techniques before proceeding to neural network training responsible for regression, similarly to the neural network training responsible for state space clustering.

Figure 7-2 depicts the state space coverage from the full dataset obtained through laminar flame simulations. These composition states enter each MLP as an input vector. All features are normalized into the [0,1] range corresponding to Case 2 of data normalization. The diagrams presented are indicative of the temperature distribution of the main reactants (CH_4, O_2), the main products (CO_2, H_2O) and the intermediate species/radicals (CO, OH) supporting the main heat release of methane-air mixture combustion process. Temperature is normalized via the progress variable, *c*.



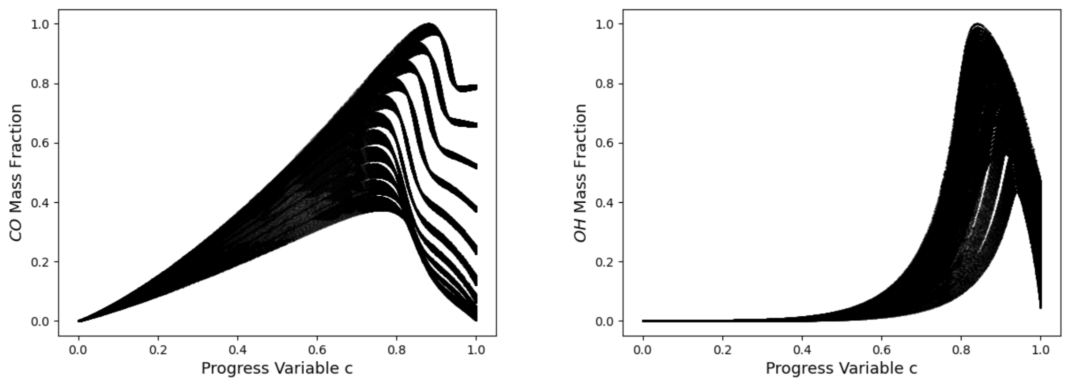
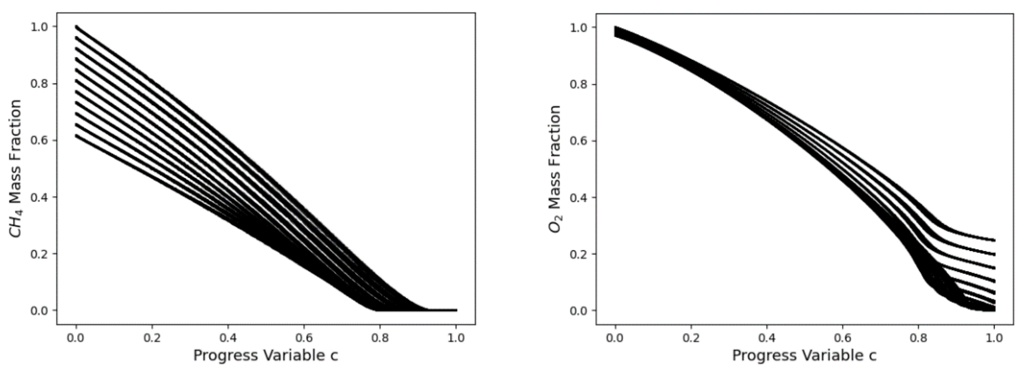


Figure 7-2: Scatter plots presenting the initial state space coverage of the data collected from methane-air premixed laminar flame simulations.

The desired output of MLPs is the state after a specified time step is elapsed. These composition states are obtained via the homogeneous reactor simulations and are certainly functions of the initial state. Due to the differences in the order of magnitude between the temperature and the species mass fractions, normalization of all features into the [0,1] range is considered mandatory. Thus, both input and output vectors of MLPs are normalized into the [0,1] range, a technique offering better behavior and robustness to the neural network, while the adjusted weights acquire values in the same order of magnitude.

Figure 7-3 depicts the state space coverage from the full dataset obtained through homogeneous reactor simulations for a specified time step $\delta t = 10^{-6}$ s. The diagrams presented are indicative of the temperature distribution of the main reactants (CH_4, O_2), the main products (CO_2, H_2O) and the intermediate species/radicals (CO, OH), supporting the main heat release of methane-air mixture combustion process. Temperature is normalized via the progress variable, *c*.



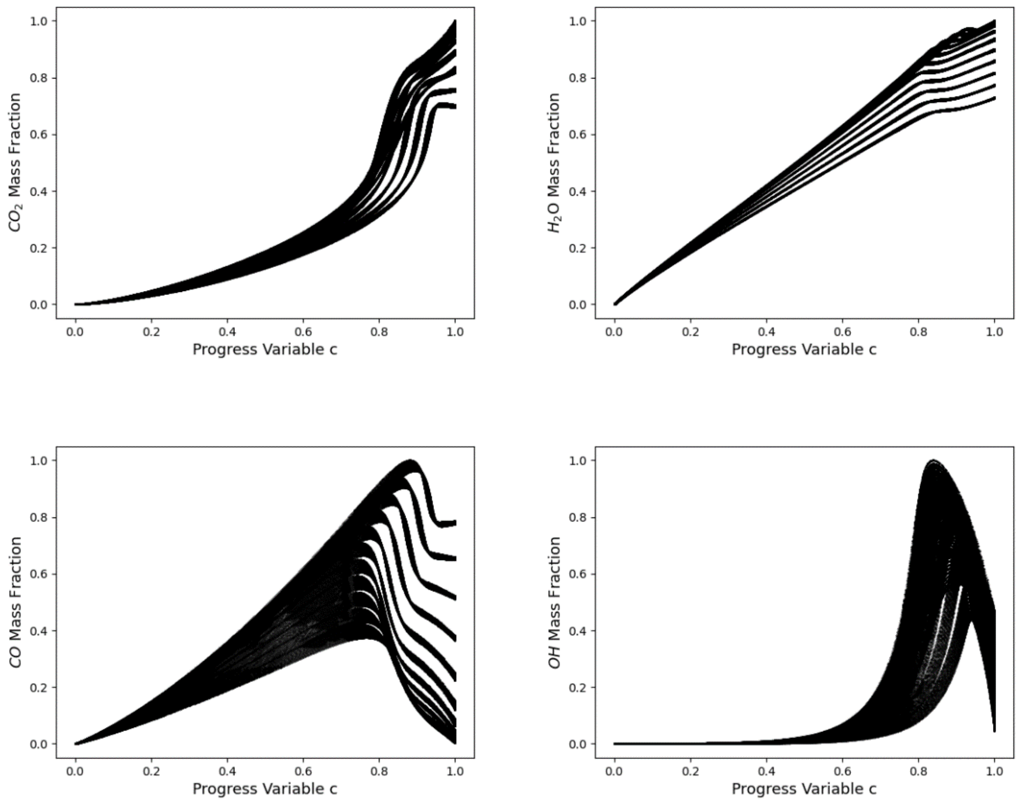


Figure 7-3: Scatter plots presenting the final state space coverage of the data collected from homogenous reactor simulations, after a specified time step δt=10^{-6} s.

An important observation extracted from Figure 7-3 is that the states after a time step δt=10^{-6} s are slightly changed in comparison to the initial ones (Figure 7-2). This observation implies that the time step is small enough, and little alterations in temperature and compositions occur. Furthermore, the neural network weights are expected to acquire values close to zero, because the output does not differ to a high degree from the input vector.

7.2 Multi-Layer Perceptron Training Process

The neural network applied for regression in the present work is the Multi-Layer Perceptron, a type of feedforward Artificial Neural Network. As discussed in Section 3.3, the network is trained through a back-propagation method with the Adam optimization algorithm.

Before the training process starts, the network hyperparameters shall be defined, similarly to the Multi-Layer Perceptrons discussed in Section 6.2. The hyperparameters of an MLP neural network are the following:

- Number of hidden layers.
- Number of neurons per layer.
- Activation function. Hyperbolic tangent is employed due to its property of being a zero-centered, smooth differentiable function.
- Maximum number of epochs. An epoch is defined as a cycle of iterations, where all training data are used exactly one time. Iterations terminate based on the loss function convergence.

The number of hidden layers and the number of neurons per layer defines the neural network architecture. Commonly, the architecture, as well as other network hyperparameters, are selected by means of trial-and-error, resulting in a configuration that balances accuracy and computational cost. A more robust method used extensively to define neural network hyperparameters is the cross validation technique, examined in Section 3.3.1.

In the context of the current study, the already developed 54-54 MLP configuration is evaluated in terms of state prediction, instead of direct derivative prediction, for purposes of comparison with the concept studied in the previous chapter (Chapter 6). Additionally, taking into account relative applications of combustion data regression and prediction of thermochemical states found in literature (e.g. [8],[9]), a 54-54 neural network configuration can be considered as a sufficient selection, balancing accuracy and computational cost.

The concept of the current implementation consists of two neural networks; firstly, a hybrid SOM – K-Means neural network is applied to cluster the state space. Next, an MLP is assigned to each cluster, in order to learn and predict the desired patterns of input data. Regarding Case 2 of data normalization, where all data features are scaled into the [0,1] range, four clusters are identified by the hybrid neural network responsible for the state space clustering. Therefore, four MLPs will be developed, each one assigned to a specific cluster.

7.3 Prediction Results

Multi-Layer Perceptrons serve the role of the basic regression algorithm of state space data. After training is finished, the trained MLPs can predict the target output, receiving a state vector as an input. The output that each MLP is assigned to predict is the state vector after a specified time step. In total, four MLPs are developed, each one assigned to each cluster generated by the hybrid SOM – K-Means neural network.

Both MLP input and output data are state vectors of dimension equal to the state space dimension, which for the considered case is equal to 54.

The following paragraphs discuss the results obtained by the trained MLPs, for each cluster generated by the hybrid neural network responsible for clustering of state space. For convenience, the clustering of state space shown in Figure 7-4 is taken from Section 5.5.

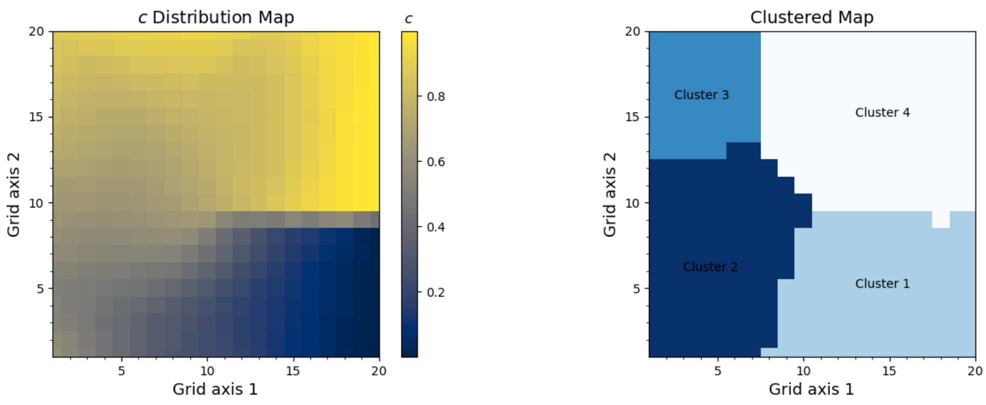


Figure 7-4: Progress variable, *c*, distribution map (left) and clustered SOM grid (right), by grouping the closest neurons into subdomains via K-Means method, for **Case 2** of data normalization.

Table 7-1 summarizes the temperature region spanned by each cluster in terms of the progress variable, *c*.

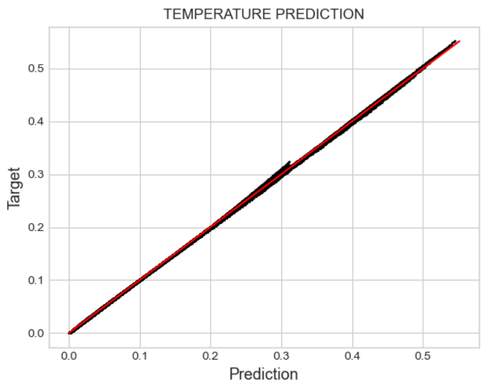
Table 7-1: Clusters of state space generated by hybrid SOM – K-Means neural network.

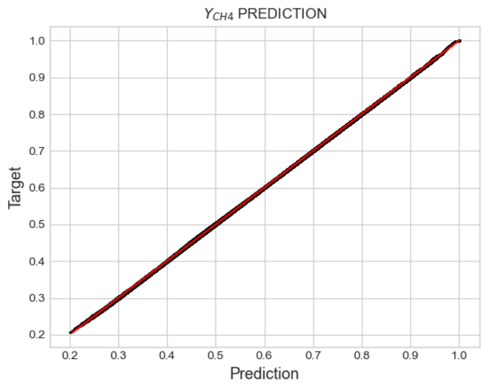
	Range of Progress Variable, *c*
Cluster 1	0 – 0.55
Cluster 2	0.27 – 0.74
Cluster 3	0.67 – 0.93
Cluster 4	0.72 – 1

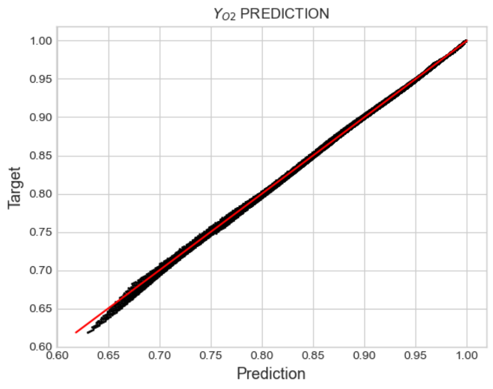
7.3.1 Results for Cluster 1

Cluster 1 ($0 \leq c \leq 0.55$) corresponds to the lower temperatures appearing in the state space covered in the present study. It spans the unburned gas region and a part of the reaction regime, where changes in temperature and species compositions start to occur.

Figure 7-5 shows the predicted temperature and species mass fractions after a specific time step versus the actual ones, as they are obtained from homogenous reactor simulations for the specified time step, $\delta t=10^{-6}$ s. The species presented are the main reactants (CH_4, O_2), the main products (CO_2, H_2O) and the intermediate species/ radicals (CO, OH) supporting the main heat release of methane-air mixture combustion process.







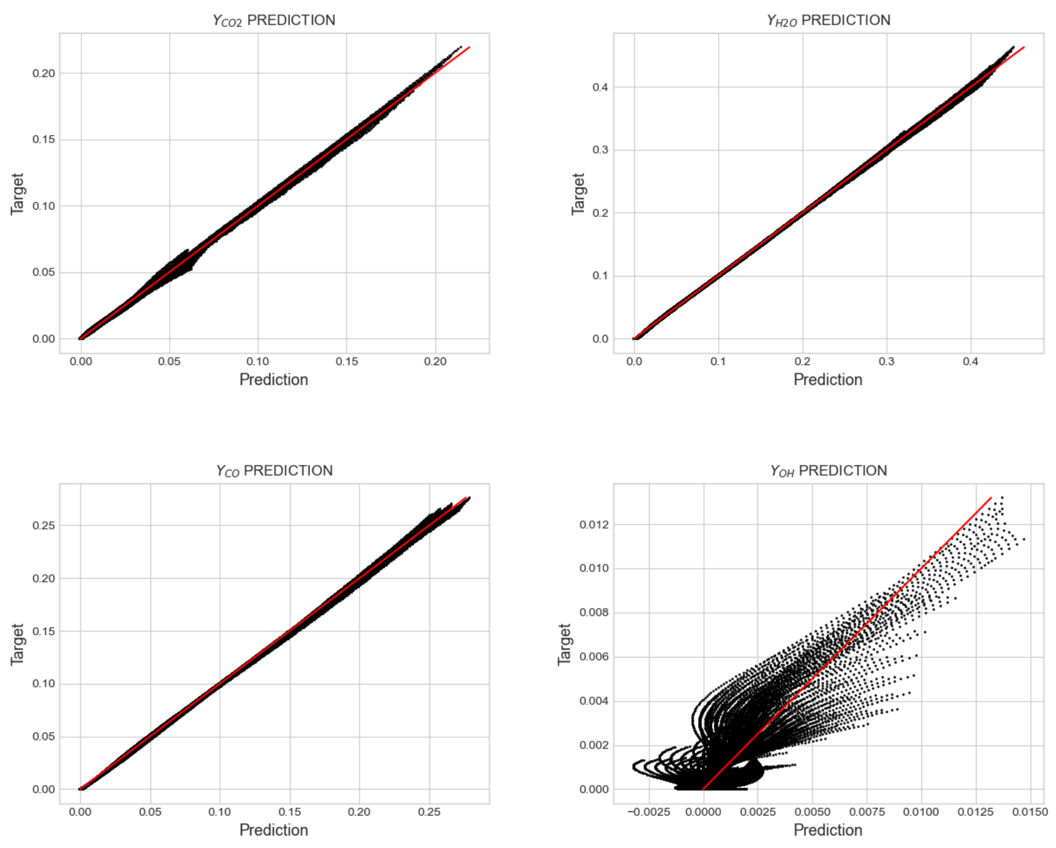


Figure 7-5: Prediction vs target output values of the MLP assigned to Cluster 1, corresponding to temperature progress variable $c = 0 – 0.55$.

The diagrams of Figure 7-5 are basically a representation of the confusion matrix. If the points lie on the straight line $y = x$, then the prediction is exactly equal to the target output, and the fit is considered excellent. Studying the above diagrams, the fit seems excellent for all main features, except for hydroxyl radical (OH), where an irregular behavior is observed for several values, which becomes more intense due to the values that the specific species acquire. In other words, the deviation of prediction from target hydroxyl radical mass fraction is optically intensified due to the fact that this species ranges in small order of magnitude values in the specific cluster. In general, the MLP prediction performance is considered satisfactory for Cluster 1, at least for the temperature and the main species temporal evolution, as the predicted data are mainly plotted on the $y = x$ line, meaning that they are approximately equal to the target data. The use of the word “approximately” is justified by the fact that a finite error exists, as it is subsequently presented.

Figure 7-6 presents the RMSE of each feature contained in the MLP output vector, regarding Cluster 1. On average, the temperature and the main species mass fractions are predicted with excellent accuracy, while several minor species, such as C_3H_7, exhibit higher error values, probably due to the actual order of magnitude of these features. The mean RMSE for the specific MLP is equal to $1.9 \cdot 10^{-3}$.

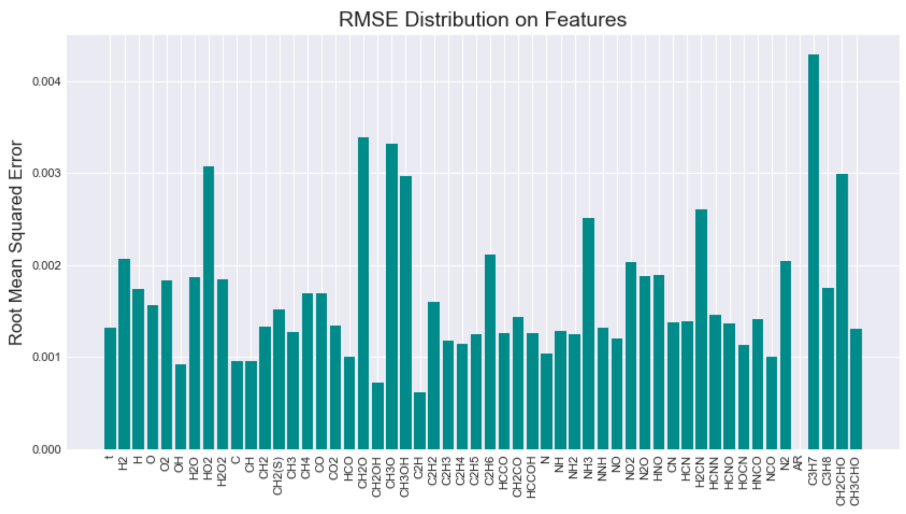
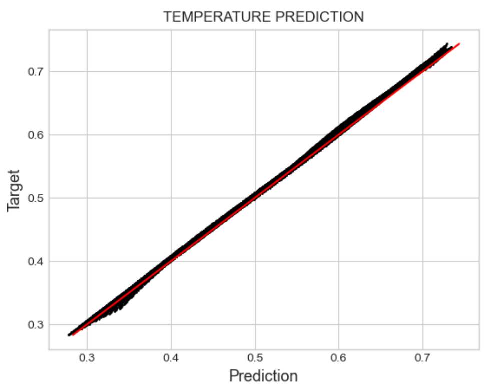


Figure 7-6: Root Mean Squared Error (RMSE) of each feature contained in the MLP output vector, regarding Cluster 1. *t* denotes the normalized temperature through the progress variable, *c*.

7.3.2 Results for Cluster 2

Cluster 2 ($0.27 \leq c \leq 0.74$) corresponds to intermediate temperatures appearing in the state space covered in the present study. It spans the preheating zone and a part of the main reaction zone, where high changes in temperature and species composition occur in both of them. A significant overlap between Cluster 1 and Cluster 2 is observed in terms of the corresponding values of the progress variable, *c*. This can be justified by the intrinsic property of the clustering algorithm to create groups of approximately equal number of data, while at the same time dealing with high-dimensional data, resulting in overlapping of features in several projections of the high-dimensional space.



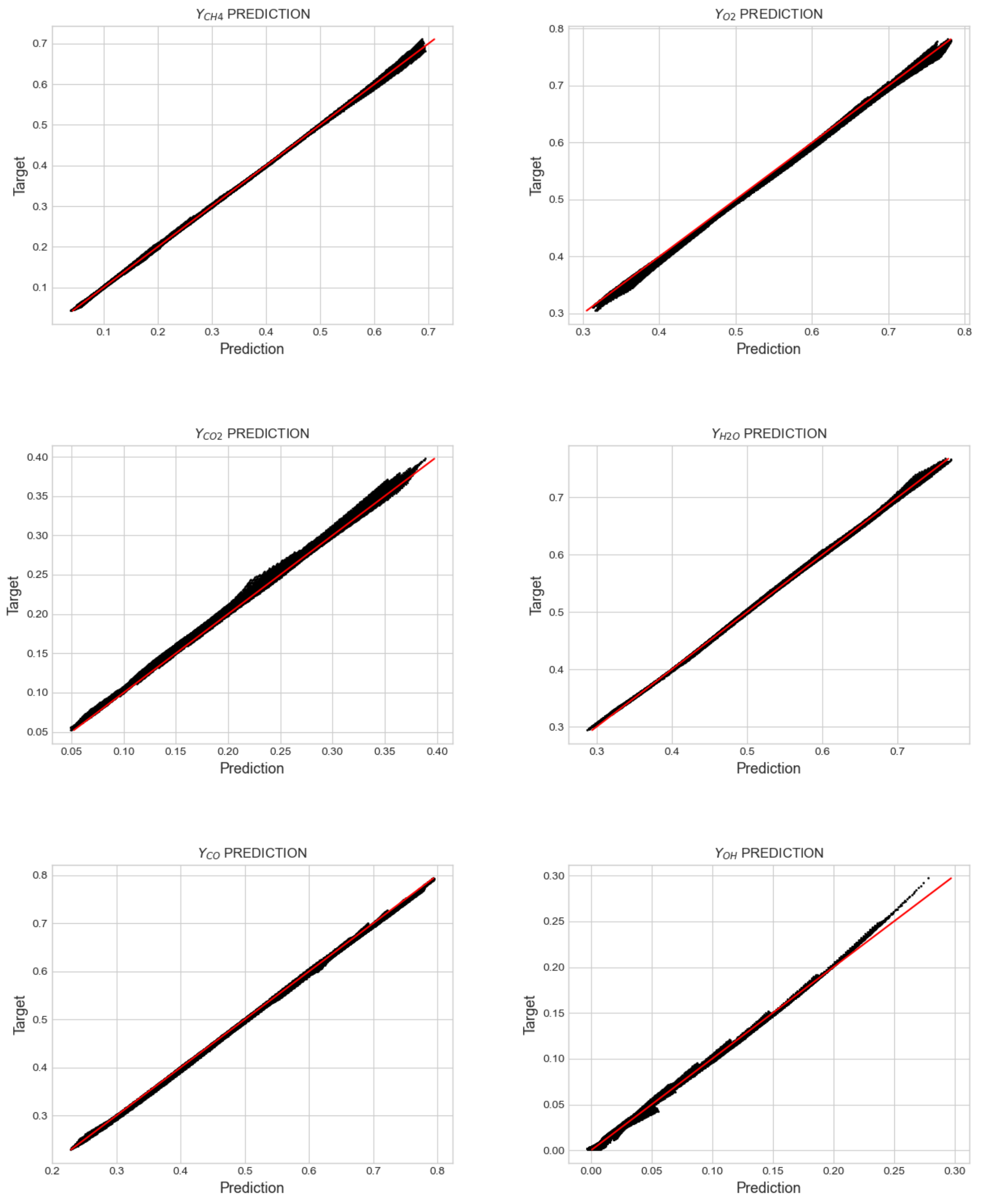


Figure 7-7: Prediction vs target output values of the MLP assigned to Cluster 2, corresponding to temperature progress variable c = 0.27 – 0.74.

Figure 7-8 represents the RMSE of each feature contained in the MLP output vector, regarding Cluster 2. On average, the temperature and the main species mass fractions are predicted with excellent accuracy, while several minor species, such as C_2H_5 and C_3H_7, exhibit higher error values, probably due to the actual order of magnitude of these features. The mean RMSE for the specific MLP is equal to $4.6 \cdot 10^{-3}$.

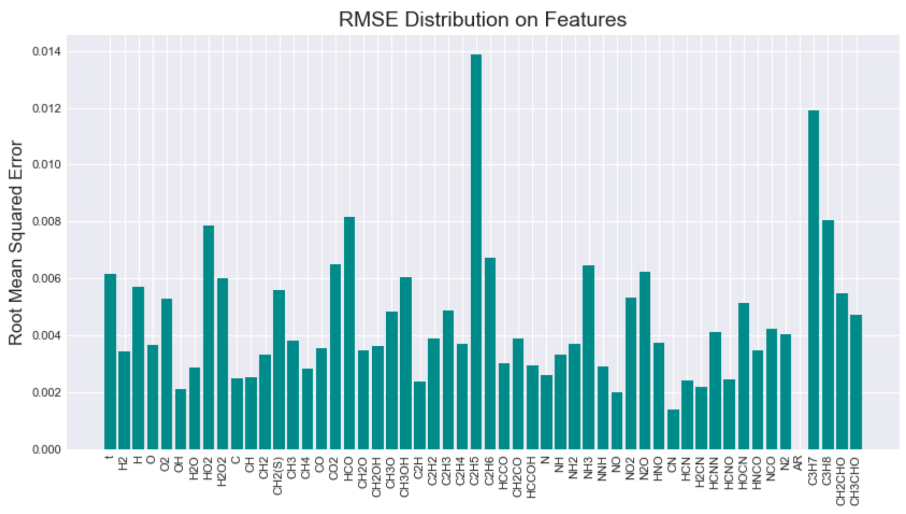
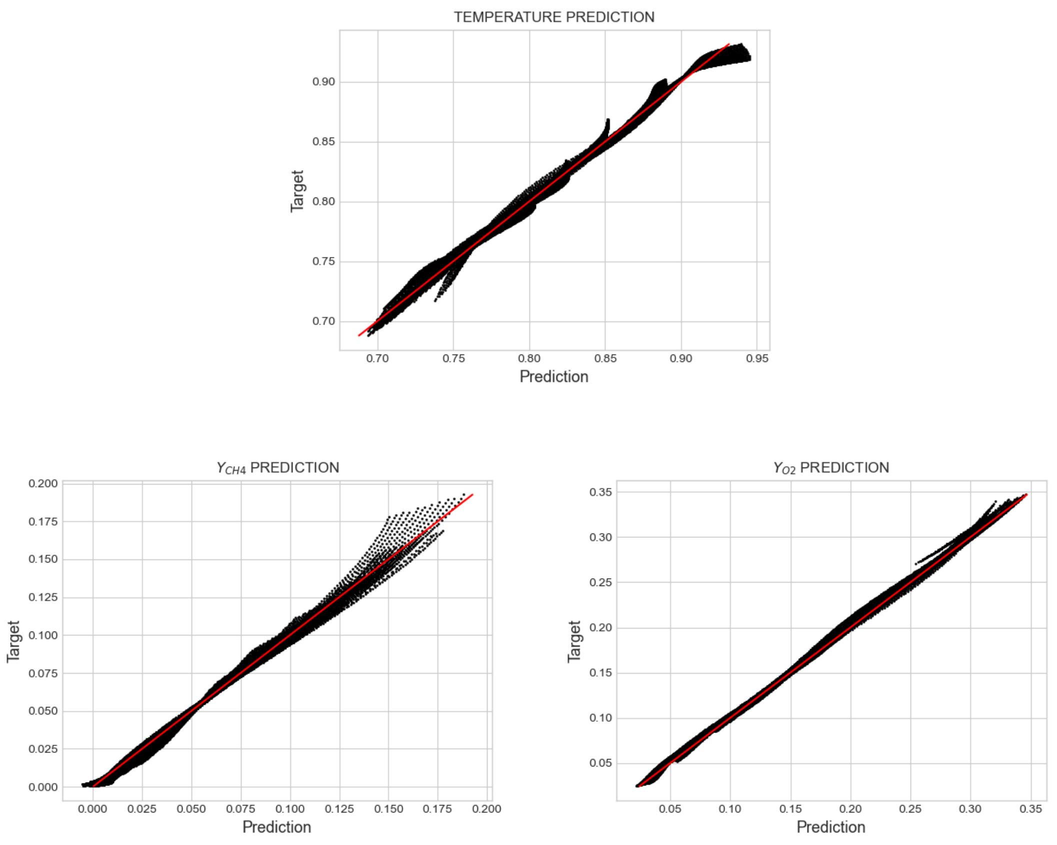


Figure 7-8: Root Mean Squared Error (RMSE) of each feature contained in the MLP output vector, regarding Cluster 2. *t* denotes the normalized temperature through progress variable, *c*.

7.3.3 Results for Cluster 3

Cluster 3 ($0.67 \leq c \leq 0.93$) corresponds to intermediate to high temperatures appearing in the state space covered in the present study. It spans the main reaction regime, where the highest changes in temperature and species compositions occur. An overlap between Cluster 2 and Cluster 3 is also observed in terms of the corresponding values of progress variable, *c*.



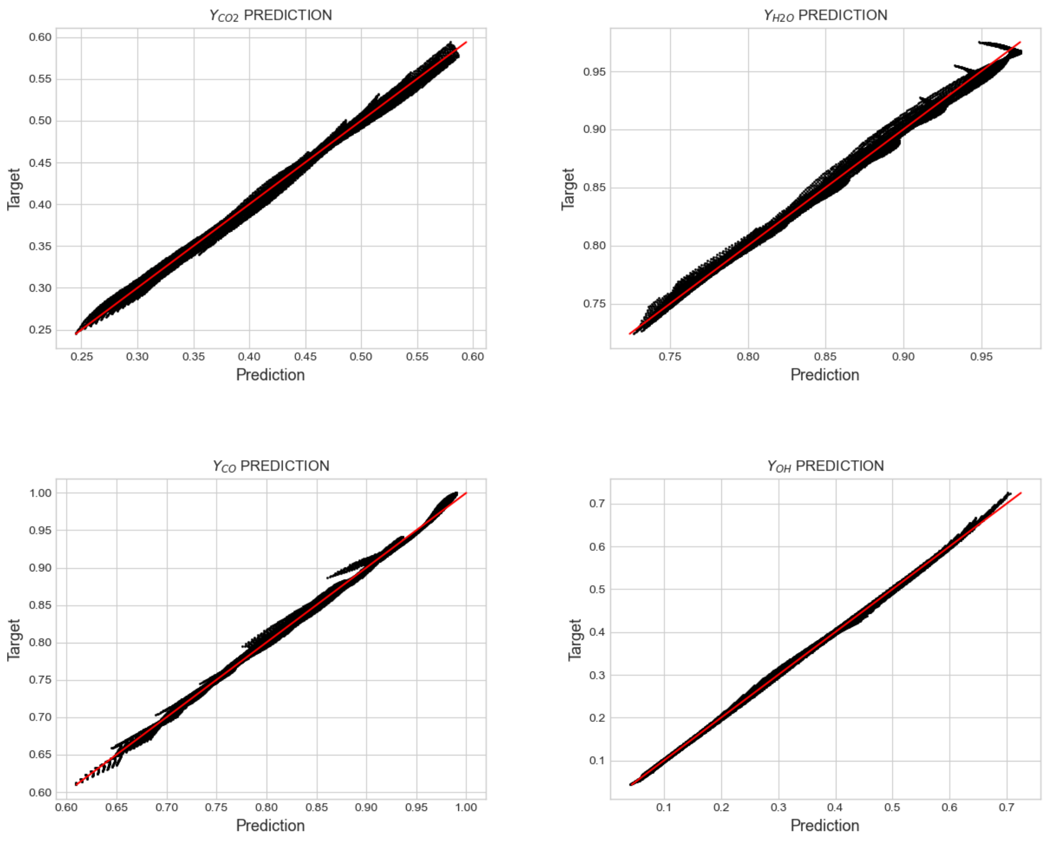


Figure 7-9: Prediction vs target output values of the MLP assigned to Cluster 3, corresponding to temperature progress variable c = 0.67 – 0.93.

Figure 7-10 represents the RMSE of each feature contained in the MLP output vector, regarding Cluster 3. There is not much deviation of the error between all features. This is probably due to the fact that radicals and minor species are activated for a small spatial window, contained in the specific cluster, hence their acquired values are in the same order of magnitude as the ones of main species mass fractions. The mean RMSE for the specific MLP is equal to $5.4 \cdot 10^{-3}$.

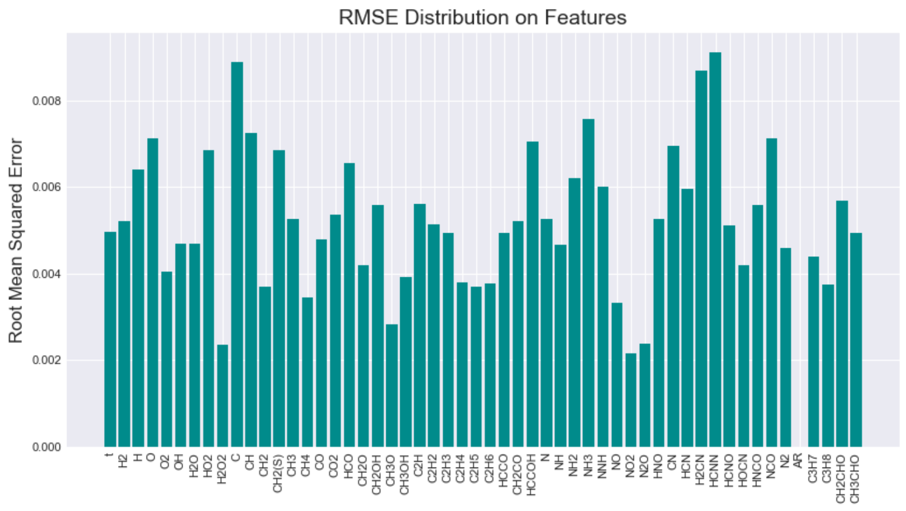
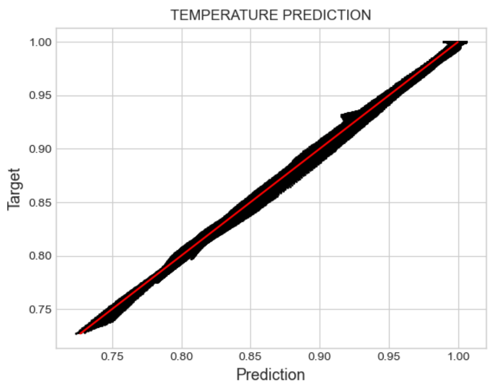
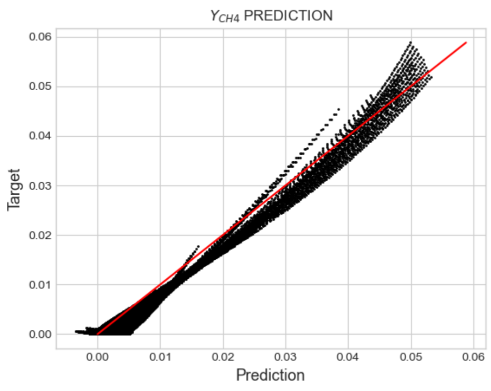


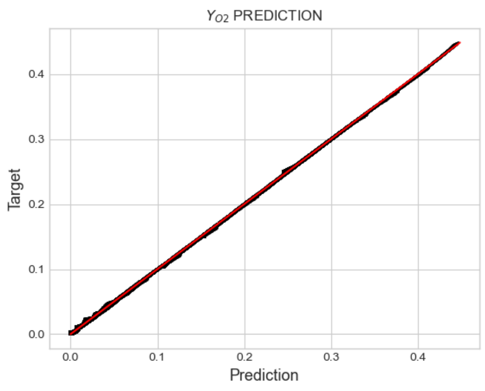
Figure 7-10: Root Mean Squared Error (RMSE) of each feature contained in the MLP output vector, regarding Cluster 3. t denotes the normalized temperature through progress variable, c.

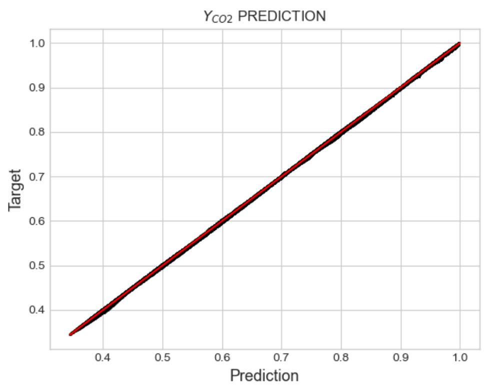
7.3.4 Results for Cluster 4

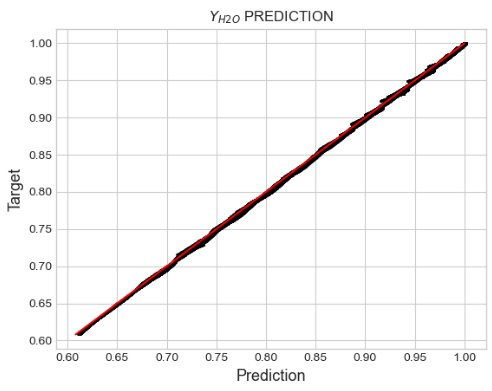
Cluster 4 ($0.72 \leq c \leq 1$) corresponds to the higher temperatures appearing in the state space covered in the present study. It spans the burned gas region and a part of the reaction regime, where several species are oxidized or recombined, and the changes occurring in temperature and species concentrations are not as high as in Cluster 3. A significant overlap between Cluster 3 and Cluster 4 is also observed in terms of the corresponding values of progress variable, *c*.











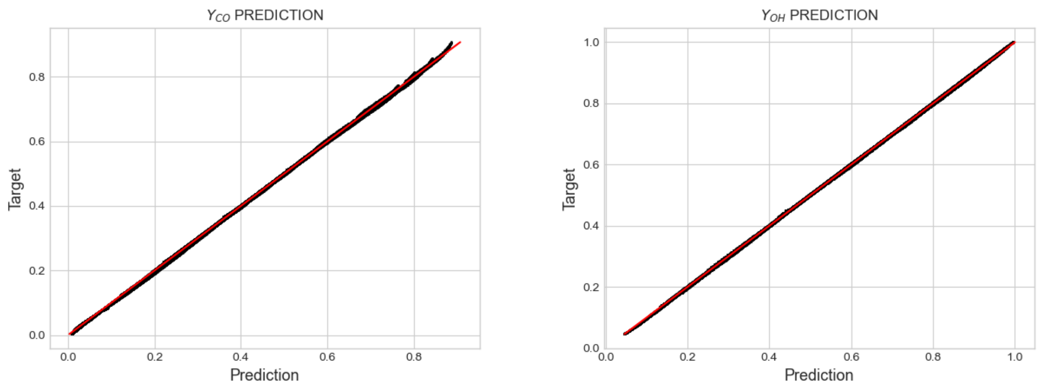


Figure 7-11: Prediction vs target output values of the MLP assigned to Cluster 4, corresponding to temperature progress variable *c* = 0.72 – 1.

Figure 7-12 presents the RMSE of each feature contained in the MLP output vector, regarding Cluster 4. There is also not much deviation of error between all features, similarly to Cluster 3. This is probably due to the fact that radicals and minor species remain activated for a small spatial window, contained in the specific cluster, and are then oxidized or recombined. Hence, their acquired values are of the same order of magnitude as the ones of main species mass fractions. The mean RMSE for the specific MLP is equal to 2.5·10^{-3} s.

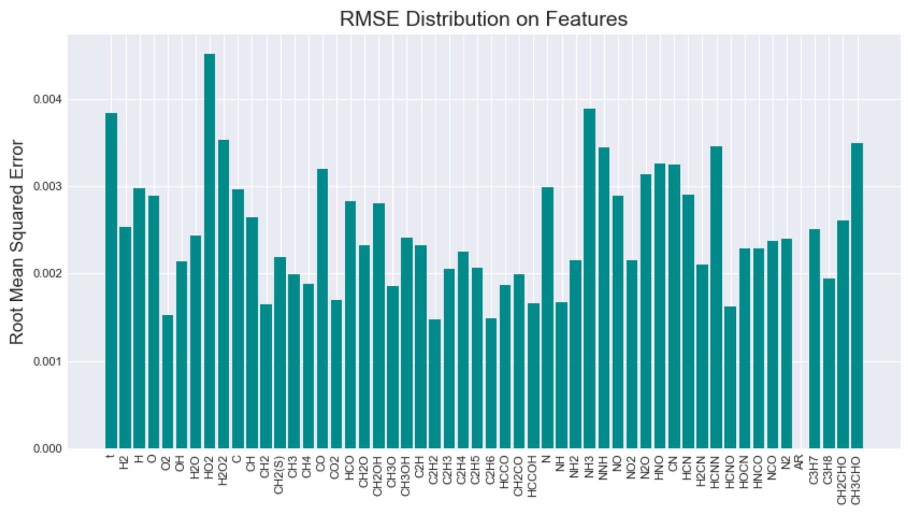


Figure 7-12: Root Mean Squared Error (RMSE) of each feature contained in the MLP output vector, regarding Cluster 4. *t* denotes the normalized temperature through progress variable, *c*.

7.4 Discussion

As previously mentioned, the state space is partitioned into four clusters, and a 54-54 MLP is assigned in each cluster. The MLP is responsible for the prediction of state after a specified time step, receiving an initial state vector as an input.

Figure 7-13 displays both target and prediction values for the temperature distribution of indicative species mass fractions, after the specified time step δt=10^{-6} s. The target data are plotted with blue color, whereas the prediction values are plotted with red color.

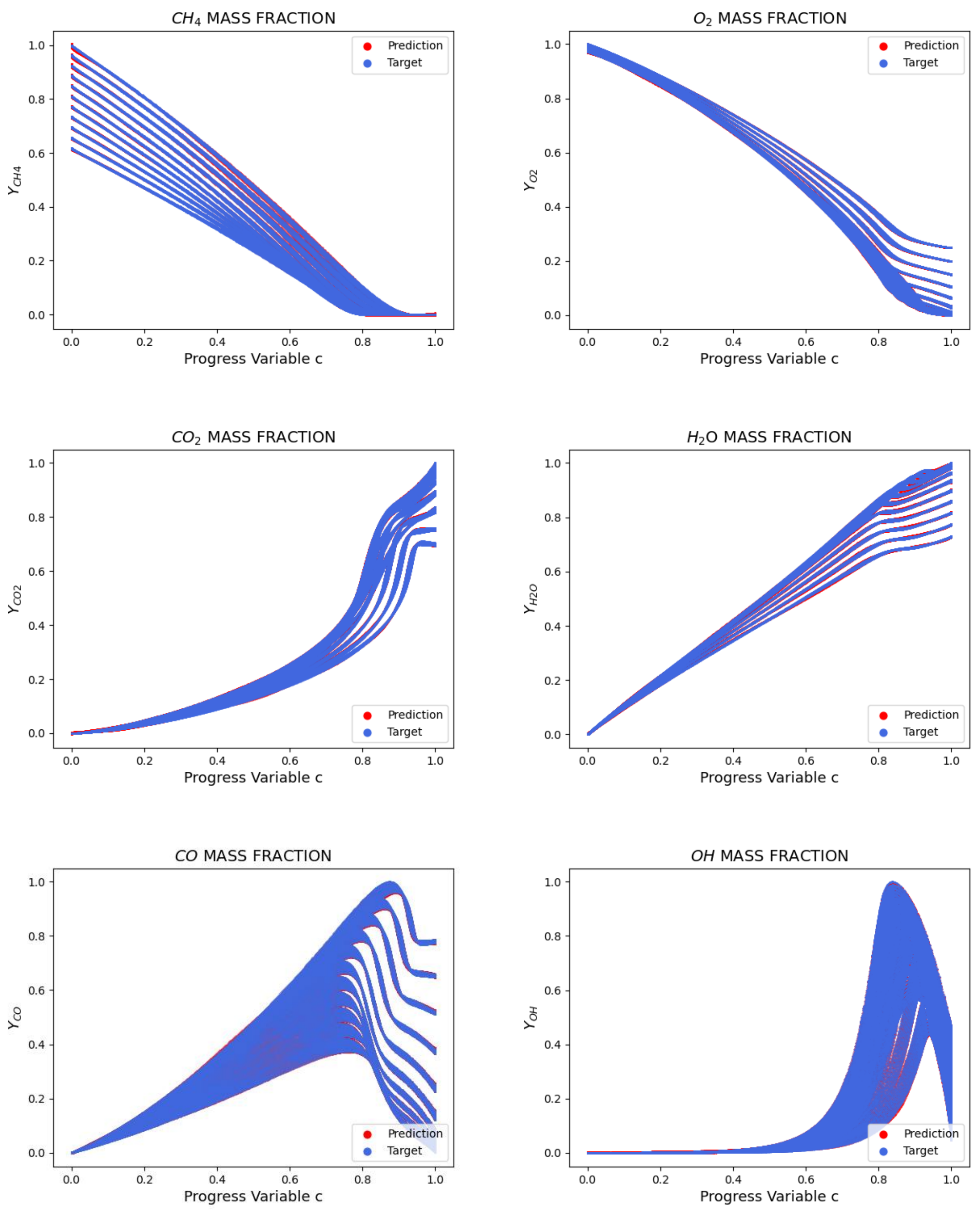
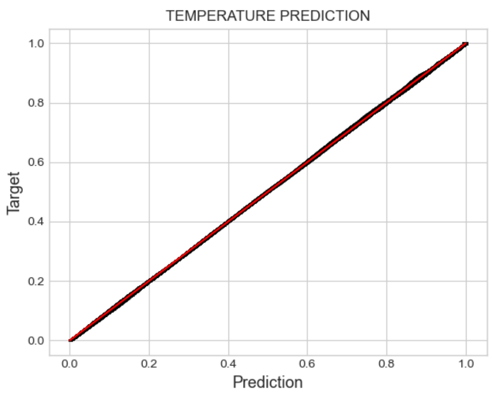
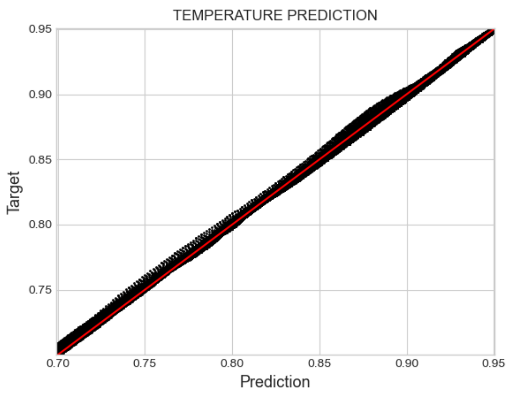


Figure 7-13: Temperature distribution of species mass fractions target and prediction values after the specified time step δt=10^{-6}, for all four clusters of the state space.

According to Figure 7-13, the target values overlap with the prediction values, meaning that the prediction is identical to the target output. If a diagram is enlarged, it is observed better that the points representing the prediction outputs (red color) lie on the points representing the target outputs (blue color). Small deviations can be observed, mainly in the borders of covered state space corresponding to the reaction regime, where the red color becomes more obvious. Overall, the MLP regression of states is considered accurate.

MLP performance can also be evaluated through prediction versus target output diagrams, presented in Figures 7-14 and 7-15, in terms of temperature and main species mass fractions, respectively.





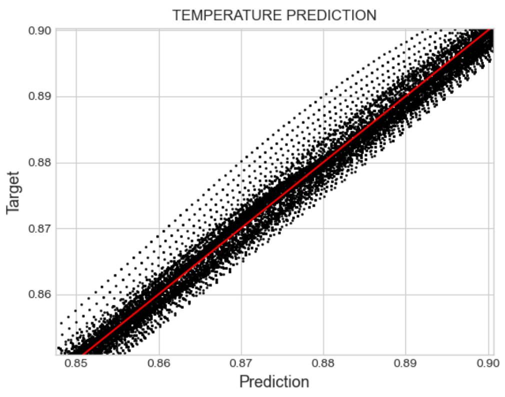
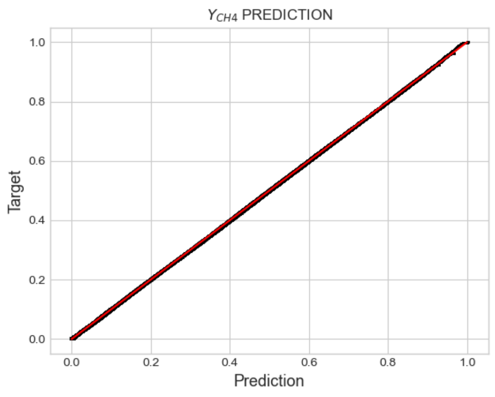
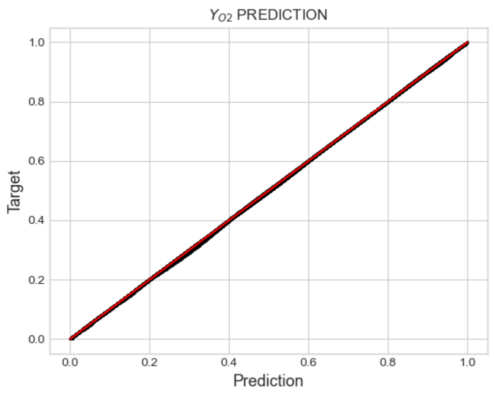


Figure 7-14: Prediction vs target output values for the temperature temporal evolution (top), for all four clusters of the state space. Partial enlargements (bottom left and bottom right) indicate that there exists an error of a low magnitude.

The prediction of temperature temporal evolution seems to be macroscopically excellent for the total state space spanned in the current implementation. However, partial enlargements in any range of temperature progress variable indicate that the prediction deviates from the target outputs, with the error being quite low. These observations are generalized for each thermochemical state feature in Figure 7-15, including the main species mass fractions.





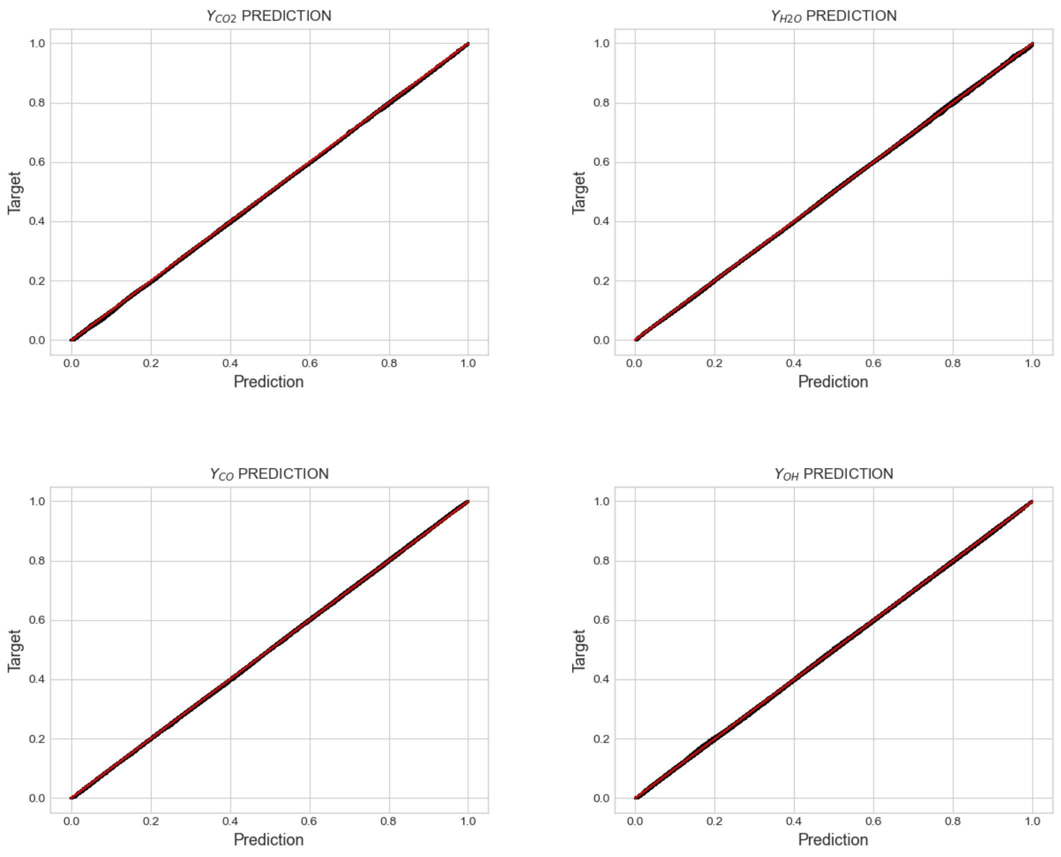


Figure 7-15: Prediction vs target output values for the main species mass fractions temporal evolution, for all four clusters of the state space.

Overall, the four developed MLPs exhibit macroscopically an excellent performance regarding the time integration of a state over a specified time step. Observations of previous sections regarding the low level of errors are also verified by the diagrams of Figures 7-14 and 7-15. Ideas associated with a further reduction of error are presented in Section 8.2.

Table 7-2 summarizes the average RMSE calculated for each one of the four developed MLPs. The accuracy can be considered sufficient, considering that the errors' order of magnitude is 10^{-3}, while the data are normalized into the [0,1] range. Higher errors appear the intermediate to high temperature zones, where the main proportion of chemical reaction occurs.

Table 7-2: Root mean squared error (RMSE) of each developed MLP.

MLP assigned to:	RMSE
Cluster 1	$1.9 \cdot 10^{-3}$
Cluster 2	$4.6 \cdot 10^{-3}$
Cluster 3	$5.4 \cdot 10^{-3}$
Cluster 4	$2.5 \cdot 10^{-3}$

CHAPTER 8: Conclusions

8.1 Summary and Discussion of Results

Chemistry acceleration via Machine Learning methods is examined in the present work using Artificial Neural Networks. The proposed methodology consists of two main objectives: to reduce the computational cost of chemistry integrations and to achieve a desired level of accuracy. The development of this methodology has been driven by the needs of the combustion kinetics tabulation problem, where different subdomains in state space can feature different dynamics. Therefore, first of all, a neural network responsible for clustering the state space is developed. In this context, Self-Organizing Maps are applied, which create two-dimensional grids representing the distribution of data features. Subsequently, Multi-Layer Perceptrons, which are a type of feedforward ANNs, are assigned, in order to regress the desired thermochemical feature.

The ANN performance is limited in learning and predicting accurately patterns visited in the thermochemical state vectors used during the training process. Therefore, the training dataset is generated through a canonical problem, in order to enhance the ANN capacity of generalization. Here, the dataset is obtained through 1-D steady premixed laminar flame simulations. It must be emphasized that the trained ANNs are not subject to the premixed laminar flame model, because several features such as spatial distribution and velocity are discarded and the dataset is shuffled prior to training. The temperature and species mass fractions are maintained as data features, whereas pressure is kept constant, creating state vectors. Data normalization plays an important role on the network behavior and performance; thus two associated cases are studied. Case 1 refers to temperature normalized into the [0,1] range while all other data features remain unchanged, whereas Case 2 corresponds to all data features scaled into the [0,1] range.

The Self-Organizing Map performs excellently in terms of clustering the state space, for both cases of data normalization; the generated maps can be used for correlation of features and quality analysis. These maps are further clustered by the K-Means algorithm, in order to reduce the number of subdomains, and a hybrid SOM – K-Means neural network is developed. Regarding Case 1 of data normalization, temperature and main species mass fractions are accounted for more than minor species, thus the developed SOM is specialized in these features of state space. Local reduction of kinetics mechanism is an application that this SOM is considered to perform well. Regarding Case 2, all data features are accounted for equally, and hence the developed SOM is not specialized to them and can be applied for mathematical calculations by assigning regression neural networks, an application examined in the present work.

The current approach is based on clustering and training ANNs specialized to particular subdomains of the state space. The present developed hybrid SOM – K-Means neural network clusters the state space into four subdomains. Therefore, a separate Multi-Layer Perceptron is assigned in each state space cluster. Two case studies of MLP outputs are examined: the first one is the heat release rate and the species net production rates prediction, whereas the second one is the temperature and the species mass fractions temporal evolution over a specified time step. The same network architecture is used for both case studies, defined via a cross validation approach for the first case, and further evaluated for the second one, where temporal evolution is calculated instead of direct integration of the ODE system. The errors are low, and considered acceptable for the purposes of the present study.

Regarding computational time, the neural network methodology reduces it by approximately 21 times. Specifically, direct integrations of the homogeneous reactor for 127099 composition states containing for a time step $\delta t=10^{-6}$ s are executed in 38613 [s], or approximately 10.7 [hours]. On the other hand, the computational time for the neural network mechanism to execute the same simulations is 1817.41 [s], or about half an hour, out of which 0.41 [s] are required for the MLP regression and 1817 [s] are spent for the determination of which state cluster contains the input vector at each case. The measurements of the execution time were conducted on a personal computer running on an i5 processor.

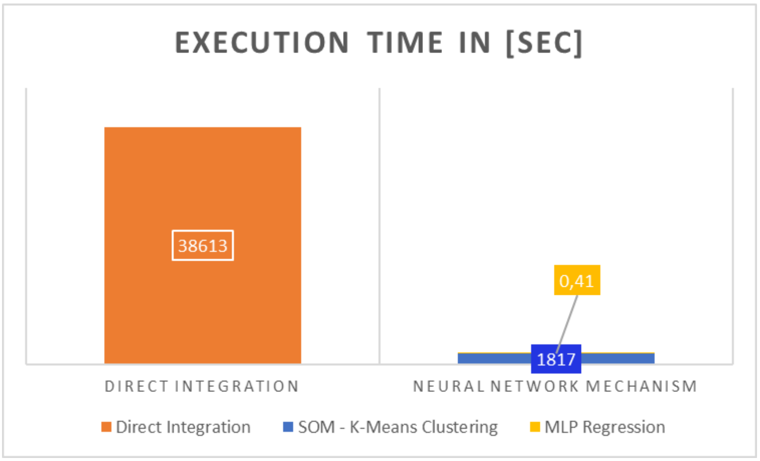


Figure 8-1: Execution time comparison between detailed numerical simulations and the developed neural network mechanism.

Therefore, the developed neural network mechanism, consisting of the hybrid SOM – K-Means and the MLP neural networks, offers a huge reduction in computational cost of chemistry integration, while achieving a desired level of predictions' accuracy.

8.2 Future Work

The Artificial Neural Networks developed in the present thesis can find several applications, such as local characterization of state space and mechanism reduction, or reduction of time for solving the ODE systems describing the chemistry term in CFD simulations. The results are very encouraging for a first engagement with this field of research, paving the way for future work. Some directions of interest for future studies are the following:

- Application of the developed hybrid SOM – K-Means neural network for local reduction of a methane combustion kinetics mechanism.
- Further optimization of MLP architecture and tests on bow type configurations.
- Application of other methods for data augmentation in order to span a larger part of the state space.
- Examination of separate MLPs development in each state space subdomain, each one responsible for specific data features.
- Examination of separate MLPs development in each state space subdomain, each one responsible for a different order of magnitude of the same feature.
- Coupling the developed neural network mechanism with a CFD code and evaluation of performance in terms of accuracy and computational cost.
- Application of the present methodology for chemistry tabulation of a more complex fuel, such as HFO, or of a fuel expected to be extensively used in the future, such as ammonia or hydrogen.

The examination of developing separate MLPs in each state space subdomain, each one responsible for specific data features, or for a different order of magnitude of the same feature, is expected to further decrease the errors identified in the present study, albeit at an increase in computational time, due to the increased number of ANNs used. Thus, a proper compromise between accuracy and computational efficiency should be also investigated.

Literature

[1] L. Kaiktsis, X. Vouvakos, D. Kazangas and P. Kontoulis, «Modeling Approaches for Internal Combustion Engine Applications». Presentation for the Course «Combustion», 2020, Division of Marine Engineering, School of Naval Architecture and Marine Engineering, National Technical University of Athens.

[2] SEA-LNG, "LNG as a marine fuel-the investment opportunity", https://sea-lng.org, accessed on August 2021.

[3] DNV-GL, https://www.dnv.com, accessed on August 2021.

[4] A. Cuoci, «Chemistry Acceleration in Computational Fluid Dynamics of combustion and heterogeneous catalysis», IMPROOF Workshop, Ghent University, January 2020.

[5] Z. Ren, Y. Liu, T. Lu, L. Lu, O. Oluwole and G. Goldin, «The use of dynamic adaptive chemistry and tabulation in reactive flow simulations», Combustion and Flame 161 (2014), pp. 127-137.

[6] J. An, G. He, F. Qin, X. Wei and B. Liu, «Dynamic adaptive chemistry with mechanisms tabulation and in situ adaptive tabulation (ISAT) for computationally efficient modeling of turbulent combustion», Combustion and Flame 206 (2019), pp. 467-475.

[7] S. Pope, «Computationally efficient implementation of combustion chemistry using in situ adaptive tabulation», Combustion Theory and Modelling 1 (1997), pp. 41-63.

[8] J. Blasco, C. Dopazo, N. Fueyo and J-Y Chen, «A self-organizing map approach to chemistry representation in combustion applications», Combustion Theory and Modelling 4 (2000), pp. 61-76.

[9] J. An, G. He, K. Luo, F. Qin and B. Liu, «Artificial neural network based chemical mechanism for computationally efficient modelling of hydrogen / carbon monoxide / kerosene combustion», International Journal of Hydrogen Energy 45 (2020), pp. 29594-29605.

[10] L. Franke, A. Chatzopoulos and S. 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 (2017), pp. 245-260.

[11] A. Chatzopoulos and S. Rigopoulos, «A chemistry tabulation via Rate-Controlled Constrained Equilibrium (RCCE) and Artificial Neural Networks (ANNs), with application to turbulent non-premixed $CH_4/H_2/N_2$ flames», Proceedings of the Combustion Institute 34 (2013), pp. 1465-1473.

[12] C. Chi, G. Janiga and D. Thevenin, «On-the-fly artificial neural network for chemical kinetics in direct numerical simulations of premixed combustion», Combustion and Flame 226 (2021), pp. 467-477.

[13] G. D' Alessio, A. Cuoci, G. Aversano, M. Bracconi, A. Stagni and A. Parente, «Impact of the Partitioning Method on Multidimensional Adaptive Chemistry Simulations», Energies 2020, 13(10), 2567, May 2020.

[14] G. D' Alessio, A. Parente, A. Stagni and A. Cuoci, «Adaptive chemistry via pre-partitioning of state space and mechanism reduction», Combustion and Flame 211 (2020), pp. 68-82.

[15] T. Zhang, Y. Zhang, W. E and Y. Ju, «A deep learning-based ODE solver for chemical kinetics», Princeton University, November 2020.

[16] A. Sharma, R. Johnson, A. Moses and D. Kessler, «Deep Learning for Scalable Chemical Kinetics», Laboratories for Computational Physics & Fluid Dynamics, U.S. Naval Research Laboratory, Washington D.C., 2020.

[17] T. Readshaw, T. Ding, S. Rigopoulos and W.P. 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, 035154 (2021).

[18] T. Ding, T. Readshaw, S. Rigopoulos and W.P. 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 (2021), 111493.

[19] O. Owoyele and P. Pal, «ChemNODE: A Neural Ordinary Differential Equations Approach for Chemical Kinetics Solvers», February 2021.

[20] J. Frankenfield, «Artificial Intelligence (AI)», www.investopedia.com , March 2021.

[21] M. Hargrave, «Deep Learning», www.investopedia.com , May 2021.

[22] G. Papalambrou, Course Notes «Artificial and Computing Intelligence for Ship Design and Operation», April 2020.

[23] N. Afentoulis and X. Pournaras, «Artificial Neural Networks and Applications in Maritime Industry», Presentation at National Technical University of Athens, 2019.

[24] A. Lorke, F. Schneider, J. Heck and P. Nitter, «Cypenko's Theorem and the capability of a neural network as a function approximator», September 2019.

[25] L. Codispoti, C. Frouzakis and K. Boulouchos, «Machine learning models for the local displacement speed of hydrogen-air premixed flames», Swiss Federal Institute of Technology (ETH), Zürich, December 2020.

[26] M. Hagan, H. Demuth, M. Beale and O. De Jesus, «Neural Network Design», E-book, 2nd Edition.

[27] J.E. Jackson, «A User's Guide to Principal Components», John Wiley & Sons, Inc., 2005.

[28] S. Wold, K. Esbensen and P. Gelaldi, «Principal Component Analysis», Chemometrics and Intelligent Laboratory Systems, 1987.

[29] Z. Jaadi, «A step-by-step explanation of Principal Component Analysis (PCA)», www.builtin.com, April 2021.

[30] A. Parente and J. Sutherland, «Principal component analysis of turbulent combustion data: Data pre-processing and manifold sensivity», Combustion and Flame 160 (2013), pp. 340-350.

[31] T. Kohonen, «The Self-Organizing Map», IEEE, 1992.

[32] T. Kohonen, «Essentials of the self-organizing map», Neural Networks, 2012.

[33] N. Zivkovic, «Implementing Self-Organizing Maps with Python and TensorFlow», Rubik's Code, July 2021.

[34] H. Garcia and I. Gonzalez, «Self-organizing map and clustering for wastewater treatment monitoring», Engineering Applications of Artificial Intelligence, March 2004.

[35] G. Breard, «Evaluating Self-Organizing Map Quality Measures as Convergence Criteria», University of Rhode Island, 2017.

[36] B. Brentan, G. Meirelles, E. Luvizotto and J. Izquierdo, «Hybrid SOM+K-Means clustering to improve planning, operation and management in water distribution systems», Environmental Modelling & Software, March 2018.

[37] K. Arvai, «K-Means Clustering in Python: A Practical Guide», www.realpython.com, July 2020.

[38] D. Xu and Y. Tian, «A Comprehensive Survey of Clustering Algorithms», Annals of Data Science, Springer, August 2015.

[39] G. Seif, «The 5 Clustering Algorithms Data Scientists Need to Know», www.towardsdatascience.com , February 2018.

[40] A. Jain, «Data clustering: 50 years beyond K-Means», Pattern Recognition Letters, September 2009.

[41] J. Vesanto and E. Alhoniemi, «Clustering of Self-Organizing Map», IEEE, May 2000.

[42] A. Ultsch, «U-Matrix: a tool to visualize clusters in high dimensional data», University of Marburg, Germany, January 2003.

[43] P. Rousseeuw, «Silhouettes: A graphical aid to the interpretation and validation of cluster analysis», Journal of Computational and Applied Mathematics, November 1986.

[44] A. Bhardwaj, «Silhouette Coefficient: Validating Clustering Techniques», www.towardsdatascience.com , May 2020.

[45] H. Rhys, «Machine Learning with R, the tidyverse and mlr», Manning Publications.

[46] S. Haykin, «Neural Networks: A Comprehensive Foundation», McMaster University, Canada, 2001.

[47] D. Rumelhart, G. Hinton and R. Williams, «Learning Internal Representations by Error Propagation», Institute for Cognitive Science, California, September 1985.

[48] I. Goodfellow, Y. Bengio and A. Courville, «Deep Learning», 2016.

[49] P. Grover, «5 Regression Loss Functions All Machine Learners Should Know», www.heartbeat.com , July 2018.

[50] T. Hastie, T. Tibshirani and J. Friedman, «The Elements of Statistical Learning: Data Mining, Inference and Prediction», Springer, 2009.

[51] D. Kingma and J. Ba, «Adam: A Method for Stochastic Optimization», University of Toronto, 2015.

[52] S. Ruder, «An overview of gradient descent optimization algorithms», www.ruder.com, January 2016.

[53] D. Kolaitis and M. Founti, «Combustion Theory», Fountas, 2014.

[54] I. Glassman and R. Yetter, «Combustion», 4th Edition, Academic Press, 2008.

[55] F. A. Williams, «Combustion Theory», 2nd Edition, 1985.

[56] GRI Mechanisms, Gregory P. Smith, David M. Golden, Michael Frenklach, Nigel W. Moriarty, Boris Eiteneer, Mikhail Goldenberg, C. Thomas Bowman, Ronald K. Hanson, Soonho Song, William C. Gardiner, Jr., Vitali V. Lissianski, and Zhiwei Qin, http://www.me.berkeley.edu/gri_mech.

[57] A. Kallieros, «Development of a skeletal mechanism for combustion of methane-hydrogen mixtures using entropy production analysis», Diploma Thesis, National Technical University of Athens, September 2020.

[58] CANTERA open-source suite, https://cantera.org/.

[59] V. M. van Essen, «Fundamental limits of NO formation in fuel-rich premixed methane-air flames», Proefschrift Rijksuniveriteit Groningen, 2007.

[60] MINISOM, https://github.com/JustGlowing/minisom.

[61] scikit-learn, https://scikit-learn.org/stable/index.html.