NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF CHEMICAL ENGINEERING
DEPARTMENT OF PROCESS ANALYSIS & PLANT DESIGN

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΧΗΜΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΑΝΑΛΥΣΗΣ ΣΧΕΔΙΑΣΜΟΥ & ΑΝΑΠΤΥΞΗΣ ΔΙΕΡΓΑΣΙΩΝ & ΣΥΣΤΗΜΑΤΩΝ

DOCTORAL THESIS

ENABLING EFFICIENT ANALYSIS OF CVD PROCESSES BY COMBINING CFD AND DATA-DRIVEN REDUCED-ORDER MODELING

by

PAVLOS GKINIS

SUPERVISOR

PROFESSOR ANDREAS G. BOUDOUVIS

ATHENS 2021

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ ΤΟΥ

ΠΑΥΛΟΥ ΓΚΙΝΗ

ΑΠΟΤΕΛΕΣΜΑΤΙΚΗ ΑΝΑΛΥΣΗ ΔΙΕΡΓΑΣΙΩΝ ΧΗΜΙΚΗΣ ΑΠΟΘΕΣΗΣ ΑΠΟ ΑΤΜΟ ΣΥΝΔΥΑΖΟΝΤΑΣ ΥΠΟΛΟΓΙΣΤΙΚΗ ΡΕΥΣΤΟΔΥΝΑΜΙΚΗ ΚΑΙ ΜΟΝΤΕΛΑ ΜΕΙΩΜΕΝΗΣ ΤΑΞΗΣ ΒΑΣΙΣΜΕΝΑ ΣΕ ΔΕΔΟΜΕΝΑ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ

ΑΝΔΡΕΑΣ Γ. ΜΠΟΥΝΤΟΥΒΗΣ

ΑΘΗΝΑ 2021

The opinions or assertions contained herein are the private opinions of the author and are not to be construed as official or reflecting the views of the School of Chemical Engineering of the National Technical University of Athens (Law 5343/1932, Article 202).

Η έγκριση της διδακτορικής διατριβής από τη Σχολή Χημικών Μηχανικών του Εθνικού Μετσοβίου Πολυτεχνείου δεν υποδηλώνει αποδοχή των γνωμών του συγγραφέα (Ν. 5343/1932, Άρθρο 202).

The thesis research was funded by the Research Committee of NTUA (ELKE-NTUA), through its annual scholarship program for 4 years, from 07/2017 to 07/2021.

Η εκπόνηση της διατριβής χρηματοδοτήθηκε από τον Ειδικό Λογαριασμό Κονδυλίων Έρευνας του Εθνικού Μετσόβιου Πολυτεχνείου (ΕΛΚΕ-ΕΜΠ) μέσω του ετήσιου προγράμματος υποτροφιών, για 4 χρόνια, από 07/2017 μέχρι 07/2021.

Acknowledgements

Throughout the work of this dissertation, I have received a lot of support and assistance, that were especially helpful during the last couple years of the pandemic.

I would first like to thank my supervisor, Professor Andreas Boudouvis, whose expertise and experience in computational fluid dynamics was invaluable in setting the pillars of my research and formulating the questions and methodology. His continuous support and feedback throughout the years of my research was beneficial to the progress of my work.

I would also like to acknowledge the members of my advisory committee, Professor Constantinos Theodoropoulos and Professor Martyn Pemble, for their comments and reviews.

I would like to thank the Fulbright Foundation for granting me the Greek Fulbright Grant that gave me the great opportunity to do research in the USA at Johns Hopkins University. Additionally, I would like to thank my US affiliation supervisor Professor Yannis Kevrekidis for our collaboration and for creating the opportunity of my stay at JHU.

I would like to especially thank my tutor Dr. Eleni Koronaki for her daily support and guidance during every single stage of my dissertation, from the initial steps of the literature review till the results of my research. Her knowledge and experience in machine learning algorithms was the main influence for the reduced order modeling part of my work. I want to thank her for being the best, most cooperative and helpful colleague I could have.

I also wish to thank my colleagues at NTUA Dr. Ioannis Aviziotis and Dr. Georgios Gakis for our collaborations and for the friendly office experience.

Finally, I would like to thank all my Greek and international friends who supported me all these years and for always being there when I needed them.

Abstract

In this dissertation, a purely data-driven, equation-free computational framework is proposed for dealing efficiently and effectively with Computational Fluid Dynamics (CFD) large-scale and complex models. The case studies utilized for the framework application are Chemical Vapor Deposition (CVD) reactors, since their CFD models present multiple challenges, such as complex geometries, variety of competing transport phenomena, multiple chemical reaction pathways and nonlinearity. The CFD models are discretized and solved with the commercial code ANSYS Fluent using the Finite Volume Method (FVM). The computational study of such problems is costly because of the studied CFD model size, which is a result of high-fidelity computational grids used and the physicochemical phenomena that occur in a CVD reactor. Therefore, the main purpose of this thesis is to develop and optimize a data-driven computational framework, which is used for the creation of equation-free Reduced Order Models (ROM) of the data obtained from the high-dimensionality models of the CVD reactors. Finally, the computational processing of the ROM is significantly cheaper than the initial studied CVD reactor model and ultimately produces the same high-fidelity results.

The core of the proposed computational framework is specifically the machine learning algorithms of Proper Orthogonal Decomposition (POD) and Artificial Neural Networks (ANN). Since, both algorithms are data-driven, the data collection takes place first, which is the solution of the CFD model several times. The data collection is executed on coarse computational grids with the chemistry removed from the CFD models, to reduce the cost of the procedure. The latter is an assumption that the chemistry does not affect the state of the reactor, because the reactive mixture of CVD processes is dilute. The data collection is a procedure of imposing step changes on a control variable of the process and capturing the trajectory of the dynamic response of the system throughout the change from the initial steady state to the target one. This trajectory is discretized in snapshots of the reactor state for every time increment. Following the data collection, the POD is used to determine the optimal low-dimensionality orthonormal basis that contains most of the spatial dependency information and shows the best accuracy at recreating parts of the data set. Since POD is not able to capture the time dependence, this is achieved by using the same data set and the previously constructed POD basis to train and simulate a Nonlinear Autoregressive Network with exogenous inputs (NARX) for different inputs of the control variable. Therefore, this results to a final ROM that can produce fast approximations of the system states for any given input with extremely low computational cost and great accuracy. Finally, considering that the ROM is built on coarse grid

data and without chemistry, its predictions are used as initial estimates after being interpolated using the Nearest Neighbor grid-to-grid interpolation to the fine grid model that includes the chemical reactions. The latter then converges in a small number of iterations and gives the desired high-fidelity solution.

Three different case studies are shown in this work, the first is the aluminum CVD from the precursor DMEAA, the second the copper CVD from copper amidinate and last a CVD reactor that presents solution multiplicity. In the first case study, the CFD model of the reactor is accompanied by a significant computational burden, due to the grid size and the complex reaction kinetics. Thus, by following the framework a ROM is created that, with negligible error on its predictions, accelerates the CFD simulations up to 2.5 times in core hours (core hours = wall clock-time x CPU cores used in parallel processing) and up to 312 times when chemistry is not considered. During the second case study of copper CVD, a novel chemical reaction network is proposed and fitted on existing experimental data, using a ROM to impressively speed up the procedure of fitting the kinetic parameters that carries a high computational burden, since a big part of it is trial and error. The reaction network consists of two temperature activated decomposition volumetric reactions and one surface deposition reaction, the first are proposed in order to explain and capture the decrease of deposition rate at high substrate temperatures, which appears in experimental findings. The two decomposition reactions are a carbodiimide deinsertion and a β-hydrogen abstraction of copper amidinate and their kinetic constants are fitted on the available experimental data. The usage of the ROM shows a 4 times acceleration in core hours with insignificant errors for its predictions, which is crucial for the fitting procedure since the model must be simulated several times. The final case study concerns a CVD reactor with solution multiplicity, which for the same operating conditions has three different states, two stable and observable experimentally, and one unstable that is not observable experimentally. This solution multiplicity increases the computational cost of the CFD study, especially close to the turning points between the stable/unstable state branches of the solution space, since the convergence of the solver “jumps” between the two different stable states. Thus, the framework was implemented and created an equation-free ROM based on data from both stable solution branches, that can capture the solution multiplicity in the studied solution space with less than 2% error, for any input on the controlled variable, thus confirming that the proposed framework can address complex nonlinear systems too.

Περίληψη

Στη παρούσα διδακτορική διατριβή, προτείνεται ένα υπολογιστικό πλαίσιο βασισμένο σε δεδομένα, χωρίς εξισώσεις, για την αποτελεσματική αντιμετώπιση σύνθετων μοντέλων Υπολογιστικής Ρευστοδυναμικής (Computational Fluid Dynamics, CFD) μεγάλης κλίμακας. Οι περιπτωσιολογικές μελέτες που χρησιμοποιούνται για την εφαρμογή πλαισίου αφορούν αντιδραστήρες Χημικής Απόθεσης από Ατμό (Chemical Vapor Deposition, CVD), καθώς τα CFD μοντέλα τους παρουσιάζουν πολλαπλές προκλήσεις, όπως πολύπλοκες γεωμετρίες, ποικιλία ανταγωνιστικών φαινομένων μεταφοράς, πολλαπλές οδούς χημικής αντίδρασης και μη γραμμικότητα. Τα μοντέλα CFD διακριτοποιούνται και επιλύονται με τον εμπορικό κώδικα ANSYS Fluent χρησιμοποιώντας τη Μέθοδο Πεπερασμένων Όγκων (Finite Volume Method, FVM). Η υπολογιστική μελέτη τέτοιων προβλημάτων είναι δαπανηρή λόγω του μεγέθους του μοντέλου CFD υπό μελέτη, το οποίο είναι αποτέλεσμα της χρήσης υπολογιστικών πλεγμάτων υψηλής ποιότητας και των φυσικοχημικών φαινομένων που εμφανίζονται σε έναν αντιδραστήρα CVD. Ως εκ τούτου, ο κύριος σκοπός αυτής της διατριβής είναι η ανάπτυξη και βελτιστοποίηση ενός υπολογιστικού πλαισίου, βασισμένο σε μεθόδους μηχανικής εκμάθησης και δεδομένα, χωρίς γνώση των εξισώσεων, το οποίο χρησιμοποιείται για τη δημιουργία Μοντέλων Μειωμένης Τάξης (Reduced Order Model, ROM) των μοντέλων υψηλής διάστασης των αντιδραστήρων CVD. Τέλος, η υπολογιστική επεξεργασία των ROM είναι σημαντικά φθηνότερη από το αρχικό λεπτομερές μοντέλο του αντιδραστήρα CVD και τελικά παράγει τα ίδια αποτελέσματα υψηλής πιστότητας.

Ο πυρήνας του προτεινόμενου υπολογιστικού πλαισίου είναι συγκεκριμένα οι αλγόριθμοι μηχανικής εκμάθησης της Ορθής Ορθογώνιας Αποσύζευξης (Proper Orthogonal Decomposition, POD) και των Τεχνητών Νευρωνικών Δικτύων (Artificial Neural Networks, ANN). Αρχικά, εφόσον και οι δύο αλγόριθμοι βασίζονται σε δεδομένα, πραγματοποιείται η συλλογή αυτών με την επίλυση του μοντέλου CFD αρκετές φορές. Η συλλογή δεδομένων εκτελείται σε CFD μοντέλα με αραιά υπολογιστικά πλέγματα χωρίς χημεία, για να μειωθεί το κόστος της διαδικασίας. Το τελευταίο είναι μια υπόθεση ότι η χημεία δεν επηρεάζει την κατάσταση του αντιδραστήρα, επειδή το μείγμα των αντιδρώντων στις διεργασίες CVD είναι αραιό. Η συλλογή δεδομένων είναι μια διαδικασία βηματικών επιβολών σε μια από τις μεταβλητές ελέγχου της διεργασίας και η καταγραφή της τροχιάς της δυναμικής απόκρισης του συστήματος κατά τη μετάβαση από την αρχική μόνιμη κατάσταση στην τελική. Αυτή η τροχιά διακριτοποιείται σε στιγμιότυπα της κατάστασης του αντιδραστήρα για κάθε χρονική στιγμή. Μετά τη συλλογή δεδομένων, η μέθοδος POD χρησιμοποιείται για τον προσδιορισμό της βέλτιστης

ορθοκανονικής βάσης χαμηλής διάστασης που περιέχει τις περισσότερη πληροφορία της χωρικής εξάρτησης και παρέχει την καλύτερη ακρίβεια στην αναδημιουργία τμημάτων του συνόλου δεδομένων. Δεδομένου ότι η POD δεν είναι σε θέση να καταγράψει την χρονική εξάρτηση των δεδομένων, αυτό επιτυγχάνεται με τη χρήση του ίδιου συνόλου δεδομένων και της προκύπτουσας βάσης POD για την εκπαίδευση και προσομοίωση ενός μη γραμμικού νευρωνικού δικτύου με ανατροφοδότηση και εξωγενείς εισόδους (Nonlinear Autoregressive Network with exogenous inputs, NARX) για διαφορετικές τιμές εισόδου της μεταβλητής ελέγχου. Επομένως, αυτό οδηγεί σε ένα τελικό Μοντέλο Μειωμένης Τάξης (Reduced Order Model, ROM) που μπορεί να παράγει γρήγορες προσεγγίσεις των καταστάσεων του συστήματος για οποιαδήποτε δεδομένη τιμής της μεταβλητής εισόδου με εξαιρετικά χαμηλό υπολογιστικό κόστος και μεγάλη ακρίβεια. Τέλος, λαμβάνοντας υπόψη ότι το ROM δημιουργείται με βάση δεδομένα που προέρχονται από υπολογισμούς σε αραιό πλέγμα και χωρίς χημεία, οι προβλέψεις του χρησιμοποιούνται ως αρχικές εκτιμήσεις μετά την παρεμβολή τους, με την παρεμβολή από-πλέγμα-σε-πλέγμα ή πλησιέστερου γείτονα, στο μοντέλο πυκνού πλέγματος που περιλαμβάνει τις χημικές αντιδράσεις. Το τελευταίο στη συνέχεια συγκλίνει σε μικρό αριθμό επαναλήψεων και δίνει την επιθυμητή λύση υψηλής ακρίβειας.

Στην παρούσα διατριβή παρουσιάζονται τρία διαφορετικά συστήματα, το πρώτο είναι η CVD αλουμίνιου από την πρόδρομη ένωση DMEAA, το δεύτερο η CVD χαλκού από το copper amidinate και το τελευταίο αποτελεί έναν αντιδραστήρα CVD που παρουσιάζει πολλαπλότητα λύσεων. Στη μελέτη του πρώτου συστήματος, το μοντέλο CFD του αντιδραστήρα συνοδεύεται από σημαντικό υπολογιστικό κόστος, λόγω του μεγέθους του πλέγματος και της πολύπλοκης κινητικής αντιδράσεων. Οπότε, ακολουθώντας το προτεινόμενο πλαίσιο δημιουργείται ένα ROM που, με αμελητέο σφάλμα στις προβλέψεις του, επιταχύνει τις προσομοιώσεις CFD κατά 2.5 φορές σε «ώρες πυρήνα» (ώρες πυρήνα = πραγματικός χρόνος ρολογιού **x** πυρήνες επεξεργαστή CPU που χρησιμοποιούνται στην παράλληλη επεξεργασία) και 312 φορές όταν δεν εξετάζεται η χημεία. Κατά τη διάρκεια της δεύτερης περιπτωσιολογικής μελέτης της CVD του χαλκού, προτείνεται και εφαρμόζεται ένα νέο δίκτυο χημικών αντιδράσεων πάνω σε υπάρχοντα πειραματικά δεδομένα. Με τη δημιουργία και χρήση ενός ROM επιταχύνεται η διαδικασία προσαρμογής των κινητικών παραμέτρων των αντιδράσεων, μια διαδικασία που φέρει υψηλό υπολογιστικό φορτίο, καθώς ένα μεγάλο μέρος της αποτελεί δοκιμή και σφάλμα. Το δίκτυο αντιδράσεων αποτελείται από δύο ογκομετρικές αντιδράσεις αποσύνθεσης που ενεργοποιούνται σε υψηλές θερμοκρασίες και μία αντίδραση επιφανειακής απόθεσης. Οι πρώτες προτείνονται με σκοπό την εξήγηση και αποτύπωση της μείωσης του ρυθμού απόθεσης σε υψηλές θερμοκρασίες υποστρώματος, η οποία εμφανίζεται στα πειραματικά ευρήματα. Οι

δύο αντιδράσεις αποσύνθεσης είναι μια αφαίρεση καρβοδιιμιδίου (carbodiimide deinsertion) και μια αφαίρεση β-υδρογόνου (β-hydrogen abstraction) του copper amidinate και οι κινητικές σταθερές αυτών προσαρμόζονται στα διαθέσιμα πειραματικά δεδομένα. Η χρήση του ROM αποδίδει 4 φορές επιτάχυνση σε πραγματικό χρόνο ρολογιού με ασήμαντα σφάλματα στις προβλέψεις, κάτι που είναι ζωτικής σημασίας για τη διαδικασία προσαρμογής, καθώς το μοντέλο πρέπει να προσομοιωθεί αρκετές φορές. Η τελευταία περιπτωσιολογική μελέτη αφορά έναν αντιδραστήρα CVD με πολλαπλότητα λύσεων, ο οποίος για τις ίδιες συνθήκες λειτουργίας έχει τρεις διαφορετικές καταστάσεις, δύο ευσταθείς και παρατηρήσιμες πειραματικά, και μία ασταθή που δεν είναι παρατηρήσιμη πειραματικά. Η πολλαπλότητα του χώρου λύσεων αυξάνει το υπολογιστικό κόστος της CFD μελέτης, ειδικά κοντά στα σημεία καμπής μεταξύ των ευσταθών/ασταθών κλάδων καταστάσεων του χώρου λύσεων, καθώς η σύγκλιση του επιλυτή «πηδά» μεταξύ των δύο διαφορετικών ευσταθών καταστάσεων. Επομένως, το υπολογιστικό πλαίσιο εφαρμόστηκε για τη δημιουργία ενός ROM με βάση τα δεδομένα από τους δύο ευσταθείς κλάδους λύσεων, άνευ γνώσης των εξισώσεων. Το ROM μπορεί να συλλάβει την πολλαπλότητα στο μελετημένο χώρο λύσεων με σφάλμα λιγότερο από 2%, για οποιαδήποτε τιμή εισόδου της ελεγχόμενης μεταβλητής, επιβεβαιώνοντας έτσι ότι το προτεινόμενο πλαίσιο μπορεί να αντιμετωπίσει και σύνθετα μη γραμμικά συστήματα.

Table of contents

1. Purpose of the thesis

The reliable modeling of Chemical Vapor Deposition (CVD) with detailed Computational Fluid Dynamics (CFD) problems can be extremely time-consuming. The main factor that makes the modeling of such systems so computationally demanding is the complex physical and chemical phenomena involved and the interactions between them [1] [2] [3]. These interactions can be computationally challenging to capture and, in some cases, they create solution multiplicity, that adds a "randomness" factor in the computational study and further increases the cost of it [4] [5] [6]. Additionally, the complex geometries of the CVD systems (reactors), paired with fine computational grids that are required for high fidelity results, significantly increase the computational burden [7] [8]. The latter, increases even more during the fitting studies of reaction kinetic parameters, where the CFD models must be solved repeatedly till the computational results match the experimental findings within a margin of error [9] [10]. Consequently, this thesis contains three different CVD cases, that present these challenges, which will be addressed one by one in the respective case studies. Thus, further validating the impact of this work by presenting a variety of problems that it can be utilized in.

In general, CFD combined with experimental studies of CVD are routinely used to investigate dominant flow patterns and chemical pathways that lead to the final product, without usually the concern of the required computational time. Nevertheless, in applications such as automation, optimization, control, or any general dynamic study of the system multiple simulation data are required in a short period of time. This creates the need for fast and accurate Reduced Order Models (ROM) of the process that can provide state predictions for these large-scale simulations. Therefore, the main goal of this work is to develop a general-purpose computational framework used for the construction of ROMs that capture the transport of mass, momentum and heat of high fidelity detailed CFD problems.

In this thesis, the ROMs are created based on the Proper Orthogonal Decomposition (POD) [11] [12] [13], more specifically by following the algorithm of a variation of POD called Method of Snapshots (MoS) [14] [15], and the use of Artificial Neural Networks (ANN) [16] [17] [18] [19]. The POD can produce a low-dimensional basis for the solution space of the studied problem based on a collection of computational or experimental observations. In this case the observations are the snapshots (states) of CVD reactors throughout the trajectory of their dynamic response to an imposed change on the operating conditions. The basis consists of vectors that contain most of the information of the data set and with the appropriate linear

combinations can produce any actual reactor state in the data set with a negligible error. These vectors contain only the spatial dependency of the system data, so the time dependency is determined using a separate machine learning algorithm. In this thesis, purely data-driven ANN are utilized for this task [20]. These are trained on the same data set as the one used in POD and ultimately their simulation can determine the time-dependent coefficients needed for the ROM. Therefore, an overall data-driven and equation-free ROM of the initial CFD reactor model is created, that can predict the dynamic and static behavior of the reactor, while its order and therefore the computational requirements for its simulation are significantly lower than the initial detailed model. [21]

A couple of other techniques that are used within the proposed computational framework for the efficient modeling of the CFD problem and the data collection, are parallel processing on multiple computational cores and coarse to fine computational grid-to-grid interpolation. Additionally, for an even more efficient data collection, the ROM may in principle be alien to the chemistry models that describe the sequence of gas and surface heterogeneous reactions that lead to the product, which in the case of CVD is a deposited solid film on a substrate surface. This is derived from the fact that in CVD reactors the inflowing gas mixture, which contains the reactants and the carrier gases, is so dilute that the influence of the depletion or production of chemical species on the development of the flow is negligible [8] [22] [23]. Therefore, the chemistry-free flow states calculated by the ROM are ultimately fed as an initial estimation into the chemistry enriched detailed model of the reactor, which results into a much faster convergence to the high-fidelity solution.

It is noteworthy that the core of the proposed framework of this thesis i.e., the POD and the ANN, can be applied to non-CVD and non-CFD problems as well, since the development of the ROM is completely data-driven and does not require the knowledge of the physicochemical phenomena and the respective governing equations of the system.

2. Chemical Vapor Deposition

2.1. Introduction

CVD is one of the most popular processes to produce thin solid films and coatings, that has developed rapidly in the recent decades and is now used in a variety of technological fields beyond laboratory limits. The process has a huge range of application in fields such as microelectronics, data storage and solar power [24] [25]. CVD can produce a huge variety of conductors, semiconductors and insulators depending on the chemical elements used and can produce high purity solid bulk materials and composite materials [26] [27].

The flexibility and adaptability of CVD, combined with its ability to coat almost any geometry and the high market price of its products led to the detailed study of its complex physical and chemical phenomena, since these determine the quality of the product [4] [28]. Even though the CVD is characterized as a complex process, due to the extensive study and analysis, the deposition of most of the chemical elements known to date has been achieved and previously impossible depositions of materials with desirable properties have been successful. Most common application are metals, while the materials produced can consist of carbon, silicon, carbides, nitrides, and oxides. Some examples where the process applies are the following [26] [29] [30]:

- In the field of microelectronics, CVD is applied on an industrial scale for the synthesis of epitaxial films and the production of films that behave as dielectric materials, such as conductors, oxidation blocks, diffusion blocks etc.

- Optical fibers used in telecommunications. These are made of silicon with an inner coating of silicon, germanium, or boron to achieve the desired refractive index.

- Coatings resistant to heat (Al2O3, Si3N4, SiC), natural corrosion (B4C, TiC, Cr7C3), electrochemical corrosion (Nb, Cr, Ta) and wear resistant coatings (TiC, TiN, Al2O3) used in many industrial applications.

- Carbon nanotubes used in detectors for applications in electronics, biological and chemical systems.

- Diamond-like carbon coatings (DLC), which are produced in plasma environment and show improved wear resistance.

- High temperature superconductors used in medicine and applications in physics that require high energy (e.g., YBCO and BSCCO).

- Conductive coatings for network connections, projectors, automated windows.

- Semiconductor lasers GaAs/(Ga,Al)As and InP/(In, Ga)As, which also find application in microwave and solar cell devices.

- Coatings for friction reduction and decoration (e.g., TiN which offers a gold exterior on watches).

In summary, the basic steps of the CVD process are as follows: First, the precursor compound is introduced into the reactor using a carrier gas. The mixture is transported towards the substrate, where the solid film is formed through heterogenous reactions. Eventually, the gaseous by-products as well as the unreacted compounds are brought to the outlet of the reactor. The design of the CVD reactor plays a very critical role in the final quality of the produced films. The main properties that determine the quality of the deposited film are its uniformity, purity, morphology and thickness. A typical CVD reactor consists of the following critical components [26]:

- Gaseous reactive mixture intake system

- Reaction chamber

- Substrate heating system

- Substrate material

- Substrate loading/unloading mechanism

- Gas exhaust and treatment system

- Vacuum system

- Process and quality (in situ) control system

A more detailed and informative description of the process and the mechanisms is developed in section 2.3.

2.2. Evaluation of CVD

The CVD carries some important characteristics, for which it is preferred in many cases instead of other deposition methods for the production of thin films. The most important of these are summarized below [26] [31]:

- Produces high uniformity effective coatings even on deep cavities, holes, recesses and generally complex geometries and three-dimensional (3D) surface configurations, in contrast to Physical Vapor Deposition processes (PVD).

- It produces high purity films, so impurities and undesirable by-products, which can degrade the quality of the film, are absent in the final product.

- It can achieve high deposition rates and production of thick layers (order of cm), which makes CVD competitive and, in some cases, more advantageous economically than other deposition processes.

- It exhibits selectivity in terms of the substrate surface. Therefore, the deposition takes place mainly on the substrate, i.e., the defined area within the reactor, while it is limited to other surfaces, such as the reactor chamber walls.

- It does not require high vacuum conditions, is flexible and easily adaptable to different equipment setups, operating conditions and reaction mechanisms (huge variety of precursors).

Inevitably, the process has in practice some limitations [26] [31]:

- The properties of the produced films often lag behind the specifications of demanding applications.

- High operating temperatures are necessary, due to possibly high activation energies of chemical reactions and to achieve high deposition rates. However, this not only leads to an increase in operating costs, but also to vital problems in cases where the

substrate is heat sensitive. This may lead to thermally unstable substrates, reducing the options of substrate materials. In addition, the produced films may be subjected to thermal shocks by interacting with substrate materials of different thermal expansion coefficients, which results to mechanical instability of the product.

- It requires precursors with a high vapor pressure, to achieve a feasible and efficient transportation and entry into the reactor. Compounds with this property appear in metalorganic CVD (MOCVD), resulting in the development and utilization of the corresponding technique.

- A large percentage of the precursor compounds are often corrosive, explosive and highly toxic, thus limiting the applications of the process. Also, the cost of some more “easy-to-use” precursors is quite high.

- The reaction by-products can be toxic and corrosive and need to be neutralized, which is a costly process.

- Due to the large number of different chemical and physical phenomena that take place in the process, their complexity and the relationships between them, the mathematical study of CVD becomes quite demanding. Although after an extensive study of the process, most of its mechanisms have been clarified, there is still major problem, which is the great demands in computational time and memory for the simulation of 3D realistic mathematical models of the process. This often plays a limiting role in the effective and efficient prediction of the product properties and in the automation of the process.

2.3. The CVD process

The operating conditions of the process are strictly controlled, with the main ones being the mass inlet flow, the mixture composition, the pressure and the temperature [4] [32]. The latter is regulated both on the surface of the substrate and on the reactor walls. The temperature range in the reactor chamber is an important characteristic that leads to various versions of CVD and significantly affects the quality of the produced films [9] [33]. The films produced under certain conditions must be reproducible and have controlled properties. The most critical properties determining the quality of a film are thickness, uniformity, surface morphology and purity. The range of the acceptable limits of the above specifications varies

according to the type of CVD process, the materials used and the target application of the film. [24] [34]

The reason the temperature of the reactor walls and the continuous heating of the substrate must be set without any room for variation throughout the process, has to do with the very principle of CVD. The whole process is based on the entry of a precursor compound in the reactor, which is the reactant of the process and contains the deposited element in its molecule. Then, after a series of complex chemical and physical phenomena, the molecules of the precursor are broken down to the atoms of the requested element on the substrate, where they are agglomerated layer-per-layer, resulting in a single film. [35] [36]

The type of CVD studied in this thesis is only that of MOCVD, which is a quite complex process, since it includes transport phenomena of mass, momentum, heat, and mechanisms of chemical kinetics. As a result, all physical and chemical mechanisms are inextricably linked. Specifically, the stages and phenomena that take place during CVD are presented in Figure 1 and are as follows [26] [37] [38] [39]:

- The mixture of reactants, i.e., the carrier gas and the precursor enter the reactor. Carrier gas is an inert substance that constitutes the greater proportion of the input flow and serves the efficient transport of the precursor in, through and out of the reactor by means of convection.

- The precursor compound diffuses through the boundary layer from the main flow field to the surface of the substrate. The potential that causes this mass transport is the difference in concentration between the two ends of the boundary layer. It is noted that within the boundary layer –as in the main gas flow field– homogeneous reactions may take place leading to unwanted by-products.

- This is followed by the adsorption of the precursor compound molecules on the substrate surface. This adsorption is mainly chemical, but also partly physical.

- A series of heterogeneous reactions occur on the surface of the substrate, which lead to the decomposition of the precursor compound and the release of deposited element atoms on the substrate. The necessary activation energy of the reactions comes from the heat released by the heated substrate. This is the main reason why the heating of the substrate is an integral part of CVD.

- The adsorbed molecules before, during and after the reactions can diffuse along the length of the substrate surface. This not only helps the reactants to come into contact, improving the performance of surface reactions, but even more importantly makes it possible to agglomerate the atoms of the deposited element and the subsequent creation of the film.

- At this point, the nucleation of the atoms of the element to be deposited is activated, followed by the growth of these nuclei resulting in the gradual formation of the film. The atoms tend to occupy positions so that the formation of the film is done layer-by-layer, thus ensuring that the products formed are relatively uniform. If it is assumed that between the substrate and the film a diffusion zone is formed, then in this zone it is possible to form intermediate products of various phases.

- Simultaneously with the heterogeneous reactions, the decomposition of molecules and the mass transport phenomena that take place on the substrate, the desorption of the by-products is carried out. At the same time intermediate products that can be found in the diffusion zone described in the previous stage, are also desorbed. It is noted that very volatile intermediates may be desorbed and then reabsorbed on the surface of the substrate.

- Then, the by-products and intermediates that have not been reabsorbed are diffused through the boundary layer to the main flow field.

- Finally, the carrier gas, the remaining precursor and the by-products of all reactions exit the reactor.

At the same time as all the above phenomena, volumetric reactions occur in the gas phase within the reactor chamber. These reactions are activated due to the heating of the substrate, so they mainly occur near it. However, in some variations of CVD or in specific types of reactors their activation can also occur by heating the walls of the reactor chamber or by completely different forms of energy, such as ultraviolet radiation, active electrons in plasma or laser photons.

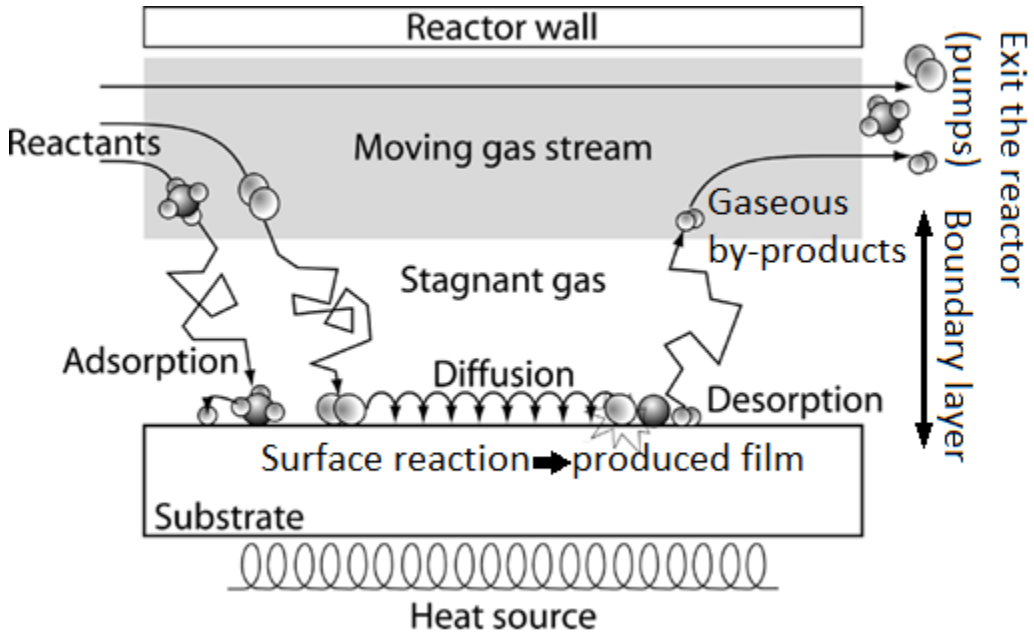

Figure 1: Schematic representation of the CVD stages and phenomena.

2.4. Temperature dependence

The study of the kinetics of the CVD and the effect of operating conditions is a valuable tool not only for the theoretical understanding of the process, but also for the prediction and control of the product properties, as well as for the effective design and efficient operation of the process. One of the most critical parameters affecting the kinetics of CVD is temperature, many studies concern the dependence of the deposition rate and uniformity of the films on the temperature of the substrate [4] [9] [33] [40]. These studies are usually illustrated in an Arrhenius plot, in which the logarithm of the deposition rate is plotted against the inverse of substrate temperature [28]. The study of reaction kinetics of CVD is usually carried out based on the rate-limiting step theory, according to which the rate of a complex reaction is determined by its slowest part. Regarding the steps of CVD analyzed in the previous section, usually, either the chemical reactions or the transport phenomena control the overall process. In an Arrhenius plot, three regions are distinguished in relation to the controlling mechanisms in the temperature range. The Figure 2 depicts a typical Arrhenius diagram of CVD, divided qualitatively into three areas, which are analyzed below. [41]

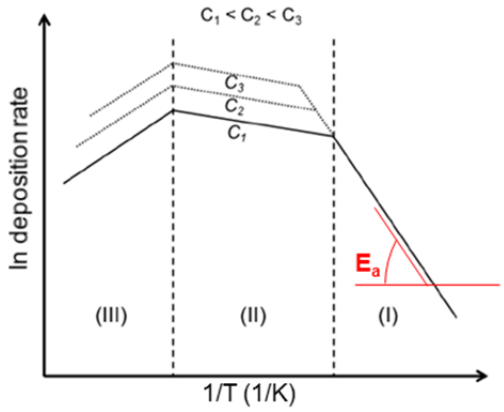

Figure 2: Typical CVD Arrhenius plot of the deposition rate natural algorithm against the reverse substrate temperature. (C1, C2, C3 precursor concentrations)

At low temperatures **(region I)** the deposition rate is controlled by the surface reactions kinetics. Due to the low substrate temperature, these reactions occur at a very slow rate, resulting in an excess amount of precursor on the surface, as it arrives through the boundary layer (by diffusion) faster than can be consumed. Therefore, an increase in the inlet flow rate of the precursor compound does not affect the deposition rate. On the other hand, the deposition rate is very sensitive to temperature changes, since the relationship linking the known kinetic constant k (reaction rate factor) follows the equation of Arrhenius below, i.e., it is exponential, which is evident in Figure 2, where the logarithm of the deposition rate depends linearly on the inverse of the temperature.

$$k = k_{ref} e^{\frac{E_a}{RT}}$$

2-1

where, k_{ref} the constant preexponential factor, E_a the activation energy, R the ideal gas constant and T the temperature. The slope of the line in area I corresponds to the total activation energy of the surface reactions. Consequently, an increase in temperature within region I leads to a significant increase in the rate of deposition. This region is characterized by low deposition rates but very high uniformity in terms of film thickness. [4] [34] [42]

Further temperature increase leads to **region II**, where the deposition rate is controlled by the mass transport phenomena and more specifically by the rate of diffusion of the precursor compound from the main mass of the gaseous flux to the surface of the substrate and by the surface reactions by-products diffusion rate in the opposite direction. Both these diffusions

occur along the boundary layer. These two diffusion rates are determined by the corresponding diffusion coefficients. Although the diffusion **coefficient** D generally increases with temperature, it does not exhibit the same sensitivity as the kinetic constant k, so a given increase in temperature results in a much greater increase in the rates of surface reactions compared to the corresponding increase in diffusion rates. Thus, in region II, there is a constant need for precursor on the substrate surface, which through diffusion is difficult to cover due to its high rate of consumption. Therefore, an increase in the input flow of the precursor compound results in a significant increase in the total deposition rate, as it increases the concentration between the main flow and the substrate, i.e., the potential of diffusion. This is also presented in Figure 2 in the form of different curves, that each corresponds to a different concentration of precursor compound in the input mixture (considering a constant total mass flow). On the other hand, as is also evident from the Arrhenius plot, an increase in temperature within this region brings about a relatively small increase in the overall rate of deposition, since, as mentioned above, the sensitivity of the diffusion coefficient to temperature changes is relatively limited. Unlike region I, this area is characterized by high deposition rates but not as uniform films as in region I. [4] [41]

For even higher temperatures, the **region III**, there is an "unexpected" decrease in the overall deposition rate. This decrease is due to the occurrence of competitive phenomena against the surface reactions, that reduce their rate. These phenomena are mostly due to the activation of volumetric reactions in the gas phase, due to the overall heating that takes place within the reactor. Therefore, it is assumed that in this area the volumetric reactions are the limiting step, although their effect on the deposition rate is opposite to the other two phenomena (an increase in the volumetric reaction rate causes a decrease in the overall rate of deposition). An important consequence of the activation of volumetric reactions, which causes this decrease in deposition rate, is the reduction of the available precursor compound for surface reactions, since a large part of it is consumed before it can reach the surface of the substrate. In addition, there is a possibility that some of the by-products produced by these reactions settle on the produced film, hindering its growth, thereby reducing the rate of deposition. Finally, such a high temperature can enhance the desorption of useful intermediate compounds (and reactants) of surface reactions from the substrate, delaying the reaction, thus reducing the overall deposition rate. Therefore, an increase in temperature causes a significant decrease in the rate of deposition, while an increase in the input flow of the precursor compound produces same results as in region II. [28] [35]

In conclusion, it is important to note the following, firstly, in most CVD cases the boundaries of each region are not as clear as they appear in Figure 2, there are also intermediate regions, which can often be designed separately. In these regions, the process is controlled by more than one phenomenon (e.g., intermediate of I-II, where both diffusion and surface reactions are the controlling steps). Secondly, the form of the Arrhenius plots is not always that of Figure 2, but also significantly depends on other operating parameters, such as pressure. Additionally, based on the analysis above, it is concluded that in CVD processes, rate and uniformity have an inverse relationship. Therefore, the process adjustment under given conditions should be done (also) based on which of the two is more important for the current product. Finally, it is noted that in some cases where the concentration of the precursor compound is sufficiently low, the limiting step is its supply. In these cases, an increase in the input flow of the precursor compound, either by increasing the concentration, or by increasing the total flow (or both), it increases the overall deposition rate.

2.5. Chemical reactions and precursor compounds

As a precursor, any compound containing in its molecule atoms of the element concerned, in this thesis a metal (MOCVD), can be selected [43]. Although pure atoms of the element are rarely used, usually the simplest cases of a precursor are those where the deposited element is connected to small organic or inorganic active chains with simple bonds, forming simple and small molecules. Of course, many times much more complex than the above precursor compounds are used, in order to exploit their properties that make them superior to their simpler counterparts. [24] [31]

The choice of the precursor compound is usually one of the primary questions to be answered when designing a CVD process. The main factors considered are the following [4] [26] [44] [45] [46] [47]:

- **Decomposition temperature**: A very important factor that determines the substrate temperature and the energy strategy of the process in general. It is sought not to be too high, which is associated with very high energy requirements, with all the negatives that this entails, but also not too low, to avoid decomposition in the gas phase.

- **Volatility**: It is expressed through vapor pressure and affects the vaporization and subsequent transport of the precursor compound in, through and outside the reactor. High values are preferred even at low temperatures.

- **High purity**: The higher the purity of the precursor compound, the higher the purity of the final film.

- **Decomposition by-products**: The by-products produced during its decomposition, either in the gaseous or in the solid phase, must be sufficiently volatile to be easily removed from the reactor and not to settle, resulting in contamination of the film and/or reactor.

- **Waste footprint**: The mixture of by-products produced from the CVD reactions and exit the reactor should meet the specifications set by the legislation on the composition of exhaust gas/waste.

- **Safety**: Avoid risk factors, such as toxicity, explosiveness, or corrosiveness.

- **Cost**: Low cost of purchase, maintenance, and cleaning.

For the chemical reactions involved in the CVD processes, a huge variety of reactions have been observed, according to the precursor compound of the application concerned. However, most of these reactions belong to one of the following three classes. [9] [10] [48] [49]

- **Pyrolysis reactions:** One of the simplest reactions observed in CVD, they are characterized by the fact that they occur mainly on a surface and not volumetrically, thus avoiding the decomposition of the precursor in the gas phase. Of course, the operating conditions and the carrier gas play an important role in the above. Regarding the conditions, a key role is played by pressure, where low pressures are preferred.

- **Reduction reactions**: They are often observed in precursor compounds containing halogen in their molecules. The reducing agent may be one of the following: the carrier gas itself (e.g., hydrogen), a separately added substance (e.g., silane), part of the precursor itself (e.g., CH3SiCl3), metal vapors (e.g., Mg), the substrate itself (e.g., silicon substrate).

- **Oxidation/hydrolysis reactions**: They occur in cases where whole oxides rather than individual elements are deposited. They are activated by the addition of oxygen or water, respectively, which remove hydrogens or halogens from the molecules of the precursor compound. In addition, ozone (O3) is sometimes added to accelerate the

reaction, and sometimes direct oxidation of the substrate is attempted instead of depositing oxides on it, a strategy that is not common as it is very time consuming.

3. Computational modeling of CVD

3.1. Purpose and benefits of CFD in CVD

A comprehensive, analytical, and successful study of a CVD process, as shown in [4] [10] [33] to name a few, requires experimental and theoretical/computational analysis. These two aspects of the overall study, although at first glance they may seem independent, are in fact closely related to each other. In fact, a strong relationship connects them, since any mathematical model developed through computational simulations of the process requires the verification of its accuracy by a plethora of experimental results. As well as, the computational study often guides the experimental system, the type of experiments and the conditions under which they will be conducted. Therefore, the CFD study does not replace the experiment, or the understanding offered by analytical methods, but ideally acts complementary. Undoubtedly, though, there are some unique advantages of CFD compared to the experimental approach for system design, that make CFD a growing and powerful technique for solving complex physical problems of CVD [50]:

- Ability to study CVD systems in which controlled experiments are difficult and sometimes impossible to be conducted (e.g., large-scale/industrial setups). [5] [7]

- Significant reduction in the preparation time and the cost of new designs. [51]

- Ability to explore flow in complex geometries and for complex models (complex physicochemical phenomena), with practically unlimited level of detail of result. [4] [5] [9]

In particular, the computational study of CVD can provide significant benefits, both at a laboratory and industrial level. Regarding new, under-development processes, where a preliminary study is carried out, the computational studies are necessary to uncover the dominant mechanisms prevailing within the reactor, phenomena that are quite complex, as explained in the previous chapters. It is usually important to study the volumetric and surface chemical reactions that take place, [9] [10] which can also determine the temperature range of the reactor operating area through the design of an Arrhenius plot of the process [41] [52],

according to what was mentioned in the section 2.4. Additionally, already existing kinetic models are often optimized, by changing the kinetic constants of their reactions, or through the addition/removal of reactions from the existing model, as shown in studies [9] [10]. Without the mathematical modeling of processes and their subsequent computational simulation, it is practically impossible to investigate these mechanisms. Of course, it is emphasized once again that whichever model is developed, without its experimental verification it lacks reliability. [4] [28] [35]

Regarding the CVD processes on the industrial level, a computational study can again offer critical benefits that would otherwise be impossible to use. For example, having already developed a verified and reliable model of the process with reliable accuracy, it is possible to determine in advance the "operating windows", that within their boundaries will lead to the production of films with the desired properties, thus receiving high quality products based on their intended application. In addition, a computational study makes it possible to optimize the process within these windows, resulting in cost and time minimization due to increased deposition rate and reduction of the required amount of raw materials and energy, etc. The operating parameters determined by such procedures include the supply and composition of the input mixture, substrate and wall temperatures. A successful study can save significant amounts of time and materials/energy, which would have been consumed in experimental studies to discover the above. Finally, the computational study through reliable mathematical models makes possible the automation and control of processes, an application very important for the efficient, effective and above all uninterrupted operation. Automation not only allows to eliminate external disturbances extremely faster than manual adjustment, but also makes easier and more direct the change from one operating point to another. [7] [10] [33]

An ideal CVD model should consist of a set of mathematical equations describing all the relevant macroscopic and microscopic physical and chemical phenomena evolving in the reactor, linking them to the corresponding properties of the deposited film. In addition, the model should be applicable to different types of CVD processes and reactors. Therefore, it should be based on fundamental principles and physicochemical laws and not in empirical constants and correlations. Despite all the progress made in this direction, to date no model has been produced that incorporates all the above characteristics with 100% success. However, by addressing the problem of modelling the CVD process from different angles, the existing CVD models have led to a remarkable improvement in the understanding of the mechanisms and the

emergence of the dominant factors that affect the operation. Thus, leading to designing more suitable configurations and geometries of CVD reactors. [4] [9] [51]

By observing the typical stages-phenomena of CVD (section 2.3), it is found that it is a process that evolves on more than one scale. Therefore, the computational/theoretical studies are also developed at different scales, depending on the phenomena of the process, the properties of the films and generally the information that the researcher aims to study. Typical scales that concern researchers and are often modeled are the macro-scale, the micro-scale and the nanoscale, where the first includes the phenomena occurring in the main volume of the reactor, on the substrate and film surfaces (m to cm), the second deals with the topography of the formed films (μm), while the third describes the nano-morphology of the films (nm). [28] [34] [53]

In view of the above, it becomes apparent that the full study and in-depth understanding of a CVD process requires not only the development of models for each scale, but also the coupling of the different models with each other, to allow the exchange of information from one to the other. However, this is often not observed in practice, as depending on the frameworks set in each research, different information is required for the successful study and the extraction of useful conclusions. Thus, there is no reason for a "meaningless" increase in computational and research cost, so the analysis is limited to only one of the scales, which is usually the macro-scale [4] [8] [54]. Nevertheless, even a general macroscopic CFD study of CVD can be significantly computationally demanding due to the complex and competing physical and chemical phenomena that occur inside a CVD reactor. Also, the study can be extremely time-consuming and sometimes even prohibitive, especially when the nonlinear behavior of CVD is combined with the necessary ultimate “dialogue” between CFD and experimental results, for the validation of the CFD model. [8]

3.2. Macroscopic modeling of CVD

A typical macroscopic model of the CVD process consists of the partial differential equations (PDEs) of the transport phenomena including chemical reactions. These are the equations for the conservation of mass, momentum, energy and chemical components as well as the kinetic ones of chemical reactions. The approximation of these equations is performed numerically using commercial (and non-commercial) software, which have built-in CFD codes and utilize methods of discretization of PDEs in space, but also in time (transient calculations), thus transforming the system of PDEs into a system of algebraic equations, which is solved

numerically through iterative methods. Finally, to solve the PDEs, it is necessary to setup the geometry of the reactor, as well as the boundary conditions of each problem, which mostly are the operating conditions of the actual reactor. Also, in case of calculations in a transient state, initial conditions of the model are necessary, while it is obvious that the model requires knowledge of the physicochemical properties of all the components involved in the process. From the solution of the macroscopic model the most important results that arise are the fields of velocity, pressure, temperatures and component concentration distributions and the deposition rate. To determine the properties of the film directly related to its quality (e.g., roughness or purity) usually requires the resolution of models of smaller scales, as the macroscopic model is unable to predict them. [7] [50] [55] [56] [57]

The application of the macroscopic model assumes that the fluid within the reactor is a continuous medium, which is determined by the dimensionless Knudsen number and is defined as follows:

$$Kn = \frac{\lambda}{L}$$

3-1

where λ the free path length of the particles and L the characteristic length of the problem. The value of the Knudsen number for the reactor must be much less than the unit (Kn<<1) and this is true for the applications in this dissertation. Therefore, the view of the continuous medium is reasonable and is taken for granted. In addition, the flow inside, through and outside the reactor is considered laminar, because the value of Reynolds number in the applications shown in this dissertation is much less than 2000. [28]

3.2.1. Governing conservation and transport equations

The general conservation equation that can be formulated for any variable ϕ, is as follows:

$$\frac{\partial(\rho\varphi)}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}\varphi) = \nabla \cdot \Gamma_{\varphi} \nabla\varphi + S_{\varphi}$$

3-2

where t the time, ρ the density of the fluid, **u** the velocity of the fluid, C_ϕ the diffusion coefficient and S_ϕ the source term. In the equation the transitional term, the convection term, the diffusion term and the source term are shown in order from left to right. As regards the variable ϕ this can be the velocity **u** for the momentum equation, the temperature T for the energy equation, the respective mass fraction ω for the equations of the chemical components and equal to one for the continuity equation (mass). [28] [58] [59]

A. Continuity

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0$$

3-3

B. Momentum

$$\frac{\partial (\rho \boldsymbol{u})}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}) = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \rho \boldsymbol{g}$$

3-4

where the operating pressure p, the acceleration of gravity $\boldsymbol{g}$ and $\boldsymbol{\tau}$ the stress tensor which is defined as follows:

$$\tau = \mu \left[(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) - \frac{2}{3} (\nabla \cdot \boldsymbol{u}) \boldsymbol{I} \right]$$

3-5

where μ the viscosity and *I* the unitary matrix.

C. Energy

$$C_p \frac{\partial (\rho T)}{\partial t} + C_p \nabla \cdot (\rho \boldsymbol{u} T) = \nabla \cdot (\lambda \nabla T) - \sum_{i=1}^{N} \left(\boldsymbol{j_i} \cdot \frac{\nabla H_i}{M_i} \right) - \sum_{i=1}^{N_g} H_i r_i^g$$

3-6

where T the absolute temperature, C_p the specific heat capacity at constant pressure, λ the thermal conductivity, i the component, $\boldsymbol{j_i}$ the diffusion rate, H_i the enthalpy of formation, M_i

the molecular weight, N the number of gaseous components taking part in the volumetric reactions and r_i^g the net rate of production or consumption of the component, which is calculated below in the equation 3-18. [58]

D. Ideal gas

$$p_i = c_i RT$$

3-7

where p_i the pressure, c_i the concentration and R the constant of the ideal gases. This constitutive equation is solved together with the fundamental conservation equations, assuming that the process fluid is an ideal gas. It is noted that the data of the equations concerning physicochemical properties of components are obtained from data libraries present in the commercial ANSYS Fluent package used for the simulations. [58]

E. Chemical components

$$\frac{\partial(\rho\omega_i)}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}\omega_i) = -\nabla \cdot \boldsymbol{j_i} + M_i \sum_{k=1}^{N_g} r_i^g$$

3-8

where ω_i the mass fraction. Since the sum of the mass fractions is equal to a unit, the equation is solved for all but one of the mixture components, which is usually chosen to be the carrier gas in a CVD process. [10] [28] [58]

In CVD, the diffusion rate $\boldsymbol{j_i}$ is calculated from the full multicomponent diffusion model. In multicomponent mixtures it is not possible to produce relationships for each diffusion rate that include the concentration gradient of only one component, as required by Fick's law. Therefore, for the calculation of the rate of diffusion, the Stephan-Maxwell equations are used, which for ideal gases take the following form: [60]

$$\sum_{\substack{j=1 \\ j\neq i}}^{N} \frac{f_i f_j}{D_{ji}} \left(\frac{\boldsymbol{j_j}}{\rho_j} - \frac{\boldsymbol{j_i}}{\rho_i} \right) = \nabla f_i - \frac{\nabla \mathrm{T}}{T} \sum_{\substack{j=1 \\ j\neq i}}^{N} \frac{f_i f_j}{D_{ji}} \left(\frac{D_{T,j}}{\rho_j} - \frac{D_{T,i}}{\rho_i} \right)$$

3-9

where f_i, f_j the molecular fractions of the components i and j respectively, D_{ji} the binary diffusion coefficient of component i in component j and $D_{T,i}$ and $D_{T,j}$ the coefficients of thermal diffusion of the components i and j. The term $\boldsymbol{j_i}$ is then calculated from the equation:

$$\boldsymbol{j_i} = -\sum_{j=1}^{N-1} \rho D_{ij} \nabla \omega_i - D_{T,i} \frac{\nabla \mathrm{T}}{T}$$

3-10

The above equation is an expression of Fick's law considering the Soret effect and has validity only when the composition of the mixture remains constant or when the coefficient D_{ij} is independent of it. The calculation of the latter is made through the equation Chapman-Enskog:

$$D_{ij} = 0.00188 \frac{[T^3 (\frac{1}{M_i} + \frac{1}{M_j})]^{0.5}}{P_{abs} \sigma_{ij}^2 \Omega_D}$$

3-11

where P_{abs} the absolute pressure, M_i and M_j the molecular weights of components i and j, σ_{ij} the cross-section of the collisions and Ω_D the collision integral, which expresses the measure of the interaction between the molecules of the system and is a function of the quantity:

$$T_D^* = \frac{T}{(\varepsilon/k_B)_{ij}}$$

3-12

where k_B the Boltzmann constant equal to $1.3807 \cdot 10^{-23} \frac{\mathrm{m^2 kg}}{\mathrm{s^2 K}}$ and the energy term $(\varepsilon/k_B)_{ij}$ calculated from the geometric mean as follows:

$$({}^{\varepsilon}/_{k_B})_{ij} = \sqrt{(\varepsilon/k_B)_i * (\varepsilon/k_B)_j}$$

3-13

The cross-section of collisions σ_{ij} is calculated from the arithmetic mean of the atomic σ as follows:

$$\sigma_{ji} = \frac{1}{2}(\sigma_i + \sigma_j)$$

3-14

The coefficients of thermal diffusion D_T are calculated through the following empirical equation, which quantifies the Soret effect:

$$D_{T,i} = -2{,}59 * 10^{-7} T^{0{,}659} \left[\frac{M_i^{0.511} f_i}{\sum_{i=1}^{N} M_i^{0.511} f_i} - \omega_i\right] \left[\frac{\sum_{i=1}^{N} M_i^{0.511} f_i}{\sum_{i=1}^{N} M_i^{0.489} f_i}\right]$$

3-15

F. Boundary conditions

The above conservation equations are solved, like all differential equations, in combination with appropriate boundary conditions, which refer in the majority to the main unknown variables of the problem, i.e., the velocity, pressure, temperature and the concentrations of the components. The boundary conditions in CVD problems belong to the following three types [4] [28]:

- **Dirichlet**: Defined values of the unknown variables.
- **Neumann**: Defined values of the derivatives of unknown variables.
- **Robin**: Linear combination of the values of unknown variables and their derivatives.

Moreover, if the above equations are solved in a transient state, initial conditions are also necessary, i.e., the distributions in the space of unknown variables for time t=0. The initial conditions are given by solving the problem in a steady state, by omitting the terms $\frac{\partial(\varphi)}{\partial t}$ in the equations above.

It is emphasized that the above PDEs can be applied either as a 3D mathematical model, or as a two-dimensional (2D). Consequently, the space dependency of the equations will involve three or two variables, respectively, as will the discretization of the reactor geometry. Obviously, 3D models have considerably greater accuracy, often to the point that while some complex phenomena may not be observed during the simulation of the process through a 2D model, the same phenomena are captured and even in sufficient detail during the simulation of the corresponding 3D one. However, the computational cost of 3D models is a lot higher than 2D models, to the extent that while the latter ones complete a full simulation cycle within a few minutes to a maximum of a few hours on an ordinary computer, 3D models require the use of parallel processing and programming technology with simulation done in Supercomputers or Computer Arrays. Therefore, if the 2D model is sufficiently precise, then its use is preferred over the corresponding 3D one. The above is often observed in the case where the geometry of the reactor shows axial symmetry. [4] [5] [6] [8]

3.2.2. Chemical reaction kinetics

In addition to transport phenomena, the mechanisms that take place within a CVD reactor include chemical reactions, which occur both in the volume of the gas phase and on the surface of the substrate. In case of laminar flow, one of the models that describes with sufficient accuracy the reactions of CVD and for this reason it is often used is the laminar finite-rate model, which is also applied in the present work. [2] [28] [58]

According to this model, chemical reactions, regardless of whether they are volumetric or surface, are described by the following general chemical equation:

$$\sum_{i=1}^{N_g} \gamma'_{ik}\, G_i \underset{r'_k}{\overset{r_k}{\rightleftharpoons}} \sum_{i=1}^{N_g} \gamma''_{ik}\, G_i + \sum_{i=1}^{N_s} \gamma''_{ik}\, S_i$$

3-16

where G_i and S_i the component i in the gas phase and solid phase (substrate surface) respectively, N_g and N_s the number of gaseous and solid chemical components respectively, γ'_{ik} the stoichiometric coefficient of the reactant i in reaction k, γ''_{ik} the stoichiometric coefficient of product i in the reaction k and r_k, r'_k the overall rates of the two opposite directions of the reaction k, whether it is volumetric or surface. The equation 3-16 applies for reversible as well as irreversible reactions, where the rhythm r'_k equals 0.

G. Volumetric reactions

The rate for an elementary reaction in the gas phase is calculated from the following Arrhenius expression:

$$r_k^g = k_{0,k} T^b \exp\left(-\frac{E_{a,k}}{RT}\right) f\left(C_1, \dots, C_{N_g}\right) \quad \left(\frac{mass}{volume * time}\right)$$

3-17

where $k_{0,k}$ is the pre-exponential coefficient of the reaction, b the exponent of reaction temperature, $E_{a,k}$ the reaction activation energy, R the global constant of gases, C_i the molecular concentration of reactants and f the function expressing the dependence of the reaction rate to the concentrations of the components.

Therefore, the net molecular rate of production or consumption of the chemical component i in the gas phase is given by:

$$r_i^g = \sum_{k=1}^{K_g} (\gamma''_{ik} - \gamma'_{ik}) r_k^g, i = 1, \dots, N_g$$

3-18

where K_g the number of volumetric reactions in the gas phase.

H. Surface reactions

The rate for an elementary reaction on the surface of the substrate is calculated from the following Arrhenius type expression:

$$r_k^s = k_{0,k} T^b \exp\left(-\frac{E_{a,k}}{RT}\right) f(C_1, \dots, C_{N_s}) \quad \left(\frac{mass}{surface * time}\right)$$

3-19

where all variables correspond to the same as 3-17, except that in this case they refer to surface rather than volumetric reactions. Therefore, the net molecular rate of production or consumption of chemical component i on the surface of the substrate is given by the following:

$$r_i^s = \sum_{k=1}^{K_s} (\gamma_{ik}'' - \gamma_{ik}') r_k^s, i = 1, \dots, N_s$$

3-20

where K_g the number of volumetric reactions in the gas phase.

For the calculation chemical components concentrations on the substrate, it is assumed that during a surface reaction, the mass flow of each chemical component from the gas phase to the surface of the substrate is equal to its rate of production or consumption due to the chemical reactions it is involved in, i.e.:

$$\rho_s D_i \boldsymbol{n} \cdot \nabla \omega_{i,s} = M_i r_i^s$$

3-21

where ρ_s is the density on the substrate, D_i the diffusion coefficient of the chemical component i, $\omega_{i,s}$ the mass fraction of the chemical component i in the substrate, $\boldsymbol{n}$ the vertical unit vector at the level of the substrate.

3.3. Spatial discretization – Solution method

As mentioned, the PDEs along with the "secondary" equations, all of which in combination with the boundary and initial conditions compose the macroscopic model of CVD, are solved numerically with the use of commercial (and non-commercial) software that have built-in CFD codes. This type of software utilizes methods of discretization of PDEs in space and time, thus transforming the system of the PDEs into a system of algebraic or algebraic-differential equations, which is solved numerically through iterative methods. The above general technique leads to the approximate solution of the equations. However, using the appropriate methods for the problem, small errors of the variable values can be achieved, thus giving a realistic picture of the reactor and sufficiently reliable results.

3.3.1. Spatial discretization

Based on the created geometry of the reactor using some technical design software (2D or 3D), the computational field is discretized in a grid of elementary cells. Each cell represents a finite volume, within which the values of each variable that are calculated from the equations

presented in section 3.2 are considered constant. That is, within each cell, it is not considered to be a change in the values of each variable in any direction of space. Each cell is delimited by its faces, which are determined when the grid is created. The set of PDEs is solved separately for each individual cell and the resulting values are then inserted into the rest of the space, resulting in the conversion of the discrete solution into a solution for the entire computational field. [50] [60]

The design of the grid is a very crucial step of the computational process, as it greatly affects both the accuracy of the simulations, the solving process itself and the computational requirements. The two most important aspects of a grid are its density and its quality. As regards to the quality of a grid, the two criteria most frequently considered are the aspect ratio, defined as the ratio of the largest to the smallest side of each cell, and the distortion, expressed by the ratio of the largest to the smallest angle of each cell. Cells with indicative values of the above indicators are shown in Figure 3. The effect of the density of a grid on the computational process, on the other hand, is more significant than the quality. The denser the grid, the more the error of the calculated solution decreases, but at the same time the demands on computational time and memory increase greatly and vice versa. In practice, the standard is that, initially, some test calculations are made with a coarse grid. Then the grid is constantly refined and gradually by relatively small percentages. This process ceases when it is found that further refining of the grid no longer improves the accuracy of the solution, i.e., the solution is independent of the grid. The above procedure is called grid independence verification.

Grids are classified into the following two categories based on the shapes of their cells:

Structured grids: They consist of quadrilateral (2-D-grids) or hexahedrons (3-D-grids), which form a grid of a uniform pattern.

Unstructured grids: They consist of triangles (2-D-grids) or pyramids (3-D-grids), which do not follow any pattern, but on the contrary are unevenly distributed in space.

Structured grids in most cases have an advantage over unstructured grids, since they not only have reduced computational requirements, but also allow better control of the density and arrangement of cells. However, unstructured grids are preferred in cases of complex geometry, since they offer higher flexibility in both the arrangement and density of the grid, allowing local refining. Two indicative spatial discretizations - one from each category - are presented in Figure 3c.

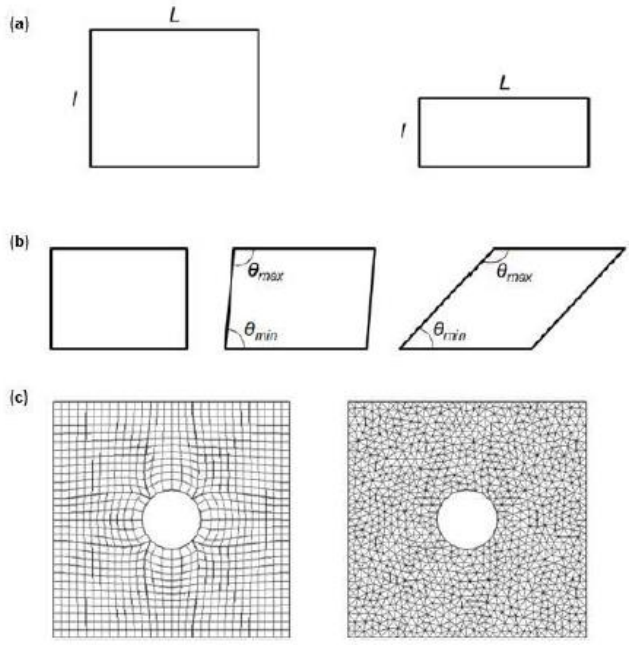

Figure 3: (a) Rectangular 2D cells with low (left) and high (right) aspect ratio, (b) Parallelogram 2D cells with increasing distortion of internal angles from left to right, (c) Structured (left) and Unstructured (right) spatial discretization.

3.3.2. Finite Volume Method

After the discretization of the computational field, the solution of the equations of the macroscopic model (3.2) follows. Usually, the method applied is one of the three that are briefly mentioned below [58] [61]:

Finite difference (FDM): It is based on the replacement of partial differentials by Taylor series expansions at each discretization point. The more terms are omitted from the Taylor series, leads to less accuracy and smaller computationally cost. It is easily applied to simple geometries due to the not particularly complex mathematics that accompany it, but its application makes it significantly difficult in cases of complex geometries.

Finite element (FEM): In this method, the unknown variables are approached by base functions which have are low-grade polynomials. In combination with the Galerkin weighted residual method, the PDEs are converted into a system of algebraic equations. The method is accompanied by relatively complex mathematics, making it more difficult to apply than the rest. However, this same disadvantage gives it enough flexibility in terms of solution parameters,

allowing to achieve very high accuracy. It also often presents quite high computational requirements, although this depends to a large extent on the problem solved.

Finite volume (FVM): The PDEs are integrated in detail for each cell separately. Their analytical integration is possible considering that all variables remain unchanged within each control volume (cell). Thus, the PDEs are transformed into algebraic equations, which are called finite volume equations, and their system can be solved by any known numerical (or even analytical) method of solving systems. Due to interpolation of the values of each variable from the center of the cells to the middle of its faces (sides), the calculations in a cell are affected the from their adjacent cells. In other words, local balances are formulated for each variable. The above logic is largely consistent with models consisting mainly of conservation equations, such as the macroscopic model of a CVD process. For this reason, FVM is often used in fluid dynamics problems. In the present dissertation, the macroscopic models were solved with the help of the commercial CFD software ANSYS Fluent 17, which is a commercial code using the finite volume method. Consequently, the FVM will be analyzed more extensively later. [4] [50] [60] [61]

Finite Volume Method

As already mentioned, according to the FVM each PDE is integrated in each elementary control volume separately. For the sake of brevity, the method will be presented only for the general conservation equation 3-2, the integration of which is obtained as [50]:

$$\int_V \frac{\partial(\rho\varphi)}{\partial t} dV + \int_V \nabla \cdot (\rho \boldsymbol{u}\varphi) dV = \int_V \nabla \cdot \Gamma_\varphi \nabla\varphi \mathrm{dV} + \int_V S_\varphi dV$$

3-22

which according to the divergence theorem becomes:

$$\int_V \frac{\partial(\rho\varphi)}{\partial t} dV + \int_A \boldsymbol{n} \cdot (\rho \boldsymbol{u}\varphi) dA = \int_A \boldsymbol{n} \cdot \Gamma_\varphi \nabla\varphi \mathrm{dA} + \int_V S_\varphi dV$$

3-23

where A the boundaries of the control volume V (cell faces) and $\boldsymbol{n}$ the unit vector perpendicular to A. The above equation is approached for a discrete cell as follows:

$$\frac{\partial(\rho\varphi)}{\partial t}V + \sum_{i}^{N_f} \rho_{f_i}\varphi_{f_i}\boldsymbol{u_{f_i}}\boldsymbol{A_{f_i}} = \sum_{i}^{N_f} \Gamma_\varphi \nabla\varphi_{f_i}\boldsymbol{A_{f_i}} + S_\varphi V$$

3-24

where the index i refers to the faces of the cell with N_f number of faces, while f_i is the middle of each face i and therefore all variables with the index f_i are calculated at the corresponding points. The above are also shown in Figure 4a, where C_0 the center of the cell, $\mathbf{r_0}$ the position vector from the center of the cell to the middle of a side and $\mathbf{A_f}$ the perpendicular to the side vector. Regarding the first term of 3-24, this refers to a time change and for the sake of simplification is omitted in this subsection. However, further reference and analysis of it will be made in the next subsection3.3.3 3.3.3.

The unknown variable, φ_{C_0} i.e., the size ϕ calculated in the center of the cell, must be inserted into the equation 3-24. This is done by developing interpolation equations of all φ_{f_i} in φ_{C_0}, which are replaced in 3-24 resulting in the creation of a (generic) nonlinear equation. The interpolations can be first order, second order, exponential law, or other even more complex ones. Furthermore, it is important to stress that the calculations of a cell do not only affect the calculations of its neighbors through the values of the variables on their common faces, but also through the value in their center. More specifically, one of the interpolation equations expresses the relationship that connects the values of ϕ in the centers of two adjacent cells Figure 4b. This relationship can be in the form of a simple average or even up to a much more complex form.

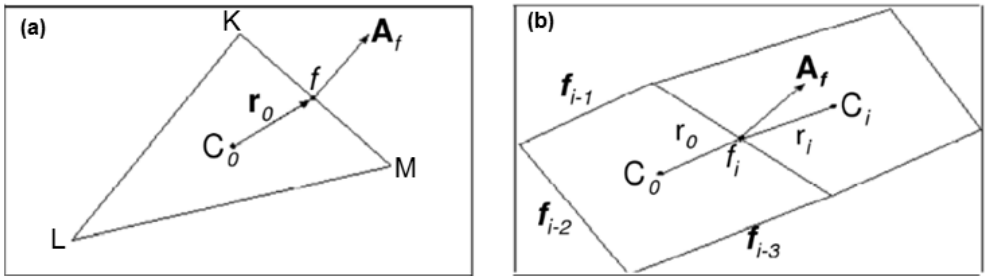

Figure 4: Typical cells of FVM application. (a) 2D Triangular cell (b) Two adjacent quadrilateral 2D cells.

Finally, by replacing the interpolation equations in 3-24, a nonlinear equation of this general form is obtained:

$$a_{C_0}\varphi_{C_0} = \sum_{nb} a_{nb}\varphi_{nb} + b$$

3-25

where the nb index refers to the adjacent cells, a_{C_0} and a_{nb} the linearization coefficients of $\varphi_{C_0}$and φ_{nb}, respectively. In each cell corresponds such an equation. Therefore, the total of the equations of each cell of the grid forms the system of algebraic equations, which are called equations of finite volumes. This system is solved with ANSYS Fluent using numerical iterative methods. The exact method to be used depends on the solver to be selected by the user and Fluent offers different options [58] [60]. Also, the method of linearization applied depends on the selected solver for the computational process.

The algorithm just described is by design iterative. Therefore, its convergence to an acceptable accuracy solution is based on reducing specific error indicators below some predefined limits. These error indicators are called residuals, they are symbolized as R_φ and are defined by the equation below:

$$R_\varphi = \frac{\sum_{C_0}\left|\sum_{nb}(a_{nb}\varphi_{nb}) + b - a_{C_0}\varphi_{C_0}\right|}{\sum_{C_0}\left|a_{C_0}\varphi_{C_0}\right|}$$

3-26

The acceptable limits are usually equal to 10^{-3}, but the user can adjust them, with a corresponding effect on solution accuracy and computational cost.

3.3.3. Time discretization and transient solution

In case the simulation of the process is carried out in a transient state, a few more steps are added to the above computational process. More specifically, apart from spatial, time discretization is also required since time is another independent variable the solution depends on. After discretizing the space and calculating equations of finite volumes, the discretization of time follows, which involves the integration of each term of these equations by a time step Dt

[58]. In more detail, a general equation that expresses the change of size ϕ through time is as follows [42] [58] [62]:

$$\frac{\partial \varphi}{\partial t} = F(\varphi)$$

3-27

where F is the function that includes any spatial discretization of ϕ (e.g., the resulting function if in equation 3-25 all terms move on one side). Discretization can follow several techniques. Fluent enables the user to choose between first-order discretization**,** which is expressed by the equation:

$$\frac{\varphi^{n+1} - \varphi^{n}}{\Delta t} = F(\varphi)$$

3-28

or second-order discretization**,** which is:

$$\frac{3\varphi^{n+1} - 4\varphi^{n} + \varphi^{n-1}}{2\Delta t} = F(\varphi)$$

3-29

where the indices n+1, n and n-1 refer to the next (t+Δt), the current (t) and the previous (t-Δt) time step.

As far as the time step is concerned, Fluent has two options for the user:

Indirect time integration: In this method F(ϕ) is calculated in the next time step, so for first-order discretization the final equation, which results after a few operations in equation 3-28, takes the form:

$$\varphi^{n+1} = \varphi^{n} + \Delta t F(\varphi^{n+1})$$

3-30

The above equation is solved repeatedly, as in the case of steady state, for each time step, before the simulation moves to the next. The method is called "indirect" because φ^{n+1} in a certain cell is associated with the φ^{n+1} of its adjacent cells through the term $F(\varphi^{n+1})$.

Direct time integration: Here F(ϕ) is calculated in the current time step, so for first-order discretization the final equation, which results after a few operations in equation 3-28, takes the form:

$$\varphi^{n+1} = \varphi^{n} + \Delta t F(\varphi^{n})$$

3-31

The above equation is solved repeatedly as in the case of a steady state for each step before the simulation moves to the next. The method is called "direct" because φ^{n+1} is defined by terms referring exclusively to the current solution (φ^{n} and $F(\varphi^{n})$).

In most cases, the first method is applied, as it shows remarkable stability in terms of time changes. On the contrary, the direct method requires certain conditions to present satisfactory time stability and is therefore used only for very specific applications. Finally, it is noted that the size of the time step is selected by the user and may even change throughout the transient calculation. Also, there is an option of choosing the number of iterations within each time step and the amount time steps as such, thus controlling how close to convergence each time step is.

4. Secondary techniques

4.1. Parallel processing and programming

4.1.1. Introduction

In modern times, solving large-scale computational problems in a reasonable time requires computing power and memory that not even the best serial computer can afford. It has now been established that such problems are solved with parallel processing techniques in computer systems known as Supercomputers, High Performance Computers or simply Parallel Computers. The most important category of basic parallel computing architecture is the Multi-Instruction – Multi-Data Architecture (MIMD), in which each processing unit operates

independently of the others, performing -in generality- different commands on different datasets, while they are all synchronized through communicating with each other. These in turn are divided into two further main subcategories, common memory computers and distributed memory computers. Their difference lies in the way the processors access memory, where in the first subcategory all processors have access to a common memory resource through a communication channel, while in the second each processor has access only to its own memory and communication between processors is done over a network. In the first case the programming of such computers is quite simple and similar to serial computer programming. However, their computing power is limited by the number of processors, since for technical reasons it is not possible to have a big number of processors accessing simultaneously in the same memory. On the contrary, in the second case, the number of processors can theoretically reach even infinity, thus outperforming in terms of the computational power they possess, but their programming is much more complex.

4.1.2. Computer arrays

The most widespread and well-established parallel processing approach is computer clusters, mainly due to the low cost that comes with them. Their principle of operation is based on the efficient interconnection of the separate computers that make up the array, which is achieved through fast local networks such as Gigabit Ethernet and Myrinet. These arrays typically use Linux as an operating system and more specifically special versions of it that have been developed exclusively for this use. The communication of the different units that make up the array requires, in addition to appropriate material (hardware), such as the abovementioned networks, and the existence of appropriate software, such as the general family of Message Passing Interface (MPI) Libraries. These are essentially a model/protocol for creating libraries that includes a core of routines, different applications of which render different libraries. Also, MPI libraries allow the programming of the communication of different computational units in programming languages such as C, C++ or Fortran.

4.1.3. Parallel solver of ANSYS Fluent

The CFD code of the ANSYS Fluent software except a serial solver, also contains a parallel one, thus enabling the user to simulate macroscopic models of fluid dynamics with parallel processing. The parallel processing technique applied by the parallel solver is the partition of the computational grid, where it is practically divided into sub-grids. The

computations in the sub-grids are shared within the available processors. The two solvers of Fluent are presented briefly in Figure 5. [60]

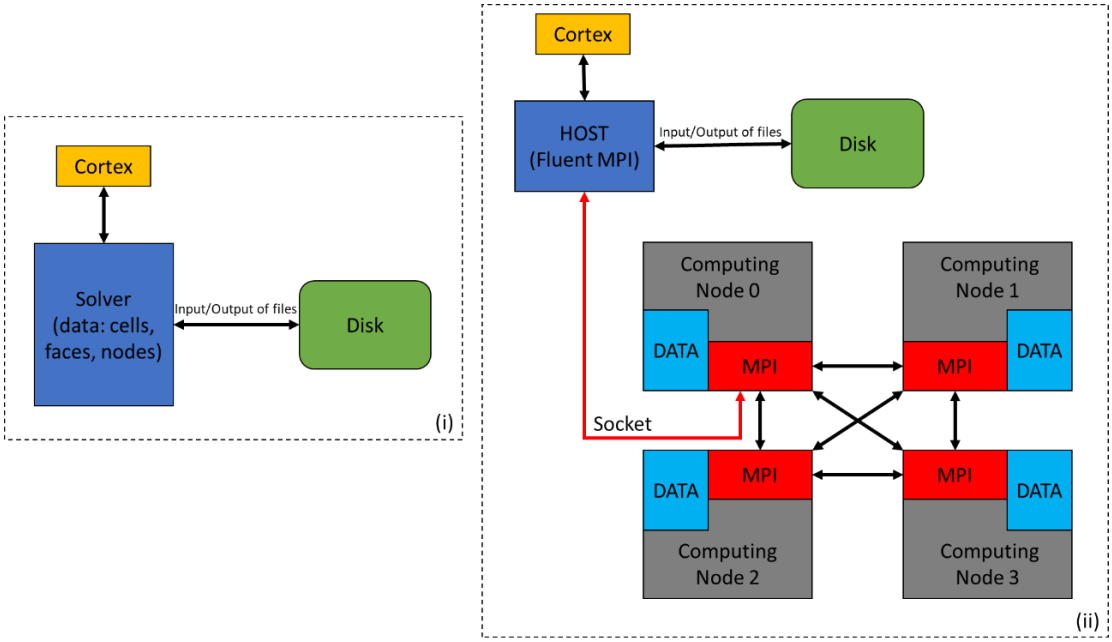

Figure 5: ANSYS Fluent solvers (i) Serial, (ii) Parallel.

The parallel solver consists of three main interactive processes: CORTEX, HOST and Computing Nodes. CORTEX performs the user's communication with the rest of the Fluent code and therefore transfers any command and data entered by the user into the HOST process, the main process of the solver. HOST on the other hand, which is itself a node, has the role of the "brain" of the solver and is no part of the grid, so it doesn't calculate anything related to Fluent. Instead, it undertakes the task of communicating and synchronizing the remaining processes through node-0. The node-0, in turn, is both a communication and calculation hub, since in addition to solving the CFD model in the sub-grid assigned to it, it also undertakes the partitioning of the commands and data received from HOST to the other computing nodes. These nodes then execute the common command list defined by the user, everyone in the respective part of the grid they are assigned to, independently of the others. Of course, computing nodes can communicate with each other in order to exchange data, synchronize or perform total functions (e.g., a sum of a variable from all cells in the grid). Finally, node-0 collects from all other computing nodes the individual parts of data of the CFD solution and transmits them to HOST, which makes them available to the user as a serial solver would. It is noted that, all these information transfers are made through the MPI library.

Regarding the computational grid it can be partitioned in many ways, depending on the partition method, optimization methods, and other parameters that affect the process. In any case, the resulting number of partitions must be an integer multiple of the number of computing nodes selected by the user to solve the problem (e.g., for 4 computing nodes the segments must be 4 or 8 or 12 and so on). Therefore, it is easily understood that each computing node can manage more than 1 part of the computational grid. In Figure 6, a 2-D grid before and after its partition into two sections is shown.

Fluent enables the user to choose between automatic or manual partitioning. In the latter case, the user is asked to select the partition parameters, the most important of which are: **Number of partitions, Partition method** and **Optimization methods.** Regarding the partition methods, these are based either on successive bisections of the grid until the desired number of partitions is reached. The difference between the methods lies in the bisection technique followed. The choice of the most effective method is an issue that depends to the studied problem, so the safest solution is comparison with trial and error.

Regarding optimization methods, these are split to the following two techniques that can be chosen independently of each other:

- **Smoothing**: Computational cells are transferred between different partitions to reduce the interface between adjacent parts of the grid.

- **Merging**: Small groups of cells that act as separate grid partitions are merged with other corresponding groups of cells, again to reduce the interface between adjacent parts.

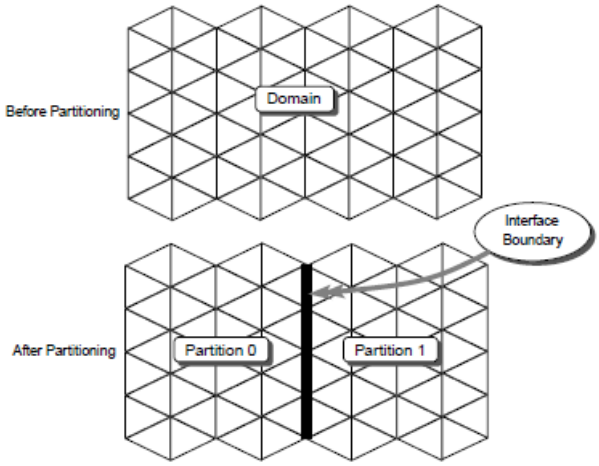

Figure 6: Partition of a 2D grid to two parts.

In Figure 7 a partitioned grid is presented: (a) before the distribution of the partitions to the computing nodes and (b) after its distribution. The yellow and blue cells are internal cells of nodes-0 and 1, respectively, while the orange and purple cells are called external and are located identically in both nodes, thus creating an overlap. This serves to maintain the continuity of the computational grid even after partitioning, while allowing the communication of the neighboring parts through their interface. However, the more the number of overlaps increases, the lower the efficiency of the partition process. [60]

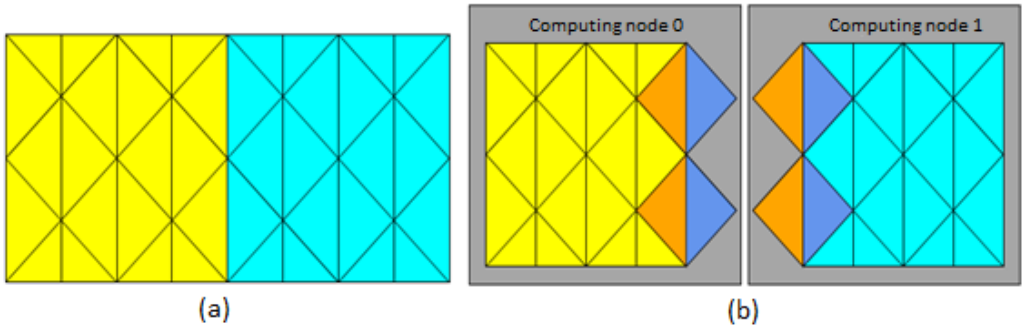

Figure 7: Partitioned grid before (a) and after (b) the distribution to the computing nodes. (example for 2 nodes)

4.1.4. Parallel processing: Parallelization of UDFs

The data mining procedure of the results of the detailed CFD model simulations is feasible in Fluent, using User Defined Functions (UDFs). One UDF is code written in C/C++, which can be paired with the main CFD code of Fluent, allowing it to perform functions that are not built in the commercial code. These functions vary and can range from writing or reading specific files to setting parameter values in different ways other than the ones Fluent has (e.g., setting a reaction rate of a different form than Arrhenius), while the time of their activation during the computational process is determined by the user through special commands at the beginning of the UDF code. Writing a serial UDF is a simple and straightforward process, although of course this also depends to a large extent on the intended function that the user is trying to incorporate into the code. On the other hand, writing a parallel UDF, or correspondingly parallelizing a serial one, is a much more complex process in terms of coding, as all the interactions and exchanges of information that take place between the computing nodes and the host must be considered. Basic principles of this process will be briefly described below. [58] [63]

First, a very important principle of parallel UDF writing is defining the type of process that makes up each part of the code. This is done through specific commands called "compiler instructions" and are as follows:

- **#if RP_HOST**: specifies that the following commands are executed only by HOST
- **#if RP_NODE**: specifies that the following commands are executed only by computing nodes
- **#if PARALLEL**: Specifies that the following commands involve both the HOST and computing nodes, but not serial corresponding processes.

Corresponding to the above commands, their opposites arise with the addition of an exclamation point (!) before the process name.

Other basic commands used exclusively when writing parallel UDFs are Commands for communication and/or data transfer between HOST and computing nodes, Commands to perform global procedures, and Commands to run a loop on the cells grid. It is required to pay attention when including commands in the latter category, as it must be considered the fact

that the partitioned grid contains internal and external cells, with the latter existing in multiple interfaces of partitions, for a reason mentioned in subsection 4.1.3.

The exact procedure applied in this thesis for the purpose of parallelizing serial UDFs for data mining is described in subsection 4.1.6.

4.1.5. Parallel processing: Utilized computer arrays

All the necessary parallel simulations carried out in the following CVD applications of this thesis ran on the Polylaos and Tyranistar computer arrays, which are housed in the School of Chemical Engineering of National Technical University of Athens. As far as the simulations themselves are concerned, the cores recruited to act as computing nodes are 12, while the grid used for the respective application is divided into also 12 partitions. Therefore, each node also corresponds to a part of the grid. It is emphasized that these values resulted from an analysis of the reduction of computational time during parallel processing between the range of 10 to 14 nodes. It should be noted that an increase in nodes above a certain limit result in a reduction in the speed of solution since the communication time between nodes is increased excessively.

Finally, the partitioning method applied is the Principal Axis, in which the computational grid is partitioned based on its main coordinates (these may be identical to the Cartesian ones). The method was chosen since it greatly facilitates the collection of data from the results of the simulations. Of course, a study was carried out to compare the speed of this method with the METIS partitioning method, which is often the preferred method, and the results were comparable. In addition, the partitioning was optimized using the smoothing technique mentioned in subsection 4.1.3.

4.1.6. Parallel programming

The data mining consists of variables in the Fluent CFD code, which can be main operating parameters, namely, the fields of velocity, pressure and temperature components in every dimension. The user can, also, include secondary variables such as molecular concentrations/fractions of compounds, reaction rates etc. These variable fields constitute the hydrodynamic characteristics of the flow within the reactor. These data are extracted from the serial solver results through a UDF, which practically creates loops on all cells in the grid, while within each loop there are commands that read and then write the values of the above chosen

fields for each cell in an array file. The UDF is activated either throughout the simulation (with trigger operation) or at the end when the Fluent code converges to a solution.

The stages of parallelization of this UDF that were followed are briefly the following:

1. HOST opens the file.
2. Each computing node receives access to the data of the cells of the grid partition that it is assigned to during the parallel simulation of the model.
3. Each computing node creates loops on its cells, reads the values of the requested variables and finally stores them in appropriate arrays within itself.
4. Node-0 is the first to send its own data to HOST.
5. The remaining nodes send their own data to node-0.
6. Node-0 sorts them to correctly represent the computational grid when combined (with no overlaps or other changes to the grid topology) and sends them to HOST in the appropriate order.
7. HOST receives them and combines them into an array for each variable.
8. Finally, HOST writes the data in the file and then closes it.

4.2. Grid-to-Grid solution interpolation

Interpolation is defined as the process of creating new values of a variable within a set of known values of that variable. The unknown values are created in between the known ones, while these may be scattered in spatial dimensions, in time or generally any vector space. The basic principle of this operation is the use of functions that correlate the values of the variable at known points of space with its values in their neighboring unknown points. The type of function used also determines the interpolation method. The interpolation methods available in the Fluent code are the constant or zeroth-order, linear, polynomial, inverse distance and least squares interpolation. [64]

Fluent applies the general technique of interpolation in several cases, some of which were reported in subsection 3.3.2. Another application of interpolation is the use of the solution obtained by simulating a model on one grid as an initial estimate when simulating the same model on another grid. The difference between the two grids can lie in various characteristics, such as the shape of the cells, while a common case is that the model is solved in a coarse grid

and the solution is inserted into a fine one as initial estimation. The benefit of the above procedure is the significant reduction of the computational cost of the latter simulation. [58]

In the case of grid-to-grid interpolation, Fluent utilizes the method of zeroth-order interpolation**,** which is otherwise called Nearest Neighbor**.** According to this method, each cell (or in the general case, point) of the new grid receives the values of the variables of the solution from the nearest cell of the initial grid, ignoring the values of the rest of its adjacent cells. This method is bound to be accompanied by large interpolation errors and is rarely used, as the linear interpolation method (or First Order) has much smaller errors, while its complexity does not increase significantly. However, its large error does not play a particular role in the case that it is simply used as an initial estimate, as the FVM will converge on the same solution regardless of the initial estimate. It is therefore considered preferable for this convergence to take a little longer than to increase the complexity of interpolation and consequently the computational cost of its development. [64]

5. Order reduction of dynamic models

5.1. Introduction

Computational simulations have now established themselves as the third way to study processes, along with theory and experiments. The main factor in this is the rapid development of technology and more specifically the increase of the available computing capabilities. At the same time as this increase, the requirements for the accuracy of the simulations have increased, which is a result of both the development of innovative computational algorithms and methods, as well as the creation of realistic detailed models. However, as admirable as the accuracy of these models may be, the computational requirements that accompany their solution are quite high and, in some cases, prohibitive, even with the modern available technology. In these cases, an attempt is made to simplify the models, by removing from them details that are not useful for the purpose of each study, while at the same time maintaining the basic properties of the complete model. A common tactic is the prior simplification of the model, either through operational order reduction or through compact modeling**.** In the first approach, any theoretical knowledge of the phenomena that make up the system is utilized, with the aim of including in the model only the phenomena and dimensions that are attempted to be studied but also those

that affect it (e.g., 2-D or 3-D modeling). Accordingly, the second approach is applied in the case of large-scale systems consisting of individual smaller subsystems, where instead of attempting to model the whole system, models are developed for the individual subsystems, which are solved in combination. [65] [66] [67]

In many cases, however, advanced simplification techniques are not feasible, so the development of Reduced Order Modeling (ROM) algorithms is required, which attempt to simplify high-complexity models consisting of a huge number of equations and variables (over 10^5). The ROM was originally developed in the field of model theory and automation, although it was later extended to the field of numerical analysis. Its main purpose is to capture as fast as possible the basic structure and properties of the original model, omitting its less important details. Thus, a ROM is obtained, which can be solved at a significantly lower computational cost, while maintaining the input-output behavior of the original model. Due to all the above, ROM has frequent application in automation and process optimization in real time, where it is required to predict the state of the system within a few minutes (or even seconds), which is in no way possible using detailed models. [65]

5.2. Principle of model order reduction

In computational mechanics, a large percentage of the models studied consist of a system of partial and/or ordinary differential equations and their respective boundary conditions. A version of those are the macroscopic fluid dynamics models studied in this work, for which ROMs are being created. In such cases, where the model consists exclusively of PDEs with independent variables the position and time, after spatial discretization the following general system of ordinary differential equations over time arises [65]:

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$$

5-1

$$\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u})$$

5-2

where $\boldsymbol{u}$ the system input, $\boldsymbol{y}$ the output and $\boldsymbol{x}$ the state vector. The above input-output system is illustrated in Figure 8.

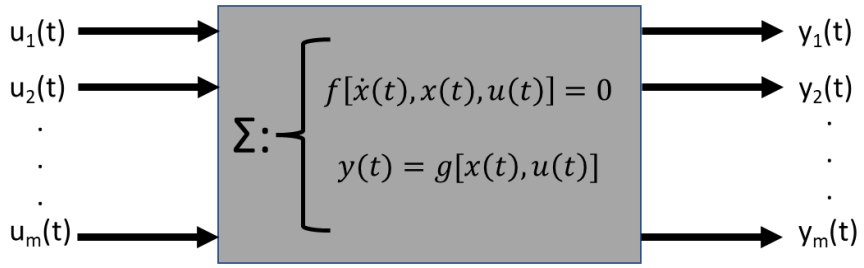

Figure 8: Dynamic Input-Output system.

The complexity of this system lies in the number of state variables, that is, the N dimension of the vector **x**. Therefore, the objective of the ROM is to reduce the dimension of the above vector while maintaining the input-output relationship of the system. The new dynamic system takes the following form:

$$\frac{d\hat{\boldsymbol{x}}}{dt} = \hat{\boldsymbol{f}}(\hat{\boldsymbol{x}}, \boldsymbol{u})$$

5-3

$$\boldsymbol{y} = \hat{\boldsymbol{g}}(\hat{\boldsymbol{x}}, \boldsymbol{u})$$

5-4

where $\hat{\boldsymbol{x}}$ the approximation of **x**, with a dimension less than N. Also, $\hat{\boldsymbol{f}}$ and $\hat{\boldsymbol{g}}$ are the approximations of the respective functions **f, g**, while **u** and **y** have remained the same. In order to achieve a satisfactory approximation of the original system it is important to meet the following criteria:

1. Small approximation error

2. Maintain basic properties and characteristics of the original system

3. Computationally efficient order reduction process

To achieve the above, a wide range of different methods and techniques has been developed, the vast majority of which are based on the concept of projection, i.e., the compacting of the solution of the detailed model towards the creation of a basis. Finally, it is noted that in the present work the input of the general system **u** can be any controlled variable

or operating condition of the CVD reactor, while its output is the state vector of the reactor [8] [68].

5.3. Proper Orthogonal Decomposition

This is a method of order reduction that was originally developed in the context of CFD, although it has now been extended to other fields such as electronics and finds frequent application in CFD problems. As has already been thoroughly analyzed, models of such problems consist mainly of nonlinear PDEs, which are discretized in space and time. The solution of such models is a computationally costly process, while even reducing their order is not a simple task. POD is not only one of the few available methods capable of simplifying models of nonlinear PDEs but is also considered the most advanced of them. The POD, which is also called Principal Components Analysis (PCA) or Karhunen–Loève decomposition, is based on the main idea that the response of a system to a particular input involves the behavior and basic properties and characteristics of that system. Thus, with appropriate mathematical processing of the output one can create a ROM containing the above, something very important for the accuracy of the model, as mentioned in the previous section. [65] [69]

The main advantages of this method are, the fact that it is based solely on data processing and not equations and the process of creating the model consists of simple linear algebra operations between matrices, even though it is applied to nonlinear systems. Thus, it can be applied even in cases with zero knowledge of the properties of the system studied or in cases where the equations describing the system are notably complex. However, the method being “equation-free” acts many times and as a limitation, as the ROM can predict the behavior of the system only within the operating area from where the data were obtained, since the global behavior of the system is unknown. [70] [71] [72]

The method is based on the fact that state x variables can be approximated by $\hat{x}$ (see 5.2) expressed as the product of base functions which maintain the spatial dependence of the state and coefficients which respectively maintain the time dependence [16] [65] [73] [74]:

$$\hat{x}(\boldsymbol{r}, t) = \sum_{i=1}^{d} a_i(t)\varphi_i(\boldsymbol{r})$$

where $\hat{x}(\boldsymbol{r}, t)$ a system state variable at position **r** and time t, $a_i(t)$ the coefficients and $\varphi_i(\boldsymbol{r})$ the basis functions. If spatial discretization is considered and the state variables are converted to the state vector (see 5.2), then the above equation is transformed into the:

$$\hat{\boldsymbol{x}}(t) = \sum_{i=1}^{d} \boldsymbol{a_i(t)\varphi_i}$$

5-6

where $\boldsymbol{\varphi_i}$ the basis vectors, which contain the spatial dependence of the model.

Although, POD is based on the two equations above, it is capable to only determine the optimal basis of the reduced model. This optimal basis is calculated by projection of the vectors containing the data, used to construct the ROM, from the total space V with dimension N to the subspace V_d with dimension d, with d<<N. To minimize the error of the approach, the projection and subsequent the basis must be orthonormal, in the sense of least squares. The choice of data to be projected also plays a very important role, although this is very often based on investigation trial and error strategies rather than a scientifically thorough algorithm. An important variation of the POD method, which is related to the way the projected data are organized is the Method of Snapshots (MoS), where the set of data is discretized in time and consists of separate states of the system for each moment in time, which are called snapshots, as opposed to the existence of a continuous trajectory as in the general method of POD. To calculate the coefficients of the approach, the Galerkin method is often applied, as in [67] [71] [74] [75], which is also based on the equations 5-5 and 5-6. According to this method, the PDEs of the detailed model are projected onto the orthonormal basis calculated by the POD, with the purpose of the least squares method. The Galerkin method, like the POD, is a projection method and are usually applied in combination for the model order reduction [76]. However, unlike the POD, the Galerkin method is based on equations rather than data. For this reason, in the context of this dissertation the coefficients of the ROM are determined using ANN, which is a statistical method that requires the knowledge of data and not equations. Therefore, the ROM is purely data oriented and entirely developed from the results of simulations of the detailed model. The ANN will be analyzed later, while the Galerkin method is omitted in this work. [16] [65] [76]

5.3.1. Design of POD basis

Suppose that the data used for the calculation of the base are organized in the matrix $\boldsymbol{Y} = [\boldsymbol{y_1}(t) \quad \cdots \quad \boldsymbol{y_m}(t)]$, which belongs to the vector space $V = \mathbb{R}^{Nxm}$ and consists of individual m trajectories $\boldsymbol{y_i}(t) \in \mathbb{R}^N \; with \; i = 1, \dots, m$ and $t \in [0, T]$. These orbits constitute the response (output) of the system to changes (input). The main components *of* the table $\boldsymbol{Y}$ are then analyzed to find a d-dimensional subspace $V_d \subset V$, in which the approximation of the above table will be located. Therefore, a projection $\Pi_d : V \to V_d$ of fixed dimension d is sought, with minimized least squares [65]:

$$\|\boldsymbol{Y} - \Pi_d \boldsymbol{Y}\|^2 = \sum_{i=1}^{m} \int_0^T \|\boldsymbol{y_i}(t) - \Pi_d \boldsymbol{y_i}(t)\|^2 dt$$

5-7

The solution to this problem is achieved through *the* correlation matrix $\mathbf{K} \in \mathbb{R}^{NxN}$:

$$\boldsymbol{K} = \sum_{i=1}^{m} \int_0^T \boldsymbol{y_i}(t) \boldsymbol{y_i}(t)^T dt$$

5-8

where $\boldsymbol{y_i}(t)^T$ the inverse matrix of $\boldsymbol{y_i}(t)$. By definition, **K** is a symmetrical, positive, semi-definite matrix with real, positive, in ascending order eigenvalues, $\lambda_1 \geq \cdots \geq \lambda_N \geq 0$ to which correspond the eigenvectors $\boldsymbol{u_j}$, which are defined by the following equation:

$$\boldsymbol{K u_j} = \lambda_j \boldsymbol{u_j}, \;\; j = 1, \dots, N$$

5-9

Due to the structure of the correlation table, these eigenvectors can be used for the construction of the orthonormal basis of the POD. Thus, the optimal subspace of dimension d, able to adequately represent the table **Y,** is the $V_d = span\{u_1, \dots, u_d\}$, whereas span is denoted the linear set of the first d eigenvectors, which are now called main POD vectors. It follows from the above procedure that the minimum error is equal to:

$$\min_{V_d} \|\boldsymbol{Y} - \Pi_d \boldsymbol{Y}\| = \sum_{j=N-d+1}^{N} \lambda_j$$

5-10

i.e., the error of the approximation equals the sum of the eigenvalues that were not considered when creating the subspace and basis. In addition, the optimal approximation $\Pi_d : V \to V_d$ is given by the formula:

$$\Pi_d = \sum_{j=1}^{d} \boldsymbol{u_j} \boldsymbol{u_j^T}$$

5-11

So, if each vector $\boldsymbol{y_i}(t)$ of the Y table is written in the form:

$$\boldsymbol{y_i}(t) = \sum_{j=1}^{N} \boldsymbol{y_{ij}}(t) \boldsymbol{u_j}$$

5-12

where:

$$\boldsymbol{y_{ij}}(t) = \boldsymbol{y_i}(t)^T \boldsymbol{u_j}$$

5-13

then the projection Π_d on the vectors $\boldsymbol{y_i}(t)$ will equal to:

$$\Pi_d \boldsymbol{y_i}(t) = \sum_{j=1}^{d} \boldsymbol{u_j} \boldsymbol{u_j^T} \sum_{j=1}^{N} \boldsymbol{y_{ij}}(t) \boldsymbol{u_j}$$

5-14

simplified in the following form, which is the final form of the ROM $\widehat{\boldsymbol{y_i}}(t)$:

$$\widehat{\boldsymbol{y}_{\boldsymbol{i}}}(t) = \Pi_d \boldsymbol{y}_{\boldsymbol{i}}(t) = \sum_{j=1}^{d} \boldsymbol{y}_{\boldsymbol{ij}}(t)\boldsymbol{u}_{\boldsymbol{j}}$$

5-15

5.3.2. Method of Snapshots

In many POD applications, the N dimension of trajectory vectors can be a very large number (10^6 – 10^{10}). This category includes fluid dynamics problems, where the number of state variables is usually big due to the spatial discretization of conservation PDEs (see 3.2). As described in the section above, the calculation of the main POD vectors, i.e., the POD basis, requires the calculation of the eigenvalues and eigenvectors of the correlation matrix $\boldsymbol{K} \in \mathbb{R}^{NxN}$, which is a computationally costly problem. In many cases, this would be completely impossible, due to the huge requirement in computational memory by a problem involving a matrix with dimension NxN, where N>10^6. The solution to this problem was given by [14], who proposed the alternative MoS, which is a variation of the general POD in the way the Y matrix is organized, and the correlation matrix used to determine the optimal basis. In the applications of this dissertation this method is applied.

Specifically, in this method the trajectories $\boldsymbol{y}_{\boldsymbol{i}}(t)$ have been discretized over time, so the matrix **Y** consists of individual snapshots of the system instead of continuous trajectories, as in the general POD. It should be noted that the term snapshot refers to the state vector $\boldsymbol{x}(t_j) \in \mathbb{R}^N$ of the system at time t_j. Therefore, each trajectory is discretized as $\boldsymbol{Y}_{\boldsymbol{i}}(t) = [\boldsymbol{x}(t_0)\boldsymbol{x}(t_1) \dots \boldsymbol{x}(t_{m_i-1})] \in \mathbb{R}^{Nxm_i}$, where m_i the number of time steps of the discretization and subsequently the number of snapshots in each trajectory $\boldsymbol{Y}_{\boldsymbol{i}}(t)$. Respectively, the total data matrix has the form $\boldsymbol{Y} = [\boldsymbol{Y}_{\boldsymbol{1}}(t) \quad \cdots \quad \boldsymbol{Y}_{\boldsymbol{N_{traj}}}(t)] \in \mathbb{R}^{N\times m}$, where N_{traj} the number of trajectories and m the number of snapshots of the matrix, which is calculated from the equation:

$$m = \sum_{i=1}^{N_{traj}} m_i$$

5-16

It should be noted that each column of matrix **Y** is also a snapshot of the system.

The correlation matrix defined in equation 5-8 takes the form of:

$$\boldsymbol{K} = \boldsymbol{Y}\boldsymbol{Y}^{\boldsymbol{T}} \in \mathbb{R}^{N\times N}$$

5-17

This is the second differentiation of the snapshot method from the general POD, since instead of the matrix $\boldsymbol{K}$, the correlation matrix $\boldsymbol{K}'$ is used, which is defined as:

$$\boldsymbol{K}' = \boldsymbol{Y}^{\boldsymbol{T}}\boldsymbol{Y} \in \mathbb{R}^{m\times m}$$

5-18

whose eigenvalues (but not the eigenvectors) are identical to those of **K**. In most physical problems $m \lll N$, so handling the above matrix is much easier than **K**, whose eigenvalues are often impossible to calculate due to limitations in computational memory.

After calculating the eigenvectors $\boldsymbol{v_j}$ of $\boldsymbol{K}'$, the eigenvectors $\boldsymbol{u_j}$ of **K** are defined**,** from which, as mentioned in the previous section, the main POD vectors that form the basis are selected. The transition from $\boldsymbol{v_j}$ to $\boldsymbol{u_j}$ is made through the equation:

$$\boldsymbol{u_j} = \frac{1}{\sqrt{\lambda_j}}\boldsymbol{Y}\boldsymbol{v_j}, \qquad j = 1, \dots, m$$

5-19

Therefore, in line with the equations 5-12 and 5-13, the equations for the approximation of each snapshot and its corresponding coefficient may be written as follows:

$$\widehat{\boldsymbol{x}_{\boldsymbol{i}}} = \sum_{j=1}^{d} a_{ij}\boldsymbol{u_j}$$

5-20

$$a_{ij} = \boldsymbol{x_i}^{\boldsymbol{T}}\boldsymbol{u_j}$$

5-21

The matrix containing the d first eigenvectors, i.e., the *main POD vectors* is also the orthonormal POD basis **Z,** i.e.:

$$\boldsymbol{Z} = [\boldsymbol{u_1} \quad \dots \quad \boldsymbol{u_d}] \ \in \mathbb{R}^{N \times d}$$

5-22

Also, the vector containing all the coefficients of the snapshot i is defined as:

$$\boldsymbol{a_i} = [a_{i1} \quad \dots \quad a_{id}] \ \in \mathbb{R}^{d}$$

5-23

so, the equations 5-20 and 5-21 are written in a table form as follows:

$$\widehat{\boldsymbol{x_i}} = \boldsymbol{Z a_i}$$

5-24

$$\boldsymbol{a_i} = \boldsymbol{Z^T x_i}$$

5-25

Finally, the last two equations are generalized to entire trajectories or even to the whole data matrix **Y,** where in the latter case they take the form:

$$\widehat{\boldsymbol{Y}} = \boldsymbol{ZA}$$

5-26

$$\boldsymbol{A} = \boldsymbol{Z^T Y}$$

5-27

where **A** the total matrix of all vectors $\boldsymbol{a_i}$, i.e.:

$$\boldsymbol{A} = [\boldsymbol{a_1} \quad \dots \quad \boldsymbol{a_m}] \ \in \mathbb{R}^{d \times m}$$

5-28

It is noted that, to accurately determine a state or a trajectory of the system with the above set of equations, the knowledge of the coefficients is required in addition to the basis. But from the equations 5-21, 5-25 and 5-27 it should be noted that their calculation presupposes a prior knowledge of the state/trajectory to be determined, which is obviously practically useless. For this reason, it was also referenced in section 5.3 that POD alone can determine only the basis and not the coefficients of the overall ROM. Of course, it should be noted that the equations 5-21, 5-25 and 5-27 are not entirely useless, as they calculate the accuracy of the basis for the approximation of the detailed model. (see 5.3.5)

5.3.3. Dimensionality of POD basis

The choice of the dimension d of the basis is always made based on the following two conflicting objectives [65]:

1. The approximation should be as best as possible, i.e., the error should be minimized.

2. The dimension should be as small as possible, to achieve the maximum possible reduction of the computational requirements for solving the model.

To simultaneously satisfy as much as possible both criteria, the choice is based on the eigenvalues of matrix **K** (and $\boldsymbol{K}'$), arranged in ascending order. The eigenvalues of the correlation matrix of a dynamic system represent characteristics of that system, with the large eigenvalues corresponding to the main properties of its behavior, while the small eigenvalues represent only some secondary characteristics of the dynamic system behavior. [16] [65]

Specifically, according to the general POD method and in fact the equation 5-10, the minimum error is equal to the sum of the eigenvalues not used in the construction of the basis. Therefore, the sum of the eigenvalues used represents the amount of information retained in the POD basis by the detailed model, or to be precise by the data of the detailed model used to create it. Based on this, the information content of the database can be set as follows:

$$I(d) = \frac{\sum_{i=1}^{d} \lambda_j}{\sum_{i=1}^{N} \lambda_j}$$

5-29

where the numerator expresses the information retained in the database, while the denominator expresses the entire information in the matrix **Y**. The desired design objective of choosing d is for $I(d)$ to tend to the unit. Of course, this is in direct conflict with the second criterion mentioned above, so usually a middle solution is optimal. Thus, a frequent and easily applied tactic followed is for the designer to select in advance the necessary percentage of the information content that is deemed satisfactory for the creation of the basis, so:

$$I(d) = min\{d \in N | I(d) \geq p\}$$

5-30

Finally, it is reported that the above quantity initially increases sharply, as the first eigenvalues are large, and it is followed by small increases due to the sharp decrease in the size of the eigenvalues. This is a main reason why the POD method (and subsequently the MoS) is considered such an efficient method of order reduction, since even with a very small number of eigenvectors, it is possible to maintain a large percentage of information content.

5.3.4. Data normalization

In cases where the state of the system consists of sets of variables that differ from each other, the values of which may differ by certain orders of magnitude, the application of the POD method is usually preceded by the normalization of matrix **Y**, to eliminate these large differences and to achieve uniformity within **Y**. This is used in this work, as the state of the reactor consists mainly of fields of velocity, pressure and temperature, variables that can differ even by 8 whole orders of size. The mathematical normalization process that is followed is as follows [8] [19]:

Firstly, the mean value of each row of **Y** is calculated, i.e., the mean value of each variable at the respective position for all snapshots.:

$$\overline{x_i} = \frac{1}{m}\sum_{j=1}^{m}(\boldsymbol{Y})_{\boldsymbol{ij}}, i = 1, \dots N$$

5-31

The standard deviation of each row of the matrix is then calculated:

$$s_i = \left(\frac{1}{m-1} \sum_{j=1}^{m} \left((\boldsymbol{Y})_{\boldsymbol{ij}} - \bar{x_\iota} \right) \right)^{0.5} \quad i = 1, \dots, N$$

5-32

Finally, from each element, the mean value of the row that belongs to it is subtracted and the result is divided by the corresponding standard deviation:

$$(\boldsymbol{Y_N})_{\boldsymbol{ij}} = \begin{cases} \dfrac{(\boldsymbol{Y})_{\boldsymbol{ij}} - \bar{x_\iota}}{s_i}, & s_i \neq 0 \\ 0, & s_i \neq 0 \end{cases} \quad i = 1, \dots, N, j = 1, \dots, m$$

5-33

where $(\boldsymbol{Y_N})_{\boldsymbol{ij}}$ the elements of the normalized data matrix $\boldsymbol{Y_N}$.

Therefore, the determination of the optimal basis, according to what was mentioned in the previous sections, is made with the normalized $\boldsymbol{Y_N}$ and not **Y**. Thus, the de-normalization of the POD basis is also required, so that it corresponds to the states and trajectories of the actual system, not the normalized one. For this, it is enough to simply follow the reverse procedure from the one described in the above three steps.

5.3.5. Accuracy of POD

The evaluation of the accuracy of the method is carried out by determining the relative error of each snapshot during the dynamic simulation of the ROM resulting from the identified basis. It is recalled that although the POD can determine only the basis and not the coefficients for unknown states/trajectories, there is the possibility of calculating the coefficients for a known trajectory, which is used to calculate the errors accompanying the identified basis (see 5.3.2). If a steady state of the reactor is calculated (i.e., for t → ∞), then the relative error of the approximation is calculated from the following:

$$\boldsymbol{Error}(\boldsymbol{x}(t)) = \frac{\|\hat{\boldsymbol{x}}(t) - \boldsymbol{x}(t)\|}{\|\boldsymbol{x}(t)\|}$$

5-34

where $\boldsymbol{x}(t)$ the state vector and $\hat{\boldsymbol{x}}(t)$ its approximation calculated by the ROM.

In the case of calculation of a dynamic response, i.e., of the entire trajectory, of the system, then the relative error is calculated over time, with the relative error being defined again by the above equation, only that now $\boldsymbol{x}(t)$ and $\hat{\boldsymbol{x}}(t)$ correspond to snapshots and their approximations, respectively, and not to steady states. In addition, the root mean square error is calculated as:

$$\boldsymbol{avg\ error} = \left(\frac{1}{m} \sum_{j=1}^{m} \boldsymbol{Error^2}(\boldsymbol{x}(t_j)) \right)^{0.5}$$

5-35

as well as the maximum associated error:

$$\boldsymbol{max\ error} = \max\left(\boldsymbol{Error}(\boldsymbol{x}(t_j)\right)$$

5-36

for the purpose of a comprehensive statistical study of the accuracy of the method.

5.4. Artificial Neural Networks

The state vector of a dynamic system can be approached by the linear combination of basis vectors containing the spatial dependence, with their coefficients containing the time dependence of the state (see eq. 5-6). The vectors of the basis, which are otherwise called the main POD vectors, are determined according to the POD procedure of a system data matrix, analyzed in previous chapters, which in generality can result either from experiments or from computational simulations of a system. This method requires only the knowledge of data of sufficient quantity and quality and not of the phenomena that govern the system and their respective equations. [16] [65] [73] [74] [77] [78]

On the other hand, the time coefficients are usually calculated by Galerkin projection of the basic equations of the detailed model on the POD basis. Although both techniques work very effectively in their combination, as both are projection methods and practically complement each other, Galerkin presupposes knowledge of the equations. In case these equations are particularly complex, uncertain, or even unknown, its implementation is from difficult to impossible. Therefore, in the context of this work, another method is proposed for the calculation of time coefficients, which can identify them purely with the knowledge of data and not equations. This method is the ANN, where its combination with the POD yields an overall approach to model order reduction of dynamic models based solely on input-output data of the system. [65] [74] [75] [76]

Although computers have for many years reached a level where they can perform mathematical functions much faster and more efficiently than the human brain, it is only a few years that they have managed to solve, simplistic, perception problems (e.g., cat/dog discrimination). The treatment of such problems by computer programs leads to the development of a new science, the Artificial Intelligence (AI), where computer systems can carry out tasks for which they are not programmed directly by the user but are instead trained through 'tangible' examples. This training process is in itself a whole scientific field, called Machine Learning. [79]

ANN are tools found both in areas such as AI, Machine Learning and Data Mining, where they work in combination with other techniques, and in more traditional areas such as Engineering or Economics, where they generally act alone to solve difficult numerical problems in an efficient way. In practice, these are computational algorithms, which consist of many elements that work in parallel and are based on the principles of Statistics. Their function resembles that of the human nervous system, which can process with great ease any external stimulus and use its information to make logical connections and draw conclusions in the future. Therefore, ANN attempts to utilize the mathematical modeling of biological neural networks (NN) resulting in the achievement of similar functions. Important applications of ANN to date are the following: [20] [80]

- Function approximation – Specify input-output relationship
- Template matching/recognition
- Data sorting

- Data forecasting
- System/process optimization and control

Finally, it is recalled that the ANN require for their training and therefore their overall operation only the existence of data from the studied system and not further knowledge of it. [81]

5.4.1. Principle of artificial neuron operation

A crucial role in the effective and efficient functioning of an ANN is played by both the individual units and their organizational structure. Initially, reference is made to the former, i.e., the artificial neurons of an ANN and the principle of their operation is presented schematically. [20] [8] [82]

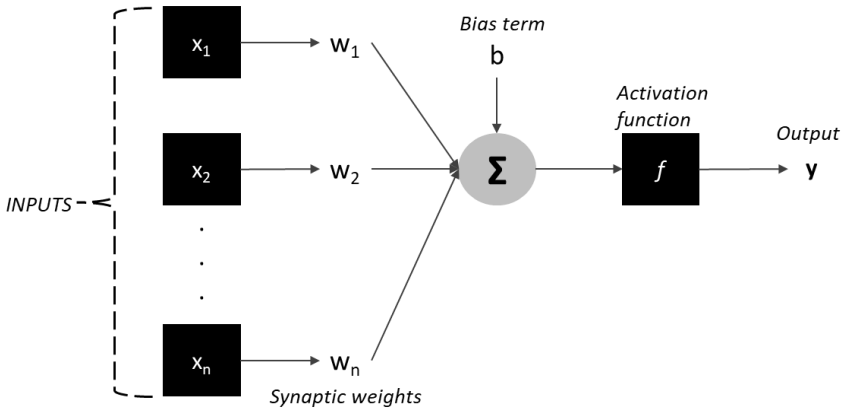

Figure 9: Schematic representation of an artificial neuron.

According to the figure above, within an artificial neuron, the following stages take place:

1. Initially, the input variables x_i, enter the neuron.
2. Each input variable is then multiplied by its corresponding coefficient, which is called the synaptic weight w_i of the variable.
3. The resulting products enter a sum operator, together with an additional fixed term, the neuron's bias term b. This can be written as:

$$net = \sum_{i=1}^{n} w_i \, x_i + b$$

5-37

where net the result of the operator. Of course, b can also be considered as a synaptic weight corresponding to a constant input variable equal to one (1).

4. Finally, the "variable" net enters the activation function f, which is also known by the name of the transfer function, resulting to the output variable of the neuron y.

This procedure is summarized in the following equation, which is also the general mathematical model of an artificial neuron:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

5-38

Expressing all the above quantities in the form of vectors, then the following equations are derived:

$$\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

5-39

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$$

5-40

where $\boldsymbol{x}$ and $\boldsymbol{w}$ the input and synaptic weights vectors, respectively. Thus, the equation 5-38 takes the following vector form:

$$y(t) = f(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x})$$

5-41

The most important parameters of a neuron are the synaptic weights, as these determine the magnitude of gravity given to each input variable when calculating the output. The reason is that these quantities practically contain the information-memory that the network acquired during its training and which it uses to solve future problems. Therefore, the values that the vector $\boldsymbol{w}$ will acquire during the training of an ANN significantly affects the effectiveness of the network. More about the training procedure will be analyzed in (see 5.4.6) However, the choice of the activation function should not be underestimated, as this is what ultimately converts the net to y, while also determining the capabilities of a network, as will be analyzed in the next section.

5.4.2. Activation function

As far as the activation function is concerned, it is a nonlinear function, the purpose of which is to obtain the sum of the weighted input variables and to give as a result the correct output variable. Its name derives from the corresponding process that occurs within the neural body, where when the signal is greater than the threshold the neuron is activated, while otherwise it remains inactive. This process is modeled by the step function [20] [82]:

$$f(u) = \begin{cases} 0, & u < 0 \\ 1, & u \geq 0 \end{cases}$$

5-42

In the case of artificial neurons, u corresponds to the variable net, which was defined by the equation 5-37. Therefore, an artificial neuron is activated when the sum of the weighted input variables is greater than the bias. The step function is not often used in ANN, mainly due to the significant limitation that output values can only get two values (0 and 1). A neuron of this type is called *Perceptron* and is used exclusively in the linear classification of data, since it has been shown that if it is possible to separate the input variables by a straight line or a plane, then a network of this nature can fulfill this operation.

Nevertheless, the need to use other activation functions rises, which will be able to identify much more complex relationships than the simple linear classification. Thus, these functions became known as Transfer Functions, as they transfer the input variable to the output

variable for the system they describe. Another relatively common but at the same time simple activation function is the linear one, which has a significant advantage over the step, as it can give output variables of any value. However, the input value must be linearly linked to the output variables, so the linear function can only determine linear input-output relationships. The neurons that use it are called Linear Filters**,** while their respective networks are called Linear Networks**.** The linear function is expressed by:

$$f(u) = u$$

5-43

Finally, another type of activation functions is the sigmoid, which are by far the most common ones observed in ANN. These functions are nonlinear, differentiable functions, a property very important when training an ANN by the general method of Backpropagation, which will be thoroughly analyzed in the following section. Moreover, another reason that makes them so useful is the fact that theoretically they can approach any input-output relationship, regardless of its complexity. In practice, logarithmic-sigmoid and tangential-sigmoid functions are usually used, which are expressed by the following equations respectively:

$$f(u) = \frac{1}{1 + e^{-u}}$$

5-44

$$f(u) = \tanh(u) = \frac{e^{u} - e^{-u}}{e^{u} + e^{-u}}$$

5-45

It should be noted that in the case of tan-sig, many times instead of the actual function, a different one is used that produces almost the same results for a given input, while it runs much faster during the simulation of ANN. Therefore, many computational environments prefer it, including Matlab, with which a large percentage of the computational work in this thesis is performed. This is given by the following:

$$f(u) = \frac{2}{1 + e^{-2u}} - 1 \simeq \tanh(u)$$

5-46

Plots of the three functions mentioned above are shown in Figure 10.

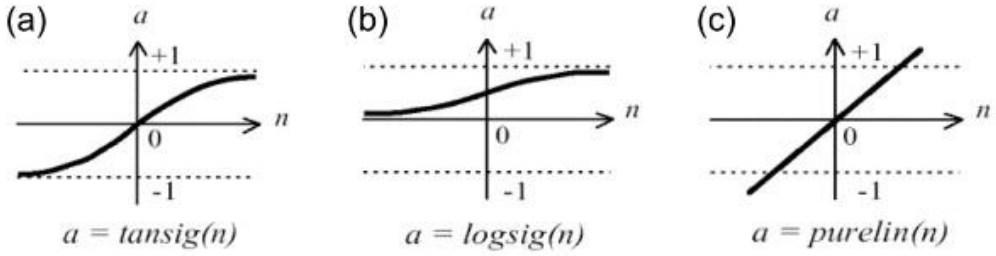

Figure 10: Activation functions diagrams: a) Tangential-Sigmoid, b) Logarithmic-Sigmoid and c) Linear.

5.4.3. Basic architecture of ANN

The artificial neurons are organized in parallel into several configurations, each of which constitutes a **layer** of the network. This parallel configuration allows the simultaneous processing of the same data by many different neurons, which give a different result, thus allowing the rapid processing of information in a manner similar to biological neural networks. An ANN consists of several layers, connected serially to each other, while the interconnection is carried out through the synaptic weights analyzed above. Therefore, the output variables (or equivalent the output vector) of a layer are also input variables of the next plane. Moreover, in each layer, according to the principle of operation of each individual neuron (see 5.4.1), the input variables are common to all neurons of the same layer, while from each separate neuron only one (different) output variable is derived. This arrangement is also presented schematically in Figure 11. [20] [79] [82]

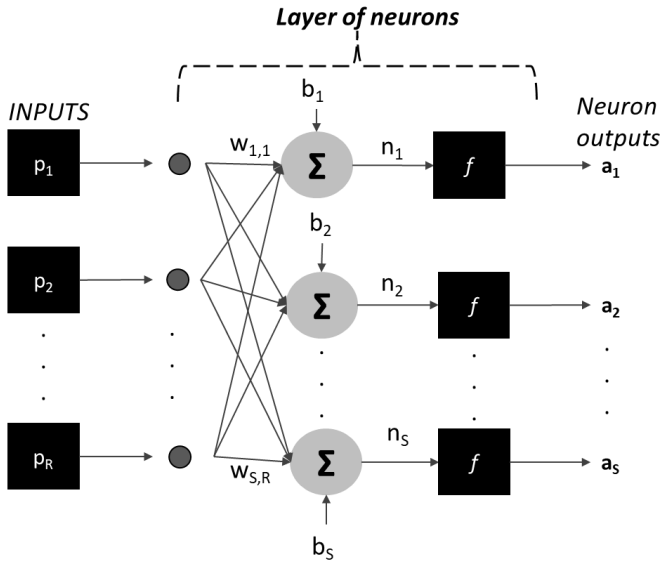

Figure 11: Basic structure of an ANN. R = number of inputs, S = number of neurons in the layer.

The three different types of layers encountered in each ANN are [20] [82]:

- **Input layer**: This is the first part of each ANN, which is responsible for feeding the input variables into the network. It has the specificity that it is the only plane that does not contain neurons and therefore no calculation is made within it. On the contrary, each unit of this layer is called an input node and corresponds to an input variable. The dimension of this layer, i.e., the number of nodes it contains, is equal to the dimension of the input vector, meaning the number of input variables.

- **Hidden layers**: The first of these receives the input variables, which it processes according to what has already been mentioned in the last sections. From each neuron exits an "intermediate" output variable, the set of which enters the next layer. This process is repeated continuously, up to the last layer of the network. These are completely connected to each other, which is achieved through synaptic weights. Each neuron in a hidden layer has the same activation function as the others, and usually the same applies to all the different hidden layers of the total network. The number and dimension of the hidden levels are defined by the user and are two of the most critical design parameters of an ANN, as they directly affect the depth, complexity, and flexibility of the network, thus indirectly its effectiveness and efficiency.

- **Output layer**: It is the last part of an ANN and works in the same way as hidden layers; it accepts as input variables the output variables of the last hidden layer, while its output variables are the output of the total network, i.e., the result of the total computational process. As in hidden layers, all neurons at the output plane have the same activation function. However, this usually differs from that of hidden planes, and in fact in most cases it is the linear function because of its simplicity. Its dimension is always equal to that of the output vector, which is obvious, since each neuron can calculate a single output variable, thus N neurons are required at the output layer to determine N final output variables.

5.4.4. Main architectures of ANN

Over the years, many different ANN architectures have been developed, which can be classified by many different criteria. A first criterion often used concerns the general training of the network, based on which ANNs are divided into the following three categories: **Supervised learning, Unsupervised learning** and **Hybrid Learning**. In the first case, the network is fed with pairs of input-output vectors and through an optimization method the synaptic weights are fine-tuned iteratively until the network can determine the correct output on its own, knowing only the input. This general category includes, feedforward networks, feedback networks and radial basis networks. In the case of unsupervised learning, the networks are fed only with input vectors, of which they analyze the basic structure by looking for patterns, which they ultimately categorize. In the third case of hybrid learning, a combination of both previous procedures is executed. [8] [20]

In the context of this dissertation, only supervised learning networks and, to be precise, feedforward and feedback networks will be analyzed, which are trained based on the general method of backpropagation. Further information about this method and its algorithms is reported in (see 5.4.7).

The simplest backpropagation network is the feedforward one, which consists of the input layer, one or more hidden layers with a sigmoid activation function and finally the output layer, which has a linear activation function. The networks of this architecture do not contain loops, so the information travels exclusively from the input to the output. This makes them, in general, static networks, i.e., for a given input vector they result unambiguously a specific

output vector, which does not depend on previous inputs, outputs or network states. Therefore, these networks are networks without memory. However, it should be noted that the indirect dynamic operation of these networks is possible but is rarely preferred. [8] [20]

The second backpropagation network architecture is that of feedback neural networks. These consist of layers of the same format as feedforward ones, but the information travels in both directions due to the existence of feedback loops within the network. These loops feed the output of a layer to previous ones, while there is no restriction on the type of connected layers. That is, loops can form between the output layer and hidden layers, the output and input layers, the hidden layer with previous hidden layers, and the hidden with input layer. Although, these networks can operate statically, they are usually used as dynamic systems. At each time step, along with the new input vector, output vectors or intermediate vectors can enter too, which alter the input-output relationship (the one observed in a feedforward network), and therefore the output does not depend only on the input. Thus, it can be assumed that, unlike feedforward networks, the feedback ones have memory. Nevertheless, during the dynamic operation of a feedback network it is usually not fed by independent input vectors but by a specific time sequence thereof, so a sequence of output vectors is respectively generated. Finally, it is noted that the memory depth of these dynamic networks is defined as output and input delay. The first determines how many past output vectors will be considered for the calculation of the current output vector, while the second refers to input vectors for that same reason. [20]

5.4.5. Three-Layer Feedforward network

Although the simplest feedforward network is that of the two layers (input + output), in practice these networks are not used very often. On the other hand, very common networks are those of three layers, which are also quite simple but at the same time computationally capable. Indeed, theoretically it has been proven that no matter how complex the relationship between input and output is, a three-layer feedforward network with feedback is able of approximating it. In practice, of course, in some cases this may be non-effective, so that it becomes practically impossible. A network of this architecture is shown in vector form in Figure 12. The input vector consists of n elements, the hidden layer of L_H neurons with a sigmoid activation function and finally the output layer of L_{out} neurons with a linear activation function. Each neuron functions in the manner described in section 5.4.1. [17] [79] [82]

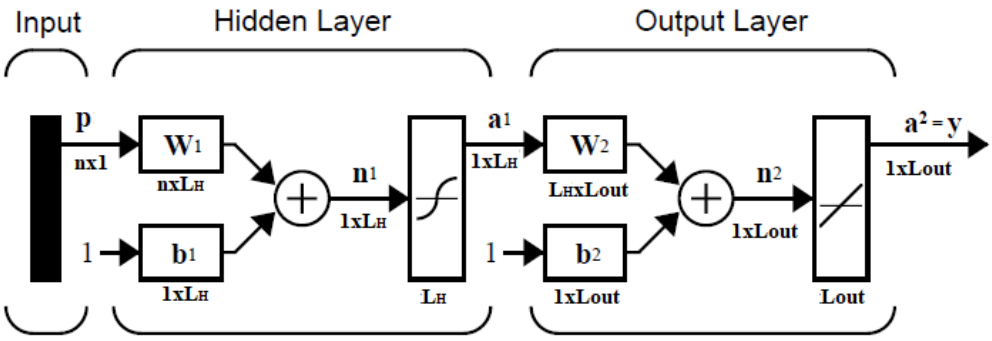

Figure 12: Schematic representation of a Thee-Layer Feedforward network.

The mathematical modelling of this network is carried out in two stages, where the first corresponds to the hidden layer and the second to the output layer. For the random k neuron of the hidden layer, the following equation applies [80]:

$$a_k^1 = f_{sigm}\left(\sum_{i=1}^{n} w_{k,i}^1 p_i + b_k^1\right), \quad k = 1, \dots L_H$$

5-47

where α_k the output variable of the neuron, f_{sigm} the sigmoid function, $w_{k,i}$ the synaptic weight of the neuron for the input variable p_i and b_k the bias of the neuron. The indicator 1 refers to the hidden layer. Accordingly, for the k neuron of the output plane, the following equation applies [80]:

$$y_k = a_k^2 = f_{line}\left(\sum_{i=1}^{L_H} w_{k,i}^2 a_i^1 + b_k^2\right), \quad k = 1, \dots L_{out}$$

5-48

where f_{line} the linear function, while the rest of the notations are identical to those of equation 5-47.

Corresponding to the vector forms of the neuron (eq. 5-41) the two above equations can be transformed into forms of matrices. Thus, by defining the synaptic weight tables $\mathbf{W^1}$ and $\mathbf{W^2}$ as follows:

$$\boldsymbol{W^1} = \begin{bmatrix} b_1 & b_2 & & b_{L_H} \\ w_{1,1} & w_{1,2} & & w_{1,L_H} \\ w_{2,1} & w_{2,2} & \dots & w_{2,L_H} \\ \dots & \dots & & \dots \\ w_{n,1} & w_{n,2} & & w_{n,L_H} \end{bmatrix}$$

5-49

$$\boldsymbol{W^2} = \begin{bmatrix} b_1 & b_2 & & b_{L_{out}} \\ w_{1,1} & w_{1,2} & & w_{1,L_{out}} \\ w_{2,1} & w_{2,2} & \dots & w_{2,L_{out}} \\ \dots & \dots & & \dots \\ w_{L_H,1} & w_{L_H,2} & & w_{L_H,L_{out}} \end{bmatrix}$$

5-50

the equations 5-47 and 5-48 take the following forms:

$$\boldsymbol{a^1} = f_{sigm}\left[\left(\boldsymbol{W^1}\right)^{\boldsymbol{T}}\boldsymbol{p}\right]$$

5-51

$$\boldsymbol{y} = \boldsymbol{a^2} = f_{line}\left[\left(\boldsymbol{W^2}\right)^{\boldsymbol{T}}\boldsymbol{a^1}\right]$$

5-52

Combining these two equations the following final equation that describes the total network of Figure 12 is formed:

$$\boldsymbol{y} = f_{line}\left[\left(\boldsymbol{W^2}\right)^{\boldsymbol{T}} f_{sigm}\left[\left(\boldsymbol{W^1}\right)^{\boldsymbol{T}}\boldsymbol{p}\right]\right]$$

5-53

5.4.6. ANN training

As already mentioned, synaptic weights are among the most critical parameters of an ANN, as they are the ones where the information acquired by the network during its training is stored. Thus, the training is defined as the process where the value of each synaptic weight is determined, all of which connect the network layers with each other, so that the network can perform a specific task effectively and efficiently. Also, it is concluded that the initial values of

synaptic weights are random and have no significance, while the values after the training indicate the relationship that connects the input and output of the network. [20] [80]

Although, it is possible for the user of the network to determine the weights directly, this rarely takes place in practice. On the contrary, their approximation is achieved through iterative optimization methods, which can be applied based on various criteria. The three main learning/training methods were mentioned in the previous section, while as pointed out in the context of this paper, only the first type of methods is studied, i.e., supervised learning. Regarding the necessary data needed for the satisfactory training of a network, these can be obtained either experimentally or computationally, while their effectiveness is judged on the basis of the following three criteria [20] [82] [83]:

- **Quantity**: A sufficient amount of data is required to include the full value range of parameters studied.

- **Quality**: Indicates if the data are sufficiently spread out in the initial model data space. If the data set is focused around a small area of the data space, the problem of overfitting arises, which is analyzed later.

- **Computational complexity**: Indicates the computational time required to complete the network's training by this data set; of course, it is significantly influenced by both the structure of the network and the training algorithm.

As far as overfitting is concerned, this is a statistical phenomenon that is very often observed in ANN training and simulation, with significant negative effects on their effectiveness, so special attention is required to the factors that cause it. As the ANN has no knowledge of the real function that connects the input to the output they attempt to determine, the approximation they calculate is highly accurate only within the data set of which they were trained. Of course, the more carefully their design and training is carried out, the higher their ability to generalize to a larger range of parameters. However, sometimes networks focus too much even on subspaces of the overall data space, meaning the phenomenon of overfitting. In these cases, while the errors of the approximate output with the actual one may be very small during training, when attempted to simulate the network at even slightly different values of the input variables, then it fails tremendously to provide high-precision predictions. The factors that determine the appearance and extent of this phenomenon are many, but the most critical that are usually considered are the quantity-quality of training data, the number of hidden layers and

its dimensions and the method of training applied. As far as the quantity and quality are concerned, reference has already been made above, while the method of training will be analyzed below. The hidden layers and the number of neurons they contain, determine the depth of the network and its ability to identify complex relationships. A small number implies inadequate training, while many layers/neurons can cause the final approximation to be defined through an unneeded overly complex relationship, which appears as the phenomenon of overfitting. Therefore, the optimal value of these design parameters should be determined with care and after extensive investigation for each problem. [17] [20] [79] [82]

5.4.7. Training algorithm: Backpropagation

The most widespread algorithms of supervised learning belong to the general category of regression or backpropagation algorithms. The name arises from the fact that the calculations start at the output level, where the error between the approximate and the actual output is calculated, and then propagates to the "back" leading to the fine-tuning of each weight separately. In the typical backpropagation algorithm, the sum of network square errors is used as an objective function, while its derivative is calculated in order to minimize it, with the weights calculated so that this derivative is continuously reduced. For this reason, the algorithms of this category are also called reduction algorithms. A mathematical form that can be written for each iteration of this algorithm is as follows [20] [79] [82] [80] [84] [85]:

$$\boldsymbol{W_{k+1}} = \boldsymbol{W_k} - a_k \boldsymbol{g_k}$$

5-54

where $\boldsymbol{W_{k+1}}$ and $\boldsymbol{W_k}$ the total matrices of the network's synaptic weights in k+1 and k iteration of the algorithm, respectively, $\boldsymbol{g_k}$ the derivative of the objective function in k and finally a_k the training rate also in k iteration.

This algorithm presents the great advantage that when applied to networks with sigmoid activation functions, it can lead to the approximation of any input-output relationship regardless of its complexity. However, it is still a very slow algorithm, especially when compared to the corresponding methods of training linear or step networks. Therefore, too many variants of it have been developed to speed up the training process. These algorithms are distinguished in heuristic algorithms and numerical optimization algorithms. Only algorithms of the second

category will be discussed and to be exact algorithms based on the Newton method. According to this method, synaptic weights can be calculated based on of the following equation for each iteration:

$$\boldsymbol{W_{k+1} = W_k - H_k^{-1} g_k}$$

5-55

where $\boldsymbol{H_k}$ the Hessian matrix, i.e., a square matrix of second-order partial derivatives of a scalar-valued function/field. Although this method greatly accelerates the training process, it has the disadvantage of requiring the calculation of the Hessian matrix, which is very complex and computationally costly. Therefore, instead of this general method, in practice variants are applied that avoid the calculation by using other functions for the approximation of the Hessian matrix, of which the Levenberg-Marquardt algorithm shows a significantly good performance in terms of speed. [80] [82]

According to this algorithm, in cases where the objective function is in the form of a sum, the following approximations apply to the Hessian matrix and the derivative of the objective function:

$$\boldsymbol{H = J^T J}$$

5-56

$$\boldsymbol{g = J^T e}$$

5-57

where $\boldsymbol{e}$ the vector of the network errors and $\boldsymbol{J}$ the Jacobean matrix, which consists of the first-order derivatives and its calculation versus the Hessian one, is much simpler and faster. Thus, based on the above relationships, eq. 5-55 takes the form:

$$\boldsymbol{W_{k+1} = W_k - (J^T J - \mu I)^{-1} J^T e}$$

5-58

where the coefficient μ determines how steep the decrease in the objective function will be.

The Levenberg-Marquardt algorithm greatly accelerates the training of a network compared to other regression techniques, so it is established as the proposed method in cases where the network consists of up to a few hundred synaptic weights. However, when the network is even larger, this algorithm causes severe problems of network overfitting. Although this problem can be addressed in other ways, such as increasing training data or modifying the size of the network, a solution is often provided by modifying the Levenberg-Marquardt algorithm to maintain its high speed but at the same time enhance the network's ability to generalize. This modification is made either through automatic shutdown or through normalization. In the first case, simultaneously with the training, the network is also simulated on independent input data, so if an increase in errors on this data set is observed while at the same time the training errors are very small, then it is indicated that the overfitting of the network has begun and therefore the training process is automatically interrupted, although the objective function has not been minimized. In the latter case, the objective function is modified so that it consists not only of the sum of squares, but also of the sum of synaptic weights squares, thus leading to the simultaneous reduction of both. It has been shown that networks consisting of small values of synaptic weights are much more stable and are less prompt to overfitting, as the avoids giving too much weight to specific variables at the expense of the others. [20] [80] [82]

The objective function F in the case of a normalized backpropagation algorithm is given by:

$$F = aE_w + \beta E_d$$

5-59

where E_w and E_d are the above-mentioned terms, which are defined as:

$$E_w = \sum {w_i}^2$$

5-60

$$E_d = \sum (\hat{y_i} - y_i)^2$$

5-61

The purpose of the coefficients α and β is to weight the two terms. If $\alpha \gg \beta$ emphasis is placed on the reduction of synaptic weights and the consequent increase in network stability, but the training of the network may not be sufficient, while if emphasis is placed on reducing training errors $\beta \gg \alpha$, it becomes more likely that overfitting might occur. Obviously, it is always the case that $\alpha + \beta = 1$, while for $\alpha = 0$ the objective function ends up in its typical form used in the classic methods of backpropagation. [20] [80] [86]

Furthermore, the issue of choosing appropriate values for the two normalization factors needs to be addressed, so that neither of the above two extreme cases occurs. The answer is given through the Bayesian framework of normalization and for this reason the method of training that uses the objective function of eq. 5-59 in combination with this framework is called **Bayesian Normalization** [82]**.** In this method, the parameters α, β are considered as random variables described by specific distributions, so the determination of their optimal values is made through the Bayes' theorem, according to which for a specific set of training data D and neural network model M the final distribution of synaptic weights is given by the relationship [80] [86]:

$$P(\boldsymbol{W}|D, \alpha, \beta, M) = \frac{P(D|\boldsymbol{W}, \beta, M)P(\boldsymbol{W}|\alpha, M)}{P(D|\alpha, \beta, M)}$$

5-62

where $P(\boldsymbol{W}|\alpha, M)$ the initial distribution of synaptic weights before training, $P(D|\boldsymbol{W}, \beta, M)$ the probability of the training data to be derived from the calculated weights and finally $P(D|\alpha, \beta, M)$ the normalization factor. Minimization of the objective function is equivalent to maximizing the term $P(\boldsymbol{W}|D, \alpha, \beta, M)$, which equals to maximizing the distribution of α and β, which is given by [80] [86]:

$$P(\alpha, \beta|D, M) = \frac{P(D|\alpha, \beta, M)P(\alpha, \beta|M)}{P(D|M)}$$

5-63

from which it is concluded that maximizing the distribution of α and β is equivalent to maximizing the probability $P(D|\alpha, \beta, M)$. After several operations, the exact relationship that gives this probability is determined, while after further operations the values of α and β that maximize it are calculated. It is reported that the Hessian matrix is also involved in these

relationships, which is given by eq. 5-56. Finally, after identifying the optimal values of the normalization parameters α and β and therefore the exact form of the objective function F, the method utilizes the Levenberg-Marquardt algorithm.

5.4.8. Dynamic NARX-ANN

As mentioned in section 5.4.4, ANNs are divided into static and dynamic networks, with the former being primarily feedforward networks while the latter are usually recursive networks, i.e., they contain at least one feedback. There are exceptions for both cases, dynamic networks are generally more powerful than static ones, as they can approximate not only input-output relationships but also time dependent patterns. This important property has led to their utilization in a wide range of applications, such as nonlinear filtering, prediction of future output values for given inputs, control and modelling of nonlinear dynamic systems. Their latter ability is used in the present work. These networks are much more complex than the static feedforward ones, as the information is also directed backwards, and each variable also depends on time. This makes their training computationally more difficult. In most cases, their training is carried out with regression algorithms, which were analyzed in the previous section. However, the calculation of the derivative is much more complex in this case, since from the second time step onwards, some of the network inputs depend on the synaptic weight calculated in the previous time step. When these "peculiarities" that arise due to the dynamic behavior of the network are considered during their training, then the method is called Dynamic Backpropagation. Finally, it is pointed out that many times dynamic backpropagation can be avoided, and a dynamic network can be trained through typical regression, something that will be explained below. [20] [82] [87] [88] [89] [90] [91]

A very powerful dynamic backpropagation network is **NARX,** which has taken its name from the homonymous statistical model «**Nonlinear autoregressive with exogenous inputs**». The NARX capabilities have been tested in literature [88] [89] and it is a common ANN architecture used for modeling the input-output relationship of nonlinear dynamical systems [87]. The NARX is described by the following equation:

$$\boldsymbol{y}(t_k) = f\left[\boldsymbol{x}(t_{k-n_x}), \dots, \boldsymbol{x}(t_{k-1}), \boldsymbol{x}(t_k), \boldsymbol{y}\left(t_{k-n_y}\right), \dots, \boldsymbol{y}(t_{k-1})\right]$$

i.e., the output of the model at the time t_k depends on the input at the same time, as well as on the inputs and outputs of previous times. The parameters n_x and n_y are also called delays of the input and output, respectively, and practically define the number of past time steps affecting the current calculated output. When the nonlinear function f is determined through training of an ANN, then that model specifically describes a dynamic recursive NARX network. [82] [88] [89]

Unlike other recursive networks, NARX contains single feedback, which starts from the output layer and ends at the input layer. This not only leads to reduction of the computing resources needed, but additionally makes the NARX networks very powerful, while also giving them several advantages, such as more effective backpropagation training and a reduction in the network's sensitivity to long-term dependencies. In addition, these networks converge faster and have an increased potential for generalization. A typical three-layer NARX network is illustrated in Figure 13. [88] [89]

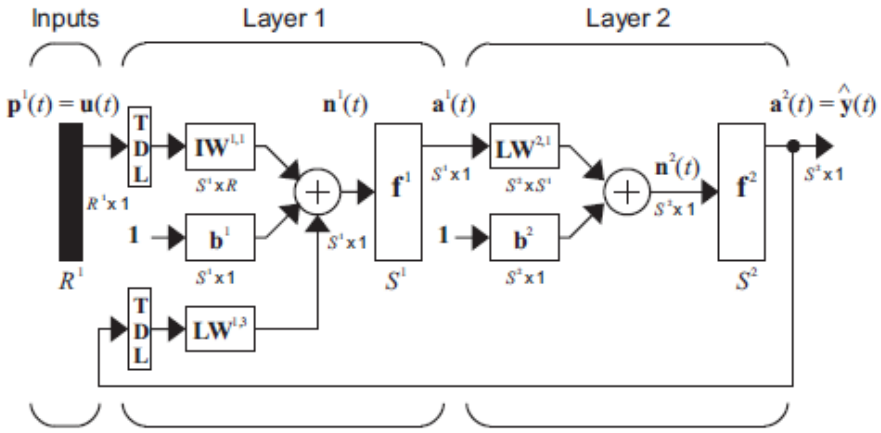

Figure 13: Schematic representation of a NARX network.

NARX networks can operate in two different architectural configurations, parallel and series-parallel, which are presented in Figure 14. The parallel configuration, which coincides with the network of Figure 13, is used during the simulation of the network, where the output of the previous time step is fed back to the input layer in the current time step, giving the property of memory and consequently the dynamic characteristic of these networks. It is noted that due to this structure there is a significant risk of errors accumulating over time. Thus, proper training and precise determination of the function f, is important for the effective operation of a NARX network. [82]

In the series-parallel configuration the feedback is removed, and the network is fed with the actual output vectors corresponding to each input vector that enters the network. Of course, both vectors are time dependent as even without feedback it is still a dynamic network. This configuration is usually applied during the network training, where actual outputs are available from the data set. Although, NARX can also be trained in parallel architecture, its training in series-parallel presents the very important advantage that the network follows a purely feedforward architecture, like those presented in Figure 11 and Figure 12, which allows the use of static regression algorithms (see 5.4.7), which are much simpler and computationally cheaper than dynamic ones.

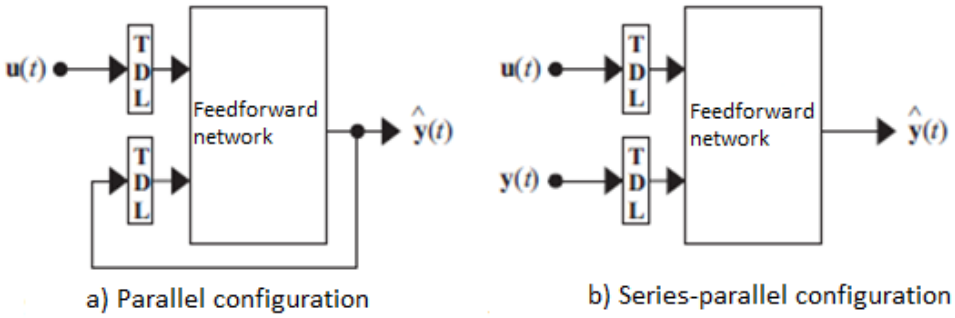

Figure 14: a) Parallel and b) Series-Parallel configuration of NARX networks.

6. Summary of Proposed Computation Framework

The Figure 15 presents the overall flowchart of the computational procedure created and followed in the context of this thesis, which essentially is the computational framework proposed for efficient and effective model order reduction of CFD consisting of 3D detailed large-scale CFD models. The core of this framework is the creation of a Reduced Order Model (ROM) by applying POD and ANN, with additional secondary methods for parallel processing and grid-to-grid interpolation techniques. The resulting model has the ability within a few minutes or even seconds to predict the state of the reactor based on coarse grid data, which is then fed into the detailed model, resulting in its faster convergence and the calculation of the deposition rate and uniformity of the films for any operating conditions. Finally, it should be noted that this framework can be applied to any CFD problem and using any CFD code, while its core, i.e., the order reduction, can be applied to any general computational problem, as it is purely data-driven and does not require a prior knowledge of the properties of the system and the equations that govern it. The following chapters are applications of this framework consisting of, first, the CVD of aluminum, on which this framework was developed and first applied for model order reduction. Second, the CVD of copper, where the framework was used for the same general reason, but with an ultimate purpose of proposing and fitting to experimental data, a new chemical reaction system for the copper deposition. Finally, the framework is applied on a CVD reactor, which presents solution multiplicity in parts of its solution space, to create a data-driven ROM that can capture this phenomenon between the stable solution branches. [8]

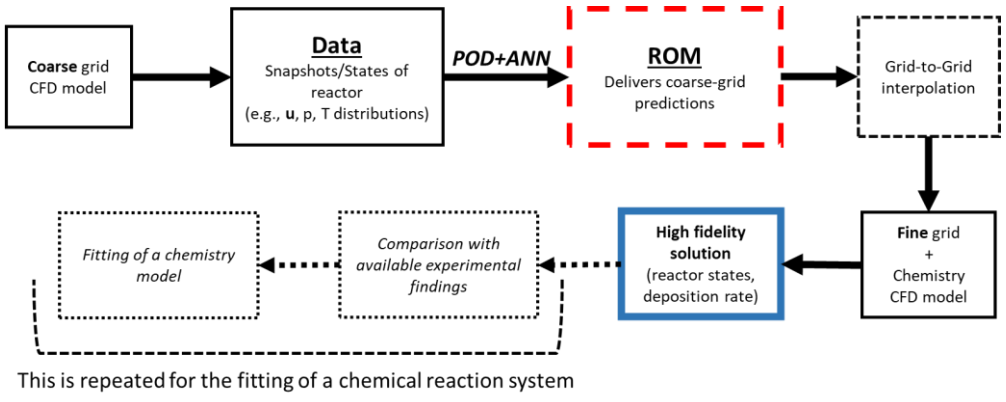

Figure 15: Flow chart of proposed computational framework.

7. Case study 1: Aluminum CVD

7.1. Introduction

CVD is a popular method for the large-scale production of semi-conductor solid films that are used for the construction of integrated circuits (IC), in the computer chip technology. A commonly desired component in these transistor applications are aluminum films, which carry properties, such as high electrical conductivity, significant resistance to electro transport and corrosion, satisfactory adhesion to silicon (a material often found in IC devices) etc. There are multiple processes used for the production of aluminum films, but this work is focused only on CVD. CVD of aluminum films is a well-studied method at a small-scale production, with countless CFD studies investigating dominant flow patterns and chemical pathways that lead to the desired deposited film. Despite of the extensive and long-term study of the aluminum CVD experimentally and computationally, the control of the process is still insufficient and its scaling from laboratory to industrial scale is limited. This reason, along with its important application in the industry of semi-conductors, but also the fact that it is a process that highlights all the most important phenomena encountered in CVD processes, make it ideal to be studied in detail and from many different perspectives [48]. The computational framework and its steps described in previous chapters was developed for this present case study. However, the framework has a general use and is also applied to other CFD problems in this thesis. Additionally, the following case study contains an evaluation of the complex kinetic model [10] of the aluminum CVD on a 3-D computational grid, by utilizing the ROM predictions, to determine if a readjustment of the kinetic parameters is necessary.

7.2. Precursor compounds

In the present work, the metalorganic CVD (MOCVD) of aluminum is studied, i.e., as a precursor, an organic compound containing an atom(s) of a metal element is used, in this case obviously aluminum. The precursor compound must meet as many of the criteria set out in section 2.5. Many precursor compounds have been tested by researchers with various results, of which the ones that now find the most widespread application are Triisobutylaluminium (TIBA), Dimethylaluminum Hydride (DMAH), Trimethylaluminium (TMA), Trimethylamine–Aluminum

Hydride (TMAA), Triethylamine-Aluminum Hydride (TEAA) and finally the Dimethyl-Ethylamine Alane/Aluminum Hydride (DMEAA). [8] [28] [33] [34]

TIBA, DMAH and TMA, among other problems that may entail their use, are significantly disadvantaged in the purity of the produced films, as for various reasons their use leads to films contaminated with carbon atoms (C). As a result, the process was inevitably led to the use of amine aluminohydrides, such as TMAA, TEAA and DMEAA, for the purpose of producing films without carbon impurities. Also, the use of the above precursors implies reduced temperature requirements, which enhances their comparative superiority. [92] [93]

Of these, DMEAA is preferred for the following reasons:

- It is liquid at room temperature, which facilitates its evaporation and transport within the reactor
- It breaks down easier than the other two
- Characterized by less pyro-fluorescence
- Has a higher vapor pressure

Therefore, the DMEAA meets most of the criteria of section 2.5. Although, this does not mean that it is an ideal precursor compound, since it must be handled quite carefully due to its sensitive nature, while its cost is relatively high. Its vapor pressure is determined by:

$$\log[P(Torr)] = 10.85 - \frac{3080}{T(K)}$$

7-1

showing that even for relatively low temperatures, the substance has high vapor pressure values.

The molecule of DMEAA consists of an aluminohydride (AlH_3) and an amino group, where nitrogen (N) bonds to two methyl (CH_3) and one ethyl group (C_2H_5). Its structure is like the other amine aluminohydrides (TMAA and TEAA), due to which it has the two advantages that lead DMEAA to outperform other aluminum-organic compounds; more specifically, the weak Al-N bond, i.e., the hydride and amino group, leads to the easy breakdown of the molecule and consequently reduces temperature requirements, while the lack of Al-C bonds ensures the production of pure films from carbon impurities. Schematic images of the molecule are shown in Figure 16.

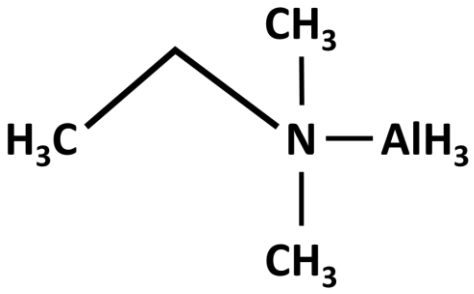

Figure 16: The structure of a DMEAA molecule.

7.3. The CVD reactor

In this chapter, the MOCVD of aluminum films from the precursor compound Dimethylethylaminealane [$AlH_3N(CH_3)_2(C_2H_5)$, DMEAA] is studied. This process has been experimentally and computationally studied in a vertical, cylindrical, stagnant flow, stainless steel CVD reactor with heated walls. The reactor is located at the laboratory of CIRIMAT (Centre Inter-universitaire de Recherche et d'Ingénierie des Matériaux) in INP Toulouse in France. A schematic of the reactor is shown in Figure 17. [8] [10] [28]

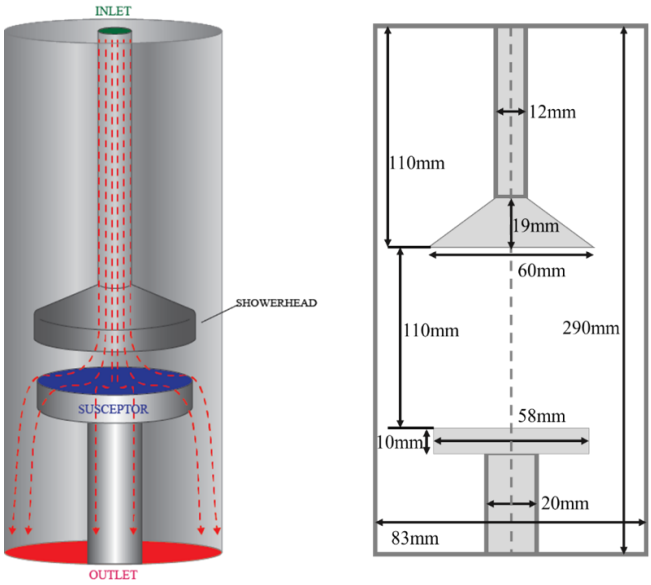

Figure 17. A schematic of the experimental CVD reactor and its relevant dimensions.

In summary, as the gas mixture of the reactants (precursor and carrier gases) enters the reactor from the top it is directed through a showerhead that spreads it evenly in the area over the susceptor, ensuring uniform flow in the volume of the reactor, thus favoring the final uniformity of the films. The product, aluminum film, is deposited on the surface of a substrate, in this case Silicon, placed on top of the susceptor in the form of small tablets. The gaseous by-products of the process, along with the carrier gases and the precursor excess, are then removed from the bottom of the reactor. This process setup and the experiments are described in more detail in previous studies [10] [34], therefore only the important information for the purpose of the present work is mentioned.

7.4. FVM Discretization

The discretization of the reactor model is created with Gambit and considers the real dimensions of the reactor, including the showerhead holes which are displayed in Figure 18. [8] [28] [33]

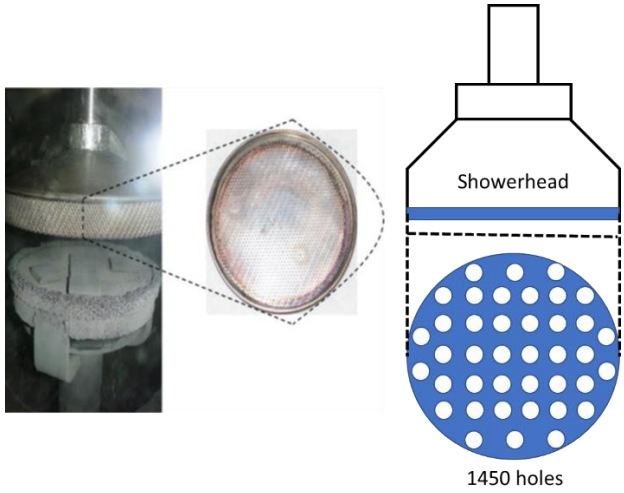

Figure 18: Left: Experimental reactor photographs, Right: Schematic of the showerhead holes.

Due to its cylindrical shape the reactor has axial symmetry that allows the use of a 2D axisymmetric computational grid for the simulation of the process, without significant loss in the accuracy of calculations and with a significantly lower computational cost. For that reason, the CFD models developed for this reactor used axisymmetric geometry, which limits the showerhead holes to only be represented by a single or multiple concentric annular holes, thus disregarding the importance of the showerhead to the uniform distribution of the incoming flow

of reactants over the susceptor. This limits the 2D model to its use only for preliminary studies, specifically testing different parameter setups to find a suitable one. Nevertheless, it is often the case that even in truly axisymmetric geometries a 3D model will have results closer to the experimental ones, thus in this work a 3D model is used, which will also provide computational insight regarding the effects of the showerhead to the deposition rate. The computational grid used for modeling the process with ANSYS Fluent 17 is shown in Figure 19, it is unstructured and uses cells with a pyramid shape. The grid consists of 1232468 cells, a grid density that emerged from a grid independence study as the optimal one.

Figure 19: Discretized computational grid of 1232468 FVM cells.

7.5. Chemical reaction system and conditions

The possible chemical pathways of the aluminum CVD and especially the DMEAA decomposition have been thoroughly studied in previous works [28] [35] [49] by comparing the deposition rate of aluminum films in computational results with experimental findings. The

latest kinetic model was developed in [10] and is used in the present work. This model consists of three gas phase reactions in the volume of the reactor and nine surface reactions on the substrate.

7.5.1. Volumetric reactions

First, the volumetric reactions include the decomposition of DMEAA into DMEA and aluminum hydride (AlH_3), a reaction that has a low activation energy and it is easily activated even at low temperatures. The other two are oligomerization reactions of the metal hydride AlH_3, a known phenomenon for this type of compounds at high temperatures that has been observed experimentally. These reactions are inhibitory and reduce the mass of the active intermediate of the process AlH_3 that reaches the substrate surface to produce solid aluminum films. The kinetic constants of the oligomerization reactions that consist of a dimerization (G2) and a trimerization (G3) of aluminum hydride, along with the decomposition of DMEAA (G1) are summarized in Table 1.

Table 1: Volumetric reactions and their kinetic constants.

A/A	***Reaction***	k_0	E_a (kJ mol^{-1})
G1	DMEAA(g) → DMEA(g)+AlH_3(g)	$2.00 \cdot 10^7$ s^{-1}	40.06
G2	2AlH_3(g) → Al_2H_6(g)	$2.55 \cdot 10^{20}$ m^3 kmol^{-1} s^{-1}	118.00
G3	Al_2H_6(g)+AlH_3(g) → Al_3H_9(g)	$7.75 \cdot 10^{20}$ m^3 kmol^{-1} s^{-1}	90.70

The rates of the three gas phase reactions are calculated by the following expression:

$$r_{G_i} = k_{0,G_i} \exp\left(-\frac{E_{a,G_i}}{RT_g}\right) \prod_{j=1}^{N_g} C_j^n$$

where G_i is the index of the gas phase reaction i, k_{0,G_i} the pre-exponential factor, E_{a,G_i} the activation energy, T_g the temperature in the gas phase, C_j the concentration of component j, N_g the number of gaseous components and n the reaction order.

7.5.2. Surface reactions

There are nine surface reactions described in [10], with four of them being adsorptions and desorptions (S1-S2, S8-S9) and the rest constructing the decomposition pathway of aluminum hydride to pure deposited aluminum (S3-S7). All the surface reactions are summarized in Table 2. Among them, the reactions S1, S3, S5, S7 and S8 are the ones that that enhance the deposition of aluminum while the others act as inhibitory reactions producing by-products.

Table 2: Surface reactions and their kinetic constants.

A/A	*Reaction*	k_0	E_a (kJ mol^{-1})
S1	$AlH_3(g)+3S \rightarrow AlH_3(ads)$	$4.42\cdot10^{8}$ m s^{-1}	19.68
S2	$AlH_3(ads) \rightarrow AlH_3(g)+3S$	$2.38\cdot10^{18}$ m s^{-1}	86.84
S3	$AlH_3(ads) \rightarrow AlH_2(ads)+H(ads)$	$1.02\cdot10^{12}$ m s^{-1}	19.25
S4	$AlH_2(ads)+H(ads) \rightarrow AlH_3(ads)$	$5.11\cdot10^{12}$ m^4 kmol^{-1} s^{-1}	18.41
S5	$AlH_2(ads) \rightarrow AlH(ads)+H(ads)$	$4.98\cdot10^{12}$ m s^{-1}	37.43
S6	$AlH(ads)+H(ads) \rightarrow AlH_2(ads)$	$1.05\cdot10^{14}$ m^4 kmol^{-1} s^{-1}	8.37
S7	$AlH(ads) \rightarrow Al(s)+H(ads)$	$2.93\cdot10^{10}$ m s^{-1}	76.82
S8	$2H(ads) \rightarrow H_2(g)+2S$	$5.50\cdot10^{16}$ m^4 kmol^{-1} s^{-1}	12.55
S9	$H_2(g)+2S \rightarrow 2H(ads)$	$3.07\cdot10^{10}$ m s^{-1}	76.97

The chemistry model initially shows the adsorption and desorption of AlH_3 on the substrate (S1-S2), that occupies three surface positions (S), which is derived from the literature based on previous stereochemistry studies. Also, the same occurs for the molecular hydrogen (S8-S9), which is formed by hydrogen atoms that are produced during the decomposition reactions of AlH_3 and takes up two surface positions on the substrate. After the AlH_3 is adsorbed on the substrate surface it is decomposed into lower order aluminohydrides down to pure aluminum. This decomposition pathway follows successive dehydrogenations, of which the first two are reversible (S3-S4 and S5-S6) and the last that leads to pure aluminum is non-reversible (S7). The two reversible reactions can be considered partially as competing phenomena, as they prevent the aluminum hydrate to reach the last step of dehydrogenation (S7), which as already mentioned is a non-reversible reaction and produces the final product. Finally, it is worth mentioning that the activation energies of the surface reactions show that the adsorption of hydrogen and the desorption of AlH_3, which are inhibitory reactions, are activated in higher temperatures than the opposite ones that favor the deposition of aluminum.

The rates of the surface reactions are given by the following form:

$$r_{S_i} = k_{0,S_i} \exp\left(-\frac{E_{a,S_i}}{RT_S}\right) \prod_{j=1}^{N_s} C_j^n$$

7-3

where S_i the index of the i surface reaction, T_S the temperature on the substrate and N_s the number of adsorbed components. It is important to mention that in this case the C_j is the concentration on the surface, not in the volume of the reactor and it is given by:

$$C_j = \theta_j * d$$

7-4

where d is the density of the active sites on the substrate in kmol/m^3 and θ the surface coverage of each chemical component, which is dimensionless and given by:

$$\theta_j = \frac{N_j}{N_{tot}}$$

7-5

where N_j the number of occupied active sites on the substrate by the component j and N_{tot} is the total number of active sites.

In summary this kinetic model is complex enough to capture the deposition rate of aluminum across the spectrum of substrate surface temperatures, since it includes competitive phenomena that are activated in high temperatures that create the reduction of deposition rate at these conditions. This phenomenon has also been measured experimentally by [28] [33] [34]and it is visible in the Arrhenius plot of the natural logarithm of deposition rate against the reverse substrate temperature Figure 20.

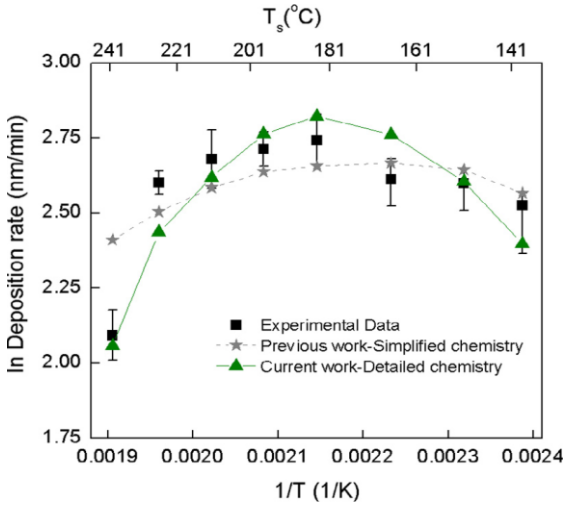

Figure 20: Arrhenius plot of experimental findings and computational data of the 2D CFD model by [10].

7.5.3. Boundary conditions

The boundary conditions of the macroscopic model are determined mainly by the operating conditions of the real process; in particular, the boundary conditions of the problem include the subgroups of inlet, outlet and wall boundary conditions. First, the inlet boundary conditions describe the flow state at the reactor inlet and can be seen in Table 3. [10] [28]

Table 3: Inlet boundary conditions.

Inlet boundary conditions	***Value***
Temperature T_{in}	100°C
Pressure P_{in}	1333 Pa (10 Torr)
Mixture mass flow rate $\dot{m}_{in}$	$6.403 \cdot 10^{-6}$ to $6.438 \cdot 10^{-6}$ kg/s
DMEAA mass fraction $y_{DMEAA,in}$	0.0167 to 0.0223
N_2 mass fraction $y_{N2,in}$	1 - $y_{DMEAA,in}$

Regarding the mass flow rate and proportion of DMEAA in the gas mixture, these are changed during the study, since the focus of this work is on the dynamic modeling of the process with the controlled variable being the precursor inlet flow. However, it is noted that the changes depend on the range of the amount of DMEAA, that can be introduced into the reactor given the amount of nitrogen that remains constant and is 330sccm. This corresponds to a maximum possible DMEAA of 2sccm calculated from the following equation:

$$Q_{DMEAA} = Q_{N_2} \frac{P_{sat}(T_{DMEAA})}{P_{reactor} - P_{sat}(T_{DMEAA})}$$

7-6

where Q_{DMEAA} and Q_{N2} the volumetric flow rates of D_{MEAA} and nitrogen respectively, P_{sat} the saturation pressure of DMEAA for given temperature T_{DMEAA} and Preactor the operating pressure of the reactor. Moreover, considering that for values of Q_{DMEAA} less than 1.5sccm the deposition process has no production value, the DMEAA flow rate is limited from 1.5 to 2sccm, thus resulting to the values of $\dot{m}_{in}$ and y_{in}. Furthermore, the outflow boundary condition is applied to the outlet and no-slip condition is applied to all reactor walls. The temperature of the substrate (set on the susceptor boundary) is set in the range of T_s=139 to 241°C, this variation of temperature is used just to compare the reliability of the models across the temperature range that has been studied experimentally. For the rest of the wall boundaries the temperature is set to T_{wall}=75°C and the operating pressure of the reactor is equal to $P_{reactor}$ = 1333Pa. [10] [28]

7.6. Design of POD basis

The construction of the basis is achieved with the POD method, more specifically with the variation MoS and it is carried out exclusively through Matlab. After the creation of the snapshot matrix **Y**, it is normalized using the equations 5-31, 5-32 and 5-33. Then the correlation matrix **K'**, its eigenvalues and eigenvectors are calculated (eq. 5-18). Finally, from the equation 5-19 the eigenvectors of the correlation matrix **K** are calculated**,** which form the orthonormal POD basis **Z** (eq. 5-22). The dimension of the basis is decided based on the simultaneous comparison of the informational content I(d) values, which are calculated by eq. 5-29, and the errors by eq. 5-34, 5-35 and 5-36. The approximated states/snapshots that are used in the error equations by eq. 5-26 and the corresponding coefficients by eq. 5-27.

During the POD method, the two most critical parameters that need to be thoroughly investigated by the user are:

1. The necessary number of snapshots in the Y matrix

2. The necessary number of main POD vectors, i.e., the dimension of the Z basis

Both criteria are considered and more specifically, scenarios independent of each other are studied regarding criterion (1), by varying the number of step changes that make up **Y**. Also, the criterion (2) is investigated in each scenario, by comparing the percentage of information as well as the errors of the different POD basis that arise for the various amounts of used eigenvectors. In the end, a total comparison of the different cases is made, to decide which basis is the best performing. Nevertheless, it should be noted that to avoid repetition and redundancy only the final selected scenario for the creation of the ROM is analyzed. Furthermore, the study is limited to the precursor supply range of DMEAA=1.5 – 2.0 sccm, since for reasons mentioned in section 7.5.3 this is the realistic range used in the experimental reactor setup. Finally, the overall procedure was carried out for three different substrate temperatures 151, 185 and 241 °C but results are presented only for the first one, as the comparison study is very much identical between these temperatures.

7.7. Data collection – Step changes

The data used to construct the ROM are the transient solution results of the CFD model of the reactor studied (see 3.2). The solution of the model is carried out in a coarse

computational grid, in order to reduce the computational cost of the data mining. The coarse grid is as coarse as it can be without losing the main characteristics and properties of the system, although its results carry a significant error compared to the detailed fine grid. The coarse grid consists of 309155 cells, so the state vector calculated each time contains 1545775 elements, which correspond to the values of the components x, y and z of the velocity, the pressure and finally the temperature, for each cell of the grid. It should be noted that the solutions resulting from the coarse grid are not real reactor situations, due to the high error they carry. Indicatively, the dense grid consists of 1232468 cells and the corresponding state vector of 6162340 elements. Also, an important aspect of this implementation is that the model does not consider chemical reactions neither in gas phase nor on the substrate surface, which greatly reduces the computational cost of the data collection process. This is made possible, as already mentioned, by the fact that the precursor gas mixture is so diluted that the concentration changes do not affect the flow field.

First, steady states of the reactor are calculated for various values of the DMEAA inlet flow rate, which is also the input variable of the dynamic input-output system analyzed in section 5.2. These steady states are then used as initial states for the simulations of the model after applying step changes on the DMEAA supply, within the range of 1.5 to 2 sccm, which was selected based on the realistic values used in the actual experimental reactor. The output variable of the dynamic system is the state vector $\boldsymbol{x}(t)$ of the reactor after each time step Δt=0.1s of the step changes, which is the snapshot of the reactor at that point in time. The set of all snapshots for a step change transient calculation of the CFD model is the reactor's trajectory in time for this corresponding step change, which is practically the dynamic response of the system $\boldsymbol{Y_i} = \left[\boldsymbol{x}(t_0) \quad \boldsymbol{x}(t_1) \quad \cdots \quad \boldsymbol{x}\left(t_{m_i-1}\right)\right]$**.** During this dynamic study of the reactor, the step changes are imposed through the Matlab programming environment, which externally used ANSYS Fluent for the simulations. For each time step k, the initial condition is the state of the previous step $\boldsymbol{x}(t_{k-1})$, while at each time step 200 repetitions of the finite volume method are performed with Fluent (see 3.3.2). The dynamic calculation begins at time, $t_0 = 0\ s$ while the snapshot $\boldsymbol{x}(t_0)$ corresponds to the steady state used as the initial condition for the step change.

The convergence of the solution to the new steady state of the step change is also controlled through Matlab, by calculating the norm of the difference of two consecutive state vectors (snapshots), which must be less than a predetermined by the user tolerance, which in this case is 10^{-6}, namely:

$$\frac{\|\boldsymbol{x}(t_{k-1}) - \boldsymbol{x}(t_k)\|}{\|\boldsymbol{x}(t_k)\|} < 10^{-6}$$

7-7

The tolerance of 10^{-6} indicates convergence (temporally) in the new steady state. To confirm this, the state vectors at the end of a step change are compared with the steady state vectors originally calculated for various values of the DMEAA flowrate. If the relative error of the two states is low, then the reactor state has converged to the new steady state and the assumption of this tolerance is therefore acceptable. Finally, the reactor trajectories through the step changes are extracted from Fluent and are combined into the total snapshot matrix Y described in detail in section 5.3.2. This matrix contains all the snapshots of all step changes and is the input data of POD.

Initially, four step changes are imposed on the DMEAA inlet flow rate and the data collected construct the snapshot matrix **Y**. These step changes were chosen to fully cover the range of DMEAA flow rates, to include small but also large changes and finally to contain changes in both "directions" (increasing and decreasing flow rate). The four step changes used are listed in Table 4:

Table 4: DMEAA mass flow rate step changes.

Initial flow rate (sccm)	***Target flow rate (sccm)***
1.5	2.0
1.9	1.5
1.5	1.8
1.8	2.0

This study is limited to 4 different bases of one to four main POD vectors, as the use of more eigenvectors is excessive. In Table 5 the information contents I(d) of each basis are shown, where d the dimension of the POD basis Z, i.e., number of POD vectors. The indicator of Z in the

table below shows this dimension. It is recalled that like the state vectors, each column (vector) of the basis contains 1545775 elements (see 7.7).

Table 5: Information contents of each POD basis.

POD basis of dimensionality d	***Information content I(d)***
Z1 (d=1)	0.4747
Z2 (d=2)	0.7846
Z3 (d=3)	0.8915
Z4 (d=4)	0.9313

All four bases contain a relatively low information content compared to usual standards of POD. More specifically, the basis $\mathbf{Z_1}$ does not retain even half of the information of the snapshot matrix **Y**, while with the addition of another eigenvector the percentage increases significantly. Further additions of eigenvectors bring smaller and smaller increases, which is to be expected as it is mentioned in section 5, the first eigenvalues are bigger than the next ones, while their respective eigenvectors contain larger percentage of information.

The accuracy of each basis is checked by calculating the errors of the snapshots of a step change, which is approximated by the under-testing basis and the coefficients resulting from it, with the actual step change calculated from the detailed model during the data collection process. The step change checked is the 1.5 to 2.0sccm. This step change will be referred to from now on as **internal**, as it is included in matrix **Y**. The following Figure 21 shows the errors of each basis for the tested step change trajectory against time.

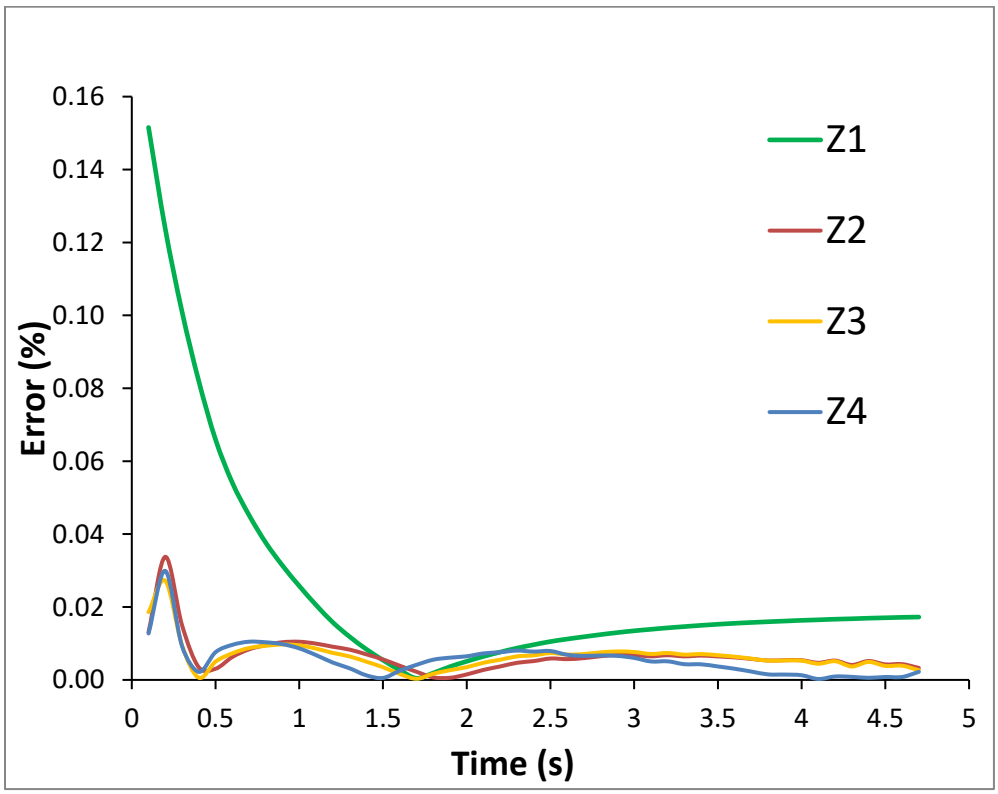

Figure 21: Evolution of error for POD basis of different dimensionalities, along the data of the internal step change trajectory.

From the previous plot, it is observed that the bases of two to four eigenvectors show much smaller errors compared to that of one eigenvector, a conclusion that is expected. Although, the overall errors seem negligible, for the shake of transparency it should be noted that an increase in the dimension from 2 to 3 and from 3 to 4 does not appear to cause a particular reduction in error, except for $\mathbf{Z_4}$ where there is a small decrease in the average error compared to the smaller bases. However, this decrease is mainly due to errors as the trajectory reaches to the new steady state (Figure 21), where the other bases have insignificant errors too. Therefore, the best is the Z_2 basis, i.e., of two main POD vectors, as a further increase in the dimension does not seem to be justified by the corresponding reductions in errors.

Furthermore, in most applications even the error of $\mathbf{Z_1}$ could be considered acceptable from a practical point of view, as it is an insignificant error. These small errors are probably due to the small range of DMEAA flow rate studied, within which there are no significant changes in the dynamic behavior of the reactor. However, because the purpose of this thesis is not simply the application of the proposed computational framework to this specific system but its general

development and evaluation, it is preferred to choose the best design parameters (for example in this case the ideal dimension of the basis) based on the comparison between them. Therefore, it was decided to use the $\mathbf{Z_2}$ basis, which is constructed by data of 4 step changes, as it combines the smallest errors with simultaneously a small dimension.

7.8. Design, training and simulation of ANN

The calculation of the ROM coefficients is carried out through simulation of ANN and specifically of NARX networks, after training them with the data of the dynamic response of the reactor to step changes on the inlet supply of DMEAA. These procedures are carried out exclusively through Matlab and its respective NARX library. During the training, input vectors are fed that include the step change on the value of DMEAA supply, while exit vectors are also fed, which include coefficient values for each snapshot of each step change and for each eigenvector of the basis, respectively. These coefficients, on a given basis, are calculated by eq. 5-27. The architectural configuration of the ANN followed during the training is the series-parallel (see 5.4.8).

During the simulation of the network, it is in parallel configuration, where it is fed with the DMEAA supply step changes as well as with the initial state, i.e., the coefficients corresponding to the initial steady state of the step change the user wants to simulate. The network then calculates the coefficients of each snapshot for each time step of the step change trajectory. These snapshots are ultimately multiplied by the POD basis according to eq. 5-26, thus giving the overall dynamic response of the ROM for this specific step change.

The most important parameters that need to be thoroughly studied when designing a NARX network are the following:

1. Number of hidden layers
2. Number of input and output delays
3. Training algorithm
4. Training data set
5. Dimension of hidden layers = number of neurons

In the network structures used in this application, a single hidden layer was used, since according to the literature, problems of this scale rarely require more. Also, the number of delays, or else the memory of the network, was chosen to be equal to 1 for both input and output (i.e.), $n_x = n_y = 1$, which is consistent with fluid dynamics problems. Thus, eq. 5-64 takes the following form for the network:

$$\boldsymbol{y}(t_k) = f[\boldsymbol{x}(t_{k-1}), \boldsymbol{x}(t_k), \boldsymbol{y}(t_{k-1})]$$

7-8

As far as the training algorithm is concerned, the Bayesian Regularization-Backpropagation is applied, i.e., the Levenberg-Marquardt algorithm with Bayesian Normalization (see 5.4.7). The reason for this method is that for a large number of snapshots, which is the case of the present problem, the Levenberg-Marquardt algorithm alone is unable to generalize the results successfully, a phenomenon called network overfitting (see 5.4.6). In addition, the network is trained based on 7 step changes, so that the entire range of DMEAA flow is studied. These contain small and large changes as well as changes in both directions. The training step changes are listed in Table 6.

Table 6: Step changes for the NARX network training.

Initial flow rate (sccm)	***Target flow rate (sccm)***
1.5	2.0
1.9	1.5
1.5	1.8
1.8	2.0
1.9	2.0
1.9	1.7
1.7	1.5

The first 4 are those that were also used for the construction of snapshot matrix **Y** when creating the POD basis, but 3 additional step changes are available for the ANN training, when it is deemed necessary. Therefore, the last ANN parameter to be determined is the dimension of the hidden layer, meaning the optimal number of neurons within it. Specifically, networks with 3, 5 and 10 neurons are trained, which are then led to a simulation of test step changes, in the same concept mentioned during the study of the POD basis, but in this case both an "internal" and an "external" one are tested. The ANN predictions are compared with the computed step changes from the detailed CFD model, and the corresponding errors are calculated. The two control/test step changes used to study the accuracy of the ANN are the internal 1.5 to 2.0sccm and the external 1.5 to 1.7sccm. The following plots show the errors of each network for the two test step changes against time.

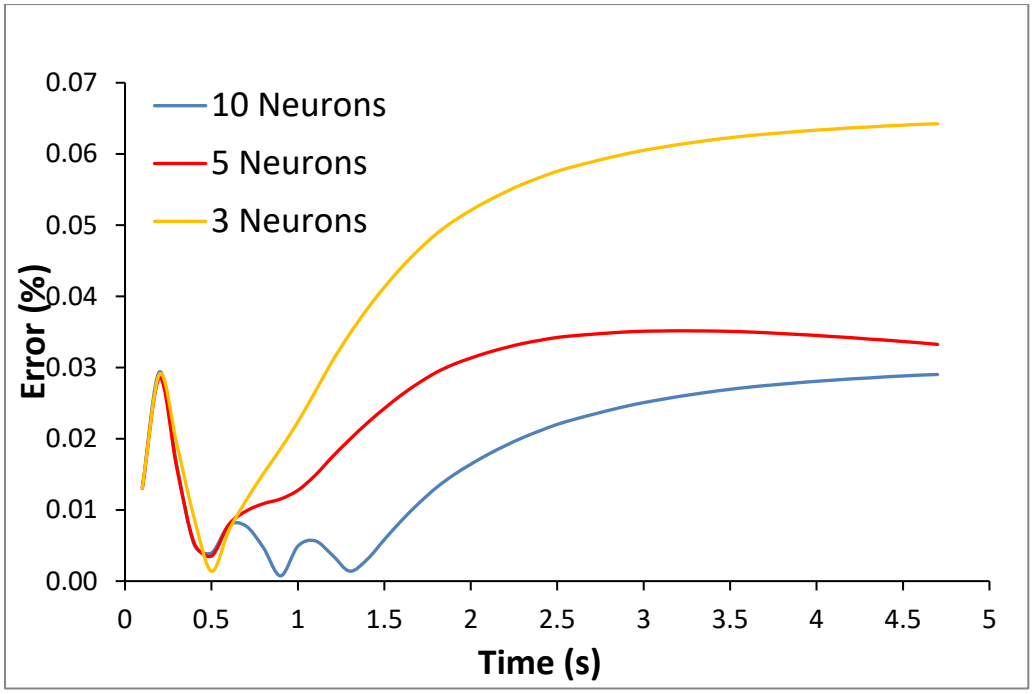

Figure 22: Prediction error of ROM along the data of the internal step change trajectory, for different number of neurons.

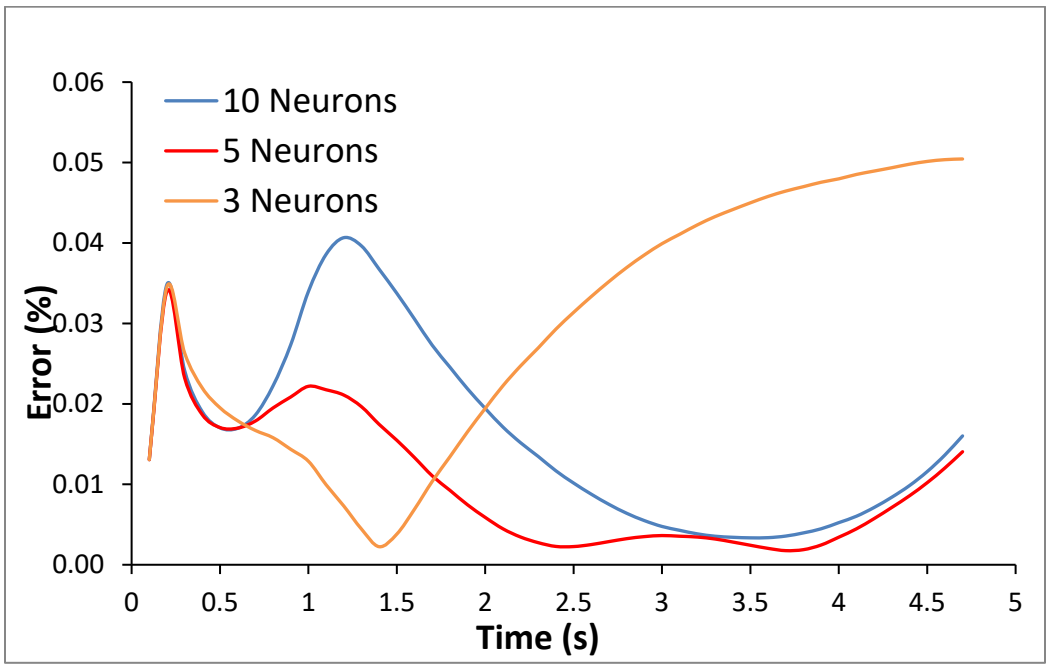

Figure 23: Prediction error of ROM along the data of the external step change trajectory, for different number of neurons.

From the analysis of the errors with the internal step change, it is observed that the networks of 5 and 10 neurons show much better training results than that of the 3 neurons, with their relative difference being almost three times. In fact, from the plots it appears that over time the error of the 3 neurons increases, while the errors of the other two networks seem to stabilize. Therefore, the network of 3 neurons is not capable of being trained as well as the other two and it is rejected from the outset. As for the remaining two networks, although the one with the 10 neurons has a significantly smaller average error than the equivalent of 5, the error of the latter can be acceptable, especially considering that its training takes almost half the time. Also, the difference is mainly visible at the middle of the step change, while at the end where the system converges to the new steady state both errors seem to be stable around the same value.

As far as the external step change is concerned, the verification of the previous rejection of the 3 neurons network is initially observed, as especially towards the end of the step change its errors are much larger than those of the other two networks. However, in this test, unlike the internal one, the network of 5 neurons not only achieves satisfactory prediction but even marginally better than that of the 10 neurons. This is probably due to the phenomenon of overfitting, where the network of 10 neurons is focused too much on training data and cannot

achieve as good generalization as that of 5 neurons. Therefore, the 10 neurons are not only judged as unnecessary, but even as damaging to the accuracy of further NARX network predictions, and therefore the optimal dimension of the hidden layer is determined to be 5 neurons. This network is depicted in the following Figure 24, in series-parallel configuration during its training and in parallel configuration during its simulation.

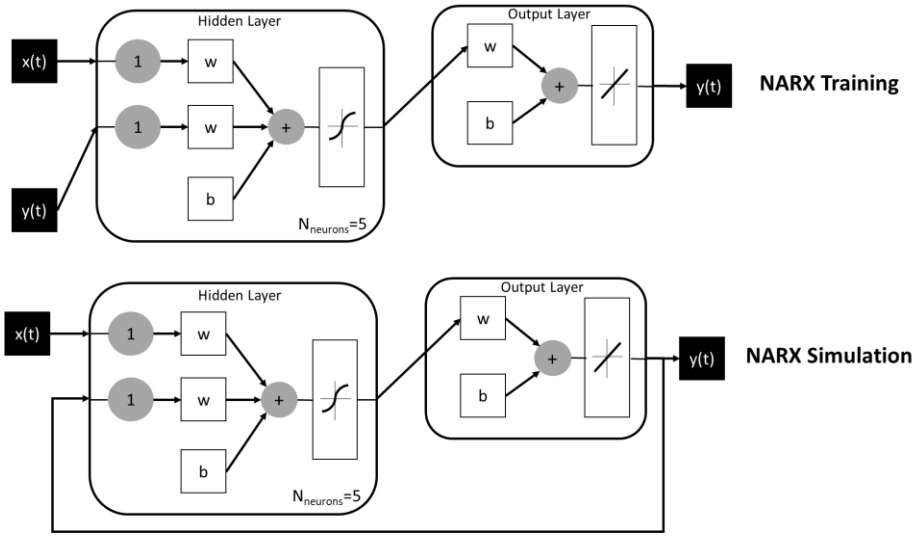

Figure 24: NARX network of the final ROM.

All the characteristics of the network can be seen in Figure 24, as they were determined in the previously mentioned tests. The output of this network is the time dependent coefficients needed in the equation 5-6 along the POD basis and give the final ROM prediction of the reactor state. The same procedure was applied for the other two substrate temperatures, where the conclusions are the same and to avoid repetition are not presented. Finally, it is important to recall that, because the data set used for the creation of POD basis and the training of ANN is collected from CFD calculations on the coarse grid model without reactions, the ROM prediction lacks information on the chemistry of the process and needs to be interpolated into the fine grid model.

7.9. Grid-To-Grid interpolation of ROM prediction

After the ROM is completed, able to predict rapidly the flow fields within the reactor, the predictions/approximations of this model are fed into the detailed fine grid model, which also includes the kinetic equations of chemical reactions (see 3.2.2 and 7.5) they are used as an initial estimate resulting in a faster convergence of the detailed CFD model. The latter is solved

in a fine grid, which as mentioned consists of 1232468 cells, while the corresponding state vector carries 6162340 elements, i.e., it is a grid about 4 times denser than the coarse one. However, to transfer the ROM predictions in the fine grid, a grid-to-grid interpolation (nearest neighbour) is used, which was described in detail in section 4.2. From the solution of the detailed model, the present application investigates the accuracy of the chemical model developed in the context of previous works. This process was of course carried out in the past, but only in a 2D grid, which is unable to include the holes of the showerhead in reactor chamber (see 7.3 and 7.4). The determination of the reliability of the kinetic model is achieved by comparing the deposition rates for various substrate temperatures with the corresponding experimental findings, as well as with those of the 2-D model from the literature.

7.10. Evaluation of chemical reaction system

As mentioned in section 7.1, the aim of the work is also to evaluate a complex kinetic model of aluminum deposition from DMEAA that had been proposed and investigated with satisfactory results in the past, in the context of a previous work [10]. However, its development was carried out exclusively on a 2D grid, which is the usual strategy for a preliminary study. This model includes, in addition to deposition reactions and inhibitory reactions, both in the gas phase and on the surface of the substrate, unlike older much more simplistic models. Therefore, it can predict the sharp decrease observed in the deposition rate at high substrate temperatures, which simple models cannot. More details are given in section 7.5.

The evaluation and reliability check of this kinetic model in a 3D grid is carried out efficiently based on the proposed framework, thus saving valuable computational time. Thus, the effect of temperature on the average deposition rate is studied. Furthermore, based on the proposed framework three different ROMs are constructed, one for each of the substrate temperatures of 151, 185 and 241 °C, the predictions of which were fed to the detailed model through interpolation to be used as an initial estimate of the flow field, resulting in a much faster convergence of the overall model (see 7.11). These three temperatures are selected to capture the reaction and diffusion limited regimes, without studying the whole temperature range. Finally, it is reported that the solution at each temperature is based on an inlet flow rate of DMEAA equal to 1.85sccm, as the corresponding experiments have been carried out on this value. The steady state of the reactor for this flow rate is predicted by the ROM simulating a step change of 1.7 to 1.85sccm in 50 timesteps. This step change is unknown to the ROM, and it has not been calculated in the past in any other way. Thus, with the error between the last 2

snapshots being less than 10^{-9} it is considered that the ROM has converged to the final steady state of the reactor for 1.85sccm.

The following Arrhenius plot Figure 25, shows the evolution of the natural logarithm of the average deposition rate (y-axis) in relation to the inverse of substrate temperature (x-axis), based on the 1.85sccm DMEAA flow rate. The new results of the 3D model agree well with both the experimental data and the results of the 2D model. This shows that the ROM, created by the proposed framework, can produce predictions that can capture the detailed, chemistry included, model.

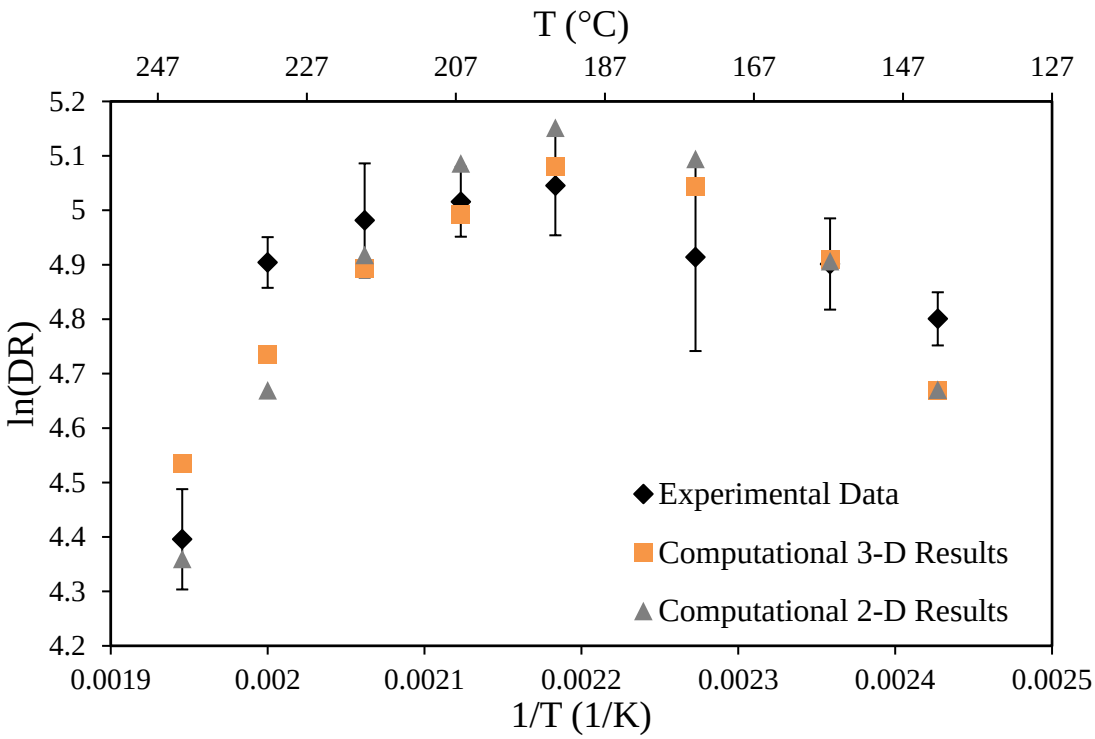

Figure 25: Arrhenius plot of the natural logarithm of the deposition rate against the reverse substrate temperature. Comparison between 3D and 2D models with experimental data.

7.11. Computational cost reduction

The proposed computational framework is not only limited to the order reduction of the detailed 3D large-scale CFD models, which nevertheless forms its core, but also extends to the use of other secondary techniques, the parallel processing, and data mining as well as the grid-

to-grid interpolation. The previous sections evaluated the accuracy of the order reduction methods and the final ROM in general, while in this section the computational benefit of the whole process is evaluated, based on all the techniques. This assessment is performed by comparing the computational costs in time and memory for different cases.

Computational cost of CFD model simulation (without reactions):

First, the benefit of using the ROM and the general framework for determining the reactor's state vector, which consists of the velocity, pressure and temperature fields, is determined. The comparison between serial and parallel computations is also mentioned. The computational time needed for each situation is in core hours, which is core hours = (CPU h in real time) x (number of CPU cores).

- **Detailed model (fine grid) – Serial simulation:** 58.5 core hours for dynamic response to a step change / 36 core hours for steady state calculation – 5 GB RAM

- **Detailed model (fine grid) – Parallel simulation (12 cores):** 78 core hours for dynamic response to a step change / 48 core hours for steady state calculation – 5 GB RAM

- **Reduced order model (coarse grid):** 2 core minutes for predicting the dynamic response and the steady state, +15 core minutes for interpolation of the prediction in the fine grid and its optimization in the detailed model – 2 GB RAM for the ROM prediction and 5 GB RAM to simulate the detailed model in the fine grid

It is obvious that there is a significant reduction in the computational real time required between the serial and parallel simulation, which makes the technique of parallel processing and data mining in parallel an integral part of the framework. However, there is also a huge reduction between the ROM simulation and the subsequent interpolation and optimization of the prediction in the fine grid, compared to the direct solution of the detailed model. S**pecifically, the acceleration achieved by using the ROM predictions is about 312 times (based on core hours), which corresponds to a 96% reduction of wall clock-time**. It should be noted that the coarse grid is not capable of providing precise solutions, so any solution must be taken from the fine grid.

Computational cost of complete detailed CFD model simulation (with reactions):

Furthermore, the benefit of using the ROM and the general framework for the solution of the kinetic model, which results to the average deposition rates of the produced films, is determined. The computational costs below are based on the detailed model with chemistry involved in the fine grid. Regarding the time needed for the ROM prediction, it is the same as above, so it is included in the following value.

- **Detailed model (fine grid + reactions) – Parallel simulation:** 90 core hours for steady state calculation – 5 GB RAM

- **Reduced order model (coarse grid without reactions)**: 36 core hours for steady state calculation, this includes the time needed for the ROM prediction, the interpolation and the simulation of the detailed model with reactions, while using the ROM prediction as estimate – 5 GB RAM

Thus, there is again a significant reduction in the required computational cost if the ROM prediction is used by the complete detailed model as an initial estimate. In this way, the detailed model needs to perform a small number of iterations to optimize the flow field, while beyond that it only has to simulate the kinetics of chemical reactions. **In particular, this is an acceleration of 2.5 times (based on core hours), which is a 60% reduction of wall clock-time.** Finally, it is important to note that in the case of CFD calculations with chemistry involved, the ROM simply acts as an accelerator. Therefore, the fact that the prediction is derived from coarse grid data it does not affect the precision of the final solution, because it is exclusively computed from the detailed CFD model, with the fine grid and reaction kinetics.

Computational cost of ROM construction:

Finally, the cost of building the ROM should be addressed, so that the value of the secondary techniques of the framework become apparent.

- **Based on the total computational framework (12 cores):** 48 core hours – 1.5 GB RAM for data collection and 6 core hours – 4 GB RAM (max) for POD+ANN

- **Without coarse grid and grid-to-grid interpolation:** 450 core hours – 5 GB RAM for data collection and 18 core hours – 12 GB RAM (max) for POD+ANN

- **Without parallel processing – UDF programming:** 4050 core hours – 5 GB RAM for data collection and 18 core hours – 12 GB RAM (max) for POD+ANN

The value of incorporating these techniques into the framework of model order reduction becomes obvious, as they significantly accelerate the solution of the CFD model and the data collection. Between the two techniques, interpolation and parallel processing, the latter is practically necessary, while the grid-to-grid interpolation further enhances the efficiency of the proposed framework.

7.12. Conclusions

The computational framework proposed in this dissertation was first applied on this case study of aluminum CVD. A ROM capable of predicting accurately the solution of the complete, detailed model was created, while at the same time the computational cost for the development of the ROM and for its simulation is very low. Regarding the first component of the framework, the construction of the POD basis, it has been determined that, for this case study, snapshots of trajectories from only 4 step changes, on the DMEAA inlet flow rate, are necessary. By ensuring the choice of the parameter values is representative of the appropriate window of operation of the reactor, it follows that only two POD vectors are capable of recreating, with small errors, the reactor states in the range of the data set. A similar investigation was carried out during the design of the ANNs, which calculate the time coefficients that are multiplied by the POD basis to obtain the final ROM. This investigation was based on several design criteria that had to be determined for the most efficient model order reduction. First, it was found that data from 7 step changes are sufficient for the training of the NARX network, provided that these extend over the entire range of DMEAA flow rate. In addition, the following have been determined:

- A single hidden layer is used and is more than sufficient for the satisfactory computation of time coefficients

- The Bayesian Regularization-Backpropagation method shows remarkable results in the training of the network

- The input and output delay are equal to 1 and is enough to give the network the required time depth/memory

- For the studied range of DMEAA flow rate, five neurons are not only sufficient but even have a smaller error compared to a larger number (10), probably due to the overfitting of the network to the training data in the latter case.

Finally, based on the above design parameters of the POD basis and the NARX ANN, the dynamic response of the reactor to an external step change that was not used at any point in the computational framework is calculated with extremely small errors.

In a second phase, the computational framework was directly applied for the primary evaluation of a complex kinetic model of aluminum CVD from DMEAA, which had been developed in the context of a previous dissertation. As the values of the kinetic parameters of this model had been determined by adapting experimental results to a 2D grid model solution, it was considered necessary to compare the experimental values with results from a 3D grid. For this purpose, three ROMs were constructed for three different substrate temperatures, which were selected in order to capture the reaction and diffusion limited regimes. The average deposition rate, for each temperature in the range set by the experiments, was calculated in a very short time with the use of ROM. As expected, a small mismatch is observed between the values of the 3D and 2D models and the experimental findings. However, the differences are within very reasonable limits, and it is assumed that the previously created kinetic model sufficiently captures the experiments and no further action is needed in this case.

Finally, the most important result of this case study is based on the computational benefits of the framework. There is a 96% reduction in the wall clock-time (x312 acceleration in core hours) required to simulate the flow field of the reactor, while there is also a 60% reduction (x2.5 acceleration in core hours) for the solution of the fine grid, chemistry included detailed model, which ultimately is the deposition rate, the uniformity of the films, etc. In addition to order reduction, the secondary techniques used, i.e., the parallel programming and solution grid-to-grid interpolation, contribute to a great extent. The latter even allows the use of practically useless results, which are obtained quickly, to determine a high-precision and high-fidelity solution. In addition, these two techniques have significantly accelerated the creation of the ROM, which would otherwise require a lot of CPU time and RAM for the data collection.

8. Case study 2: Copper CVD

8.1. Introduction

The CVD of copper Cu from the precursor compound Copper(I) N,N'-diisopropylacetamidinate (known as Copper Amidinate or $[Cu(amd)]_2$ where amd is $CH(CH_3)_2NC(CH_3)NCH(CH_3)_2$) is the subject of the following application of the thesis proposed computational framework. This case study, at first, will simulate the experimental findings collected from previous work[1], by utilizing the ANSYS Fluent CFD code. As it will be analyzed in upcoming chapters, the experimental data [41] in question show a decrease in the deposition rate of copper above a certain substrate temperature. From the previous work on this case, this decrease was explained with a variety of reasons, such as buoyancy or even surface site deactivation from the high temperature, reversible deposition reactions etc. Thus, the previous studies required two different kinetic mechanisms to describe the process and failed to produce a unified kinetic model, which can simulate the relationship between the competing phenomena that result into the two different deposition regimes. Therefore, the ultimate purpose is the fitting of a new unified kinetic model of the copper CVD across the entire experimental temperature range. In this application the proposed framework is applied in order to create a Reduced Order Model (ROM), which is used as a tool that reduces the convergence time in the CFD code, improving the efficiency and effectiveness of the fitting of new chemical reactions on the experimental data.

8.2. Precursor compounds

The Copper Amidinate is a precursor used to create think films of solid elemental copper and is the focus of this case study. Because of its low resistivity and high thermal and electrical conductivity, copper has begun to substitute aluminum as the connecting metal in microelectronics in the recent years [94]. While there are other precursors available for copper deposition, the advantages of Copper Amidinate are considerable, making it a particularly appealing alternative. [95] originally investigated amidinate precursors for deposition procedures in a study that indicated the use of said precursors for the deposition of a vast set of pure metals that were previously difficult to deposit with the available precursors. The metals included copper, cobalt, nickel and iron. The amidinate-based precursors required just

[1] By reverse engineering, the experimental findings in [41] were found to be misprinted, therefore the original deposition rates are utilized in this case study.

molecular hydrogen rather than atomic, which was commonly utilized but had bad design limitations that made it considerably more difficult to incorporate into commercial processes. The reason behind this is the fact that hydrogen atoms are recombined quickly on surfaces and they cannot be delivered deep through the substrate surfaces, thus limiting the efficacy of said precursors in applications where small trenches are involved. Also, atomic hydrogen can further negatively affect the efficiency of the process and the quality of the film, because of the possible plasma damage it can cause to the deposited surface [96]. In addition, other common copper precursors can have disadvantages such as, high contamination of the produced films with impurities, which are the result of reducing agents or intermediate products and they can have relatively low reactivity and volatility, so higher operating temperatures are required.

Therefore, the main advantages of the amidinate precursors that make them more acceptable in copper deposition processes compared to other common precursors, is first, the sufficiently high reactivity and volatility and their ability to deposit films at relatively low operating temperatures at around 200°C. Also, the films they produce are free of impurities since their molecule lacks oxygen or halogen. Finally, the precursor used in this case study, copper amidinate, carries the above advantages and its assumed deposition surface reaction, in the presence of molecular hydrogen, is the following [97]:

$$[Cu(amd)]_2 + H_2 \rightarrow 2Cu_{surf} + 2H(amd)_{gas}$$

Further investigation of the precursor and the reaction pathways is done in next chapters where the fitting of a new chemical reaction system of copper deposition takes place.

8.3. The CVD reactor

The case study here, is based on the same vertical MOCVD cylindrical reactor that was described in section 7.3 and was used for the aluminum CVD application. This reactor was also used for the experiments in [41], so to avoid repetition further details of the structure of the reactor are omitted in this case study.

The Fluent case of this application is built based on the previous study of [41], where the deposition rate for the substrate temperature range was captured by two kinetic mechanisms as it is shown in Figure 26. In that approach the decrease in deposition rate at the temperature range of 275°C to 350°C was modelled with a Langmuir-Hinshelwood transport limited regime. For the lower temperatures Arrhenius kinetics were assumed to govern the surface reactions.

The same experimental findings are used in this thesis, but with the addition of experiments on two higher substrate temperatures (320°C and 350°C) that were made available by (C. Vahlas personal communication 2020) [9]. In this work, combined with the Arrhenius kinetics regime, two additional temperature activated reactions are fitted, to describe the decrease in deposition rate at the high temperature range.

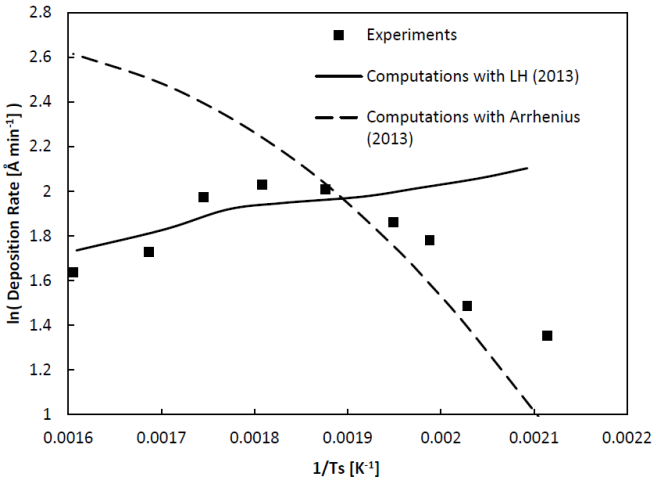

Figure 26: Arrhenius plot of Cu deposition rate as a function of substrate temperature Ts, both reaction models are shown. [41]

As mentioned, the reactor is the same as the one used in section 7, so only the operating conditions of the experiments and the respective boundary conditions will be mentioned. First, the substrate temperature was varied in each experiment in the range of 200°C - 350°C, thus it is the control variable of this CFD investigation [95]. The rest of the main operating conditions are summarized in Table 7. [41] [94]

Table 7: Operating conditions of Cu CVD reactor.

Operating condition	***Value***
Substrate temperature (K)	473, 493, 513, 533, 553, 573, 593, 623
Walls temperature (K)	368
Total pressure (Pa)	1333
Inlet temperature (K)	368
Inlet mass flowrate (kg/s)	$7.473 \cdot 10^{-6}$
Amidinate $[Cu(amd)]_2$ mass fraction	0.001016
Hydrogen H_2 mass fraction	0.004107
Nitrogen N_2 mass fraction	0.255600
Argon Ar mass fraction	0.739277

The CFD model of the reactor consists of a 3D geometry, mentioned in section 7.4, to account for the effect of the showerhead in this application too. The discretization and transient solution of transport equations is carried by FVM within ANSYS Fluent, as mentioned in the previous case study (see 7.4). Also, for the CFD model setup, certain boundary conditions for the computations must be determined, to achieve a flow field that matches the experimental findings. These conditions are mainly based on the operating conditions of the reactor with certain laws being followed for the area close to the boundary layer as described in section 3.2.1. The following Table 8 lists all the necessary boundary conditions, which combined with the operating conditions, setup the CFD model.

Table 8: Boundary conditions of the CFD model.

Boundary	***Condition***
Walls	Stationary wall No slip Fixed temperature = 368K
Susceptor-Substrate	Stationary No slip Temperature range = 473-623K
Inlet	Mass flow rate = $7.473 \cdot 10^{-6}$kg/s Initial gauge pressure = 0Pa Temperature = 368K
Outlet	Outflow

8.4. Literature review and proposed chemical reaction network

Based on the previous work of modelling the kinetics of copper CVD from copper amidinate it was assumed that no additional reactions occur in the presence of hydrogen, with the only main reaction being the following surface reaction [97]:

$$[\mathrm{Cu(amd)}]_{2(\mathrm{g})} + \mathrm{H}_{2(\mathrm{g})} \rightarrow 2\mathrm{Cu}_{(\mathrm{s})} + 2\mathrm{H(amd)}_{(\mathrm{g})} \qquad \text{(S1)}$$

Where a bimolecular elementary second order Arrhenius rate expression was set as [41]:

$$\text{Deposition rate (DR)} = \mathrm{k}_0 \exp\left(-\frac{66\,\mathrm{kJ/mol}}{\mathrm{RT}}\right) \mathrm{C}_{[\mathrm{Cu(amd)}]_2} \mathrm{C}_{\mathrm{H}_2} \qquad \text{(S1)}$$

where, k_0 the pre-exponential factor of S1, $C_{[Cu(amd)]_2}$ and C_{H_2} the molecular concentrations of amidinate precursor and hydrogen respectively. Whilst this Arrhenius reaction successfully captures the deposition rate for the range of substrate temperature 473K to 550K, it fails to describe the phenomena for higher temperatures, as it can be seen in Figure 26 [41]. Regarding this Arrhenius rate, the activation energy was calculated using the slope of the deposition rate curve and the pre-exponential factor was determined by gradual adjustment on the CFD model until the error between the computational and experimental deposition rate data was as low as possible. However, the experiments in [41] were carried up to 573K, so in the previous study in order to explain the reduction in deposition rate for the experiments of 550K to 573K an alternative Langmuir-Hinshelwood (LH) scheme was [98]. The issue with this was the suggested parameters of the LH rate denominator were too small, which reduced the scheme to an almost linear relationship between the deposition rate and the concentration of precursor, something also visible in Figure 26. The slight variation on the LH scheme is due to changes in the flow field at different substrate temperatures w. Thus, the need for a new kinetic system, arise, that can describe the decreasing rate at high temperatures. Finally, the original experimental results, which are utilized for the rest of this case study, are shown in Figure 27.

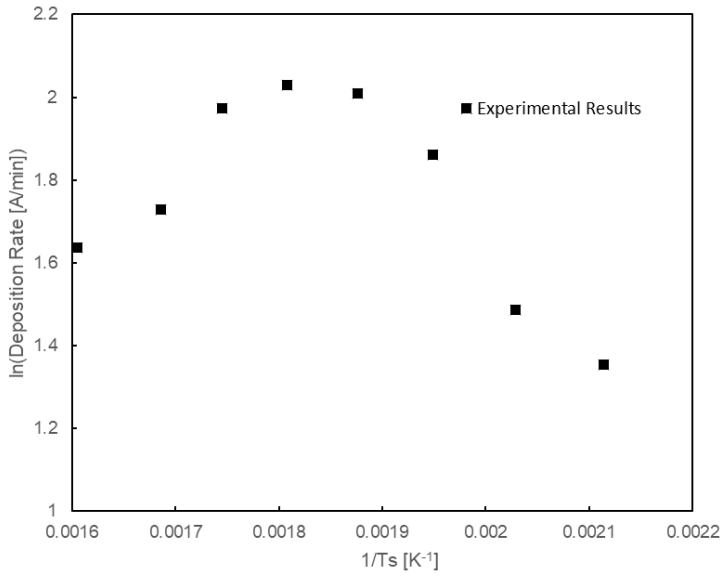

Figure 27: Arrhenius plot of the original and correct Cu deposition rates.

Therefore, the chemical reaction pathway considered in this work at first, was an equilibrium reaction on the surface of the substrate. Although, this could easily describe the decrease of deposition rate at high temperatures by the system leaving the equilibrium state, it

implies that solid deposited copper would sublimate from the surface [99] [100], something that is deemed unlikely. Also, other similar processes were considered as a possibility to capture the experimental findings, such as surface site removal, but since copper can be deposited on various transition metal surfaces [95] [101] in this case the stainless-steel substrates or the previously deposited copper, this was also disregarded [97]. The other alternative was a high temperature activated volumetric nucleation or decomposition reaction of the precursor, which would reduce the available precursor in the gas phase of the reactor before it reaches the surface of the substrate and ultimately reduce the produced deposited copper.

The studies of [43], [44], [94], [102], [103] and [104] were significantly applicable in this case for surface temperatures over 300°C, considering the possibility of volumetric reactions being the case for the reduced deposition rate. The study of [102] shows the decomposition of metallic precursors with amidinate or amidinate-type ligands with two possible pathways. Based on the latter the precursor copper amidinate could decompose by an abstraction of β-hydrogen or by deinsertion of the carbodiimide. The first one is activated at high gas phase temperatures, while the second is more common at a lower temperature solution-based thermolysis. Similar pathways are also suggested by [44], who studied the decomposition of the precursor Copper Guanidinate, a chemical component that has similar structure to the amidinate used in this work Figure 28 [102] [104].

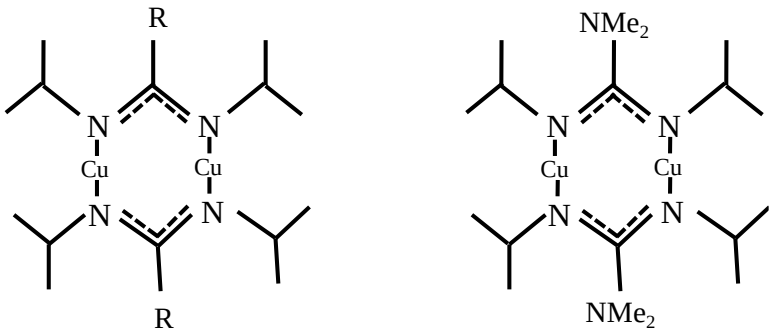

Figure 28: Schematic representation of the structural similarity between Copper amidinate (left) and copper guanidinate (right).

Therefore, based on these studies [102] [104] the mentioned decomposition pathways are adopted for the amidinate precursor as a probable cause of the decreasing deposition rate

at high temperatures. These pathways rely on an initial endothermic ligand shifting and a demonstration of this step is shown in Figure 29.

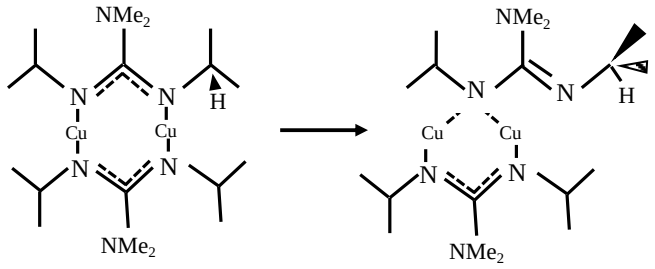

Figure 29: Ligand shifting of copper (I) guanidinate.

The activation energy of this step was assumed to be the same as in the case of guanidinate since there is absence of data for the case of amidinate. The value of said energy is 94.1 kJ mol^{-1} and compared to the surface deposition reaction activation energy at 66 kJ mol^{-1} [95], it assures the idea of the decomposition taking place only at higher temperatures. Following this ligand shift, the β-hydrogen abstraction and carbodiimide deinsertion occur, which are illustrated in Figure 30. The activation energies for each reaction are shown in the Figure 30, where the β-hydrogen and carbodiimide pathways require 138.1 kJ mol^{-1} and 90.4 kJ mol^{-1} activation energies, respectively. Both pathways are modelled in the CFD case study, using the same pre-exponential factors, since there is an absence of information on which one of the two is preferred for systems like this. Additionally, this helps the fitting process of the kinetic parameters and makes the model more flexible, since the degrees of freedom are reduced from three, i.e., the deposition reaction and the two decomposition pathways, to two as the decomposition pathways have the same kinetic parameter that undergoes optimization. [102]

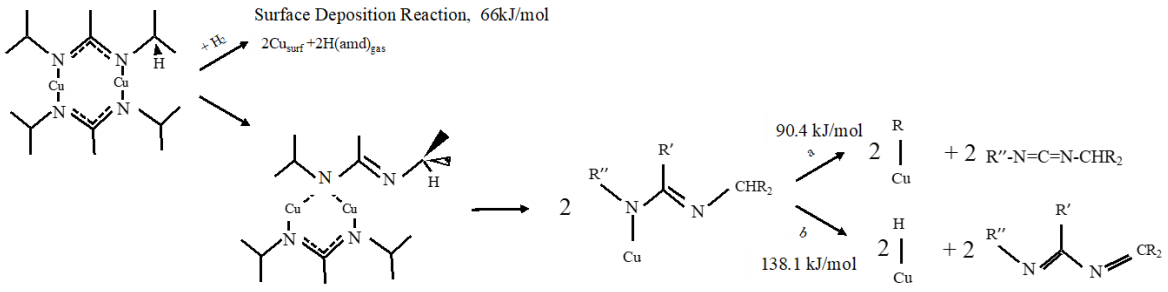

Figure 30: Schematic illustration of the potential decomposition pathways for copper amidinate. (a) Carbodiimide deinsertion, (b) β-hydrogen abstraction.

Furthermore, by not focusing the model on one of the two available decomposition pathways, the computational burden of the fitting process is reduced significantly, compare to the opposite case of a two-parameter optimization problem, which even with the help of the upcoming ROM would still be a lengthy process. It should be noted that this setup is speculative, so it is beneficial to assume as little as possible about the system and try to capture the experimental findings in a rudimentary level. Finally, the following two volumetric decomposition reactions are added to the CFD model along with the mentioned surface reaction S1. The carbodiimide deinsertion G1 and β-hydrogen elimination G2.

$$[\mathrm{Cu(amd)}]_{2(\mathrm{g})} \rightarrow 2\mathrm{Cu(CH_3)}_{2(\mathrm{g})} + 2\mathrm{CH(CH_3)_2} - \mathrm{NCN} - \mathrm{CH(CH_3)}_{2(\mathrm{g})} \quad \text{(G1)}$$

$$[\mathrm{Cu(amd)}]_{2(\mathrm{g})} \rightarrow 2\mathrm{CuH}_{(\mathrm{g})} + 2\mathrm{CH(CH_3)_2} - \mathrm{N} = \mathrm{C(CH_3)} - \mathrm{N} = \mathrm{C(CH_3)}_{2(\mathrm{g})} \quad \text{(G2)}$$

The activation energies of S1, G1 and G2 are assumed to be as in literature and are added in the ANSYS Fluent case as they are. On the other hand, the pre-exponential factors of S1 and G1-G2, k_0 and k_1=k_2 respectively, go through a fitting process in order to capture the experimental findings for the whole range of substrate temperatures. The Arrhenius reaction rates of the volumetric reactions are:

$$\text{Reaction rate (RR)} = \mathrm{k_1 exp}\left(-\frac{90.4\,\mathrm{kJ/mol}}{\mathrm{RT}}\right)\mathrm{C}_{[\mathrm{Cu(amd)}]_2} \quad \text{(G1)}$$

$$\text{Reaction rate (RR)} = \mathrm{k_2 exp}\left(-\frac{138.1\,\mathrm{kJ/mol}}{\mathrm{RT}}\right)\mathrm{C}_{[\mathrm{Cu(amd)}]_2} \quad \text{(G2)}$$

This detailed 3D CFD model that additionally includes the species transport equations of the previously mentioned reactions S1, G1 and G2 has over 16 million degrees of freedom, since it is discretized on a fine grid of 1232468 cells as described in the previous application (see 7.4). A case study of this size amounts a significant computational cost, especially when considering the fitting of the kinetic parameters, which is usually a trial-and-error procedure. The usual loop during a fitting process starts by assuming a set of kinetic parameters values, then the CFD solution is compared to experimental data and based on the errors the kinetic parameters are readjusted until a satisfactory accuracy is achieved. Therefore, a need of an efficient model order reduction that will reduce the computational burden of the fitting process arises, this is achieved by using the proposed computational framework of this thesis.

8.5. Data collection – Step changes

As shown in the previous application, the first step in the proposed framework is the data collection of snapshots for the POD method. These snapshots contain the state vector of the CVD reaction, which consists of the velocity components, temperature, pressure, and mass fraction of precursor values on each discrete FVM cell. The snapshots are collected through the dynamic response of the reactor to a step change on the substrate temperature (the altered parameter of this application) on each discrete time instance. Each snapshot is provided by solving the discretized governing equations of momentum, heat and mass transport in the reactor with the FVM in ANSYS Fluent. This computation is performed with a transient solver for a fixed number of 100 iterations for each time step. The latter is set to be $\Delta t=0.1s$. For the data collection, a 3D coarse grid is used, which is the same as the one in aluminum application (see 7.7), it consists of 309155 cells, thus making each snapshot a vector of 1854930 elements. It should be noted that the CFD calculations are all done using parallel processing, as it is common for case studies of this size.

The initial states for the data collection are the reactor steady states at the lowest and highest experimental substrate temperature. The end point of each step change in terms of substrate temperature is selected in a way so they are evenly distributed in the temperature range of the study. By using only two initial states for the data collection means that only two steady states need to be calculated in advance by the CFD model. Therefore, any other state used in the rest of this study was computed completely by the ROM predictions, which reduces the computational cost of the study significantly. The applied step changes on the substrate temperature are shown in Figure 31, with the starting point and direction for each one. Finally, the trajectory for each step change consists of 10 snapshots on average.

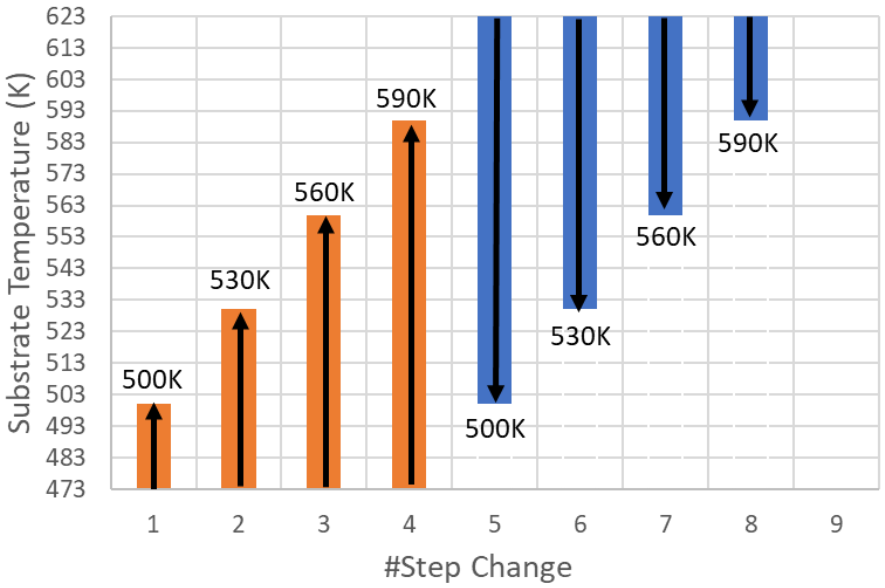

Figure 31: The eight step changes on the substrate temperature with their starting points, ending points and direction of the change.

8.6. Design of POD basis

To begin with the POD basis creation, the snapshot matrix Y is made with the data from the step changes. This matrix has the dimension of N x k, where N is the number of degrees of freedom (the number of elements of each snapshot) and k the total number of snapshot vectors collected from the dynamic responses. To collect the data needed for this matrix the 3D CFD model needs to be simulated multiple times as it is shown in the case study of aluminum. This process can be extremely time consuming if a detailed fine FVM grid is used. In addition, even the manipulation of the resulting data would be daunting if the detailed model was used to produce them. Therefore, a coarse grid of 309k finite volumes is used instead (see 7.7), this reduces of course the accuracy of the CFD solution for the sake of efficiency, but as analyzed in previous chapters of the proposed computational framework, the accuracy can be recovered. This grid produces snapshots of 1.86 million elements and the entire data set was collected in almost 2 days of CPU time. It should be noted that, for further reduction of the data collection computational cost, the chemical reactions and the respective chemical compounds of the process are not included in the simulations during this procedure. This is because in CVD cases, like this one, the inlet gas mixture contains a very small amount of precursor, thus the depletion or product/by-product production does not affect the overall flow or temperature distribution.

Finally, it is obvious that the predictions from the created ROM are not going to be very accurate, but this issue will be addressed later, along with the connection of these predictions with the information about the Cu deposition.

The construction of the POD basis uses the data of the step changes mentioned in Figure 31, which are listed in the Table 9 below.

Table 9: The 8 step changes utilized for the ROM production and accuracy testing.

Initial Temperature / K	***Final Temperature / K***	***No. of Snapshots***
473	500	10
473	530	10
473	560	11
473 – *Internal*	590	11
623 – *External*	500	11
623	530	10
623	560	10
623	590	9

Regarding the step change 473 to 590K, the name "internal" is due to the reason that the data of this specific step change are used for the creation of the ROM along the rest of step changes, but they are also used to test the accuracy of the ROM in simulating data that were used for its creation (internal data). On the contrary, the step change 623 to 500K is excluded from the development of the ROM and it's used as an accuracy test of the ROM to data outside of its "knowledge" (external data). Therefore, with the latter step change removed, the snapshot matrix Y of the POD has the dimension of 1.86 10^6 x 71, since 71 snapshots are used from the seven step changes. The POD by using the equation [8-1] reduces the eigenvalue problem from the previous snapshot matrix to the correlation matrix 71 x 71, which reduces the computational complexity significantly. The solution of the eigenproblem gives a set of 71 eigenvalues and their respective eigenvectors and then the information content captured by the POD basis for different dimensionalities is calculated (eq. 5-29). These information contents are

plotted against the number of eigenvectors used in POD basis (dimensionality) in the Figure 32 below.

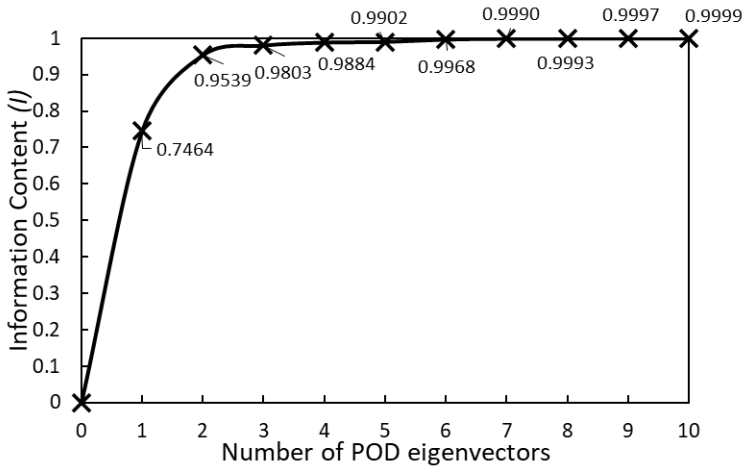

Figure 32: Information content for the first 10 POD eigenvectors with the higher eigenvalues.

Following the Figure 32 and based on the theory presented in section 5.3.3, four different POD basis are formed with dimensionalities 1 to 4, with 4 being the max since it is the smallest possible number of eigenvectors that contains almost 99% of the information. Then, based on the procedure analyzed in section 5.3.5 and in order to demonstrate the effect of the dimensionality of the POD basis to the resulting ROM accuracy, the predicted snapshots by the ROM are compared to the CFD data for the different basis size in the next subchapter.

8.7. Design, training and simulation of ANN

After the creation of the POD basis, the ANN is trained by using the different dimensionality POD basis in conjunction with the same data set. The neural network used is the NARX network and it is trained and simulated in MATLAB. The training procedure is the same as the one followed in section 7.8, so it will not be analyzed fully again in this case study. The difference here is that the input/control variable is the substrate surface temperature and the NARX setup parameters, which is summarized in Table 10.

Table 10: NARX network design parameters.

Artificial Neural Network Properties	***Value***
Hidden Layers	1
Neurons in Hidden Layer	5
Input Delays (n_x)	1
Output Delays (n_y)	1
Training algorithm	Bayesian Regularization Backpropagation

A schematic representation of the NARX network during the training procedure and its simulation, along the input and output parameters is shown in the Figure 33 below.

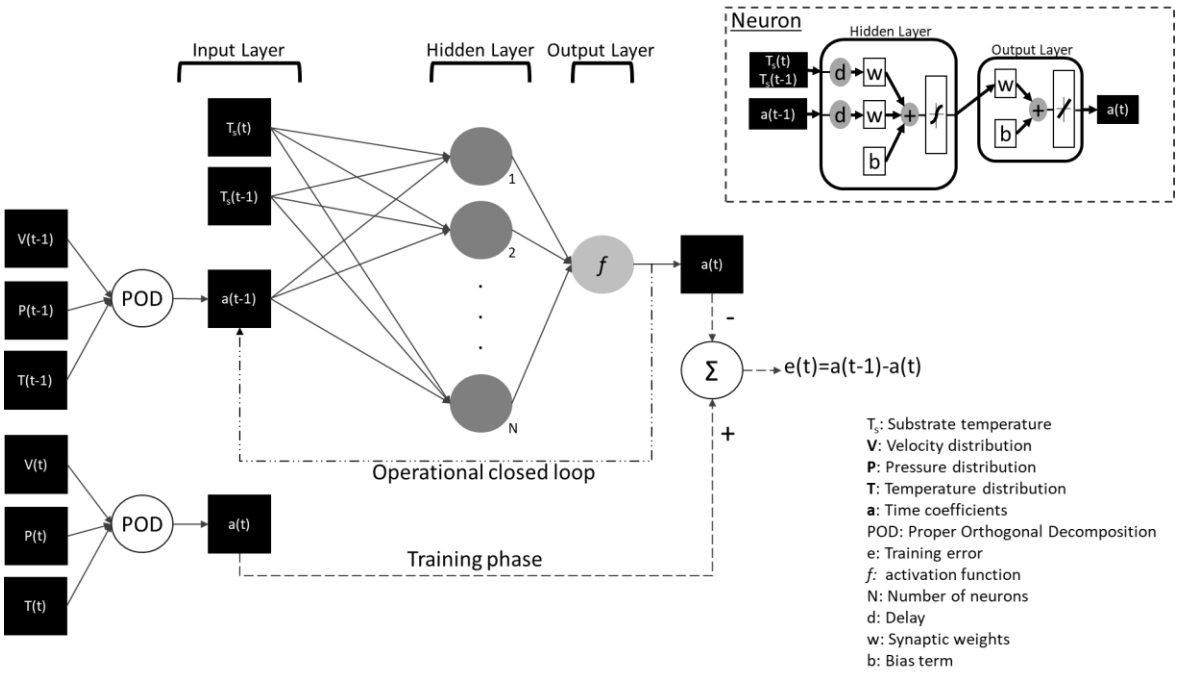

Figure 33: Schematic representation of the NARX network structure during the training phase and the operational closed loop simulation. (top right: structure of a single neuron).

During the simulation of the trained network, the input data include the initial condition, which is a snapshot at the starting temperature of the step change that is going to be simulated, the initial and the final substrate temperature of the step change. The output data of the network is the group of predicted time-dependent coefficients, which combined with the POD basis they can recreate the entire trajectory of the reactor throughout the simulated step change (eq. 5-24). These predicted data are compared to the CFD ones as previously mentioned for the 4 different POD basis, to decide on the final dimensionality of the latter. This comparison is shown in the Figure 34, where the predicted ROM data are compared to the CFD data for the external step change, which is unknown to the ROM. The error decreases by increasing the POD dimensionality, but it is obvious that for all POD basis the average error is less than 1%, which shows a good generalization of the trained data-driven ROM and it is not affected significantly by the POD basis size. Therefore, for the rest of this application a predictive ROM based on a POD basis of 3 eigenvectors is used.

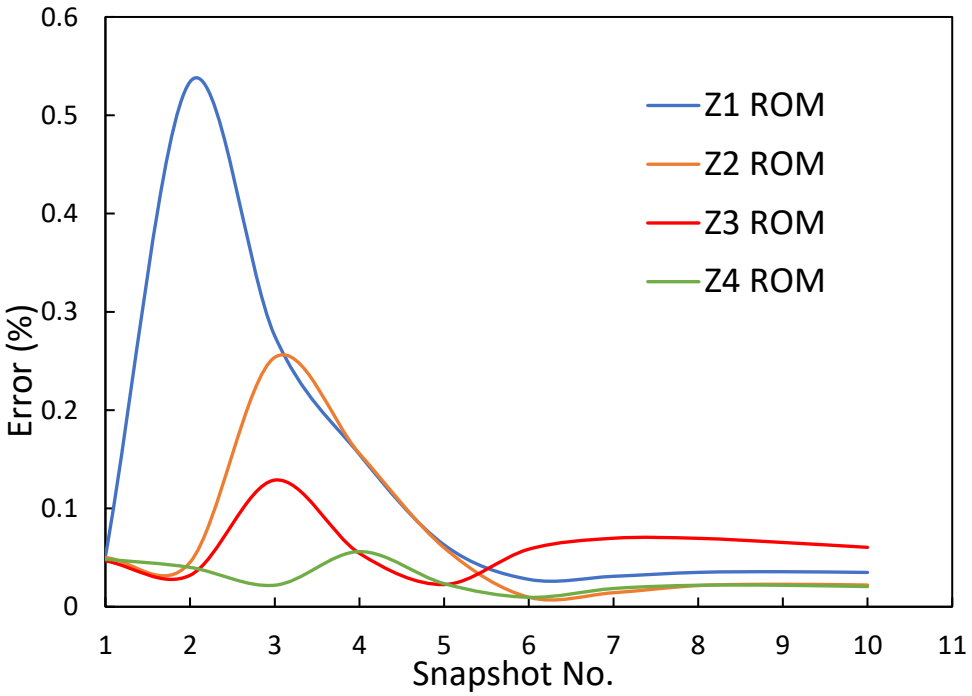

Figure 34: Prediction error of ROM along the data of an external step change trajectory, for different POD basis dimensionalities.

The recovery of the lost information from collecting data using a coarse grid, without chemical reactions, CFD model must be addressed. Mainly because the ROM predictions are

also in said form and they have a significant error against the fine grid, chemistry involved CFD model results. The procedure used is the grid-to-grid interpolation (see 4.2), where the ROM predictions, in the coarse grid form, are first interpolated into the fine grid using the Nearest Neighbor technique. In this technique the value of a variable in a cell of the fine grid is set equal to the value in the nearest coarse grid neighbor cell. These interpolated predictions are used as an initial estimate for the CFD computations of the detailed model, including chemistry, which converges to the high-fidelity solution in only a few iterations of the solver. This procedure is also shown schematically in Figure 35 (TOP).

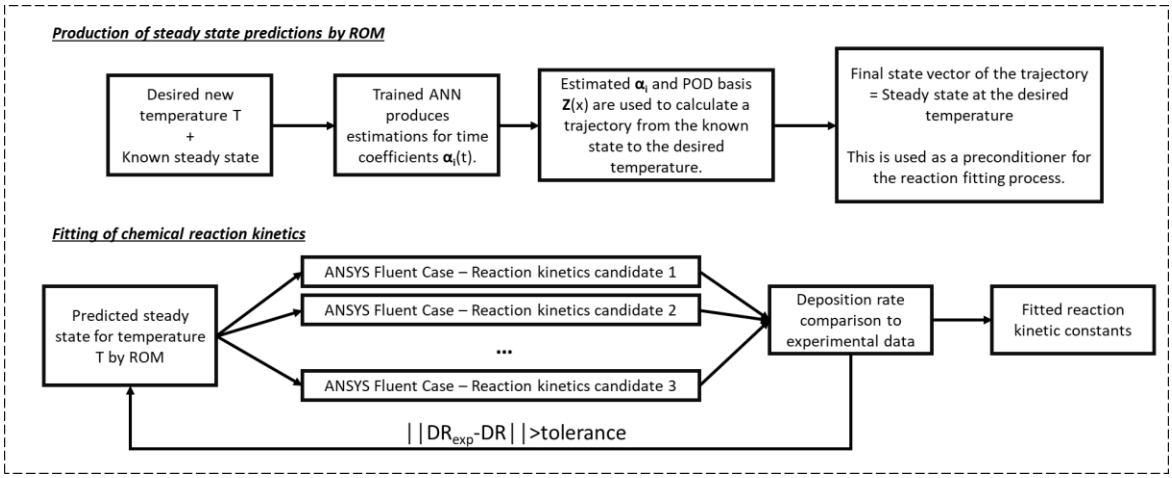

Figure 35: Schematic illustration of (TOP) the production of detailed model stead state and deposition rate predictions by ROM, (BOTTOM) the fitting procedure of chemical reaction kinetics.

The computational cost reduction by using the developed ROM is significant and especially for the case of kinetic parameters fitting procedure, which is usually based on trial and error and its extremely computationally expensive. The computational time needed for each case is calculated in core hours, which is the wall clock-time needed multiplied by the amount of CPU nodes used. Thus, a CFD simulation of CVD reactor detailed model (fine grid and chemistry included) in parallel on 12 CPU nodes requires 864 core hours. On the other hand, a ROM prediction of the solution for the same parameter setup, takes less than a minute on a single CPU node and for the detailed model to converge to the high-fidelity solution by using this prediction as initial estimation, only 216 core hours are needed (on 12 CPU nodes). Therefore, there is a 4 times acceleration in core hours. Finally, considering that in a kinetic parameter fitting procedure, a model like this has to be simulated several times, for several different parameter setups, the overall computational benefit is even more significant and increases with the number of unknown kinetic constants.

8.8. Fitting of chemical reaction kinetics

The chemical reaction system proposed in section 8.4 is fitted using the ROM predictions as initial states of the detailed CFD model, to achieve faster convergence. Each state predicted by ROM at a certain temperature of the substrate can be used as many times needed for all the different values of pre-exponential factors being tested. For every run of a given surface temperature and set of kinetic parameters the deposition rate of copper is extracted and is compared to the experimental findings for the same temperature. The CFD simulated deposition rates for different setups are plotted against the experimental values for the range of temperatures studied, until a setup of kinetic parameters captures with an error as small as possible the curve of experimental data shown in Figure 27. Thus, resulting to the final fitted set of kinetic parameters for the surface and volumetric reactions. A schematic summary of this fitting procedure using the ROM predictions is described in Figure 35 (BOTTOM).

The values of the three fitted pre-exponential factors of the surface and volumetric reactions described in section 8.4 are summarized in Table 11. Finally, the computational deposition rate of copper is compared to the experimental findings in the following Arrhenius plot Figure 36. The proposed chemical reaction system appears to accurately describe the CVD of copper for the experimental substrate temperature range, without using two separate kinetic models. Additionally, it captures accurately the trend of experimental data even for higher temperatures than 573K. The kinetic constants of the three reactions system are summarized in Table 11.

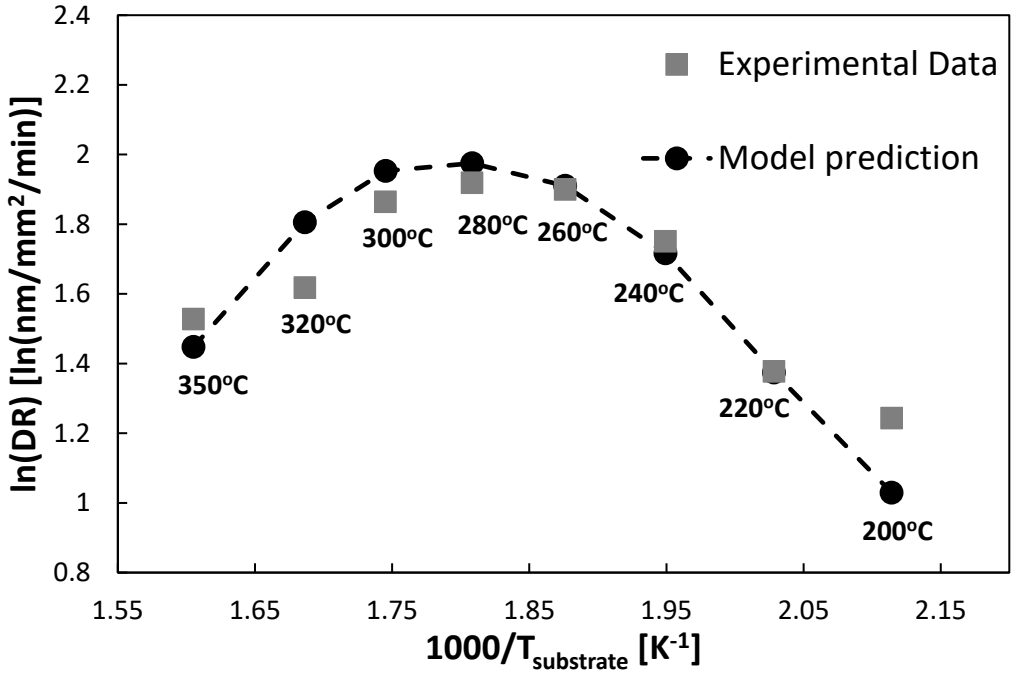

Figure 36: Arrhenius plot of the natural logarithm of the deposition rate against the reverse substrate temperature. Comparison between ROM predictions and experimental data.

Table 11: Fitted kinetic parameters of the chemical reaction system.

Reaction	***Pre-exponential factor***	***Activation energy (kJ/mol)***
Deposition reaction (S1)	$1.35 \cdot 10^{10}$ m^3 $kmol^{-1}$ s^{-1}	66.0
Carbodiimide deinsertion (G1)	$8.5 \cdot 10^{8}$ s^{-1}	90.4
β-hydrogen abstraction (G2)	$8.5 \cdot 10^{8}$ s^{-1}	138.1

8.9. Conclusions

A novel chemical reaction system is presented for the CVD of copper from precursor copper amidinate. The addition of two new volumetric reactions, with high activation energy results in a significant drop of copper deposition rate at temperatures over 573K due to precursor depletion. Based on the literature on copper amidinate and a structurally comparable

precursor, copper guanidinate, two methods of decomposition are considered for the amidinate. These are the β-hydrogen abstraction and the carbodiimide deinsertion. The suggested kinetic parameters of the chemical reaction system are derived from the comparison of the data-driven and CFD assisted ROM predictions to the experimental observations. This fitting procedure is computationally costly and is only possible in this circumstance by following the proposed machine learning framework of model order reduction based on POD and ANN. Throughout the case study, further reduction on the computational cost of developing the ROM are realized, by using a low-fidelity, coarse grid and without chemistry CFD model for the data collection. The sacrificed accuracy during the data collection, is then restored by utilizing the ROM predictions as initial estimates for the detailed CFD model, which speeds up its convergence. Finally, since the ROM has no information of the chemistry, it can be reused for future fitting procedures on the same reactor for different kinetic parameters setups without the need of ROM reconstruction, which further increases the efficiency of the latter.

9. Case study 3: CVD reactor with solution multiplicity

9.1. Introduction

The following application of the proposed computational framework is another case study of CVD. As mentioned, the process of CVD is characterized by complex transport phenomena and a complex network of chemical reactions, which makes it necessary to study it computationally, for its proper understanding, and therefore the production of a product (films) with desired specifications. The reactor studied is used for various MOCVDs, but in this example only the transport phenomena are concerned, and the species transport and chemistry are not included. The reactor studied is a vertical, axisymmetric CVD reactor, which according to previous computational research [5] [22], its CFD model has three states, two stable (observable) and one unstable (non-observable) for the same operating conditions. The control of a system like this, is a difficult to impossible task, which is carried out almost empirically or even intuitively by the operator in an industrial or experimental setting. A dynamic study of the system becomes necessary for the control of the reactor and the production of the desired product. The aim of this work is to construct a Reduced Order Model, following the proposed framework, that will describe the dynamic behavior of the system for the two stable branches of steady states, as well as alternations between them.

9.2. The CVD reactor

In the context of this work, the reactor presented in Figure 37 was used. It is a cylindrical reactor with a heated substrate at 1200K and cooled walls at 300K. The gas mixture enters the reactor from the top side of the infusion tube and then is driven towards the substrates, as it shown in the Figure 37. There, the film is formed, as presented in section 2.3, while the by-products and inert gases exit from the reactor outlet. The operating pressure in the chamber is kept constant at 1300Pa and it can be assumed that the gases in the volume of the reactor behave as ideal and the flow is laminar. It is worth noting that the reactor walls are cooled to avoid the activation of deposition reactions on them and therefore no product is deposited on the walls. [22] [55] [105] [106]

This reactor is discretized and solved on a 2D axisymmetric grid using the commercial software ANSYS Fluent. The governing equations presented in section 3.2.1 are solved in a system of cylindrical coordinates using the finite volume method and a transient solver. The computational grid is divided into 15066 cells, in which the pressure, temperature and velocity (the axial and radial components) are calculated for each time increment, thus creating a x(t) state vector of 60264 elements [5]. This FVM grid is shown in the Figure 37 (c) below.

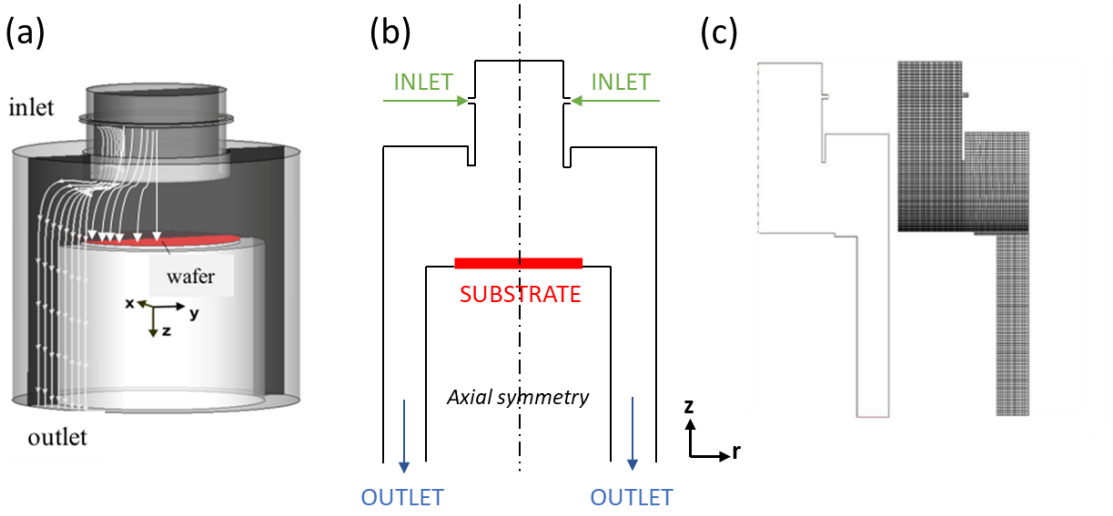

Figure 37: CVD reactor with axial symmetry and solution multiplicity. (a) 3D geometry, (b) 2D axial symmetry geometry and (c) FVM computational grid of the 2D geometry.

The discretized system of PDEs described in sections 3.2.1, 3.3.2 and 3.3.3, requires a setup of boundary conditions for their solution. The boundary conditions are summarized as follows: [22] [55] [105]

- The gaseous mixture enters at the reactor inlet boundary at a constant temperature of 298K and at a uniform, perpendicular to the boundary, velocity. The inlet velocity is directly dependent to the mass inlet flow rate, which changes with time as it will be described later:

$$\boldsymbol{n} \cdot \boldsymbol{u} = \frac{\dot{m}(t)}{2\pi L_{in}} \; and \; \boldsymbol{n} \times \boldsymbol{u} = \boldsymbol{0}$$

9-1

where, n the unit vertical vector on the boundary, $\dot{m}$ the mass flow rate and L_{in} the annular inlet width. In this study the mass flow rate ranges between 1.5·10-5 kg/s and 6.0·10-5 kg/s.

- At the reactor outlet, a velocity vector perpendicular to the boundary, a zero temperature and mass flow gradient and a reference value (1300Pa) for the pressure are imposed.

$$\boldsymbol{n} \cdot (\nabla \rho \boldsymbol{u}) = 0 \, , \boldsymbol{n} \times \boldsymbol{u} = 0 \; and \; \boldsymbol{n} \cdot \nabla T = 0$$

9-2

- The no-slip condition for the velocity applies to the reactor walls and the temperature is kept constant to 300K for the cooled walls and 1200K for the heated substrate.

- On the symmetry axis, the velocity has a zero component perpendicular to the axis and the gradient of all variables is zero on the direction perpendicular to the axis.

It should be noted that the only boundary condition that changes over time during the transient simulation is the flow rate $\dot{m}(t)$. In addition, the flow rate is the only input variable that can be used as a control variable in the system. This is because the other input variables, i.e. the temperature of the walls and the temperature of the substrate are kept stable throughout the process, due to restrictions set by chemical reactions. Finally, for the solution of the PDEs the

existence of initial conditions is necessary. These are the known spatial distributions of the pressure, velocity and temperature fields at time t=0. The initial conditions can either be set at default/zero values or be a previously obtained steady state for a certain value of the mass flow rate. The steady state is a vector containing the pressure, velocity and temperature fields at time t.

Regarding the chemical reaction system of this application, it is not included and only the transport phenomena concerning the inert gas nitrogen (N_2) are studied (which is usually used as a carrier gas). Since the main goal of this study is to create a ROM based on the proposed framework, which as shown in the previous applications does not require the information of the reactor chemistry, the reactions are fully omitted from this study. This assumption can be made because CVD systems are usually dilute, meaning the reactants are 1% or less of the total gas mixture, so the depletion or production of chemical compounds does not affect the flow and heat transfer.

9.3. Solution multiplicity

A characteristic of the CVD system studied in this application, is the multiplicity of solutions, which is due to the nonlinear terms of the transport equations in section 3.2.1. For each mass flow rate value ranging between $2.3 \cdot 10^{-5}$kg/s and $5.0 \cdot 10^{-5}$kg/s the system can converge to three different states, two stable and one unstable, depending on the initial setup. The Figure 38 (left) shows a diagram of the three branches of solutions versus the Reynolds number, which is proportional to the inlet flow, and the Nusselt number, which characterizes the steady state of the system [22] [55] [105]. The natural cause of the multiplicity of solutions presented above lies in the fact that in the physical system two mechanisms compete, the free convection, where buoyancy predominates due to the temperature gradient between the heated substrate and the main volume of the reactor, and the forced convection where the velocity of the incoming fluid is dominant. Specifically, the upper stable branch contains the steady states of force convection and the lower stable branch the free convections ones. [22] [55]

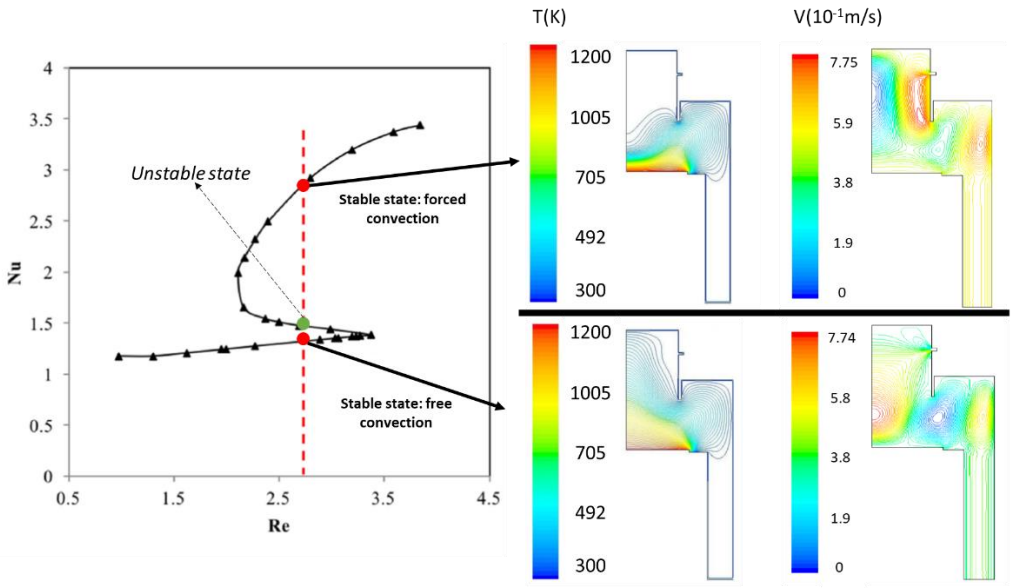

Figure 38: (Left) The CVD reactor solution space based on Nusselt number above the substrate against the Reynolds number for substrate temperature 1200K. (Right) The same operating conditions, Re=2.77 and $\dot{m}$= 3.5·10-5kg/s, can result to two possible observable stable solutions and one non-observable unstable.

The Figure 38 (right) shows comparisons of the temperature and velocity distributions in the reactor for the steady states on the two stable branches for the same inlet flow rate, which is obviously located within the multiplicity range. These two different states can be experimentally observable, unlike the unstable branch state and they can affect the quality of the product randomly, without the user changing the parameter setup. This is a huge problem in an experimental or industrial setting, since the user has no control over the final steady state that the reactor is going to reach inside the multiplicity range. Thus, this is the main reason for the computational study of this system and the further application of the proposed framework to reduce its cost.

Moreover, during a step change on the mass flow rate, starting from a steady state outside of the multiplicity range with a target flow rate inside the multiplicity range, the Fluent solver will usually converge on the respective solution to the closest stable branch. Finally, the convergence time increases significantly as the Fluent solver tries to converge to a point where a shift from one stable branch to another takes place, i.e., as it approaches the area of instability. [22] As mentioned for most CVD cases, the flow and temperature fields are not affected by components other than the carrier gas. This applies in cases where the reacting components are in traces in the reactor [8]. In contrast, the deposition rate as well as the uniformity of the

produced film depend significantly on the flow mechanism prevailing in the reactor, as shown in Figure 39 from a study in [22].

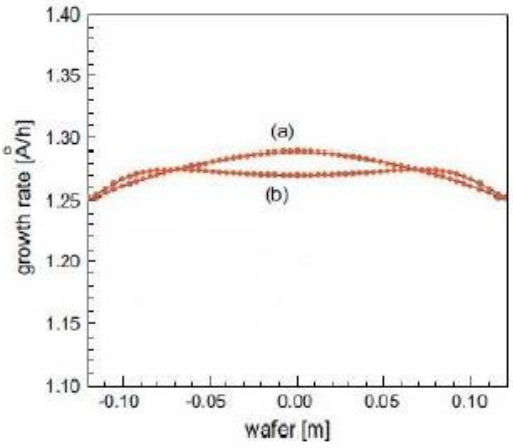

Figure 39: Plot of deposition rate of the product versus the position on the substrate for a constant flow rate in (a) lower stable branch and (b) in the upper stable branch.

The Figure 39 shows the differences in the uniformity of the film produced when the reactor is subject to free or forced convection. Films produced under conditions of forced convection are characterized by greater uniformity. As it is obvious, the industry wants to achieve as much uniformity as possible, but it is difficult to control it due to the multiplicity [4] [22]. Therefore, the deposition rate and the uniformity of the film is not fully determined by the flow rate of the reactive mixture. This makes the detailed study of the process necessary both at a theoretical level and in practice to produce films that meet the specifications of each application, since a small change in the inlet mixture flow rate can radically change their characteristics, if the reactor is operating close to the branches turning points. In the context of this work, a ROM will be designed that can predict these changes between the stable branches with sufficient accuracy and in a short period of time.

9.4. Data collection – Step changes

During the data collection, the elements of a steady state vector are taken as initial conditions combine with the respective initial flow rate. The dynamic behavior of the system is captured from one steady state to another after a step change is imposed to the inlet mass flow rate. In each timestep of the trajectory, 200 steps of the iterative solver are performed. The number of iterations in each time step does not have to ensure convergence, but in any case, is

large enough that the norm of the difference of two consecutive iterations is about 10^{-5} for the continuity equation. It is noted that, the time is discretized at intervals of Δt=0.1s. Specifically, in a step change with n time steps, n state vectors are computed and each of them is a snapshot of the reactor state at the respective time increment. All these snapshots create the trajectory of the dynamic response of the system to the step change.

Initially, steady states for mass flow rates in the range $1.5 \cdot 10^{-5}$kg/s to $6.5 \cdot 10^{-5}$kg/s, for the two stable branches presented by the system, are calculated. It is recalled that the bottom branch corresponds to a flow dominated by buoyancy (free convection) and the upper branch by forced convection. Simulations were then carried throughout step changes from a steady state of a particular $\dot{m}(t)$ and multiplicity branch to another steady state, which may be located either in the same or in the other branch. The range of these step changes on the mass flow rate was chosen to adequately describe the entire range of the multiplicity area. It should be noted that the convergence time increases significantly as the state of the system tends to the turning points, and therefore the step changes with alternation between branches are accompanied by long convergence times. It is recalled that the step changes are imposed, and their data are collected by coupling Fluent with Matlab. Some of the data collected from step changes are combined into the snapshot matrix, which will be used to design the POD basis according to the methodology developed in section 5.3. Another part of the data set is used for the training of the neural network as presented in section 5.4 and others for the evaluation of the above.

The main objective of this study is to design a model that satisfactorily describes the two stable branches of multiplicity, as well as the alternations between them. As far as alternation step changes are concerned, starting from a known state in one of the two stable multiplicity branches, the step change target should be imposed outside of the starting branch, so that the Fluent solver converges in the other stable multiplicity branch. This is because the solver tends to converge on the closest branch to the initial state. If for example the initial state corresponds to a value of the mass flow rate of $4 \cdot 10^{-5}$kg/s in the upper branch and given a step change to $3 \cdot 10^{-5}$kg/s, the solver will converge in the closest state, i.e., in the upper branch of the multiplicity of solutions. Therefore, it is difficult to collect computational data describing the alternation between the multiplicity branches and at the same time stay in the studied range.

9.5. Design of POD basis

The first step of this study is to construct the snapshot matrix Y that will be used to design the POD basis. For the snapshot matrix, data of 8 step changes are satisfactory, 4 of these

are based on the upper branch and 4 on the lower branch. The specifics of starting and ending points, the number of snapshots, as well as the corresponding branch of each step change are presented in Table 12.

Table 12: Step changes on the mass inflow rate.

Mass inflow rate (kg/s) step change			
Starting point	***Ending point***	***Number of snapshots***	***Solution branch***
$1.5 \cdot 10^{-5}$	$5.2 \cdot 10^{-5}$	74	Lower
$4.0 \cdot 10^{-5}$	$2.5 \cdot 10^{-5}$	60	Lower
$3.5 \cdot 10^{-5}$	$5.0 \cdot 10^{-5}$	54	Lower
$5.2 \cdot 10^{-5}$	$4.9 \cdot 10^{-5}$	47	Lower
$3.5 \cdot 10^{-5}$	$5.5 \cdot 10^{-5}$	55	Upper
$5.5 \cdot 10^{-5}$	$3.3 \cdot 10^{-5}$	138	Upper
$3.3 \cdot 10^{-5}$	$4.0 \cdot 10^{-5}$	75	Upper
$6.0 \cdot 10^{-5}$	$3.2 \cdot 10^{-5}$	232	Upper

Therefore, the snapshot matrix consists of 735 state vectors (snapshots), from 8 step changes. The matrix undergoes the POD procedure described in section 5.3.2 and ultimately the eigenvalues and eigenvectors of the correlation matrix are calculated. From these the first 4 eigenvectors with the higher eigenvalues are chosen to construct the final POD basis. The informational content contained in those vectors is 98%. Thus, by using the POD the order of the model is reduced from Nxk, where N the degrees of freedom (=60264) and k the number of snapshots (=735), to Nxm, where m the dimensionality of the basis (=4). Also, 2 step changes were used to test the accuracy of the POD basis and later the ANN accuracy. These are 2 internal step changes, meaning they are included for the design of POD basis. These are the step change $3.3 \cdot 10^{-5}$ kg/s to $4 \cdot 10^{-5}$ kg/s for the upper and $1.5 \cdot 10^{-5}$ kg/s to $5.2 \cdot 10^{-5}$ kg/s for the lower branch.

The accuracy testing is based on the procedure described in section 5.3.5. The average relative error of the 4-eigenvector basis on capturing the test step changes is no more than 1%, which is accepted. Thus, the next step is the training of an ANN (NARX) with the 4-eigenvector basis and the same data set, which will be able to predict reactor states with great accuracy and with low computational cost.

9.6. Design, training and simulation of ANN

In the previous section, a basis was designed using 8 step changes that accurately describe the entire range of solution multiplicity. Therefore, the Matlab toolbox [82] was used to train and simulate a NARX network, as described in section 5.4. The main design parameters of the network are presented in Table 13.

Table 13: NARX network design parameters.

Artificial Neural Network Parameter	***Value***
Hidden Layers	1
Neurons in Hidden Layer	5
Input Delays (n_x)	1
Output Delays (n_y)	1
Training algorithm	Bayesian Regularization Backpropagation

The setup is pretty much alike previous applications, so in this study no more further details of the design parameters will be discussed. The network by maintaining the time-dependency of the system can predict the reactor states for mass flow rate values beyond the initial data set. These predictions can be obtained in a matter of seconds with an extremely low computational cost. The reliability test of the trained network is performed by comparing the predicted states or trajectories to the already CFD calculated (known) ones, for the respective mass flow rate in both multiplicity branches. The results of the test are very satisfactory.

First, for a steady state either on the upper branch or the lower, the complete ROM (ANN+POD) can predict the reactor states with an average error of 1.2%, when compared to the known CFD calculated steady states. These are states for mass flow rates that are not a starting

or ending point of the training step changes, thus the ANN has no memory/information about them. Second, to determine if the ROM can capture the whole trajectory of the dynamic response of the reactor to a mass flow rate step change, the same two step changes used for testing the POD basis are utilized here too. Regarding the lower branch step change trajectory, the error hovers around 0.2%, with only one swing to 1.2% at the beginning of the trajectory, which is not a concern. For the upper branch the error settles to an average of 2% towards the end of the trajectory, but it shows a major swing to 16% after the first 20 time increments. Thus, another step change was tested for the upper branch, $3.5 \cdot 10^{-5}$kg/s to $5.5 \cdot 10^{-5}$kg/s, which shows a consolidation of the error at around 1%. Finally, for a balanced testing procedure, a second lower branch step change was tested, with an error of 0.4%. It seems that the ROM captures the lower branch better than the upper one, but the difference is not significant. The plots of the two main testing step changes trajectory errors are summarized in Figure 40.

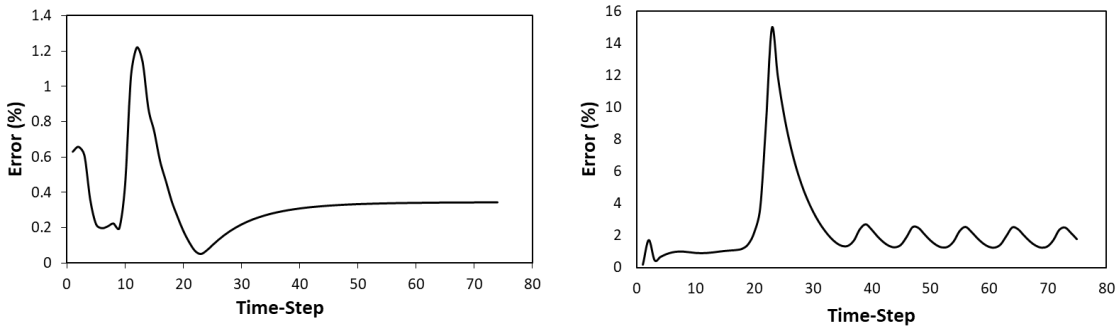

Figure 40: Prediction error of ROM along the trajectory of lower (left) and upper (right) solution branch step changes.

9.7. Conclusions

The computational framework proposed in this thesis is applied in this case study, to design a data-driven ROM using the machine learning algorithms of POD and ANN. This ROM can predict the system dynamic response to step changes on the inlet mass flow rate fast and accurately. In this study the reactor system presents solution multiplicity, which means that for the same parameter setup it has two different stable experimentally observable states and one unstable non-observable state. This solution multiplicity is captured in CFD studies, and it is the result of the nonlinearity of PDEs and the competing natural mechanisms of free and force convection. The equation-free ROM developed in this application uses POD and ANN to capture

the spatial and time dependency, respectively. Although, the ROM is based on a limited amount of data, compared to the solution space of the reactor, it performs well with the average errors not even exceeding 2%. Therefore, this application confirms that the proposed computational framework, additionally to previous applications, can be also used to develop purely data-driven ROMs for complex distributed nonlinear systems.

10. Summary and conclusions

The CFD studies are a standard approach nowadays in the spectrum of engineering with the purpose of simulating real-world applications. They offer a great way of understanding the principles and the key phenomena within the system that lead to the final product, without having to experiment and waste a significant amount of high-quality raw materials. Nevertheless, a complete CFD study of some applications can be a daunting and sometimes even an impossible task, based on the computational cost that accompanies them even with parallel processing. This cost is extremely affected by the size of the problem and the large number of degrees of freedom, and it is also linked to unexplored and unresolved aspects of the process that are usually not observable in the real scenario, but they still crucially affect the final product. A good example of such application, hence the one studied in this work, is that of CVD reactors, which are characterized by complex geometries, a variety of competing transport phenomena that occur in the reactor chamber and multiple chemical reaction pathways. The importance of CVD in the industry of state-of-the-art semiconductors for computer chips and the wide variety of coating applications, led to the popularity of the process and to the frequent studies with CFD models that try to determine the dominant flow patterns and chemical mechanisms that affect the final, expensive in most cases, produced film/coating.

This dissertation deals with the challenges that arise from the need of reliable CFD modeling of realistic experimental and industrial systems, by creating a purely data-driven and equation-free computational framework. The framework built in this work can create ROMs of the CFD models of the real-world systems, which have a significantly lower computational cost. These ROMs contain the input-output relationship of the initial system and can produce predictions of the system output for any given input within the range of the data set with almost no error. They can also give predictions outside of the data set area, but with obviously an increasing error while moving away from it. The core of the computational framework developed in this dissertation are two data-driven machine learning algorithms, the POD and the ANN.

The start of the framework is the data collection in the desired part of solution space of the system, in this case the studied CVD reactors. This is a costly procedure, so certain assumptions and modifications were made in order to reduce it. First, the size of the computational FVM grids used for the CFD modeling of the reactors was reduced as much as possible, without losing the main characteristics and properties of the initial system. This creates an accuracy problem of the final ROM predictions later, which is addressed by using a grid-to-grid interpolation, specifically the Nearest Neighbor interpolation, that "translates" the prediction derived from the coarse computational grid data to the initial high-fidelity one. Second, the chemistry aspect of the system was not included during the data collection, since in this work, the computational framework is applied to CVD processes, which usually have a dilute inlet mixture of reactants, thus the depletion or production of chemical species and the energy released or used by the reactions has a negligible effect to the reactor state. Furthermore, the method of POD and specifically its variation MoS is applied on the collected data set, which creates a low-dimensional basis that describes the data and contains more than 90% of the information. Thus, reducing the dimensionality of the problems studied in this work, more than 100 times. Although, the POD can capture the spatial dependency of the data set, it struggles with the time dependency and for the latter the ANN are used to predict the dynamic (and static) behavior of the systems. The ANN used in this work is the NARX network, which is trained with the POD basis and the same data set, thus is completely data-driven and with no information of the governing equations. Consequently, after the network is trained the final ROM (based on POD and ANN) can predict any steady state or dynamic response to an input of the studied system with low errors generally and almost no error within the data set. It is noted, that moving further from the data set the error increases.

The developed computational framework of this thesis is used on three different CVD systems and each of them propose an individual challenge. First, the framework was applied on the CVD of aluminum films from the precursor DMEAA. The challenge in this case is that the real cylindrical reactor carries a showerhead for the even distribution of the inlet gas mixture, which destroys the potential axial symmetry of the reactor, thus making the 3D CFD modeling necessary to achieve realistic results. The 3D CFD model of this reactor consists of 1.2million FVM cells, which leads to over 6million unknowns during the CFD calculations. Additionally, this case study carries a complex chemical reaction system of 3 volumetric and 9 surface reactions, which when added to the CFD model explodes the number of unknowns to more than 13million. Therefore, by applying the first steps of the developed computational framework the data were collected on a coarser grid of 300k cells and reactions were not included, to speed up the data

collection procedure. These data are snapshots of the reactor state for each time step of 0.1s throughout the trajectory of its dynamic response to a step change on the precursor DMEAA supply in the range of 1.5 to 2sccm (range of experimental study). A POD basis was created with data from 4 step changes and a basis of 2 eigenvectors that contains 80% of the information was decided from a comparison on the accuracy of the representation of a trajectory within the data set by four different bases (1, 2, 3 and 4 eigenvectors). Then the NARX network is designed with different number of neurons, 3, 5 and 10 and is trained using the backpropagation algorithm with the data from 7 step changes. The trained networks of 3, 5 and 10 neurons are tested on their accuracy of capturing the dynamic response of the CVD reactor to the DMEAA supply change, both during an internal (within the training data set) and an external (outside of training data set) step change. This comparison showed that the network of 5 and 10 neurons captures the dynamic behavior of the system for both tests, but the network of 10 neurons displayed a slight overfitting, which increased its error for the external step change. Thus, the final ROM was created for this case study that consists of a 2 eigenvectors POD basis and a NARX network of 5 neurons, that can predict the CVD reactor states for a given DMEAA flow rate input. This ROM was used to evaluate a previously developed chemical reaction network of a 2D CFD model and existing experimental findings. Although, there were no necessary changes on the reaction kinetics, since the 3D results were close to those of the 2D model, the evaluation of the chemistry was extremely sped up by using the ROM. Finally, an acceleration of 2.5 times in core hours (considering parallel processing) was achieved when reactions are included, and a 312 times acceleration when reaction are excluded, which is obvious since the ROM has no information on the chemistry. It is noted that many of the design parameters of the ROM were used in the other case studies, since this was the first one that the framework was applied, thus some are not included in the upcoming conclusions.

The second case study that the proposed computational framework was applied is the CVD of copper from the precursor copper amidinate, on the same CVD reactor of the previous case study. A new chemical reaction system was proposed for the deposition of copper films with the addition of two volumetric decomposition reactions, to capture the decrease of the copper deposition rate at temperatures higher than 573K. These volumetric reactions that achieve a depletion of the precursor copper amidinate at high temperatures are the β-hydrogen abstraction and the carbodiimide deinsertion. The kinetic parameters of each volumetric and the surface deposition reaction were fitted to existing experimental findings. The fitting procedure is extremely computationally expensive because it is mainly a trial-and-error strategy and in this case is applied on a 3D CFD model with chemistry. Thus, the use of a ROM developed

by the proposed framework sped up the fitting procedure by giving reliable predictions of the reactor states efficiently and with low computational cost. The ROM in this case study consists of a NARX network of 5 neurons that is trained on a data set of 7 step changes on the substrate temperature and with a POD basis of 3 eigenvectors. This ROM achieves a 4 times acceleration in core hours, which is a significant reduction in computational cost and is even greater when considering that in a fitting procedure the CFD model is simulated several times. Finally, the fitted chemical reaction network captures astonishingly the experimental findings throughout the studied temperature range and shows the increase and decrease of copper deposition rate at low and high substrate temperatures, respectively.

The third and final case study is a cylindrical CVD reactor that presents solution multiplicity when modelled computationally. The 2D axisymmetric CFD model of this reactor presents three distinctively different states for the same operating conditions. Two of them are observable experimentally and are on the stable branches of the solution space. The third is non-observable and lies upon the unstable branch that connects the two stable ones. This multiplicity occurs because of two competing phenomena in the chamber, the free convection and the forced convection, and because of the nonlinearity of governing PDEs. The first appears when the buoyancy is dominant, which is created due to the temperature gradient between the heated substrate and the cooled walls. The second appears when the inflow velocity of the reactive mixture is dominant. The solution multiplicity affects the deposition rate and uniformity of the produced films and without the CFD study, two different films can be produced during an experiment under the same conditions with no explanation whatsoever. Therefore, the CFD modeling of this reactor is crucial in order to capture the different solutions and be able to predict the outcome in each run. The framework in this case creates a ROM based on a limited amount of data, compared to the actual solution space, from both stable solution branches, which can efficiently, in a matter of minutes, predict both observable reactor states for any given input. The ROM was trained on a data set of well distributed 8 step changes on the inlet mass flow rate, combined with a POD basis of 4 eigenvectors. The ROM was able to predict the reactor states in a matter of seconds on both stable branches with less than 2% error for given input of the mass flow rate. Although, this case study is based on a 2D axisymmetric CFD model, which implies that there is no significant need of computational acceleration, this application is still crucial since it confirms that the proposed framework can additionally create ROMs that capture the behavior of complex nonlinear systems.

In summary, a computational framework is developed that can create ROMs of the CFD models, efficiently and with low cost, for three different CVD case studies. These ROMs deal with the challenges that appear in each study, which consist of reduction of computational cost, fitting procedure of new reaction networks and capturing the behavior of complex nonlinear systems with solution multiplicity.

Additionally, in the future, the proposed framework could be coded into one fully automated and functional plugin for the commercial CFD software ANSYS Fluent, with the use of the parallel processing UDFs. This plugin would be a fast, and user-friendly tool for the creation of ROMs of the studied CFD model, which would either utilize the ROMs for the automated computational cost reduction of the CFD simulations or even give the option to export ROMs for use in other applications. It is important to note that, since the proposed framework of this dissertation is completely data-driven and equation-free, it can be applied to other chemical processes rather than CVD and even non-CFD problems with data from a different type of source (e.g., experimental data). Finally, the developed ROMs have the potential to be used in model predictive control applications and optimizations of processes, due to their near perfect accuracy and rapid computational speed.

Communications and publications from the thesis

Journal publications

- Gkinis, P. A., Koronaki, E. D., Skouteris, A., Aviziotis, I. G., Boudouvis, A. G. (2019), *"Building a data-driven Reduced Order Model of a Chemical Vapor Deposition process from low-fidelity CFD simulations."* Chemical Engineering Science, Vol 199, pp. 371-380.
- Koronaki, E. D., Gkinis, P.A., Beex, L., Bordas, S.P.A., Theodoropoulos, C., Boudouvis, A. G. (2019), *"Classification of states and model order reduction of large scale Chemical Vapor Deposition processes with solution multiplicity."* Computers & Chemical Engineering, Vol 121, pp. 148-157.
- Spencer, R., Gkinis, P., Koronaki, E. D., Gerogiorgis, D. I., Bordas, S. P. and Boudouvis, A. G. (2021), *"Investigation of the chemical vapor deposition of Cu from copper amidinate through data driven efficient CFD modelling."* Computers & Chemical Engineering, Vol 149, pp. 107289.

Conference communications

- Gkinis, P. A., Koronaki, E. D., Boudouvis, A. G., *"A computational framework for development of Reduced Order Models of Chemical Vapor Deposition processes."* Oral Presentation at the 12th Panhellenic Scientific Conference in Chemical Engineering, Athens, Greece 29-31 May 2019.
- Gkinis, P. A., Koronaki, E. D., Boudouvis, A. G., *"Reduced Order Modeling of reactive transport: Application in CVD processes."* Oral presentation at the ECMetAC Days Poznań, Poland, 3-5 December 2018.
- Gkinis, P. A., Skouteris, A., Koronaki, E. D., Boudouvis, A. G., *"A Reduced-Order Model for efficient CFD analysis of Chemical Vapor Deposition Processes."* Oral presentation at the 9th GRACM International Congress on Computational Mechanics Chania, Greece, 4-6 June 2018.

References

[1] V. S. Ban and S. L. Gilbert, "The chemistry and transport phenomena of chemical vapor deposition of silicon from SiCl4," *Journal of Crystal Growth,* vol. 31, pp. 284-289, 1975.

[2] M. E. Coltrin, R. J. Kee and J. A. Miller, "A mathematical model of the coupled fluid mechanics and chemical kinetics in a chemical vapor deposition reactor," *Journal of the Electrochemical Society,* vol. 131, p. 425, 1984.

[3] K. F. Jensen, E. O. Einset and D. I. Fotiadis, "Flow phenomena in chemical vapor deposition of thin films," *Annual review of fluid mechanics,* vol. 23, pp. 197-232, 1991.

[4] P. A. Gkinis, I. G. Aviziotis, E. D. Koronaki, G. P. Gakis and A. G. Boudouvis, "The effects of flow multiplicity on GaN deposition in a rotating disk CVD reactor," *Journal of Crystal Growth,* vol. 458, pp. 140-148, 2017.

[5] E. D. Koronaki, P. A. Gkinis, L. Beex, S. P. Bordas, C. Theodoropoulos and A. G. Boudouvis, "Classification of states and model order reduction of large scale Chemical Vapor Deposition processes with solution multiplicity," *Computers & Chemical Engineering,* vol. 121, pp. 148-157, 2019.

[6] E. D. Koronaki, N. Cheimarios, H. Laux and A. G. Boudouvis, "Non-axisymmetric flow fields in axisymmetric CVD reactor setups revisited: influence on the film's non-uniformity," *ECS Solid State Letters,* vol. 4, p. P37, 2014.

[7] M. Kim, S. Park, D. Lee, S. Lim, M. Park and J. M. Lee, "Modeling long-time behaviors of industrial multiphase reactors for CO2 capture using CFD-based compartmental model," *Chemical Engineering Journal,* vol. 395, p. 125034, 2020.

[8] P. A. Gkinis, E. D. Koronaki, A. Skouteris, I. G. Aviziotis and A. G. Boudouvis, "Building a data-driven reduced order model of a chemical vapor deposition process from low-fidelity CFD simulations," *Chemical Engineering Science,* vol. 199, pp. 371-380, 2019.

[9] R. Spencer, P. Gkinis, E. D. Koronaki, D. I. Gerogiorgis, S. P. Bordas and A. G. Boudouvis, "Investigation of the chemical vapor deposition of Cu from copper amidinate through data driven efficient CFD modelling," *Computers & Chemical Engineering,* vol. 149, p. 107289, 2021.

[10] G. M. Psarellis, I. G. Aviziotis, T. Duguet, C. Vahlas, E. D. Koronaki and A. G. Boudouvis, "Investigation of reaction mechanisms in the chemical vapor deposition of al from DMEAA," *Chemical Engineering Science,* vol. 177, p. 464–470, February 2018.

[11] G. Berkooz, P. Holmes and J. L. Lumley, "The proper orthogonal decomposition in the analysis of turbulent flows," *Annual review of fluid mechanics,* vol. 25, pp. 539-575, 1993.

[12] A. E. Deane, Kevrekidis, I. G., G. E. Karniadakis and S. A. Orszag, "Low-dimensional models for complex geometry flows: application to grooved channels and circular cylinders," *Physics of Fluids A: Fluid Dynamics,* vol. 3, pp. 2337-2354, 1991.

[13] M. Fahl, "Computation of POD basis functions for fluid flows with Lanczos methods," *Mathematical and computer modelling,* vol. 34, pp. 91-107, 2001.

[14] L. Sirovich, "Turbulence and the dynamics of coherent structures. Part I-III," *Quarterly of Applied Mathematics,* pp. 561-590, 1987.

[15] B. Moore, "Principal component analysis in linear systems: Controllability, observability, and model reduction," *IEEE transactions on automatic control,* vol. 26, pp. 17-32, 1981.

[16] W. Xie, I. Bonis and C. Theodoropoulos, "Linear MPC based on data-driven Artificial Neural Networks for large-scale nonlinear distributed parameter systems," in *Proceedings of the 22nd European Symposium on Computer Aided Process Engineering*, London, 2012.

[17] K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks,* vol. 2, pp. 359-366, 1989.

[18] Lewis FW, Jagannathan S and Y. A., Neural network control of robot manipulators and non-linear systems, CRC PRESS, 2020.

[19] W. Xie, I. Bonis and C. Theodoropoulos, "Data-driven model reduction-based nonlinear MPC for large-scale distributed parameter systems," *Journal of Process Control,* vol. 35, pp. 50-58, 2015.

[20] A. K. Jain, J. Mao and K. Mohiuddin, "Artificial neural networks: a tutorial," *Computer,* vol. 29, no. 3, pp. 31-44, March 1996 .

[21] J. Moyne and J. Iskandar, "Big data analytics for smart manufacturing: Case studies in semiconductor manufacturing," *Processes,* vol. 5, p. 39, 2017.

[22] N. Cheimarios, E. D. Koronaki and A. G. Boudouvis, "Enabling a commercial computational

fluid dynamics code to perform certain nonlinear analysis tasks," *Computers & chemical engineering,* vol. 35, pp. 2632-2645, 2011.

[23] N. Cheimarios, E. D. Koronaki and A. G. Boudouvis, "Illuminating nonlinear dependence of film deposition rate in a CVD reactor on operating conditions," *Chemical Engineering Journal,* vol. 181, pp. 516-523, 2012.

[24] H. O. Pierson, Handbook of chemical vapor deposition: principles, technology and applications, Noyes Publications/ William Andrew, 1999.

[25] X. Hou and K.-L. Choy, "Processing and Applications of Aerosol-Assisted Chemical Vapor Deposition," *Chemical Vapor Deposition,* vol. 12, no. 10, pp. 583-596, October 2006.

[26] J.-H. Park and T. S. Sudarshan, Chemical Vapor Deposition, 2nd ed., Materials Park, OH: ASM International, 2001.

[27] H. Pedersen and S. D. Elliott, "Studying chemical vapor deposition processes with theoretical chemistry," *Theoretical Chemistry Accounts,* vol. 133, pp. 1-10.

[28] I. G. Aviziotis, "Chemical vapor deposition of Al, Fe and of the Al13Fe4 approximant intermetallic phase: Experiments and multiscale simulations," Athens, 2016.

[29] N. Mikoshiba, K. Tsubouchi and K. Masu, "Plasma CVD of aluminum films". USA Patent US5091210 A, 25 February 1992.

[30] R. M, "High Temperature Materials, Electronics, and Dielectics and Insulation Divisions," in *Proceedings of the Ninth International Conference on Chemical Vapor Deposition*, 1984.

[31] M. J. Hampden-Smith and T. T. Kodas, "Chemical vapor deposition of metals: Part 1. An overview of CVD processes," *Chemical Vapor Deposition,* vol. 1, no. 1, p. 8–23, July 1995.

[32] T. D. Manning, I. P. Parkin, M. E. Pemble, D. Sheel and D. Vernardou, "Intelligent window coatings: atmospheric pressure chemical vapor deposition of tungsten-doped vanadium dioxide," *Chemistry of Materials,* vol. 16, pp. 744-749, 2004.

[33] I. G. Aviziotis, T. Duguet, K. Soussi, G. Kokkoris, N. Cheimarios, C. Vahlas and A. G. Boudouvis, "Investigation of the kinetics of the chemical vapor deposition of aluminum from dimethylethylamine alane: experiments and computations," *Physica status solidi,* vol. 12, pp. 923-930, 2015.

[34] I. G. Aviziotis, N. Cheimarios, T. Duguet, C. Vahlas and A. G. Boudouvis, "Multiscale

modeling and experimental analysis of chemical vapor deposited aluminum films: linking reactor operating conditions with roughness evolution," *Chemical Engineering Science,* vol. 155, pp. 449-458, 2016.

[35] T. C. Xenidou, A. G. Boudouvis, N. C. Markatos, D. Samélor, F. Senocq, N. PrudHomme and C. Vahlas, "An experimental and computational analysis of a MOCVD process for the growth of Al films using DMEAA," *Surface and Coatings Technology,* vol. 201, pp. 8868-8872, 2007.

[36] D. Vernardou, M. E. Pemble and D. W. Sheel, "The Growth of Thermochromic VO2 Films on Glass by Atmospheric-Pressure CVD: A Comparative Study of Precursors, CVD Methodology, and Substrates," *Chemical Vapor Deposition,* vol. 12, pp. 263-274, 2006.

[37] T. C. Xenidou, N. Prud'homme, C. Vahlas, N. C. Markatos and A. G. Boudouvis, "Reaction and transport interplay in Al MOCVD investigated through experiments and computational fluid dynamic analysis," *Journal of The Electrochemical Society,* vol. 157, p. D633, 2010.

[38] F. E. Rosenberger, Fundamentals of Crystal Growth I: Macroscopic Equilibrium and Transport Concepts, vol. 5, Springer Science & Business Media, 2012.

[39] D. E. Rosner, "Thermal/Soret/diffusion effects on interfacial mass transport rates," *Physicochemical Hydrodynamics,* vol. 1, pp. 159-185, 1980.

[40] D. I. Fotiadis, S. Kieda and K. F. Jensen, "Transport phenomena in vertical reactors for metalorganic vapor phase epitaxy: I. Effects of heat transfer characteristics, reactor geometry, and operating conditions," *Journal of crystal growth,* vol. 102, pp. 441-470, 1990.

[41] I. G. Aviziotis, N. Cheimarios, C. Vahlas and A. G. Boudouvis, "Experimental and computational investigation of chemical vapor deposition of Cu from Cu amidinate," *Surface and Coatings Technology,* vol. 230, pp. 273-278, 2013.

[42] B. Mitrovic, A. Gurary and W. Quinn, "Process conditions optimization for the maximum deposition rate and uniformity in vertical rotating disc MOCVD reactors based on CFD modeling," *Journal of crystal growth,* vol. 303, pp. 323-329, 2007.

[43] P. G. Gordon, A. Kurek and S. T. Barry, "Trends in copper precursor development for CVD and ALD applications," *ECS Journal of Solid State Science and Technology,* vol. 4, p. N3188, 2014.

[44] J. P. Coyle, P. A. Johnson, G. A. DiLabio, S. T. Barry and J. Müller, "Gas-phase thermolysis of a guanidinate precursor of copper studied by matrix isolation, time-of-flight mass spectrometry, and computational chemistry," *Inorganic chemistry,* vol. 49, pp. 2844-2850, 2010.

[45] M. Rasadujjaman, M. Watanabe, H. Sudoh, H. Machida and E. Kondoh, "Supercritical fluid chemical deposition of Cu in Ru and TiN-lined deep nanotrenches using a new Cu (I) amidinate precursor," *Microelectronic Engineering,* vol. 137, pp. 32-36, 2015.

[46] V. Krisyuk, L. Aloui, N. Prud'Homme, B. Sarapata, F. Senocq, D. Samelor and C. Vahlas, "CVD of pure copper films from a novel amidinate precursor," *ECS Transactions,* vol. 25, p. 581, 2009.

[47] N. Prud'homme, V. Constantoudis, A. E. Turgambaeva, V. V. Krisyuk, D. Samélor, F. Senocq and C. Vahlas, "Chemical vapor deposition of Cu films from copper (I) cyclopentadienyl triethylphophine: Precursor characteristics and interplay between growth parameters and films morphology," *Thin Solid Films,* vol. 701, p. 137967, 2020.

[48] B. E. Bent, R. G. Nuzzo and L. H. Dubois, "Surface organometallic chemistry in the chemical vapor deposition of aluminum films using triisobutylaluminum:. beta.-hydride and. beta.-alkyl elimination reactions of surface alkyl intermediates," *Journal of the American Chemical Society,* vol. 111, pp. 1634-1644, 1989.

[49] T. Nakajima, M. Nakatomi and K. Yamashita, "Quantum chemical calculations on Al-CVD using DMEAA: surface reaction mechanism of AlH3 on Al (111)," *Molecular Physics,* vol. 101, pp. 267-276, 2003.

[50] H. K. Versteeg and W. Malalasekera, An introduction to computational fluid dynamics: the finite volume method, Pearson education, 2007.

[51] C. Theodoropoulos, T. J. Mountziaris, H. K. Moffat and J. Han, "Design of gas inlets for the growth of gallium nitride by metalorganic vapor phase epitaxy," *Journal of crystal growth,* vol. 217, pp. 65-81, 2000.

[52] X. Gao, T. Li, W. A. Rogers, K. Smith, K. Gaston, G. Wiggins and J. E. Parks II, "Validation and application of a multiphase CFD model for hydrodynamics, temperature field and RTD simulation in a pilot-scale biomass pyrolysis vapor phase upgrading reactor," *Chemical Engineering Journal,* vol. 388, p. 124279, 2020.

[53] M. Crose, J. S. I. Kwon, A. Tran and P. D. Christofides, "Multiscale modeling and run-to-run

control of PECVD of thin film solar cells," *Renewable Energy,* vol. 100, pp. 129-140, 2017.

[54] A. C. Antoulas and D. C. Sorensen, "Approximation of large-scale dynamical systems: An Overview," 2001. [Online]. Available: https://hdl.handle.net/1911/101964.

[55] E. D. Koronaki, G. P. Gakis, N. Cheimarios and A. G. Boudouvis, "Efficient tracing and stability analysis of multiple stationary and periodic states with exploitation of commercial CFD software," *Chemical Engineering Science,* vol. 150, pp. 26-34, 2016.

[56] X. Lu, P. Xie, D. B. Ingham, L. Ma and M. Pourkashanian, "A porous media model for CFD simulations of gas-liquid two-phase flow in rotating packed beds," *Chemical Engineering Science,* vol. 189, pp. 123-134, 2018.

[57] G. Wahl, "Hydrodynamic description of CVD processes," *Thin Solid Films,* vol. 40, pp. 13-26, 1977.

[58] ANSYS Inc, Ansys Fluent Theory Guide, 12 ed., 2009.

[59] G. P. Gakis, H. Vergnes, E. Scheid, C. Vahlas, B. Caussat and A. G. Boudouvis, "Computational fluid dynamics simulation of the ALD of alumina from TMA and H2O in a commercial reactor," *Chemical Engineering Research and Design,* vol. 132, pp. 795-811, 2018.

[60] ANSYS Inc, Ansys Fluent User's Guide, 12 ed., 2009.

[61] B. Sjodin, April 2016. [Online]. Available: http://www.machinedesign.com/fea-and-simulation/what-s-difference-between-fem-fdm-and-fvm.

[62] P. G. Huang, B. E. Launder and M. A. Leschziner, "Discretization of nonlinear convection processes: a broad-range comparison of four schemes," *Computer methods in applied mechanics and engineering,* vol. 48, pp. 1-24.

[63] ANSYS Inc, Ansys Fluent UDF Manual, 12 ed., 2009.

[64] R. Franke, "Scattered data interpolation: tests of some methods," *Mathematics of Computation,* no. 38, pp. 181-200 , 1982.

[65] W. H. A. Schilders, H. A. Vorst and J. Rommes, Model Order Reduction: Theory, Research Aspects and Applications, Springer, 2008.

[66] P. S. B. Nigro, M. Anndif, Y. Teixeira, P. M. Pimenta and P. Wriggers, "An adaptive model

order reduction by proper snapshot selection for nonlinear dynamical problems," vol. 57, pp. 537-554, 2016.

[67] K. Li, T. Z. Huang, L. Li and S. Lanteri, "POD-based model order reduction with an adaptive snapshot selection for a discontinuous Galerkin approximation of the time-domain Maxwell's equations," *Journal of Computational Physics,* vol. 396, pp. 106-128, 2019.

[68] B. Koo, T. Jo and D. Lee, "Modified inferential POD/ML for data-driven inverse procedure of steam reformer for 5-kW HT-PEMFC," *Computers & Chemical Engineering,* vol. 121, pp. 375-387, 2019.

[69] S. Dey and A. Dhar, "On proper orthogonal decomposition (POD) based reduced-order modeling of groundwater flow through heterogeneous porous media with point source singularity," *Advances in Water Resources,* vol. 144, p. 103703, 2020.

[70] M. Rathinam and L. R. Petzold, "A new look at proper orthogonal decomposition," *SIAM Journal on Numerical Analysis,* vol. 41, pp. 1893-1925, 2003.

[71] M. Hinze and S. Volkwein, "Proper orthogonal decomposition surrogate models for nonlinear dynamical systems: Error estimates and suboptimal control," in *Dimension reduction of large-scale systems*, 2005.

[72] K. Afanasiev and M. Hinze, "Adaptive control of a wake flow using proper orthogonal decomposition," *Lecture Notes in Pure and Applied Mathematics,* vol. 216, pp. 317-332, 2001.

[73] A. J. Newman, "Model Reduction via the Karhunen-Loeve Expansion Part I: An Exposition," College Park, MD, 1996.

[74] M. F. Barone, D. J. Segalman, H. Thornquist and I. Kalashnikova, "Galerkin Reduced Order Models for Compressible Flow with Structural Interaction," in *46th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 2008.

[75] Y. Wang, H. Ma, W. Cai, H. Zhang, J. Cheng and X. Zheng, "A POD-Galerkin reduced-order model for two-dimensional Rayleigh-Bénard convection with viscoelastic fluid," *International Communications in Heat and Mass Transfer,* vol. 117, p. 104747, 2020.

[76] D. Sipp, M. F. de Pando and P. J. Schmid, "Nonlinear model reduction: A comparison between POD-Galerkin and POD-DEIM methods," *Computers & Fluids,* vol. 208, p. 104628, 2020.

[77] J. H. Lee, J. Shin and M. J. Realff, "Machine learning: Overview of the recent progresses and implications for the process systems engineering field," *Computers & Chemical Engineering,* vol. 114, pp. 111-121, 2018.

[78] I. Bright, G. Lin and J. N. Kutz, "Compressive sensing based machine learning strategy for characterizing the flow around a cylinder with limited pressure measurements," *Physics of Fluids,* vol. 25, p. 127102, 2013.

[79] O. Baghirli, "Comparison of Lavenberg-Marquardt, scaled conjugate gradient and Bayesian regularization backpropagation algorithms for multistep ahead wind speed forecasting using multilayer perceptron feedforward neural network," Uppsala, 2015.

[80] M. Kayri, "Predictive Abilities of Bayesian Regularization and Levenberg–Marquardt Algorithms in Artificial Neural Networks: A Comparative Empirical Study on Social Data," *Mathematical and Computational Applications,* vol. 21, no. 2, p. 20, May 2016.

[81] A. S. Alshehri, R. Gani and F. You, "Deep learning and knowledge-based methods for computer-aided molecular design—toward a unified approach: State-of-the-art and future directions," *Computers & Chemical Engineering,* p. 107005, 2020.

[82] The MathWorks, Neural Network Toolbox™ User's Guide (MATLAB), 6th ed., 2009.

[83] M. Bracconi and M. Maestri, "Training set design for machine learning techniques applied to the approximation of computationally intensive first-principles kinetic models," *Chemical Engineering Journal,* vol. 400, p. 125469, 2020.

[84] F. D. Foresee and M. T. Hagan, "Gauss-Newton approximation to Bayesian learning," in *Proceedings of international conference on neural networks (ICNN'97)*, 1997.

[85] D. J. MacKay, "Bayesian interpolation," *Neural computation,* vol. 4, pp. 415-447, 1992.

[86] P. Kumar, S. Merchant and U. Desai, "Improving performance in pulse radar detection using Bayesian regularization for neural network training," *Digital Signal Processing,* vol. 14, no. 5, pp. 438-448, September 2004.

[87] H. Xie, H. Tang and Y. H. Liao, "Time series prediction based on NARX neural networks: An advanced approach," in *2009 International conference on machine learning and cybernetics*, 2009.

[88] H. T. Siegelmann, B. G. Horne and C. L. Giles, "Computational Capabilities of Recurrent NARX Neural Networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*

(Cybernetics), vol. 27, no. 2, pp. 208 - 215, April 1997.

[89] T.-N. Lin, C. L. Giles, B. G. Horne and S. Kung, "A delay damage model selection algorithm for NARX neural networks," *IEEE Transactions on Signal Processing,* vol. 45, no. 11, pp. 2719 - 2730, November 1997 .

[90] X. Zhu, Z. Wan, D. C. Tsang, M. He, D. Hou, Z. Su and J. Shang, "Machine learning for the selection of carbon-based materials for tetracycline and sulfamethoxazole adsorption," *Chemical Engineering Journal,* vol. 406, p. 126782, 2021.

[91] Y. Yan, T. Mattisson, P. Moldenhauer, E. J. Anthony and P. T. Clough, "Applying machine learning algorithms in estimating the performance of heterogeneous, multi-component materials as oxygen carriers for chemical-looping processes," *Chemical Engineering Journal,* vol. 387, p. 124072, 2020.

[92] B. Kiran, A. K. Kandalam, J. Xu, Y. H. Ding, M. Sierka, K. H. Bowen and H. Schnöckel, "Al6H18: A baby crystal of γ-AlH3," *The Journal of chemical physics,* vol. 137, p. 134303, 2012.

[93] F. M. Mwema, O. P. Oladijo, S. A. Akinlabi and E. T. Akinlabi, "Properties of physically deposited thin aluminium film coatings: A review," *Journal of alloys and compounds,* vol. 747, pp. 306-323, 2018.

[94] Z. Li, S. T. Barry and R. G. Gordon, "Synthesis and characterization of copper (I) amidinates as precursors for atomic layer deposition (ALD) of copper metal," *Inorganic chemistry,* vol. 44, pp. 1728-1735, 2005.

[95] B. S. Lim, A. Rahtu, J. S. Park and R. G. Gordon, "Synthesis and characterization of volatile, thermally stable, reactive transition metal amidinates," *Inorganic chemistry,* vol. 42, pp. 7951-7958, 2003.

[96] Z. Li, A. Rahtu and R. G. Gordon, "Atomic layer deposition of ultrathin copper metal films from a liquid copper (I) amidinate precursor," *ournal of The Electrochemical Society,* vol. 153, p. C787, 2006.

[97] A. Turgambaeva, N. Prud'homme, V. Krisyuk and C. Vahlas, "Decomposition Schemes of Copper (I) N, N’-Diisopropylacetamidinate During Chemical Vapor Deposition of Copper," *ournal of nanoscience and nanotechnology,* vol. 11, pp. 8198-8201, 2011.

[98] J. Wang, R. B. Little, W. G. Lai and G. L. Griffin, "Reactor transport effects in copper

APCVD," *Thin solid films,* vol. 262, pp. 31-38, 1995.

[99] M. Juppo, A. Rahtu, M. Ritala and M. Leskelä, "In situ mass spectrometry study on surface reactions in atomic layer deposition of Al2O3 thin films from trimethylaluminum and water," *Langmuir,* vol. 16, pp. 4034-4039, 2000.

[100] R. L. Puurunen, "Surface chemistry of atomic layer deposition: A case study for the trimethylaluminum/water process," *Journal of applied physics,* vol. 97, p. 9, 2005.

[101] Q. Ma, H. Guo, R. G. Gordon and F. Zaera, "Surface chemistry of copper (I) acetamidinates in connection with atomic layer deposition (ALD) processes," *Chemistry of Materials,* vol. 23, pp. 3325-3334, 2011.

[102] S. T. Barry, "Amidinates, guanidinates and iminopyrrolidinates: Understanding precursor thermolysis to design a better ligand," *Coordination Chemistry Reviews,* vol. 257, pp. 3192-3201, 2013.

[103] S. T. Barry, A. V. Teplyakov and F. Zaera, "The chemistry of inorganic precursors during the chemical deposition of films on solid surfaces," *Accounts of chemical research,* vol. 51, pp. 800-809, 2018.

[104] J. P. Coyle, A. Kurek, P. J. Pallister, E. R. Sirianni, G. P. Yap and S. T. Barry, "Preventing thermolysis: precursor design for volatile copper compounds," *Chemical Communications,* vol. 48, pp. 10440-10442, 2012.

[105] C. R. Kleijn, "A mathematical model of the hydrodynamics and gas-phase reactions in silicon LPCVD in a single-wafer reactor," *Journal of the Electrochemical Society,* vol. 138, p. 2190, 1991.

[106] H. van Santen, C. R. Kleijn and H. E. van den Akker, "On multiple stability of mixed-convection flows in a chemical vapor deposition reactor," *International journal of heat and mass transfer,* vol. 44, pp. 659-672, 2001.

[107] R. Curley, T. McCormack and M. Phipps, "Low-pressure CVD and Plasma-Enhanced CVD," College Park, MD, 2011.

[108] N. Li, Y.-H. A. Wang, M. N. Iliev, T. M. Klein and A. Gupta, "Growth of Atomically Smooth Epitaxial Nickel Ferrite Films by Direct Liquid Injection CVD," *Chemical Vapor Deposition,* vol. 17, no. 7-9, p. 261–269, September 2011.

[109] L. G. Hubert-Pfalzgraf and H. Guillon, "Trends in precursor design for conventional and aerosol-assisted CVD of high- T c superconductors," *Applied Organometallic Chemistry,* vol. 12, no. 3, pp. 221-236, 1998.

[110] M. L. Green, R. A. Levy, R. G. Nuzzo and E. Coleman, "Aluminum films prepared by metal-organic low pressure chemical vapor deposition," *Thin Solid Films,* vol. 114, no. 4, pp. 367-377, April 1984.

[111] J. H. Yun, M. Y. Park and S. W. Rhee, "Fourier transform infrared diagnostics of gas phase reactions in the metalorganic chemical vapor deposition of aluminum from dimethylethylamine alane," *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films,* vol. 16, pp. 419-423, 1998.

[112] T. Nishikawa, K. Horiuchi, T. Joutsuka and S. Yamauchi, "Low-pressure chemical vapor deposition of Cu on Ru using CuI as precursor.," *Journal of Crystal Growth,* vol. 549, p. 125849, 2020.

Appendix

The UDFs developed for the serial and parallel data collection from the ANSYS Fluent simulations are the following.

The "writing" UDF, which writes any variable value for the whole computational grid or for a selected area and it can be used in both serial and parallel processing.

```
/************************************************************************
This function will write the values of velocity components (XYZ),
pressure and temperature of each cell center for a fluid zone to a
file on the host machine. Run serial or parallel.
Pavlos Gkinis
************************************************************************/
#include "udf.h"
#include <stdio.h>

# define FLUID_ID 2 /* Define the zone ID for the fluid zone */
# define TOTAL_CELLS 1232468 /*Define the total number of cells*/
```

```
DEFINE_EXECUTE_AT_EXIT(Writing_parallel_pg)
{
 /* For #<command> refer to pre-processor directives */

  /* Different variables are needed on different nodes */

/* Action-1 initializing the domain,thread and cells */
#if !RP_HOST
 Domain *domain=Get_Domain(1);
 Thread *thread;
 cell_t c;
#else
 int i;
#endif

/* Action-2 creating the data file */
#if !RP_NODE
 FILE *fp = NULL;
 char filename[]="Initial_Solution.txt";
#endif

/* Action-3 iniatializing the buffer arrays */
#if PARALLEL
 int size;  /* data passing variables */
 real *array;
 real *array1;
 real *array2;  /* BUFFERS */
 real *array3;
 real *array4;

 int pe;
#endif

 /* Only Serial and Compute Nodes have data on threads */
#if !RP_HOST
 thread=Lookup_Thread(domain,FLUID_ID);
#endif

/* Action-4 opening the file */

#if !RP_NODE /* SERIAL or HOST */
 if ((fp = fopen(filename, "w"))==NULL)
     Message("\n Warning: Unable to open %s for writing\n",filename);
 else
     Message("\nWriting Variables to %s...",filename);
#endif
```

```
 /* UDF Now does 3 different things depending on SERIAL, NODE or HOST */

#if !PARALLEL /* SERIAL */
/* Simply write out the data */

/* loop over all cell threads in the domain  */
thread_loop_c(thread,domain)
{
/* loop over all cells */
 begin_c_loop_all(c,thread)
   fprintf(fp,"%8.6e\n", C_U(c,thread));
 end_c_loop_all(c,thread)
}
/* loop over all cell threads in the domain  */
thread_loop_c(thread,domain)
{
/* loop over all cells */
 begin_c_loop_all(c,thread)
   fprintf(fp,"%8.6e\n", C_V(c,thread));
 end_c_loop_all(c,thread)
}
/* loop over all cell threads in the domain  */
thread_loop_c(thread,domain)
{
/* loop over all cells */
 begin_c_loop_all(c,thread)
   fprintf(fp,"%8.6e\n", C_W(c,thread));
 end_c_loop_all(c,thread)
}
/* loop over all cell threads in the domain  */
thread_loop_c(thread,domain)
{
/* loop over all cells */
 begin_c_loop_all(c,thread)
   fprintf(fp,"%8.6e\n", C_P(c,thread));
 end_c_loop_all(c,thread)
}
/* loop over all cell threads in the domain  */
thread_loop_c(thread,domain)
{
/* loop over all cells */
 begin_c_loop_all(c,thread)
   fprintf(fp,"%8.6e\n", C_T(c,thread));
 end_c_loop_all(c,thread)
}
```

```
#endif /* !PARALLEL */

 /* Action-5 Each Node loads up its data passing array */
#if RP_NODE
 size=THREAD_N_ELEMENTS_INT(thread);
 array = (real *)malloc(size * sizeof(real));
 array1 = (real *)malloc(size * sizeof(real));
 array2 = (real *)malloc(size * sizeof(real));  /* BUFFERS */
 array3 = (real *)malloc(size * sizeof(real));
 array4 = (real *)malloc(size * sizeof(real));

 begin_c_loop_int(c,thread)
  array[c]= C_U(c,thread);
 end_c_loop_int(c,thread)

 begin_c_loop_int(c,thread)
  array1[c]= C_V(c,thread);
 end_c_loop_int(c,thread)

begin_c_loop_int(c,thread)
 array2[c]= C_W(c,thread); /*Fluent variables to buffer arrays*/
 end_c_loop_int(c,thread)

 begin_c_loop_int(c,thread)
  array3[c]= C_P(c,thread);
 end_c_loop_int(c,thread)

 begin_c_loop_int(c,thread)
  array4[c]= C_T(c,thread);
 end_c_loop_int(c,thread)


  /* Set pe to destination node */
  /* If on node_0 send data to host */
  /* Else send to node_0 because */
  /*   compute nodes connect to node_0 & node_0 to host */
 pe = (I_AM_NODE_ZERO_P) ? node_host : node_zero;

 PRF_CSEND_INT(pe, &size, 1, myid);
 PRF_CSEND_REAL(pe, array, size, myid);
 PRF_CSEND_REAL(pe, array1, size, myid);
 PRF_CSEND_REAL(pe, array2, size, myid); /*Sending arrays*/
 PRF_CSEND_REAL(pe, array3, size, myid);
 PRF_CSEND_REAL(pe, array4, size, myid);

 free(array);
```

```
 free(array1);
 free(array2);   /* free arrays on nodes once data sent */
 free(array3);
 free(array4);
 /* node_0 now collect data sent by other compute nodes */
 /*   and sends it straight on to the host */
 if (I_AM_NODE_ZERO_P)

/*!!!!THIS LOOP RUNS for node-1 -> .... -> node-n IN ODRER!!!!*/

/*Action-6 send data to host*/
  compute_node_loop_not_zero (pe)
  {
    PRF_CRECV_INT(pe, &size, 1, pe);
    array = (real *)malloc(size * sizeof(real));
    array1 = (real *)malloc(size * sizeof(real));
    array2 = (real *)malloc(size * sizeof(real));/*BUFFERS*/
    array3 = (real *)malloc(size * sizeof(real));
    array4 = (real *)malloc(size * sizeof(real));

    PRF_CRECV_REAL(pe, array, size, pe);
    PRF_CRECV_REAL(pe, array1, size, pe);
    PRF_CRECV_REAL(pe, array2, size, pe);
    PRF_CRECV_REAL(pe, array3, size, pe);
    PRF_CRECV_REAL(pe, array4, size, pe);



    PRF_CSEND_INT(node_host, &size, 1, myid);
    PRF_CSEND_REAL(node_host, array, size, myid);
    PRF_CSEND_REAL(node_host, array1, size, myid);
    PRF_CSEND_REAL(node_host, array2, size, myid);
    PRF_CSEND_REAL(node_host, array3, size, myid);
    PRF_CSEND_REAL(node_host, array4, size, myid);


          free((char *)array);
          free((char *)array1);
          free((char *)array2);
          free((char *)array3);
          free((char *)array4);
  }

/*Send with correct order, now that all data has been collected*/
```

```
#endif /* RP_NODE */

/*Action-6 the host receives the data and prints them into the file*/

#if RP_HOST
real *total_array_U;
real *total_array_V;
real *total_array_W; /*Initializing total arrays for each var*/
real *total_array_P;
real *total_array_T;

/*running_size is used to insert the data of the next node after
the last one in the file (cause each node has its own number of cells)*/

int running_size = 0;

/*BUFFERS*/
total_array_U=(real *)malloc(TOTAL_CELLS * sizeof(real));
total_array_V=(real *)malloc(TOTAL_CELLS * sizeof(real));
total_array_W=(real *)malloc(TOTAL_CELLS * sizeof(real));
total_array_P=(real *)malloc(TOTAL_CELLS * sizeof(real));
total_array_T=(real *)malloc(TOTAL_CELLS * sizeof(real));

/*pe only acts as a counter in this loop*/
 compute_node_loop (pe)
  {
    /* Receive data sent by each node and write it out to the file */
    PRF_CRECV_INT(node_zero, &size, 1, node_zero);
    array = (real *)malloc(size * sizeof(real));
    array1 = (real *)malloc(size * sizeof(real));
    array2 = (real *)malloc(size * sizeof(real)); /*BUFFERS*/
    array3 = (real *)malloc(size * sizeof(real));
    array4 = (real *)malloc(size * sizeof(real));

    PRF_CRECV_REAL(node_zero, array, size, node_zero);
    PRF_CRECV_REAL(node_zero, array1, size, node_zero);
    PRF_CRECV_REAL(node_zero, array2, size, node_zero);
    PRF_CRECV_REAL(node_zero, array3, size, node_zero);
    PRF_CRECV_REAL(node_zero, array4, size, node_zero);

    /*Concatenate data into 1 array for each variable*/
    memcpy(total_array_U+running_size, array, size*sizeof(real));
    memcpy(total_array_V+running_size, array1, size*sizeof(real));
    memcpy(total_array_W+running_size, array2, size*sizeof(real));
    memcpy(total_array_P+running_size, array3, size*sizeof(real));
    memcpy(total_array_T+running_size, array4, size*sizeof(real));
```

```
    running_size += size;

    free(array);
    free(array1);
    free(array2);
    free(array3);
    free(array4);
  }

/* Printing the total arrays into the file for each variable */
for (i=0; i<running_size; i++)
        fprintf(fp, "%8.6e\n", total_array_U[i]);

for (i=0; i<running_size; i++)
        fprintf(fp, "%8.6e\n", total_array_V[i]);

for (i=0; i<running_size; i++)
        fprintf(fp, "%8.6e\n", total_array_W[i]);

for (i=0; i<running_size; i++)
        fprintf(fp, "%8.6e\n", total_array_P[i]);

for (i=0; i<running_size; i++)
        fprintf(fp, "%8.6e\n", total_array_T[i]);

free(total_array_U);
free(total_array_V);
free(total_array_W);
free(total_array_P);
free(total_array_T);

#endif /* RP_HOST */

/* Action-7 closing the file */
#if !RP_NODE /* SERIAL or HOST */
 fclose(fp); /* Close the file that was only opened if on SERIAL or HOST */
 Message("Done\n");
#endif

}
```

The next UDF can additionally (to the previous one) write the coordinates of the respective computational cell for every variable, thus giving indicators to each value for their position in space.

```
/*********************************************************************
This function will write the XYZ position, velocity components (XYZ),
pressure and temperature of each cell center for the fluid zone to a
file on the host machine. Run serial or parallel.
Pavlos Gkinis
*********************************************************************/
#include "udf.h"
#include <stdio.h>
#include <string.h>

# define FLUID_ID 3 /* Define the zone ID for the fluid zone */
# define TOTAL_CELLS 534802 /*Define the total number of cells*/

DEFINE_EXECUTE_AT_END(Writing_parallel_pg)
{
 /* For #<command> refer to pre-processor directives */

  /* Different variables are needed on different nodes */

/* Action-1 initializing the domain,thread and cells */
#if !RP_HOST
 Domain *domain=Get_Domain(1);
 Thread *thread;
 cell_t c;
#else
 int i;
#endif

/* Action-2 creating the data file */
#if !RP_NODE
 FILE *fp = NULL;
 char time_step[100];
 snprintf(time_step,10,"%f",CURRENT_TIME);
 char filename[]="Initial_Solution";
 strcat(filename,time_step);

#endif

/* Action-3 iniatializing the buffer arrays */
#if PARALLEL
 int size;  /* data passing variables */
 real *array;
 real *array1;
 real *array2;  /* BUFFERS */
 real *array3;
 real *array4;
```

```
 real *arrayx;
 real *arrayy;
 real *arrayz;

 int pe;
#endif

 /* Only Serial and Compute Nodes have data on threads */
#if !RP_HOST
 thread=Lookup_Thread(domain,FLUID_ID);
#endif

/* Action-4 opening the file */

#if !RP_NODE /* SERIAL or HOST */
 if ((fp = fopen(filename, "w"))==NULL)
     Message("\n Warning: Unable to open %s for writing\n",filename);
 else
     Message("\nWriting Variables to %s...",filename);
#endif

 /* UDF Now does 3 different things depending on SERIAL, NODE or HOST */

#if !PARALLEL /* SERIAL */
/* Simply write out the data */

/* loop over all cell threads in the domain  */
thread_loop_c(thread,domain)
{
/* loop over all cells */
 begin_c_loop_all(c,thread)
   fprintf(fp,"%8.6e\n", C_U(c,thread));
 end_c_loop_all(c,thread)
}
/* loop over all cell threads in the domain  */
thread_loop_c(thread,domain)
{
/* loop over all cells */
 begin_c_loop_all(c,thread)
   fprintf(fp,"%8.6e\n", C_V(c,thread));
 end_c_loop_all(c,thread)
}
/* loop over all cell threads in the domain  */
thread_loop_c(thread,domain)
{
/* loop over all cells */
```

```
 begin_c_loop_all(c,thread)
   fprintf(fp,"%8.6e\n", C_W(c,thread));
 end_c_loop_all(c,thread)
}
/* loop over all cell threads in the domain  */
thread_loop_c(thread,domain)
{
/* loop over all cells */
 begin_c_loop_all(c,thread)
   fprintf(fp,"%8.6e\n", C_P(c,thread));
 end_c_loop_all(c,thread)
}
/* loop over all cell threads in the domain  */
thread_loop_c(thread,domain)
{
/* loop over all cells */
 begin_c_loop_all(c,thread)
   fprintf(fp,"%8.6e\n", C_T(c,thread));
 end_c_loop_all(c,thread)
}
#endif /* !PARALLEL */

 /* Action-5 Each Node loads up its data passing array */
#if RP_NODE

 real xc[ND_ND];
 size=THREAD_N_ELEMENTS_INT(thread);
 array = (real *)malloc(size * sizeof(real));
 array1 = (real *)malloc(size * sizeof(real));
 array2 = (real *)malloc(size * sizeof(real));  /* BUFFERS */
 array3 = (real *)malloc(size * sizeof(real));
 array4 = (real *)malloc(size * sizeof(real));
 arrayx = (real *)malloc(size * sizeof(real));
 arrayy = (real *)malloc(size * sizeof(real));
 arrayz = (real *)malloc(size * sizeof(real));

 begin_c_loop_int(c,thread)
   array[c]= C_U(c,thread);
 end_c_loop_int(c,thread)

 begin_c_loop_int(c,thread)
   array1[c]= C_V(c,thread);
 end_c_loop_int(c,thread)

begin_c_loop_int(c,thread)
  array2[c]= C_W(c,thread); /*Fluent variables to buffer arrays*/
```

```
end_c_loop_int(c,thread)

begin_c_loop_int(c,thread)
  array3[c]= C_P(c,thread);
end_c_loop_int(c,thread)

begin_c_loop_int(c,thread)
  array4[c]= C_T(c,thread);
end_c_loop_int(c,thread)

begin_c_loop_int(c,thread)
C_CENTROID(xc,c,thread);
  arrayx[c]=xc[0];
  arrayy[c]=xc[1];
  arrayz[c]=xc[2];
end_c_loop_int(c,thread)

  /* Set pe to destination node */
  /* If on node_0 send data to host */
  /* Else send to node_0 because */
  /*   compute nodes connect to node_0 & node_0 to host */
pe = (I_AM_NODE_ZERO_P) ? node_host : node_zero;

PRF_CSEND_INT(pe, &size, 1, myid);
PRF_CSEND_REAL(pe, array, size, myid);
PRF_CSEND_REAL(pe, array1, size, myid);
PRF_CSEND_REAL(pe, array2, size, myid); /*Sending arrays*/
PRF_CSEND_REAL(pe, array3, size, myid);
PRF_CSEND_REAL(pe, array4, size, myid);
PRF_CSEND_REAL(pe, arrayx, size, myid);
PRF_CSEND_REAL(pe, arrayy, size, myid);
PRF_CSEND_REAL(pe, arrayz, size, myid);

free(array);
free(array1);
free(array2);   /* free arrays on nodes once data sent */
free(array3);
free(array4);
free(arrayx);
free(arrayy);
free(arrayz);

/* node_0 now collect data sent by other compute nodes */
/*   and sends it straight on to the host */
if (I_AM_NODE_ZERO_P)
```

```
/*!!!!THIS LOOP RUNS for node-1 -> .... -> node-n IN ODRER!!!!*/

/*Action-6 send data to host*/
  compute_node_loop_not_zero (pe)
  {
    PRF_CRECV_INT(pe, &size, 1, pe);
    array = (real *)malloc(size * sizeof(real));
    array1 = (real *)malloc(size * sizeof(real));
    array2 = (real *)malloc(size * sizeof(real));/*BUFFERS*/
    array3 = (real *)malloc(size * sizeof(real));
    array4 = (real *)malloc(size * sizeof(real));
    arrayx = (real *)malloc(size * sizeof(real));
    arrayy = (real *)malloc(size * sizeof(real));
    arrayz = (real *)malloc(size * sizeof(real));

    PRF_CRECV_REAL(pe, array, size, pe);
    PRF_CRECV_REAL(pe, array1, size, pe);
    PRF_CRECV_REAL(pe, array2, size, pe);
    PRF_CRECV_REAL(pe, array3, size, pe);
    PRF_CRECV_REAL(pe, array4, size, pe);
    PRF_CRECV_REAL(pe, arrayx, size, pe);
    PRF_CRECV_REAL(pe, arrayy, size, pe);
    PRF_CRECV_REAL(pe, arrayz, size, pe);



    PRF_CSEND_INT(node_host, &size, 1, myid);
    PRF_CSEND_REAL(node_host, array, size, myid);
    PRF_CSEND_REAL(node_host, array1, size, myid);
    PRF_CSEND_REAL(node_host, array2, size, myid);
    PRF_CSEND_REAL(node_host, array3, size, myid);
    PRF_CSEND_REAL(node_host, array4, size, myid);
    PRF_CSEND_REAL(node_host, arrayx, size, myid);
    PRF_CSEND_REAL(node_host, arrayy, size, myid);
    PRF_CSEND_REAL(node_host, arrayz, size, myid);


          free((char *)array);
          free((char *)array1);
          free((char *)array2);
          free((char *)array3);
          free((char *)array4);
          free((char *)arrayx);
          free((char *)arrayy);
          free((char *)arrayz);
  }
```

```
/*Send with correct order, now that all data has been collected*/


#endif /* RP_NODE */

/*Action-6 the host receives the data and prints them into the file*/

#if RP_HOST
real *total_array_U;
real *total_array_V;
real *total_array_W; /*Initializing total arrays for each var*/
real *total_array_P;
real *total_array_T;
real *total_array_x;
real *total_array_y;
real *total_array_z;
/*running_size is used to insert the data of the next node after
the last one in the file (cause each node has its own number of cells)*/

int running_size = 0;

/*BUFFERS*/
total_array_U=(real *)malloc(TOTAL_CELLS * sizeof(real));
total_array_V=(real *)malloc(TOTAL_CELLS * sizeof(real));
total_array_W=(real *)malloc(TOTAL_CELLS * sizeof(real));
total_array_P=(real *)malloc(TOTAL_CELLS * sizeof(real));
total_array_T=(real *)malloc(TOTAL_CELLS * sizeof(real));
total_array_x=(real *)malloc(TOTAL_CELLS * sizeof(real));
total_array_y=(real *)malloc(TOTAL_CELLS * sizeof(real));
total_array_z=(real *)malloc(TOTAL_CELLS * sizeof(real));

/*pe only acts as a counter in this loop*/
 compute_node_loop (pe)
  {
    /* Receive data sent by each node and write it out to the file */
    PRF_CRECV_INT(node_zero, &size, 1, node_zero);
    array = (real *)malloc(size * sizeof(real));
    array1 = (real *)malloc(size * sizeof(real));
    array2 = (real *)malloc(size * sizeof(real)); /*BUFFERS*/
    array3 = (real *)malloc(size * sizeof(real));
    array4 = (real *)malloc(size * sizeof(real));
    arrayx = (real *)malloc(size * sizeof(real));
    arrayy = (real *)malloc(size * sizeof(real));
    arrayz = (real *)malloc(size * sizeof(real));
```

```
    PRF_CRECV_REAL(node_zero, array, size, node_zero);
    PRF_CRECV_REAL(node_zero, array1, size, node_zero);
    PRF_CRECV_REAL(node_zero, array2, size, node_zero);
    PRF_CRECV_REAL(node_zero, array3, size, node_zero);
    PRF_CRECV_REAL(node_zero, array4, size, node_zero);
    PRF_CRECV_REAL(node_zero, arrayx, size, node_zero);
    PRF_CRECV_REAL(node_zero, arrayy, size, node_zero);
    PRF_CRECV_REAL(node_zero, arrayz, size, node_zero);

    /*Concatenate data into 1 array for each variable*/
    memcpy(total_array_U+running_size, array, size*sizeof(real));
    memcpy(total_array_V+running_size, array1, size*sizeof(real));
    memcpy(total_array_W+running_size, array2, size*sizeof(real));
    memcpy(total_array_P+running_size, array3, size*sizeof(real));
    memcpy(total_array_T+running_size, array4, size*sizeof(real));
    memcpy(total_array_x+running_size, arrayx, size*sizeof(real));
    memcpy(total_array_y+running_size, arrayy, size*sizeof(real));
    memcpy(total_array_z+running_size, arrayz, size*sizeof(real));

    running_size += size;

    free(array);
    free(array1);
    free(array2);
    free(array3);
    free(array4);
    free(arrayx);
    free(arrayy);
    free(arrayz);
  }

/* Printing the total arrays into the file for each variable */
for (i=0; i<running_size; i++)
        fprintf(fp, "%8.6e\n", total_array_x[i]);

for (i=0; i<running_size; i++)
        fprintf(fp, "%8.6e\n", total_array_y[i]);

for (i=0; i<running_size; i++)
        fprintf(fp, "%8.6e\n", total_array_z[i]);

for (i=0; i<running_size; i++)
        fprintf(fp, "%8.6e\n", total_array_U[i]);

for (i=0; i<running_size; i++)
        fprintf(fp, "%8.6e\n", total_array_V[i]);
```

```
for (i=0; i<running_size; i++)
        fprintf(fp, "%8.6e\n", total_array_W[i]);

for (i=0; i<running_size; i++)
        fprintf(fp, "%8.6e\n", total_array_P[i]);

for (i=0; i<running_size; i++)
        fprintf(fp, "%8.6e\n", total_array_T[i]);

free(total_array_U);
free(total_array_V);
free(total_array_W);
free(total_array_P);
free(total_array_T);
free(total_array_x);
free(total_array_y);
free(total_array_z);
#endif /* RP_HOST */

/* Action-7 closing the file */
#if !RP_NODE /* SERIAL or HOST */
 fclose(fp); /* Close the file that was only opened if on SERIAL or HOST */
 Message("Done\n");
#endif

}
```

Finally, this "reading" UDF reads any variable value for the whole computational grid or for a selected area and it is used in serial processing, since ANSYS Fluent only reads data in serial and partitions them itself. Therefore, the user only need to keep track of the indicators or the order the data are written.

```
/************************************************************
                        **serial**
  This function will read velocity (XYZ),pressures and temperatures
   for a fluid zone from a file (with the values in cell id order)
Pavlos Gkinis
************************************************************/

#include <stdio.h>
#include "udf.h"
# define FLUID_ID 2 /* Define the zone ID for the fluid zone */
# define TOTAL_CELLS 309155 /*Define the total number of cells*/
```

```
DEFINE_INIT(Initial_Solution,d)
{
Domain *domain=Get_Domain(1);
Thread *t;
cell_t c;
t=Lookup_Thread(domain,FLUID_ID);
FILE *fp = NULL;
char filename[]="Initial_Solution.txt";

 if ((fp = fopen(filename, "r"))==NULL)
     Message("\n Warning: Unable to open %s for reading\n",filename);
 else
     Message("\nReading Variables from %s...",);

/* loop over all cells in order in the fluid zone for each variable */

for (c=1; c<=TOTAL_CELLS; c++)
          fscanf(fp,"%lf\n", &C_U(c,t));

for (c=1; c<=TOTAL_CELLS; c++)
          fscanf(fp,"%lf\n", &C_V(c,t));

for (c=1; c<=TOTAL_CELLS; c++)
          fscanf(fp,"%lf\n", &C_W(c,t));

for (c=1; c<=TOTAL_CELLS; c++)
          fscanf(fp,"%lf\n", &C_P(c,t));

for (c=1; c<=TOTAL_CELLS; c++)
          fscanf(fp,"%lf\n", &C_T(c,t));

fclose(fp);
Message("Done\n");
}
```