

Utilization of artificial neural networks to resolve chemical kinetics in turbulent fine structures of an advanced CFD combustion model

by

Ryno Laubscher



*Dissertation presented for the degree of Doctor of Philosophy
in the Faculty of Engineering at Stellenbosch University*

Supervisor: Dr. JE Hoffmann

March 2017

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2017

Copyright © 2017 Stellenbosch University
All rights reserved.

Abstract

Utilization of artificial neural networks to resolve chemical kinetics in turbulent fine structures of an advanced CFD combustion model

R. Laubscher

*Department of Mechanical and Mechatronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.*

Dissertation: Doctor of Philosophy

March 2017

This work investigates an alternative chemistry integration approach to be used with the eddy dissipation concept (EDC) advanced combustion model for large-scale industrial applications where detailed or reduced mechanisms are utilised. The goal of the study was to reduce the computational resources required to solve reduced or detailed mechanisms using the EDC model. The unique approach uses artificial neural networks (ANNs) as a chemistry integrator for the reactions that take place in the fine structure regions created by the turbulence field. The ANNs are therefore utilised to predict the incremental species changes that occur in these fine structure regions as a function of the initial species composition, temperature and the residence time of the mixture in the fine structure regions. The ANN's weights- and bias matrices were changed to minimise the network's prediction error using the back-propagation algorithm and datasets generated using the results of separate ideal plug-flow reactor simulations. The effect that the ANN's architecture has on its ability to accurately predict the temporal evolution of the species was also investigated and the best-performing configuration was selected. The novel chemistry integration approach for the EDC model was implemented to model a piloted methane/air turbulent jet diffusion flame (Sandia Flame D) at a Reynolds number of 22400. To prove the concept, a 5-step methane combustion mechanism was used to model the chemical reactions of the experimental flame. The results of the new approach were benchmarked against experimental data and the simulation results using the standard integration approaches in Fluent. It was shown that once the ANN is well-trained, it can predict the species mass

ABSTRACT

iii

fractions with relative accuracy in both a time and computer memory efficient manner compared to using traditional integration procedures.

Uittreksel

Benutting van kunsmatige neurale netwerke om chemiese kinetika op te los in turbulente fyn strukture van 'n gevorderde NVD verbrandingsmodel

(“Utilization of artificial neural networks to resolve chemical kinetics in turbulent fine structures of an advanced CFD combustion model”)

R. Laubscher

*Departement Meganiese en Megatroniese Ingenieurswese,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.*

Proefskrif: Doktor van Filosofie

Maart 2017

Hierdie werk ondersoek 'n alternatiewe chemie-integrasie metode wat gebruik maak van die *EDC* (*eddy dissipation concept*) gevorderde verbrandingsmodel vir grootskaalse industriële toepassings waar gedetailleerde of verminderde chemiese meganismes gebruik word. Die doel van die studie is om die rekenaar hulpbronne wat benodig is om verminderde of gedetailleerde meganismes met die *EDC*-model op te los, te verlaag. Dié unieke benadering gebruik kunsmatige neurale netwerke (KNN) as 'n chemie-integreerder vir die reaksies wat plaasvind in die fyn struktuur streke wat deur die vloeiveld se turbulensie geskep is. Die KNN'e is dus aangewend om die inkrementele spesiesveranderinge wat in hierdie fyn struktuur streke plaasvind as 'n funksie van die aanvanklike spesiesamestelling, temperatuur en die reaksietyd van die mengsel in die fyn struktuur streke te voorspel. Die geweegde veranderlikes matrikse van die KNN is verander om die voorspellingsfout van die netwerk te minimeer deur die terug-voortplanting algoritme te gebruik deur middel van datastelle wat gegenereer is met behulp van die resultate van ideale propvloeireaktor simulaties. Die effek van die KNN se argitektuur op sy vermoë om die tydelike evolusie van die spesies akkuraat te voorspel is ook ondersoek en die beste presterende opset is gekies. Die chemie-integrasie benadering vir die *EDC*-model is geïmplementeer om 'n geloodsde metaan / lug onstuimige straler diffusie vlam (*Sandia Flame D*) met 'n Reynoldsgetal van 22400 te modelleer. Ten einde die konsep te bewys is 'n 5-stap metaan-verbranding meganisme gebruik om die

reaksies van die eksperimentele vlam te modelleer. Die resultate van die nuwe benadering is vergelyk met eksperimentele data en die simulasiereultate wat met behulp van Fluent se standaard integrasie benaderings verkry is. Daar is bewys dat wanneer die KNN goed opgelei is, dit met redelike akkuraatheid die spesies massa-fraksies kan voorspel op 'n wyse wat doeltreffend is beide ten opsigte van tyd en rekenaar geheue.

Acknowledgements

- Thank you my God and heavenly Father for giving me all the abilities, perseverance, strength and knowledge to finish this thesis.
- My promoter Dr Jaap Hoffmann, for his guidance, tutelage, support and willingness to help when necessary.
- John Thompson for giving me the time, support and financial support to undertake this thesis, especially Philip Du Toit, Quintus Engelbrecht and Etienne De Villiers.
- My soon-to-be wife and best friend, Joëlle, for her unconditional love and support.
- My parents, brother and sister for their eternal love and support.
- My family and friends for their interest and support.

Dedications

To Joëlle

Contents

Declaration	i
Abstract	ii
Uittreksel	iv
Acknowledgements	vi
Dedications	vii
Contents	viii
List of Figures	xi
List of Tables	xv
Nomenclature	xvi
1 Introduction	1
2 Problem statement and objectives	6
3 Literature study	9
3.1 Combustion modelling	9
3.2 Artificial neural network applications to chemical reaction prediction	12
4 Combustion modelling and artificial neural networks theory	15
4.1 Transport equations for mass, species, momentum and energy in reactive flow	15
4.2 Turbulence modelling (closure of averaged conservation equations)	23
4.3 Chemical reaction theory and modelling	28
4.4 Turbulence-chemistry interaction modelling	34
4.5 Artificial neural networks	46
5 Experimental setup CFD modelling using traditional methods	54

<i>CONTENTS</i>	ix
5.1 Geometry	55
5.2 Boundary conditions	55
5.3 Flow simulation setup	57
5.4 Mesh independence investigation	57
5.5 Results and post-processing	58
6 Plug-flow reactor simulation and data generation	70
6.1 Random sample generation	70
6.2 Reactor solver	71
7 Neural network architecture selection and training	75
7.1 Architecture selection	75
7.2 Neural network training of plug-flow reactor generated data	87
8 Implementation of artificial neural network in Fluent[®] 17.0	92
9 Results and computational performance	97
9.1 Results comparison	97
9.2 Computational resource comparison	108
10 Summary, conclusion and recommendations	114
10.1 Summary and conclusions	114
10.2 Recommendations and future work	119
Appendices	120
A Backward-differencing variable-coefficient ordinary differential equation solver	121
B NASA polynomials for fluid properties	123
C Mixture fraction calculation UDF	125
D Plug-flow reactor computer program	127
D.1 Sample calculation	127
D.2 Code listing	132
E Neural network programs and sample calculations	136
E.1 Forward propagation procedure	136
E.2 Back-propagation procedure	141
F Artificial neural network implementation code	159
F.1 Neural network main function	159
F.2 Species source macros	161
F.3 Energy source macro	167

<i>CONTENTS</i>	x
G Contour plots of turbulent Reynolds numbers for RSM and realizable k-epsilon models	168
List of References	170

List of Figures

4.1	Methane/Air turbulent piloted jet flame - Barlow and Frank (2007)	16
4.2	Left: Jet flow velocity distribution, Right: Mixing layer formed between fast- and slow- moving fluids Versteeg and Malalasekera (2007)	24
4.3	Graphic representation of species transfer rate from the surrounding fluid into the fine structure regions	31
4.4	Schematic of a plug-flow reactor, Turns (2000)	32
4.5	Non-premixed and premixed eddies mixing rate representation	37
4.6	Fine structure regions in a fluid volume - Magnussen (2009)	39
4.7	Energy cascade model for the transfer of mechanical energy from the mean flow, through turbulence energy to heat Ertesvag and Magnussen (1999)	41
4.8	Graphic representation of the energy cascade model, with larger eddies feeding the smaller ones.	44
4.9	Biological representation of a single neuron - Winston (1993)	46
4.10	Schematic of a single artificial neuron	47
4.11	Activation functions used in artificial neural networks	48
4.12	Schematic of a multi-layer regression neural network	49
5.1	Two-dimensional axisymmetric computational domain used to model the piloted jet flame	55
5.2	Temperature and mixture fraction distribution for the centreline normalised axial location. Blue solid line: CFD, Black dots: Experimental	60
5.3	Temperature and mixture fraction distribution for normalised radial locations at $x/d = 0.15$ m. Blue solid line: CFD, Black dots: Experimental	60
5.4	Temperature and mixture fraction distribution for normalised radial locations at $x/d = 0.30$ m. Blue solid line: CFD, Black dots: Experimental	60
5.5	Species mass fractions for centreline normalised axial locations. Blue solid line: CFD, Black dots: Experimental	61
5.6	Species mass fractions for normalised radial locations at $x/d = 0.15$ m. Blue solid line: CFD, Black dots: Experimental	62

5.7	Species mass fractions for normalised radial locations at $x/d = 0.3$ m. Blue solid line: CFD, Black dots: Experimental	63
5.8	CO mass fraction results on centre line for various turbulence models	64
5.9	Temperature results on centre line for various turbulence models . .	65
5.10	Temperature contour of the methane/air piloted jet flame solved using traditional EDC model, $[K]$	66
5.11	Heat of reaction contour of the methane/air piloted jet flame solved using traditional EDC model, $[W]$	67
5.12	Mixture fraction contour plot	68
5.13	Mixture fraction upper and lower bound superimposed on the heat of reaction contour plot	68
6.1	Plug-flow reactor simulation results for single scenario	73
6.2	Plug-flow reactor simulation flowchart	74
7.1	(Input) Sigmoid \rightarrow (Hidden) Sigmoid \rightarrow (Output) Sigmoid: configured network	77
7.2	(Input) Sigmoid \rightarrow (Hidden) Sigmoid \rightarrow (Output) Linear: configured network	78
7.3	(Input) Tanh \rightarrow (Hidden) Tanh \rightarrow (Output) Tanh: configured network	79
7.4	(Input) Tanh \rightarrow (Hidden) Tanh \rightarrow (Output) Linear: configured network	79
7.5	CH_4 mass fractions for varying amount of neurons and hidden layers	81
7.6	CO mass fractions for varying amount of neurons and hidden layers	82
7.7	CO_2 mass fractions for varying amount of neurons and hidden layers	83
7.8	H_2O mass fractions for varying amount of neurons and hidden layers	84
7.9	H_2 mass fractions for varying amount of neurons and hidden layers	85
7.10	O_2 mass fractions for varying amount of neurons and hidden layers	86
7.11	In-sample error as a function of reaction time-step for single hidden layer (7 neurons) neural network	87
7.12	Theoretical plot of in-sample error and out-of-sample error for simple and complex statistical models, Abu-Mostafa <i>et al.</i> (2012) . . .	88
7.13	In-sample and out-of-sample error for 0.5×10^{-6} network model of increasing dataset sizes	89
7.14	In-sample and out-of-sample error for 0.5×10^{-3} network model of increasing dataset sizes	90
7.15	Solving time as a function of amount of species compositions solved to nearly steady -state	91

9.1	Temperature and mixture fraction distribution at axial locations on the centreline of the flame. Θ (black) - Exp, \ominus (blue) - DI/ISAT EDC, $\omin�$ (red) - EDC/ANN, $\omin�$ (green) - EDC/ANN on EDM temperature and flow field, $\omin�$ (magenta) - EDC/ANN on EDC two-step temperature and flow field and $\omin�$ (orange)- Kempf <i>et al.</i> (2005) . . .	98
9.2	Temperature and mixture fraction distribution at radial locations at $x/d = 0.15$ m	98
9.3	Temperature and mixture fraction distribution at radial locations at $x/d = 0.3$ m	98
9.4	Species mass fractions at axial locations along centreline of flame. Θ (black) - Exp, \ominus (blue) - DI/ISAT EDC, $\omin�$ (red)- EDC/ANN, $\omin�$ (green) - EDC/ANN on EDM temperature and flow field, $\omin�$ (magenta) - EDC/ANN on EDC two-step temperature and flow field and $\omin�$ (orange) - Kempf <i>et al.</i> (2005)	99
9.5	Species mass fractions at radial locations at $x/d = 0.15$ m. Θ (black) - Exp, \ominus (blue) - DI/ISAT EDC, $\omin�$ (red) - EDC/ANN, $\omin�$ (green) - EDC/ANN on EDM temperature and flow field, $\omin�$ (magenta)- EDC/ANN on EDC two-step temperature and flow field and $\omin�$ (orange) - Kempf <i>et al.</i> (2005)	100
9.6	Species mass fractions at radial locations at $x/d = 0.3$ m. Θ (black) - Exp, \ominus (blue)- DI/ISAT EDC, $\omin�$ (red)- EDC/ANN, $\omin�$ (green) - EDC/ANN on EDM temperature and flow field, $\omin�$ (magenta) - EDC/ANN on EDC two-step temperature and flow field and $\omin�$ (orange) - Kempf <i>et al.</i> (2005)	101
9.7	Residuals of species fractions for Sandia flame D simulation using ANN chemistry integrator and ISAT/DI integrators	102
9.8	Predicted <i>CO</i> mass fractions by ANN chemistry integrator for different upper bound mixture fraction limits with models 3,4. $\omin�$ - $f = 0.9$, $\omin�$ - $f = 0.8$ and $\omin�$ - $f = 0.6$	104
9.9	Predicted <i>CO</i> mass fractions by ANN chemistry integrator for different upper bound mixture fraction limits with model 2. $\omin�$ - $f = 0.9$, $\omin�$ - $f = 0.8$ and $\omin�$ - $f = 0.6$	105
9.10	Species mass fractions at axial locations along centreline of flame predicted using the RSM turbulence model. Θ (black) - Exp, $\omin�$ (red) - EDC/ANN and $\omin�$ (magenta) - EDC/ANN on EDC two-step temperature and flow field	107
9.11	Temperatures at axial locations along centreline of flame predicted using the RSM turbulence model. Θ (black)- Exp, $\omin�$ (red) - EDC/ANN and $\omin�$ (magenta) - EDC/ANN on EDC two-step temperature and flow field	108
9.12	Speed-up ratio over the direct integration method. $\omin�$ - ANN-EDC (Model 2), $\omin�$ - EDC models using ISAT, $\omin�$ - EDM-EDC-ANN (Model 3) and $\omin�$ - EDC-EDC-ANN (Model 4)	111

9.13	Computer memory usage for ISAT, DI and ANN models. \ominus - ANN, \ominus - DI, \ominus - ISAT	112
9.14	Solving time comparison between EDM and EDC-ANN combustion models for the WD1 mechanism. \ominus - ANN, \ominus - EDM	113
E.1	Program flowchart for forward-propagation procedure	137
E.2	Program flowchart for back-propagation procedure	143
G.1	Contour plot of turbulent Re number for RSM turbulence model . .	168
G.2	Contour plot of turbulent Re number for realizable $k - \epsilon$ turbulence model	169

List of Tables

4.1	WD1 reaction mechanism kinetic data	34
5.1	Burner dimensions	55
5.2	Boundary conditions	56
5.3	Mesh refinement data	58
5.4	Richardson extrapolation calculation for first refinement stage . . .	58
8.1	Under-relaxation factors for ANN chemistry integrator and standard EDC simulations with DI/ISAT activated	95
9.1	Computational performance of integration techniques: CPU time .	109
9.2	Computational performance of integration techniques: RAM requirements	110
B.1	NASA -7 polynomial coefficients	124

Nomenclature

Constants

$A_0 =$	4.04
$C_2 =$	1.9
$C_{1\epsilon} =$	1.44
$C_\gamma =$	2.1377
$C_\tau =$	0.4082
$g =$	9.81 m/s ²
$Pr_t =$	0.85
$R =$	8314.459 J/kmol.K
$Sc =$	0.7
$\sigma_k =$	1.0
$\sigma_\epsilon =$	1.2
$\gamma =$	0.7
$\eta =$	0.35

Variables

A	Pre-exponential factor	[1/s]
B	Body forces	[N]
c_P	Specific heat of fluid	[J/kgK]
D	Binary diffusion coefficient	[m ² /s]
e	In-sample error of neural network prediction	[]
E	Total energy	[W]
E_a	Activation energy	[J/kmol]
e_{ij}	Strain rate	[]
f	Mixture fraction	[]
h	Enthalpy of fluid	[J/kg]
\bar{h}	Ensemble averaged fluid enthalpy	[J/kg]
\tilde{h}	Favre averaged fluid enthalpy	[J/kg]
h'	Fluctuating component of the enthalpy	[J/kg]

h''	Density averaged fluctuating component of the enthalpy	[J/kg]
$h(x)$	Neural network output signal	[]
k	Thermal conductivity of gas mixture or kinetic energy	[W/mK, m ² /s ²]
L'	Turbulent length scale	[m]
m^*	Fine structure mass transfer rate	[1/s]
\dot{m}	Mass flow rate	[kg/s]
M	Molecular weight	[kmol/kg]
n	Observation in dataset	[]
\bar{M}	Mean molecular weight	[kmol/kg]
P	Absolute pressure	[Pa]
q	Dissipation of turbulent kinetic energy or reaction rate of progress variable	[kmol/m ³ s]
Re	Reynolds number	[]
S_h	Energy source from radiation and/or chemical reactions	[W/m ³]
s_j	Cumulative input signal into layer in a neural network	[]
S_m	Source term for dispersed phase mass transport	[kg/m ³ s]
S_k	Source term for dispersed phase mass transport of species k	[kg/m ³ s]
t	Time	[s]
t_k	Kolmogorov time scale	[]
T	Temperature or user-specified global training iterations	[K]
u	Velocity	[m/s]
\bar{u}	Ensemble averaged fluid velocity	[m/s]
\tilde{u}	Favre averaged fluid velocity	[m/s]
u'	Turbulent velocity or fluctuating velocity component	[m/s]
u''	Density averaged fluid velocity fluctuating component	[m/s]
u_k	Kolmogorov velocity scale	[]
V	Volume	[m ³]
V_k	Diffusion velocity of species k	[m/s]
w	Neural network weight variable	[]
w'	Turbulent energy production	[]
x	Coordinate	[m]
$x^{(l)}$	Signal in synapse of neural network	[]
X	Symbol for species	[]
[X] Species concentration	[kmol/m ³]
y	Coordinate	[m]
Y	Species mass fraction	[]
\bar{Y}	Ensemble averaged species mass fraction	[]

\tilde{Y}	Favre averaged species mass fractions []
Y'	Fluctuating species mass fraction component []
Y''	Density averaged species mass fraction fluctuating component []
Y^*	Fine structure regions species mass fractions []
\bar{Y}	Cell averaged species mass fraction []
z	Coordinate [m]
Z	Total number of reactions []
β_j	Temperature exponent []
δ_{ij}	Kronecker delta coefficient []
ϵ	Turbulence energy dissipation rate [m ² /s ³]
ϵ_i	Random number between 0-1 in a normal distribution . []
η	Kolmogorov length scale or neural network learning rate parameter []
ϕ	Arbitrary fluid property []
γ	Fine structure length fraction []
γ^*	Fine structure volume fraction []
ρ	Density of gas mixture [kg/m ³]
σ_{ij}	Fluid stress tensor [N]
τ^*	Fine structure time scale [s]
$\theta()$	Activation function []
μ	Molecular viscosity [Pa.s]
μ_t	Eddy viscosity []
$\dot{\omega}$	Chemical reaction rate [kg/m ³ s]
ν'	Stoichiometric coefficients of the reactants []
ν''	Stoichiometric coefficients of the products []

Vectors and Tensors

\mathbf{A}	Surface (normal) area [m ²]
\mathbf{F}_s	Surfaces forces [N]
\mathbf{q}	Heat flux [W/m ²]

Subscripts

x, y, z	Rectangular coordinates
r	Radial coordinate
k	Species designation
t	Turbulent

j Reaction designation

Abbreviations

ANN	Artificial Neural Networks
BDF	Backward Differencing Formulation
CEA	Chemical Equilibrium Analysis
CFD	Computational Fluid Dynamics
CPU	Central Processing Unit
DI	Direct Integration
DNS	Direct Numerical Simulations
EBU	Eddy Break-Up
EDC	Eddy Dissipation Concept
EDM	Eddy Dissipation Model
FR	Finite-Rate
IEA	International Energy Agency
ISAT	In-Situ Adaptive Tabulation
LES	Large-Eddy Simulations
ODE	Ordinary Differential Equations
PDF	Probability-Density Function
RANS	Reynolds Averaged Navier-Stokes
RSM	Reynolds Stress Models
RAM	Random Access Memory
UDF	User Defined Functions
WD1	Westbrook and Dryer

Chapter 1

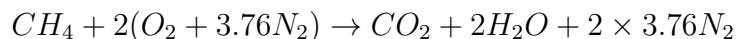
Introduction

The present research brings together two distinct fields of science and engineering, namely artificial intelligence and combustion modelling. Artificial intelligence is the study of computations that make it possible to perceive, reason and act, according to Winston (1993); where combustion modelling uses disciplines such as thermodynamics, chemical kinetics, fluid mechanics, heat and mass transfer, and turbulence to predict the quantities involved in combustion processes. The central topic of the current research will be the application of these two distinct fields to develop a combustion modelling approach capable of resolving detailed chemical mechanisms and turbulence-chemistry interaction without the extensive computer resources required by the current techniques used in commercial computational fluid dynamics codes. The required computer resources are reduced by using artificial neural networks rather than traditional ordinary differential equation solvers or look-up tabulation methods. The approach is developed for future implementation in combustion modelling of industrial scale biomass, coal, oil, and gas-fired boilers using large multi-step chemical mechanisms.

The ability to harness thermal energy from combustion is essential to human existence on earth. There is a broad range of applications of combustion, such as power generation where coal particles are burned in furnaces of power stations to produce steam for driving turbines, district heating, process heat for production of iron, steel and glass, internal combustion engines, gas turbines combustion for powering air planes, etc. According to the IEA (2013) report, 92 % of the world's energy demand is supplied by combustion processes (natural gas, coal/peat, oil, biofuels and waste) and the percentage of total energy supplied by coal/peat and natural gas combustion is on the increase. It is therefore worthwhile studying combustion processes. The popular fuels used in boiler systems are coal, natural gas, oil and biomass. Liquid fuels, such as oil, heat up when introduced into a high temperature zone due to radiation and convection from the surroundings. Heat transfer to the droplet increases the vapour pressure, and thus evaporation of the liquid into the gas phase. Once the fuel is evaporated, the fuel and oxidiser species can molecu-

larly mix and react. At first glance solid fuels such as biomass and coal burn much like liquids. They are initially heated to a temperature where they generate gas-phase volatiles that burn in the same manner as reactants of a gas during combustion (Warnatz *et al.*, 2006). The difference between solid fuel combustion and liquid fuel combustion is that when the volatiles are driven off the fuel, a residual fraction of combustible matter left in the particle can be oxidised. The combustion process of solid fuels can be divided into five distinct processes, namely: (1) initial/sensible heating; (2) moisture evaporation; (3) devolatilisation; (4) volatile combustion; and (5) char combustion. Volatile matter plays a crucial role in coal combustion, in that the heat from the volatile combustion is used to assist in char ignition of the coal particles. The major difference between biomass and coal is the fractions of volatiles, moisture, char and ash. Biomass usually has much higher volatile fraction than coal and therefore has a higher percentage of energy being released due to gas phase reactions. In all the above-mentioned fuels, gas phase combustion plays a critical role in energy release, species and pollutant formation.

Historically, boiler gas phase combustion modelling was aimed at fluid mechanics that included global heat release by chemical reaction, where the combustion systems are lumped into a single control volume or modelled as multiple control volumes. The heat release is described using the first law of thermodynamics and the assumption of infinitely fast chemistry. The enthalpies of the products and reactants of the global reaction are used to determine the global heat release rate. An example of such a complete combustion reaction is:



This approach is useful to some extent for designing stationary combustion processes, but is not sufficient for treating transient processes like ignition and quenching or pollutant formation (Warnatz *et al.*, 2006). The assumption of complete combustion, as above, is in practice incorrect. Some reactions occur in reverse at high temperatures, thus forming reactants. This process is called dissociation. Dissociation reactions are endothermic; therefore the actual gas temperature calculated by the global complete combustion approach (adiabatic flame temperature calculation) is in reality slightly higher. This leads us to the next simplified approach to resolve combustion, namely Chemical equilibrium analysis (CEA). CEA is used to determine the composition of the gas mixture after dissociation and is based on the second law of thermodynamics (Versteeg and Malalasekera, 2007). In reality, combustion of a fuel does not occur in a single reaction and the reaction takes time to reach equilibrium. Chemical kinetics determines the time a reacting system of multiple reactions takes to reach equilibrium and is therefore the study of reaction mechanisms and reaction rates. Many combustion processes are controlled by the rate of molecular and turbulent diffusion. However, chemical kinetics can play an important role under certain conditions such as low pressure combustion, re-

stricted oxygen supply to fuel, and large radiative heat losses (Versteeg and Malalasekera, 2007). The discussed methods (complete combustion, CEA and chemical kinetics) are the simplified methods that are used in zero- or one-dimensional models to predict the gas mixture temperature and composition. The mixture temperature and properties are then used to determine the heat flux to the environment (boiler furnaces, superheaters, etc.) using empirical correlations for radiation and convection heat transfer. Using these fluxes, the combustion equipment is sized accordingly. The methods discussed above does not take into account the effect of the fluid flow, and in many combustion situations the fluid flow is an integral part of the combustion processes. Implementing these gross simplifications and assumptions usually leads to furnaces and superheaters that are over- or undersized and species formations that are incorrectly predicted, due to the averaging of heat fluxes, species formations, fluid velocity, etc. in the calculations.

Computational fluid dynamics (CFD) modelling is therefore becoming increasingly important in the development and optimisation of processes involved with designing combustion systems for boilers, especially applicable to gas phase combustion modelling. Where gas phase combustion includes the volatile combustion of coal or biomass fuels and vaporised liquid fuels, CFD modelling of combustion yields the ability to predict flow, temperature and species formation in boilers with relative accuracy, making it an ideal platform to investigate:

- Fuel and air mixing, which is used to design the over-fire air system for higher combustion rates and furnace efficiency. Higher furnace efficiency leads to the cost-effective design of furnace sizes.
- Determine heat transfer rate to furnace walls and superheaters, used to ensure designed components are sized correctly, and also using heat fluxes to determine boiler two-phase circulation.
- Model fuel particle spread in furnace and burner flame length. This aids the engineer in sizing specific dimensions of the combustion chamber.
- Erosion modelling of particles on heat transfer equipment to prolong boiler component lifetime.
- Determine fuel particle residence time in furnace and temperature history, to investigate and identify possible slagging and fouling regions and mitigate the problem areas through altering designs.
- Pollutant formation, such as unburned hydrocarbons, SO_x and NO_x. More stringent emission requirements, dictates that the combustion process needs to be designed to lower the emissions. CFD is therefore used to predict the emissions of the combustion in the furnace. One method

to lower emissions is to design a staged combustion system with CFD in an effort to reduce thermal NO_x formation.

To accurately model some of the above-mentioned processes such as pollutant formation, fouling and slagging, and furnace heat fluxes, an advanced combustion model capable of accurately handling multiple step chemical mechanisms is required. The flow in majority of industrial combustion systems, especially boiler furnaces, are turbulent due to the high velocity over-fire air injections. The increased mixing rate due to turbulence increases the combustion rate and therefore system efficiency. Various models, such as the eddy break-up models (EBU) and eddy dissipation concept model (EDC) have been developed to account for the turbulence-chemistry interaction. The EBU models are the most prevalent Reynolds Averaged Navier-Stokes (RANS) based combustion models that have been successfully used to model various industrial boiler systems. The low computational costs of the EBU models have made it a very popular combustion modelling approach, but the model lacks the ability to fundamentally account for turbulence-chemistry interaction and complex chemical mechanisms. The EDC model enables the consideration of the complex interaction of turbulence and detailed reaction kinetics, thus making it an excellent candidate for advanced combustion modelling. The major drawback of using the EDC model with large chemical mechanisms is the computer resource requirements for industrial problems. This is because in addition to solving the conservation and species transport equations for each of the species in the mechanism, the stiff chemical ordinary differential equations need to be integrated. These numerical processes are in turn implemented on industrial models that usually have large geometries and require large meshes to be able to capture all the flow and combustion effects accurately. These processes at industrial scale are very resource-intensive, even when using the RANS approach. The present research is aimed at developing a novel approach to predict the solution of the stiff chemical kinetics differential equations without solving the system of differential equations during the CFD simulation. This process was used to greatly speed up the solving procedure by learning the incremental changes of the species for specified time-steps of the chemical mechanism beforehand and calculating the required rates with simple mathematical expressions, thus exhibiting artificial intelligence by understanding the involved processes.

Artificial intelligence finds its origin in fields such as computer science, mathematics and neuroscience. The engineering goal of artificial intelligence is to solve problems using artificial intelligence as a collection of tools to learn simple or complex systems and then act or predict based on what has been learned. The goal is to build an intelligent system that can learn from examples, therefore mine data to understand and exploit complex relationships and regularities. The sub-field of machine learning emerged from artificial intelligence. Machine learning takes these complex relationships and regularities

and learns from the observations through the use of computer algorithms. The machine learning model used in the present research is artificial neural networks. The use of artificial neural networks is an emerging and promising, low-resource alternative for solving chemical reaction differential equations in turbulent combustion flows as discussed by Christo *et al.* (1996b). The majority of the work found in literature where neural networks have been used in turbulent combustion simulations, were using the non-premixed combustion approach for large-eddy simulations (LES) and RANS simulations as seen in Christo *et al.* (1996b), Sen and Menon (2009), Kempf *et al.* (2005) and Enami and Fard (2012). The novel chemistry integration approach developed in the present research uses a trained artificial neural network to predict the fine scales species composition for the EDC model after the fine structure regions have reacted over a specified time-step. The EDC model then uses the fine scales species mass fractions to determine the net reaction rate for each of the mixture species. The traditional EDC model uses either the in-situ adaptive tabulation (ISAT) or direct integration (DI) method to solve the temporal evolution of the species in the fine scales. Both of these traditional methods have serious drawbacks, namely large computer memory (ISAT method) and CPU (DI method) requirements.

The current document is comprised of the following sections to investigate and develop the novel chemistry integration approach for the EDC model: problem statement, where the motivation for the research is given along with a brief history of species transport combustion modelling techniques. In the literature study, a critical review is performed of previous work. Next, the relevant theory of combustion and artificial neural networks is covered. In Chapter 5 the experimental setup is modelled with the traditional chemistry integration procedures in Fluent[®] 17.0. Chapter 6 covers modelling of plug-flow reactors, which was used to generate the training data for the neural networks. The following chapter covers the actual training of the neural networks and the selection of the best-performing architecture. Chapter 8 discusses the implementation of the neural network chemistry integrator. Chapter 9 is the results and comparison section of the document, where the results for the species mass fractions and the temperature of the new approach is compared to the results of the traditional approaches and the experimental data. The performance benefits of the artificial neural network chemistry integration approach are also investigated and compared to the direct integration and in-site adaptive tabulation methods.

Chapter 2

Problem statement and objectives

The current convention in gas phase combustion modelling is the utilisation of the EDM and finite rate chemistry (FR) combustion model hybrid with two-step global reaction mechanisms, according to Scharler (2013). This model works under the assumption that the combustion rate ($\frac{kmol}{m^3.s}$ or $\frac{kg}{m^3.s}$) is either reaction-rate limited (net reaction rate of the chemical reactions, usually determined as a function of the Arrhenius rate), or mixing-rate limited (rate of intermolecular mixing of oxidiser with fuel and/or radical species (Magnussen and Hjertager, 1977)). The EDM-FR model's low computational cost has made it a popular modelling technique for industrial combustion (Shiehnejadhesar *et al.*, 2014), however, the model's empirical constants need to be adjusted depending on the application and are not universally valid (Scharler *et al.*, 2003). The other major shortcoming of this model, according to ANSYS (2016), is that the model will yield incorrect results when using more than two reactions, due to the infinitely fast chemistry or mixing limit assumption. The other combustion models used in industry such as the conserved scalar PDF approach (non-premixed combustion) which can only account for infinitely fast chemistry and their shortcomings will also be discussed.

The EDC combustion model is an extension of the EDM, but is able to incorporate detailed chemical mechanisms. The EDC is based on the turbulent energy cascade (larger eddies break/cascade into smaller eddies due to viscous and inertial forces), and assumes that the species are molecularly mixed in the volume that the smallest eddies occupy (fine structures). The position and volume fraction that these fine structure regions occupy is determined from the turbulent field parameters (k and ϵ). The temporal evolution of the species in the fine structures is then calculated using ideal reactor theory (well-stirred or plug-flow reactors).

The set of differential equations used to resolve the temporal evolution of the species within the fine structure reactors are said to be numerically stiff. The wide variation in chemical time scales has a severe impact on the numerical solution of the set of differential equations that govern the reaction system. The eigenvalues of the Jacobians of these ordinary differential equa-

tions (ODE) systems reflect the time scales. The stiffness characterises the maximal differences in these time scales. In simpler terms, a numerical solution is said to be stiff if the rate of a reaction is very fast in comparison to the other reaction rates. The stiff chemistry differential equations in the fine structure regions are solved in ANSYS Fluent[®] 17.0 either through DI where a backward differencing (BDF) ODE solver is utilised, or through the ISAT method where a look-up table is generated from the results of the chemistry ODEs integration. The look-up tables are populated by the DI solver and are then used to speed up the solving time for regions with similar species and turbulence conditions as pre-solved cells. The DI method is the most accurate and does not have large random access memory (RAM) requirements, but the required CPU time becomes intolerable due to the amount of calculations the processor is required to perform. This imposes limitations on the applicability of the DI scheme, according to Christo *et al.* (1996a). The ISAT method, on the other hand, reduces the CPU load by preventing repeated integration of the same set of ODEs. The resolution of the table of composition increments is restricted by the available RAM of the computer. The required RAM for the ISAT table therefore becomes extremely large if the number of reactions exceeds five (Christo *et al.*, 1996a).

Christo *et al.* (1996a) suggests that the optimum numerical scheme for representing the chemistry should offer the advantages of the ISAT and the DI, at low CPU and RAM resource requirements. The purpose of the present research is therefore to develop a chemistry integration modelling approach to resolve the incremental species changes in the fine structure regions of a multi-step chemical mechanism model with reduced CPU and RAM requirements. The proposed approach will enable engineers to solve industrial scale models using multi-step chemical mechanisms with the EDC model at roughly the same computational requirements as the EDM approach, making the EDC and multi-step chemical mechanisms a generic and viable industrial modelling approach to solve all types of combustion problems (this statement will be further discussed in the literature study). The approach will use artificial neural network (ANN) chemistry integrators to predict the species changes in the fine structure regions over specified chemical time-steps. The objective of the study is therefore to develop the novel chemistry integrator and implement it into user-defined functions that bypass Fluent[®] 17.0's chemical integration routines. The SANDIA Flame D experimental setup is used as validation of the model along with the solutions generated by the standard numerical techniques in Fluent[®] 17.0. The objectives of this study are:

- Research previous work involved with ANNs and combustion.
- Research physics involved with combustion (conservation equations, turbulence chemistry interaction of the EDC and chemical kinetics modelling) and ANN theory.

- Experimental setup modelling and solution.
- Reactor modelling of the chemical ODEs to generate inputs and response variable sets for the ANN to train on.
- Development of own ANN code for training (back propagation algorithm) and predictions (forward propagation).
- Investigate the best-performing ANN architecture.
- Implement ANN memory (weights and bias matrices) and forward propagation into user-defined functions (UDFs) that hook into Fluent[®] 17.0's species transport and energy equations. This will entail rewriting the entire EDC turbulence-chemistry interaction model in Fluent[®] 17.0.
- Generate results to compare the experimental data to the various modelling methods.

The concepts (EDC, ANN, ISAT, DI, etc.) mentioned in the above section will be discussed in depth in the following sections of the document. The hypothesis for the current research states that artificial neural networks could be used to resolve the chemical kinetics in the fine structure regions by replacing the current traditional initial value problem ODE solvers (direct integration and look-up tables) as the chemistry integrator. The hypothesis further states that the ANN integrators should have lower computational requirements than the current chemical reaction kinetics solvers, and is possible of solving combustion simulations within a solving time comparable to the widely used turbulence-chemistry interaction model, the EDM.

Chapter 3

Literature study

In this section a critical discussion of important research pertaining to combustion modelling and the application of ANNs to reaction modelling will be presented. Combustion of solid, liquid and gaseous fuels in boilers are generally non-premixed combustion, thus the literature study will be limited to modelling approaches that address these types of combustion flow configurations (the other types of combustion flow configurations will be briefly discussed in the theory section of the present document).

3.1 Combustion modelling

The simplest approach to model combustion with CFD was proposed by Spalding in 1979 (Versteeg and Malalasekera, 2007). Spalding's proposed method involves only the global nature of the combustion process, therefore detailed kinetics are ignored and the reaction process is modelled as a global one-step infinitely fast reaction where oxidant combines with the fuel in stoichiometric proportions to form products. The mass fractions of the fuel and oxidiser species are used to develop a passive scalar called the mixture fraction. The mixture fraction variable is used to track the species through the computational domain, having its origin in the fuel and oxidiser stream. The model can be extended to be used to predict species formation and energy release for turbulent flows. To account for turbulent fluctuations of the \tilde{Y}_i and \tilde{T} , we need to know the statistics of the variables as a function of the mixture fraction. The probability density function (PDF) is used to calculate the mean quantities of the fluctuating scalars, such as the mixture fraction (Versteeg and Malalasekera, 2007). This approach is only applicable in combustion systems where the fuel and oxidiser species are non-premixed and the chemistry is close to equilibrium (fast chemistry) everywhere in the computational domain. This approach is not able to predict intermediate species (cannot be used for pollutant formation and unburned hydrocarbon prediction) and assumes a linear relationship between the mixture fraction and the oxidiser and

fuel mass fractions. If the oxidiser and fuel species do not react completely (dissociation and slow chemical reactions due to restricted oxygen and large radiative heat losses) these linear relationships no longer hold (Warnatz *et al.*, 2006), and therefore the model cannot be used to accurately resolve all the complex chemical reactions in a boiler.

To circumvent some of the shortcomings of a single-step global reaction of Spalding's proposed model, Bilger (1976) assumed that the gas mixture is in equilibrium everywhere in the domain and reversible multi-step reactions are allowed. This approach can be employed to predict the equilibrium species concentrations, including minor species. The individual species concentrations can be determined as a function of the mixture fraction by using CEA of the reaction mechanisms as an alternative to the global one-step fast chemistry model. Kent and Bilger (1973) successfully showed that the model accurately predicts a hydrogen/air jet diffusion flame. The modelling approach still has some of the shortcomings of Spalding's simple approach, in that it is only applicable to non-premixed flames and chemistry must be close to equilibrium (infinitely fast). The equilibrium assumption causes the model to incorrectly predict pollutant and minor species formation. Jones and Priddin (1982) showed that the model severely over-predicted species such as CO and H_2 in fuel-rich areas. This is due to the local and turbulent time scales that were much smaller than the time required for chemical equilibrium to be reached. The model therefore has certain shortcomings, meaning it is not a general combustion model for simulations in boiler furnaces to accurately predict species distributions

Spalding (1971) also proposed the EBU combustion model. This model also assumes infinitely fast chemistry and was originally developed for premixed flame combustion problems, and works on the assumption of mixed is burnt meaning if the fuel and oxidiser species comes in contact they will burn. The EBU model does not predict the species concentrations based on the mixture fraction as the two non-premixed models mentioned above, but rather solves a transport equation for each of the species in the gas mixture (except nitrogen). The mixing controlled rate of reaction is expressed in terms of the turbulence time scale (k/ϵ , [s]) and empirical constants. The turbulence time scale can be described as the time scale at which large eddies break into smaller eddies. Spalding's original EBU model was later modified by Magnussen and Hjertager (1977) to be able to predict non-premixed combustion, making it a generic turbulence-chemistry interaction model for all types of combustion flow configurations with the assumption of infinitely fast chemistry. To include finite rate chemistry effects, the laminar finite rate of the reactions are calculated. The reaction rate used in the species transport equation is selected as the minimum between the laminar finite reaction rate and the mixing reaction rate. The shortcomings of this approach are: (1) the model becomes unreliable when mixing and kinetic time scales are comparable, due to the mixed is burnt assumption; (2) the model is unable to incorporate chemical mechanisms with more than two steps; (3) the model tends to over-predict mixture temperature;

and (4) the model relies on empirical constants that needs to be tuned for specific combustion problems (thus we may find the model correctly predicts the combustion of volatiles around the over-fire air jet in an industrial boiler, but is unable to account for the low Reynolds number region just above a grate). The EDM-FR model of Magnussen and Hjertjager is therefore not a general combustion model that can be used for pollutant and minor species prediction in boiler furnaces and other combustion systems.

Peters (1984) developed the laminar flamelet model, which is an extension of the mixture fraction non-premixed combustion models that incorporate, to a certain extent, chemical non-equilibrium. The model views the turbulent flame as consisting of an ensemble of stretched laminar flamelets (Versteeg and Malalasekera, 2007). Turbulent flames can be visualised as multiple moving laminar sheets of small reaction zones, called flamelets. In a turbulent flow the flamelets are considered to be stretched and strained by the flow and turbulence. To incorporate the effect of the flame stretching, the scalar dissipation rate or strain rate, or both, are incorporated in the prediction of the species and mixture temperature. The advantage of using the laminar flamelet model is that detailed chemical mechanisms can be used that allow engineers to predict formation of pollutants. The model only moderately implements non-equilibrium and the steady laminar flamelet model is not applicable to flows or combustion conditions where detailed chemical kinetics plays an important role in the prediction of the reaction rates. Given that the goal of the present research is to develop a tool for practising engineers to model industrial problems such as boilers, the transient laminar flamelet model is not considered because of excessive solving time for industrial problems.

Magnussen (2005) presented the EDC model. This model attempts to incorporate the importance of the fine structures at which fuel and oxidiser species are molecularly mixed in turbulent combustion flows. The model treats the fine structure regions as well-stirred or plug-flow reactors and can incorporate detailed chemical mechanisms. The fine structure regions are developed from the energy cascade model and have the same order length and time scales as the Kolmogorov scales. The EDC model therefore resolves the turbulence-chemistry interaction without the need of tuning empirical constants, as the case is for the EDM model. The EDC model (apart from the composition PDF combustion model) is the only industrial combustion model that can incorporate detailed chemistry for all types of combustion flow configurations. The only major drawback of the EDC model is that it is a species transport combustion approach, meaning a transport equation must be solved for each of the species used in the chemical mechanism along with the integration of the species reaction equations in the fine structure regions, making this model very computationally intensive. The present research, as mentioned, sets out to alleviate the computational requirements of the EDC model by using an ANN to predict the incremental species changes in the fine structure regions, rather than integrating the reaction equations that are CPU- and RAM-intensive

(depending on the integration method used).

3.2 Artificial neural network applications to chemical reaction prediction

Blasco *et al.* (1998) investigated the use of neural networks to alleviate the heavy computational resource requirements (CPU load for DI and computer memory load for tabulation methods) of chemistry integrators in laminar direct numerical simulations. Blasco *et al.* modelled only the chemical reaction solution (ordinary differential equations for species transport) for a methane reduced mechanism, and did not model the mass, momentum and energy equations. Four neural networks were trained, three of which were used to predict the composition increment of the species for different reaction time-steps encountered in typical combustion problems. The other network was used to return the temperature and density of a gas based on the species mass fractions. The training datasets, which is the evolution of the species fractions (over the different reaction time-steps), is created by solving the chemistry to steady state for random initial species compositions within pre-set boundary conditions, with a traditional chemical reaction solving technique (DI). Their research found that the ANN was successful in capturing the behaviour of the chemical species evolution during combustion. The computer memory reduction over the tabulation method was 831 times, and the reduction of the CPU time was in the order of 165 – 511, depending on the amount of species in the mechanism. One interesting observation made by Blasco *et al.* (1998) was that the larger the reaction time-step, the larger the error between the direct numerical solution and the ANN. This is due to a more complex relationship between initial and final species concentrations.

Christo *et al.* (1996a) investigated using artificial neural networks with a modelled velocity-scalar joint PDF transport equation for H_2/CO_2 (three-step mechanism) turbulent jet diffusion flames. The training algorithm used was a backward-propagation supervised learning procedure with stochastic gradient descent. The joint PDF transport equation is solved by Monte Carlo technique, in which the PDF is represented by a large number of stochastic particle injections. These particles evolve according to the joint PDF transport equation that simulates diffusion, convection, chemical reaction and molecular mixing. The neural network was then accessed to solve the chemical reactions rather than accessing the chemistry ODE solver or the look-up table. Multiple neural networks were used similarly to Blasco *et al.* (1998) for the various reaction time-steps. Good correlation was found between the look-up table results and the ANN. Christo *et al.* (1996a) found that the computational benefits of the neural network approach over the look-up table and DI methods, both in CPU time and RAM storage requirements, are not substantial for a chemical

mechanism of less than three reactions. However, neural network approach becomes increasingly more superior for increasing reaction mechanism complexity. Christo *et al.* (1996b) further built on their research by proposing a statistical mapping approach to create training datasets by small-scale PDF/Monte Carlo pre-simulations using DI. This produced a set of input samples that were used to generate a set of composition increments over given reaction time-steps.

Kempf *et al.* (2005) investigated the structure of a diffusion flame in terms of length scales, scalar dissipation and flame orientation by using LES. The simulations were performed on the Sandia Flame D methane/air setup (similar to the present research) at a Reynolds number of 22400 (Barlow and Frank, 2007). The flame was modelled using a non-premixed assumption with the steady flamelet model, which was represented by an ANN. ANNs are used for the storage and interpolation of the flamelet libraries. The ANNs are trained with the steady flamelet solutions, integrated with the presumed β -PDF. Kempf *et al.* (2005) used multilayer perceptrons, consisting of two hidden layers with non-linear activation functions and a linear activation function for a single neuron in the output layer. For each species component an ANN was constructed with input variables to network, being: \tilde{f} mixture fraction, \tilde{f}''^2 mixture fraction variance and \tilde{X} scalar dissipation rate. Kempf *et al.* (2005) also stated that this technique reduces storage size of the chemistry library in the computer memory by three orders of magnitude. Good agreement between the LES-ANN/PDF model results and the experimental results was achieved. One shortcoming of the steady flamelet model is that it cannot model slow chemistry and is only suitable for fast chemistry problems (ANSYS, 2016).

Sen and Menon (2009) generated training data using direct numerical simulation of a laminar flame vortex interaction model for a 10-step chemical mechanism with 14 species (syngas). The neural network type used by Sen and Menon (2009) was a multi-layer perceptron network with hyperbolic-tangent sigmoid activation functions. In addition, they added the momentum modification to the stochastic gradient ascent procedure to ensure the global minimum is reached by assisting the model to escape local minima in the high dimensional tuning variable (weights) space. The inputs to the neural network are the species mass fractions and temperature of the control volume, and the output is the reaction rate of a single species. Thus, a neural network was trained for each of the species. The turbulence closure model used was LES. A closure at the resolved scales is not appropriate, according to Sen and Menon (2009), and for the subgrid scale combustion modelling since heat release, volumetric expansion and small-scale turbulent mixing all occur at the small scales that are not resolved. Sen and Menon (2009) used the linear eddy mixing model as a subgrid scale model to account for the combustion processes in the fine scales of the flow field. The trained ANN model was then directly implemented into the subgrid model, without the requirement of filtering. Good results were

generated using the ANN chemistry integrator approach. The speed-up over the DI method was between 4.71 – 11.22, depending on the amount of hidden layers and neurons in each layer. The more neurons used slowed down the ANN integrator, but the speed-up over the DI-method was still substantial. Sen and Menon (2009) also found that the more hidden layers/neurons they used, the closer the results of the new technique corresponded to the traditional approach. The increased flexibility (more weights/tuning variables), therefore enabled the back-propagation procedure to fit the training data more accurately. The ANN method reduced the amount of memory usage by a factor of a thousand. The modelling methodology followed by Sen and Menon (2009) is very comprehensive and also requires large computational resources to solve the LES model; therefore this approach is not a viable industrial application at this stage.

Enami and Fard (2012) used a similar approach as Kempf *et al.* (2005). by training an ANN on the flamelet library and using the mixture fraction and scalar dissipation rate as the input variables and the species mass fractions and temperature as outputs. The major difference between their work is that Enami and Fard (2012) used the RANS turbulence modelling approach, not the LES. Enami and Fard (2012) found a reduction in CFD computational time of 25 times.

Chapter 4

Combustion modelling and artificial neural networks theory

This chapter sets out to cover in depth the theory behind the concepts discussed in the previous sections, which are:

1. Conservation equations used to model fluid flow, species transport and heat transfer
2. Closure of averaged conservation equations
3. Chemical reaction theory and modelling
4. Turbulence-chemistry interaction theory
5. Artificial neural network theory

The theory stipulated above will set the foundation used to later discuss the modelling methodologies utilised to develop the EDC-ANN turbulence-chemistry interaction model of the methane/air piloted jet flame. Figure 4.1 below shows a photograph of the experimental setup modelled in this study.

4.1 Transport equations for mass, species, momentum and energy in reactive flow

There are three main differences between conservation equations for reacting flows and for non-reacting flows (Poinsot and Veynante, 2005):

1. A reacting gas is a non-isothermal mixture of multiple species (hydrocarbons, oxidising species and moisture) where each species has its own continuity equation that needs to be solved. The heat capacities of the reacting gas change significantly through the computational domain, thus making the simulation much more complex and unstable.

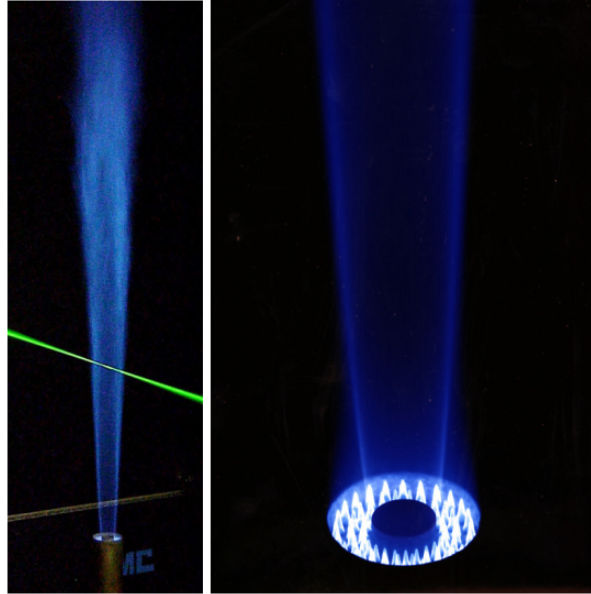


Figure 4.1: Methane/Air turbulent piloted jet flame - Barlow and Frank (2007)

2. Species react chemically, and the combustion/reacting rate is a function of the turbulent mixing and chemical kinetics involved at that specific spatial and temporal position in the domain.
3. Since the gas is a mixture, the transport coefficients such as heat diffusivity, species diffusion, viscosity and so forth needs special attention.

This subsection will define the basic conservation equations used in modelling reactive flow along with their time-averaged forms.

4.1.1 Global continuity equation

The mass conservation/continuity states that the rate of increase of mass in a fluid element must equal the net rate of flow of mass into the fluid element; therefore the continuity equation is defined by:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho u_x) + \frac{\partial}{\partial y}(\rho u_y) + \frac{\partial}{\partial z}(\rho u_z) = S_m \quad (4.1)$$

where S_m [$kg/m^3.s$] is the mass added to the continuous phase from the solid or dispersed second phase, for example fuel particle devolatilisation or moisture evaporation; and u_i [m/s] is the velocity vector in Cartesian coordinates, where $i = 1, 2, 3$ and $1 = x, 2 = y$ and $3 = z$. The density, ρ [$kg/m^3.s$] in combustion flows is variable and depends on pressure, temperature and species concentration (Versteeg and Malalasekera, 2007). The density is calculated using the state equation (ideal gas law).

$$P = \rho \frac{R}{M} T \quad (4.2)$$

where \bar{M} is the mean molecular weight of the mixture and is calculated by the following expression:

$$\frac{1}{\bar{M}} = \sum_{k=1}^N \frac{Y_k}{M_k} \quad (4.3)$$

Equations (4.2) and (4.3) have the following variable definitions: P [Pa] is the absolute pressure of the domain; R [8314.4598J/kmol.K] is the universal gas constant; T [K] is the temperature; Y_k [kg/kg] is the species k mass fraction; and M_k [kmol/kg] is the molecular weight of species k in the mixture with N species in total. For deflagrations (subsonic combustion wave, similar to the model used in this research), flame speeds are small compared to the speed of sound. This leads to a useful simplification (Poinsot and Veynante, 2005): for combustion flows with a low Mach number (usually lower than 0.3), the variations of pressure is negligible and can be assumed to be constant in the state equation (4.2). The density change can be assumed to be directly related to the temperature changes in the combustion flow field.

4.1.2 Species equation

The species transport equation must ensure that the sum of the rate of change in mass of species k , the net rate of decrease of the mass of species k due to convection, and the net rate of decrease of the mass of species k due to diffusion must equal the increase of species k due to sources from the solid particle phase and generation due to chemical reactions (Versteeg and Malalasekera, 2007). The species transport equation for species k is given by the following expression for rectangular coordinates:

$$\rho \left(\frac{\partial Y_k}{\partial t} + u_x \frac{\partial Y_k}{\partial x} + u_y \frac{\partial Y_k}{\partial y} + u_z \frac{\partial Y_k}{\partial z} \right) + \frac{\partial}{\partial x} (\rho Y_k V_{kx}) + \frac{\partial}{\partial y} (\rho Y_k V_{ky}) + \frac{\partial}{\partial z} (\rho Y_k V_{kz}) = \dot{\omega}_k + S_k \quad (4.4)$$

where in equation (4.4) the mass diffusion velocities are $V_{kx} = -\frac{D}{Y_k} \frac{\partial Y_k}{\partial x}$, $V_{ky} = -\frac{D}{Y_k} \frac{\partial Y_k}{\partial y}$ and $V_{kz} = -\frac{D}{Y_k} \frac{\partial Y_k}{\partial z}$. Further in equation (4.4) the following variable definitions are: $\dot{\omega}_k$ [kg/m³.s] is the species k generation source term due to chemical reactions; S_k is the species k source term for generation due to mass transport between the gas phase and the solid phase; and D [m²/s] is the binary mass diffusion coefficient and is obtained by solving a very complex transport equation for full multicomponent diffusion, which can be seen in Kuo (2005) and Poinsot and Veynante (2005). Mathematically, this task is difficult and costly and most commercial codes use simplified approaches to calculate the binary mass diffusion coefficient. It is common practice to assume constant diffusion coefficient for all the species. Whilst not very accurate, it greatly simplifies the simulations (Versteeg and Malalasekera, 2007) for laminar flows where species diffusion plays an important role. For turbulent flows, the turbulent convection modelled as diffusion dominates over the laminar dif-

fusion; therefore the assumption of constant diffusion coefficient is reasonable. This assumption was used in the methane jet flame modelled in later sections of this document, given that the objective of this study is to investigate the accuracy and performance benefits of an EDC-ANN model compared to the traditional EDC-DI or EDC-ISAT models.

4.1.3 Momentum equation

Newton's second law states that the rate of change of momentum of a fluid particle equals the sum of the forces acting on that fluid particle. Three approaches to derive the momentum conservation equation exist, namely:

1. infinitesimal particle approach (Lagrangian approach)
2. infinitesimal control volume approach (Eulerian approach)
3. finite control volume approach

The latter will be used here to derive the momentum conservation equation. The finite control volume approach considers a fixed control volume of finite size and arbitrary shape and uses the Gauss' divergence theorem to relate the surface and volume integrals (Kuo, 2005). The momentum equation for a finite control volume fixed in space is:

$$\frac{\partial}{\partial t} \int_V \rho \mathbf{u} dV + \int_A \mathbf{u} (\rho \mathbf{u} \cdot d\mathbf{A}) = \mathbf{F}_s + \int_V \mathbf{B} dV \quad (4.5)$$

where in (4.5): V is the volume of the arbitrary shaped control volume; \mathbf{u} is the velocity vector in three dimensional rectangular coordinates; \mathbf{A} is the surface area of the control volume; \mathbf{F}_s is the surface forces acting on the control volume; and B is the body forces such as gravity and buoyancy forces acting on the mass of the control volume. Applying Gauss' theorem, the momentum equation in tensor notation is:

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_i u_j)}{\partial x_j} = \frac{\partial \sigma_{ji}}{\partial x_j} + B_i \quad (4.6)$$

where σ_{ji} is the stress tensor acting on the surface of the control volume. The stress tensor in a three-dimensional space has nine components as seen in (4.7). The subscripts 1, 2 and 3 are the dimensional components x, y and z respectively.

$$\sigma_{ji} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} \quad (4.7)$$

The Newtonian fluid assumption, which requires the shear stress (off-centre components σ_{ji} where $i \neq j$) to be linearly proportional to the rate of angular

deformation, yields the following equation:

$$\sigma_{ji} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \quad i \neq j \quad (4.8)$$

Next, let us consider the linear deformation of the fluid control volume. As for shear stresses, there is a linear relationship between the normal or linear stress (σ_{11}) and the normal or linear strain rate (e_{11} where $e_{ij} = \frac{1}{2}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i})$) in the same direction. In addition to normal strain rate affected by the normal stress, it is also influenced by the strain rate in other directions (e_{22} and e_{33}) as seen in the following expressions. Because there is no preferred direction between e_{22} and e_{33} , they are related by the λ coefficient. The fluid is therefore assumed to be isotropic. With some derivation, as seen in Kuo (2005), the diagonal stress components can be expressed by:

$$\begin{aligned} \sigma_{11} &= -P + \lambda(e_{11} + e_{22} + e_{33}) + (c - \lambda)e_{11} \\ \sigma_{22} &= -P + \lambda(e_{11} + e_{22} + e_{33}) + (c - \lambda)e_{22} \\ \sigma_{33} &= -P + \lambda(e_{11} + e_{22} + e_{33}) + (c - \lambda)e_{33} \end{aligned} \quad (4.9)$$

where in equation (4.9) the coefficients ($c - \lambda = 2\mu$) are valid for isotropic Newtonian fluid and $\lambda = \mu' - \frac{2}{3}\mu$. Here μ' is called the bulk viscosity and caused by relaxation effects between the translational motion and the various integral degrees of freedom (Kuo, 2005). According to Kuo (2005), the bulk viscosity is negligible in combustion processes. Applying these simplifications (4.9) can be condensed into the following general stress tensor for combustion flow (and basic fluid flow):

$$\sigma_{ij} = P\delta_{ij} + \left(\mu' - \frac{2}{3}\mu\right)(e_{11} + e_{22} + e_{33})\delta_{ij} + 2\mu e_{ij} \quad (4.10)$$

The stress tensor equation (4.10) can be substituted into (4.6) to yield the general form of the Navier-Stokes equation. It should be noted that the density variation is very pronounced for combustion flow problems; therefore the incompressible form of the Navier-Stokes equations has very limited usage and the compressible form of the equation must be used even for low Mach number flows.

4.1.4 Energy equation

The temperature of a gas volume in combustion flows depends on the thermodynamic state and mass fractions of each fluid constituent. The model used in the present research, as shown, uses the species transport approach in modelling the species concentration and energy distribution through the computational domain.

The major cause of heat fluxes in most non-reactive fluid flow problems (\mathbf{q}) is conduction between zones in the fluid domain where temperature gradients (∇T) occur. In reacting- or multiple species flows, additional effects exist

that generate heat, namely inter-diffusion - and Dufour fluxes. The former is caused when the average velocity \mathbf{u}_k of species component k is different from the mass-average velocity of the mixture. There is then a diffusion mass flux $\rho_k \mathbf{V}_k$ of species component k relative to the bulk mass flux of the mixture flowing at the mass-average velocity in the local control volume. If the average enthalpy associated with the k component is h_k [$J/kmol.K$], then the additional energy flux transported by the k fluid mass from one control volume to the other is equal to $\rho h_k Y_k \mathbf{V}_k$. The total energy flux due to inter-diffusion is $\rho \sum_{k=1}^N h_k Y_k \mathbf{V}_k$, (Kuo, 2005). The Dufour effect is where concentration gradients generate a heat flux, and is based on Onsager's reciprocal relation of irreversible thermodynamic processes. According to Kuo (2005), the Dufour effect is negligible in combustion flows. The total heat flux in reacting or multi-species flows is therefore:

$$\mathbf{q} = -k \nabla T + \rho \sum_{k=1}^N h_k Y_k \mathbf{V}_k \quad (4.11)$$

where in (4.11), k [$W/m.K$] is the thermal conductivity of the mixture. The law of conservation of energy states that the rate of accumulation of internal energy and kinetic energy must equal the sum of the following energy flows (Kuo, 2005):

1. net rate of influx of internal energy and kinetic energy by convection
2. net rate of heat addition due to heat flux \mathbf{q}
3. rate of heat added by heat sources (chemical reactions and radiation)
4. net rate of work done on fluid element by surroundings

The steady-state energy equation can therefore be written in tensor form as:

$$\frac{\partial}{\partial x_i} [u_i (\rho E + P)] = \frac{\partial}{\partial x_j} \{k_{eff} \frac{\partial T}{\partial x_j} + u_i (\tau_{ij})_{eff}\} + S_h \quad (4.12)$$

where in equation (4.12); E [W/m^3] is the total energy of the fluid; k_{eff} is the effective thermal conductivity which is $k_{eff} = k_t + k$; where, in turn, k_t is the turbulent conductivity which is calculated by the turbulence model (discussed later on); τ_{eff} is the viscous dissipation term; and S_h is the energy source term used to account for sources such as radiation, chemical reactions and discrete phase energy transfer (ANSYS, 2016).

The total energy term is a function of the kinetic energy and the internal energy (Cengel and Boles, 2015), which can be written for compressible flows as:

$$E = h - \frac{P}{\rho} + \frac{u^2}{2} \quad (4.13)$$

The sensible enthalpy h is defined as follows for ideal gases:

$$h = \sum_{k=1}^N Y_k h_k \quad (4.14)$$

where in (4.14) species k sensible enthalpy is calculated by:

$$h_k = \int_{T_{ref}}^T c_{P,k} dT \quad (4.15)$$

where in equation (4.15) $c_{P,k}$ is the constant pressure specific heat of species k . The energy source term due to chemical reaction in equation (4.12), according to ANSYS (2016), is:

$$S_{h,reaction} = - \sum_{k=1}^N \frac{h_k^0}{M_k} \dot{\omega}_k \quad (4.16)$$

where in (4.16) h_k^0 [$J/kmol.K$] is the formation enthalpy of species k and $\dot{\omega}_k$ [$kg/m^3.s$] is the net volumetric generation term of species k due to chemical reactions. For the purposes of this study, radiation is ignored because there are no radiating particles such as soot due to the dilution of the fuel stream, and there is no wall boundaries that interact with the flame through radiation exchange. It will be shown later on in the document that the results generated without radiation activated is still satisfactory and this simplification decreases the solving time of the simulations.

4.1.5 Time-averaged form of conservation equations

The methane piloted jet flame used to implement and validate the new ANN chemistry solver is highly turbulent, thus the conservation equations discussed in this section to this point must be modified to account for turbulent fluctuations. Turbulence is characterised by random fluctuations of all the fluid properties through time and space at sufficiently high Reynolds numbers, depending on the geometry of the fluid domain (Poinsot and Veynante, 2005). Mathematically, any property in the turbulent flow field ϕ is usually comprised of a mean quantity $\bar{\phi}$ and a fluctuating quantity ϕ' (Kays *et al.*, 2005).

$$\phi = \bar{\phi} + \phi' \quad (4.17)$$

The fluid variables' instantaneous forms can be written in their respective mean and fluctuating components (velocity and stresses are in tensor notation)

$$\begin{aligned} u_i &= \bar{u}_i + u'_i \\ Y_k &= \bar{Y}_k + Y'_k \\ h &= \bar{h} + h' \\ P &= \bar{P} + P' \\ \sigma_{ij} &= \bar{\sigma}_{ij} + \sigma'_{ij} \end{aligned} \quad (4.18)$$

Equations (4.18) for the fluid variables are referred to as the Reynolds decomposition. Modifying the conservation equations to be able to model turbulent combustion flow, the fluid variable decompositions must be inserted into their respective transport equations. This yields their respective turbulent forms. The insertion of these decompositions creates quantities in the conservation equations that are difficult to solve; thus turbulence models like $k - \epsilon$ are used to solve the conservation equations and close off these new quantities.

In addition to the closure of these new quantities, combustion flows as mentioned have large density variation through the domain; thus Reynolds averaging for variable density flows introduces many other terms in the turbulent conservation equations that need to be handled by the turbulence models. To avoid difficulty, mass weighted averages (Favre averages) are usually used for combustion flows (Poinsot and Veynante, 2005):

$$\tilde{\phi} = \frac{\overline{\rho\phi}}{\bar{\rho}} \quad (4.19)$$

The decomposition can now be written as:

$$\phi = \tilde{\phi} + \phi'' \quad (4.20)$$

Equation (4.20) can now be used similarly to equation (4.17) to create the decompositions in (4.18). Inserting the Favre-averaged decompositions into the conservation equations and assuming steady-state conditions yields the following unclosed-conservation equations; for the derivations not shown, see Poinsot and Veynante (2005), Kays *et al.* (2005) and ANSYS (2016):

Mass:

$$\frac{\partial}{\partial x_i}(\bar{\rho}\tilde{u}_i) = 0 \quad (4.21)$$

Momentum:

$$\frac{\partial}{\partial x_i}(\bar{\rho}\tilde{u}_i\tilde{u}_j) + \frac{\partial\bar{P}}{\partial x_j} = \frac{\partial}{\partial x_i}\left\{\mu\left[\frac{\partial\tilde{u}_j}{\partial x_i} + \frac{\partial\tilde{u}_i}{\partial x_j} - \frac{2}{3}\delta_{ij}\frac{\partial\tilde{u}_i}{\partial x_i}\right]\right\} + \frac{\partial}{\partial x_i}(-\bar{\rho}\widetilde{u_j''u_i''}) \quad (4.22)$$

Species:

$$\frac{\partial}{\partial x_j}(\bar{\rho}\tilde{u}_j\tilde{Y}_k) = -\frac{\partial}{\partial x_j}(\bar{V}_{k,i}Y_k + \bar{\rho}\widetilde{u_j''Y_k''}) + \bar{\omega}_k \quad k = 1, 2, \dots, N \quad (4.23)$$

Energy:

$$\frac{\partial}{\partial x_i}(\tilde{u}_i[\bar{\rho}\tilde{E} + \tilde{P}]) = \frac{\partial}{\partial x_j}\left[k\frac{\partial T}{\partial x_j} + \bar{\rho}\widetilde{u_j''E''} + \tilde{u}_i(\tilde{\tau}_{ij})_{eff}\right] + \tilde{S}_h \quad (4.24)$$

The averaging of the conservation equations introduces three new terms, namely:

1. Reynolds turbulent stresses ($-\overline{\rho u_i'' u_j''}$), which is the shear force per unit area due to eddy motion perpendicular to the mean flow (Kays *et al.*, 2005).
2. Species turbulent flux ($\overline{\rho u_j'' Y_k''}$), which occurs due to the turbulent fluctuations, which in turn is caused by the turbulent stresses.
3. Energy turbulent flux ($\overline{\rho u_j'' E''}$).

Performing turbulent calculations to evaluate these new flow quantities is referred to as the turbulent "closure" problem (Kays *et al.*, 2005). For variable-density flows, in Fluent[®] 17.0, the normal RANS species and energy equations are interpreted as Favre-averaged Navier-Stokes equations, with the velocities representing mass-averaged values. Fluent[®] 17.0 therefore uses the normal RANS equations for variable density flows (ANSYS, 2016).

One method of modelling these extra stresses and fluctuating quantities employs the Boussinesq hypothesis, which relates the Reynolds stresses to mean velocity gradients and the kinetic/dissipation energy in the flow field (for the rest of the document the tilde and over bar will be dropped for sake of simplification) (Hinze, 1975).

$$-\overline{\rho u_i'' u_j''} = \mu_t \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} (\rho k + \mu_t \delta_{ij} \frac{\partial u_k}{\partial x_k}) \quad (4.25)$$

where μ_t is the turbulent dynamic viscosity, which is a proportionality factor used to relate the Reynolds stresses to the velocity gradients and turbulent energy ($k = \frac{1}{2} \sum_{k=1}^3 \overline{u_k'' u_k''}$). A turbulence model (such as $k - \epsilon$, $k - \omega$ and *RSM*) is used to evaluate the turbulent dynamic viscosity term. Thus, using the turbulent dynamic viscosity along with the correct turbulence model for the problem, the new terms in the conservation equations can be "closed off" as shown in the next section.

4.2 Turbulence modelling (closure of averaged conservation equations)

There are three computational approaches for turbulent combustion modelling, namely RANS (Reynolds/Favre Averaged Navier-Stokes), LES (Large Eddy Simulation) and DNS (Direct Numerical Simulation) according to Poinot and Veynante (2005). The latter two of these approaches are very computationally intensive and are not feasible solutions for industrial simulations of turbulent combustion flows because they must be solved transiently on very fine computational meshes. The present research will therefore focus on the RANS approach and, more specifically, the two-equation approaches such as the $k - \epsilon$

and realizable $k - \epsilon$ models where the kinetic and dissipation energy are solved in parallel to the conservation equations.

The Boussinesq hypothesis is usually used along with a two-equation turbulence models (Poinsot and Veynante, 2005), where the turbulent viscosity is solved as a function of k and ϵ .

$$\mu_t = \rho C_\mu \frac{k^2}{\epsilon} \quad (4.26)$$

The turbulent viscosity and classical gradient assumption are used to close the turbulent fluxes in the conservation equations. The correct turbulence model, capable of resolving the turbulence field for the specified type of flow (jet, wake and confined) with relative accuracy, must be selected for the problem. As mentioned in the introduction, the validation case to be solved in the current research is a piloted turbulent combustion jet flame. Turbulent jet flows classify as one of the most important types of turbulent flows in engineering, namely free turbulent flows. Another type of turbulent flow which falls into this classification are turbulent wake flows (Versteeg and Malalasekera, 2007). In these types of flows, a mixing layer is formed between the fast- and slow-moving fluid, as seen in figure 4.2 below.

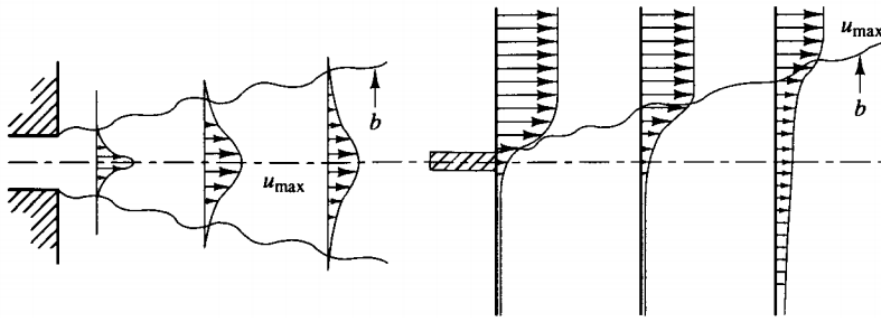


Figure 4.2: Left: Jet flow velocity distribution, Right: Mixing layer formed between fast- and slow- moving fluids Versteeg and Malalasekera (2007)

In a jet, a region of high-speed flow is surrounded by a stationary or slow-moving fluid that creates an initially thin layer of high-velocity fluid. The fast-moving fluid gradually dissipates along the centerline away from the nozzle. The transition to turbulence occurs after a very short distance in the flow direction from the point where the fast and slow streams initially meet. The turbulence causes vigorous mixing of the fast-moving fluid with the slow-moving medium, as seen in figure 4.2, where the mixing layer gradient is large (close to the jet origin). This excessive mixing causes a rapid widening of the jet. From experimental measurements by Gutmark and Wygnanski (1976), it was seen that the largest fluctuating stress components ($-\overline{u_i'' u_j''}$) are found in the region where the mean velocity gradient is the largest (close to the origin

of the jet). The widening of the jet highlights the connection between turbulence production and the large mean sheared flows. Where shear is high turbulence quantities such as the RMS velocity fluctuations are high and their distribution anisotropic. In contrast, where the shear is low, the turbulence production lowers and the turbulent eddies dissipates and the flow becomes more isotropic.

The standard $k - \epsilon$ turbulence model performs well for confined or boundary layer flows (Versteeg and Malalasekera, 2007), but does not perform well for flows with a high mean shear rate or a large separation (Shih *et al.*, 1995), which is the case for jet flows. The inaccuracy lies in the method by which the standard approach models the dissipation rate. The standard model severely over-predicts the spreading rate of axisymmetric jets due to the over-prediction of the eddy viscosity [equation (4.26)] according to Shih *et al.* (1995). The over-prediction is due to the isotropic assumption of the $k - \epsilon$ two-equation turbulence model and the fact that the actual turbulence field is anisotropic due to large mean shear rates. The Reynolds stress model, which is a six-equation model, is suited to deal with anisotropic turbulence; however, Zahirovic *et al.* (2006) showed that the RSM shows no significant improvement over the realizable $k - \epsilon$ turbulence model for jet flames. As mentioned frequently throughout the current document, the goal is to develop an industrial modelling tool. The RSM model is computationally more intensive because it solves more transport equations than the realizable $k - \epsilon$ model. To model the turbulence more accurately, the realizable $k - \epsilon$ model is used rather than the standard model. The realizable $k - \epsilon$ turbulence model improves the prediction accuracy by using an enhanced model dissipation rate equation, which has been derived from the exact equation for the transport of the mean-square vorticity fluctuation, and the new eddy viscosity formulation ensures realizability and contains the effect of mean rotation and turbulent stresses. The eddy dissipation equation's production term is similar to the spectral energy transfer concept and is believed to capture turbulent vortex stretching and dissipation terms more accurately, according to Shih *et al.* (1995). The $k - \omega$ family of turbulence models can also be used to model the jet flow phenomena (free shear flows), and will be compared to the realizable $k - \epsilon$ model later on along with the RSM (6-equation models) turbulence models.

The model's equation will be presented and discussed along with the closure of the turbulent fluctuating quantities in the conservation equations. The steady-state realizable $k - \epsilon$ transport equations for kinetic energy and dissipation rate are:

$$\frac{\partial}{\partial x_j}(\rho k u_j) = \frac{\partial}{\partial x_j} \left[\left(\mu + \frac{\mu_t}{\sigma_k} \right) \frac{\partial k}{\partial x_j} \right] - \rho \epsilon + G_k + G_b - Y_M \quad (4.27)$$

$$\frac{\partial}{\partial x_j}(\rho \epsilon u_j) = \frac{\partial}{\partial x_j} \left[\left(\mu + \frac{\mu_t}{\sigma_\epsilon} \right) \frac{\partial \epsilon}{\partial x_j} \right] + \rho C_1 S_\epsilon - \rho C_2 \frac{\epsilon^2}{k + \sqrt{\nu \epsilon}} + C_{1\epsilon} \frac{\epsilon}{k} C_{3\epsilon} G_b \quad (4.28)$$

$$C_1 = \max\left(0.43, \frac{\eta}{\eta + 5}\right), \quad \eta = S \frac{k}{\epsilon}, \quad S = \sqrt{2S_{ij}S_{ij}}$$

where in equations (4.28) and (4.27) the following variables is are used:

1. G_k is the turbulence production term due to velocity gradients.
2. G_b is the turbulence production term due to buoyancy forces.
3. Y_M is the term that accounts for fluctuating dilatation in compressible turbulence flows.
4. $C_2 = 1.9, C_{1\epsilon} = 1.44, \sigma_k = 1.0$ and $\sigma_\epsilon = 1.2$ are model constants.

The realizable $k - \epsilon$ model also uses equation (4.26) to calculate the eddy viscosity. The difference between the standard $k - \epsilon$ model and the realizable model is that the C_μ is not constant as for the standard model, but calculated as follows:

$$C_\mu = \frac{1}{A_0 + A_S \frac{kU^*}{\epsilon}}$$

$$U^* = \sqrt{S_{ij}S_{ij} + \tilde{\Omega}_{ij}\tilde{\Omega}_{ij}} \quad (4.29)$$

$$\tilde{\Omega}_{ij} = \Omega_{ij} - 2\epsilon_{ijk}\omega_K$$

$$\Omega = \bar{\Omega}_{ij} - \epsilon_{ijk}\omega_K$$

where $\bar{\Omega}_{ij}$ is the mean rate-of-rotation tensor viewed in a moving reference frame with angular velocity ω_K . The model constant for the eddy viscosity calculation is:

$$A_0 = 4.04 \quad A_S = \sqrt{6}\cos\phi$$

$$\phi = \frac{1}{3}\cos^{-1}(\sqrt{6}W) \quad W = \frac{S_{ij}S_{jk}S_{ki}}{\tilde{S}^3} \quad \tilde{S} = \sqrt{S_{ij}S_{ij}} \quad S_{ij} = \frac{1}{2}\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right) \quad (4.30)$$

The turbulent production due to velocity gradients in (4.27) is calculated by:

$$G_K = \mu_t S^2 \quad S \equiv \sqrt{2S_{ij}S_{ij}} \quad (4.31)$$

where S_{ij} is the same as the variable in (4.30), which is the modulus of the mean rate-of-strain tensor. The turbulent production due to buoyancy is assumed to be negligible in the developed model due to the fact that the flow is driven mainly by convection from the inlet jet velocity and the effects of buoyancy driven flow is small in comparison.

To close the Reynolds turbulent stresses term in equation (4.22), the Boussinesq equation (4.25) is inserted into the momentum equation. This equation is then solved along with (4.27), (4.28), (4.29) and (4.30) to determine the turbulent velocity field of the computational domain.

The turbulent energy flux in equation (4.24) can be modelled using the gradient assumption as shown in Poinso and Veynante (2005):

$$\overline{\rho u_j'' E''} = \frac{c_P \mu_t}{Pr_t} \frac{\partial T}{\partial x_j} \quad (4.32)$$

where the default value for the turbulent Prandtl number in Fluent[®] 17.0 is 0.85. Experimental data shows that the turbulent Prandtl number for turbulent flows of fluids with Prandtl numbers ranging between 0.7 – 5.9 is scattered around 0.85 (Kays *et al.*, 2005) in the logarithmic region close to walls. The Prandtl number for the validation case modelled is in the range of 0.7 – 0.89, thus in reality the turbulent Prandtl number will be slightly higher than the Fluent[®] 17.0 default, but because the problem solved is a jet, there is an absence of wall effects and the assumption of 0.85 will lead to a small error. Inserting (4.32) in (4.24) yields the "closed" turbulent energy equation that is solved for the jet flame model. This is shown in equation (4.12) where $\frac{c_P \mu_t}{Pr_t} = k_t$.

The last turbulent quantity that needs to be addressed is the turbulent species flux in equation (4.23). Here a similar approach is followed as in the case of the energy turbulent flux closure. The turbulent species flux can be equated to the gradient of the species mass fraction times a proportionality quantity, as follows:

$$\overline{\rho u_j'' Y_k''} = -\frac{\mu_t}{Sc} \frac{\partial \tilde{Y}_k}{\partial x_j} \quad (4.33)$$

where in equation (4.33), Sc is the turbulent Schmidt number. In Fluent[®] 17.0, the Schmidt number is set to 0.7 (ANSYS, 2016) as default. Generally in turbulent combustion, the turbulent diffusion overwhelms the laminar diffusion, and the effect of laminar diffusion is almost negligible. Inserting (4.33) into (4.23) and with some derivation, the general steady-state turbulent form of the species transport for the k species equation used in Fluent[®] 17.0 is:

$$\frac{\partial}{\partial x_j} (\rho u_j Y_k) = -\frac{\partial}{\partial x_j} \left[-(\rho D_{k,m} + \frac{\mu_t}{Sc_{kt}}) \frac{\partial Y_k}{\partial x_j} \right] + \dot{\omega}_k \quad (4.34)$$

In this section, all the turbulent conservation equations were closed using the two-equation approach refined for axisymmetric jet flows. Using these equations and methods mentioned above, the mass, momentum, species and energy transport equations of the mixture fluid can be modelled. The only phenomena left to be investigated are the species generation/destruction and energy sources due to chemical reactions ($\dot{\omega}_k$ and $S_{h,reaction}$ in the species and energy

conservation equations respectively). This investigation into workings of these phenomena is split into two sections; firstly, the theory governing chemical reactions and rate of creation/destruction; and finally, the interaction between turbulence mixing and chemical reactions.

4.3 Chemical reaction theory and modelling

The study of the elementary reactions and their rates is a specialised field of physical chemistry, where chemists define the chemical reaction pathways leading from reactants to products. The study also involves measuring the rates of the various chemical reactions that make up a chemical mechanism and, using these measured rates, developing empirical correlations to calculate the rates (Turns, 2000). The developed reaction rate equations are then used by scientists and engineers to construct computer models of the reacting system.

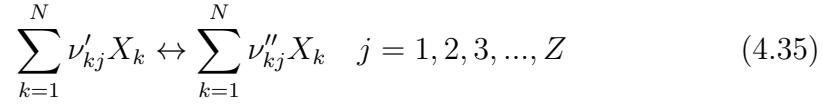
In turbulent combustion systems, the shift of species mass fractions and release of energy due to chemical reactions is a function of turbulent mixing of the oxidiser with fuel or radical species and the chemical kinetics (also known as chemical reaction rate). There are various approaches that are used to account for the turbulence-chemistry interaction. The two models we will be focusing on are the EDM-FR and EDC models, both of which account for the chemical kinetics in a different manner. The former uses the laminar finite-rate chemistry approach, and the latter uses plug-flow/well-stirred reactor approach to determine the temporal evolution of the species mass fractions and energy release rate. Therefore, in this section of the dissertation, the theory governing finite-rate chemistry and reactor modelling applicable to combustion will be discussed, along with a discussion of the various types of reaction mechanisms and the type selected for the current research. Note that the implementation of these chemical reaction modelling methodologies in turbulent flows will be discussed in the next section.

4.3.1 Finite-rate chemistry formulation for volumetric (gas-phase) reactions

Chemical reactions take place at definite rates that are influenced by the following conditions of the system: (1) concentration of each species; (2) temperature; (3) pressure; (4) presence of catalyst or inhibitor; and (5) radiative effects (Kuo, 2005). The rate of reaction [$kmol/m^3.s$] may be expressed as the rate of increase of a product species as a function of the reactant species concentrations, or the rate of decrease of a reactant species as a function of all the reactant species' concentrations for a given reaction (Kuo, 2005). The finite-rate chemistry model is used to calculate the chemical source term $\dot{\omega}_k$ in equation (4.4) for laminar flow problems. This approach is inaccurate for tur-

bulent combustion problems due to the highly non-linear modelling approach to chemical kinetics (ANSYS, 2016).

Consider a chemical system of N species reacting through Z reactions:



where X is the symbol for the chemical species k , ν'_k and ν''_k are the molar stoichiometric coefficients of species k as reactant or product respectively in reaction j . The chemical source term (net rate of production) for species k , $\dot{\omega}_k$ is the sum of all rates for the specific species produced by all Z reactions:

$$\dot{\omega}_k = M_k \sum_{j=1}^Z (\nu''_j - \nu'_j) q_j \quad (4.36)$$

where q_k [$kmol/m^3.s$] is the rate of progress for reaction j and is defined by:

$$q_j = k_{fj} \prod_{k=1}^N [X_k]^{\nu_{kj}'} - k_{rj} \prod_{k=1}^N [X_k]^{\nu_{kj}''} \quad (4.37)$$

In equation (4.37) $[X_k]$ is the molar concentration of species k ($[X_k] = \frac{\rho Y_k}{M_k}$) and k_{fj} , k_{rj} are the forward and reverse specific reaction rate constant (Kuo, 2005). The specific reaction rate constants for the Z reactions are modelled using the Arrhenius law. The Swedish chemist/physicist Svante Arrhenius postulated that only those molecules that possess energy greater than a certain amount of E_a activation energy will react (Glassman and Yetter, 1987). Using Arrhenius' Boltzman factor, the forward and reverse specific reaction rate constants for the j reaction can be calculated using:

$$k_j = A_j T^{\beta_j} \exp\left(\frac{-E_{aj}}{RT}\right) \quad (4.38)$$

where in equation (4.38) A_j [$1/s$] is the pre-exponential factor which is assumed to include the effect of the collision terms and the steric factor associated with orientation of the colliding molecules (Kuo, 2005); β_j is the temperature exponent for the j reaction; R is the universal gas constant; and T is the temperature of the system. For elementary reactions, the activation energy and pre-exponential factor are experimentally obtained. These values are often greatly disputed in the kinetics community according to Poinso and Veynante (2005). Further for global and multi-step mechanisms (quasi-global), the stoichiometric values ν_{kj}' , ν_{kj}'' are replaced by reaction orders, which are also experimentally obtained, but this will be discussed later on in this section. For laminar flow problems equation (4.36) can be directly inserted into equation (4.4) to account for net species creation or destruction due to

chemical reactions, and to account for the energy release, $\dot{\omega}_k$ is substituted into equation (4.16) and this equation in turn inserted into equation (4.12). The EDM-FR turbulence-chemistry interaction model uses the laminar finite rate chemistry approach as discussed in the current section, and also accounts for turbulent mixing. The model then switches between the minimum of the chemical reaction rate and the mixing rate, but this will be discussed in more detail later on. The EDC model, on the other hand, uses the chemical reactor modelling approach to account for the chemical kinetics in the fine structure regions (molecularly well-mixed volumes) of the fluid domain, therefore in the next section reactor modelling and the solution of stiff chemistry mechanisms will be discussed.

4.3.2 Chemical reactor modelling

Ideal reactors allow us to simplify the energy, momentum and species transport equations by assumptions (perfectly stirred and no mass diffusion) thus focussing the analysis on the chemical kinetics. Species and energy state changes due to chemical reactions in ideal reactors depend on the reactor type, scale, geometry, mode of operation and operating conditions (Perry, 2008). Figure 4.3 shows a representation of the small reactors formed in a control volume by the turbulent mixing. The species from the surrounding fluid is swept into the small reactors and allowed to react over the specified reaction time. Once the reaction time has elapsed, the gas mixture constituents are ejected from the reactor into the surrounding fluid and the surrounding fluid composition updated. This process is repeated until steady state is reached. The variables m^* , Y_k^* and γ^* represent the fine scales mass transfer rate, species mass fraction and volume fraction respectively and Y_k is the surrounding fluid mass fraction.

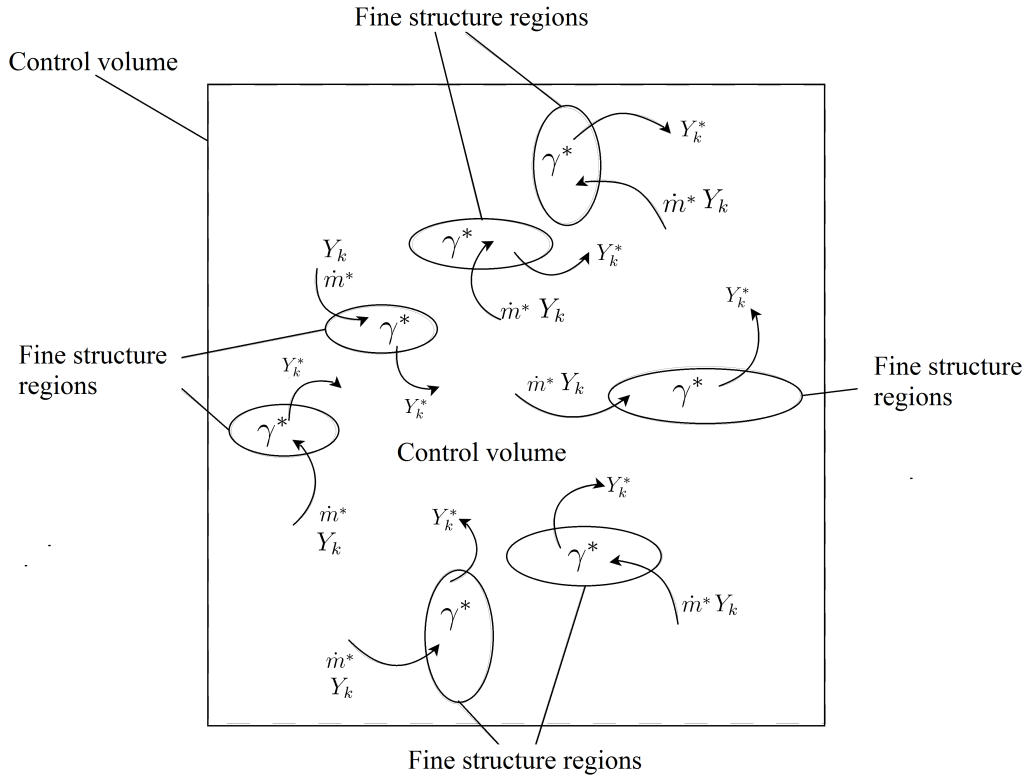


Figure 4.3: Graphic representation of species transfer rate from the surrounding fluid into the fine structure regions

In literature, two types of ideal reactors are utilised to solve the species and energy equations of the fine structure regions in the EDC model. Magnussen (1981) originally proposed using well-stirred (perfectly-stirred) reactors, and later on Jessee *et al.* (1993) showed that the well-stirred reactor might lead to convergence problems during the iterative solution of the reaction equations. They proposed using the plug-flow reactor, which considerably simplified the numerical solution; therefore the discussion below will be limited to the plug-flow reactor model (this is also the model used by Fluent[®] 17.0's EDC combustion model). The temporal evolution of the species and energy in the fine structure regions of the EDC model occurs over the fine structure time scale τ^* (discussed in next section); thus the conservation equations under consideration must be solved over the period $0 \rightarrow \tau^*$ seconds. For the EDC turbulence-chemistry model, only the species and energy transport equations are required to be solved (discussed in next section). A plug-flow reactor is, as mentioned, an ideal reactor that has the following attributes, according to Turns (2000):

1. Steady-flow, $\dot{m}_{in} = \dot{m}_{out}$
2. No mixing in the axial direction

3. Uniform properties in control volume
4. Ideal gas behaviour

The PFR can be assumed to have a pseudo-batch reactor behaviour, meaning the species mixture is transferred into the reactor by the turbulent mixing and then reacts over the period $0 - \tau^*$ before leaving the reactor, as seen in the schematic below.

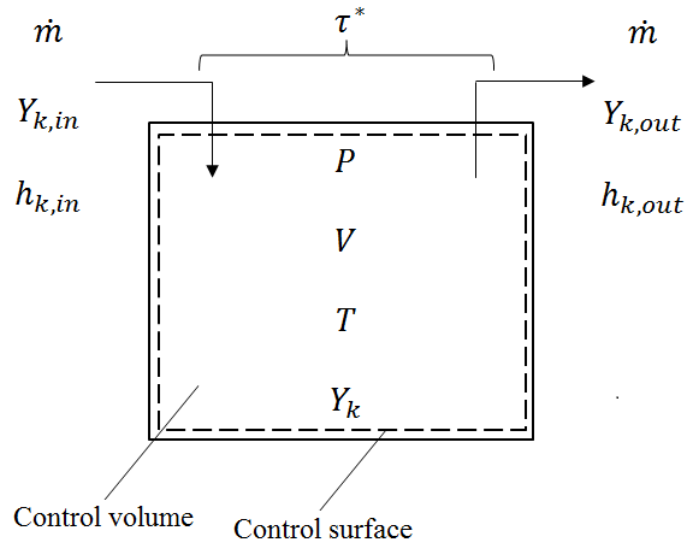


Figure 4.4: Schematic of a plug-flow reactor, Turns (2000)

The species transport equation for the PFR is:

$$\frac{dY_k}{d\tau^*} = \frac{\dot{\omega}_k}{\rho} \quad (4.39)$$

The energy equation is:

$$\frac{dT}{d\tau^*} = -\frac{1}{\rho c_P} \sum_{k=1}^N \frac{h_k}{M_k} \dot{\omega}_k \quad (4.40)$$

In equation (4.40) the enthalpy term h_k is the partial molar enthalpy of the species k . As mentioned, the species equation must be solved for each of the species in the reaction mechanism. This leads to a numerical issue, namely the stiffness of the solution. The species equations form a stiff chemistry problem. The degree of stiffness is determined by the reactant concentrations, pre-exponential factors and activation energies of reactions. A high concentration and/or large pre-exponential factor and/or low activation energy will lead to a stiff reaction equation. A set of differential equations are considered stiff when one or more variables (mass fractions) change very rapidly over time while the

remaining variables change very slowly (Turns, 2000). This disparity in time scales is due to large variation in activation energies and pre-exponential factors in reaction formulations (Poinsot and Veynante, 2005). Hindmarch (1983) and Kee *et al.* (1991) offer solution strategies to stiff chemistry problems. The discretisation and solution technique used in the present work is the BDF along with the variable coefficient ordinary differential equation (VODE) solver respectively. The reader is advised to see Appendix A for further discussion of the solution strategy of the differential equations.

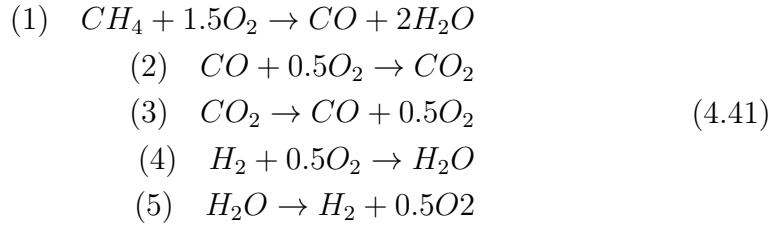
4.3.3 Reaction mechanisms

There are multiple levels of detail that can be incorporated into the reaction mechanisms used in combustion simulations. Starting from the most complex, here are some of the approaches used in research and the industry today:

1. Elementary (or detailed) chemical reaction mechanisms where the entire chemical reaction spectrum is solved. An example of this is the GRI MECH 3.0, which is comprised of 53 species and 325 chemical reactions. These are not viable approaches in industrial CFD problems such as engines or furnaces due to the prohibitive amount of computational work in turbulent combustion flow simulations (Warnatz *et al.*, 2006).
2. Reduced chemical reaction mechanisms use quasi-steady-state (radical production rate and destruction rate is assumed to be equal, thus its species fraction remains constant - usually when the reaction forming the intermediate radical is slow) and partial equilibrium assumptions to filter out non-rate-limiting reactions. Even these mechanisms are not viable solutions on an industrial scale due to the same reason as for detailed mechanisms, unless the mechanisms are severely reduced.
3. Multi-step quasi-global reaction mechanisms that usually have more than two steps with empirical constants for the activation energy and the pre-exponential factor in the specific reaction rate equations (4.38) and fitted constants for the reaction orders (replaces stoichiometric values in equation (4.37)).
4. Global reaction mechanism, which is usually a single or two-step mechanism with a similar approach to multi-step quasi-global mechanisms.

As mentioned, the outcome of the present research is to showcase and prove an alternative method for solving chemical reactions in turbulent species transport simulations using the EDC model, which reduces the required computational resources. To, therefore, simplify the implementation and testing (discussed later on) of this new chemistry integration technique, a multi-step quasi-global mechanism for methane combustion was selected. The mechanism that was selected is the Westbrook and Dryer mechanism (WD1), which is comprised of

a single fuel and oxygen reaction that forms CO and H_2O . Next, the mechanism has a reversible $CO - CO_2$ reaction. The rate constants originated from studies of CH_4 and CO oxidation under fuel lean conditions in a turbulent flow reactor, and they are now widely used for hydrocarbon combustion modelling as discussed by Wang *et al.* (2012), Westbrook and Dryer (1984). The reverse reaction step for the CO_2 decomposition is used in order to reproduce proper heat of reaction and pressure dependence of the $[CO]/[CO_2]$ equilibrium. The mechanism, in addition, includes a H_2 oxidation reaction.



The table below contains the kinetic data for the reaction mechanism in (4.41).

Table 4.1: WD1 reaction mechanism kinetic data

No.	$A[\frac{1}{s}]$	β	$E_a[\frac{J}{kg}]$	Reaction order
1	5.03×10^{11}	0	2×10^8	$[CH_4]^{0.7}[O_2]^{0.8}$
2	2.24×10^{12}	0	1.703×10^8	$[CO][O_2]^{0.25}[H_2O]^{0.5}$
3	5.0×10^8	0	1.703×10^8	$[CO_2]$
4	5.69×10^{11}	0	1.464×10^8	$[H_2][O_2]^{0.5}$
5	2.41×10^{14}	0	3.979×10^8	$[H_2O]$

Applying the reaction constants of table 4.1, Wang *et al.* (2012) showed that the WD1 mechanism is able to predict, with reasonable accuracy, the major species of a methane combustion PFR problem. They compared WD1 reaction mechanism (and various other multi-step quasi-global reaction mechanisms) with results generated using the complex mechanism GRI 3.0.

4.4 Turbulence-chemistry interaction modelling

In laminar flows, mixing of the oxidiser and the fuel or radical species is driven by diffusion, whereas the mixing rate in turbulent reactive flows is dominated by the convective flow, which is a result of turbulent eddies. This convection process brought about by the eddy motion greatly accelerates the mixing process (Warnatz *et al.*, 2006). The increase in molecular transfer rates between fuel and oxidiser results in increased reaction rates (and thus increased heat release rates). The growth of these vortices is the result of competition between

the turbulence production process and the eddy destruction caused by viscous dissipation (Warnatz *et al.*, 2006). The eddy production term grows larger than the eddy dissipation term when the flow's turbulent Reynolds number ($Re_t = \frac{k^2}{\nu_e}$) exceeds a certain value. Flow is said to be fully turbulent when Re_t is above or equal to 64.

The previous sections in this chapter discussed fluid flow-, turbulence- and chemical reaction modelling. These subject areas are all important for the development of turbulence-chemistry interaction models such as the EDM and the EDC. Due to the complexity of turbulent flows and solutions of multi-step chemical mechanisms, it is impractical to perform direct calculations on an industrial scale (DNS or LES) without hugely expensive computer resources available, because these models must be solved unsteady and require extremely fine meshes, according to Magnussen (1981). The flow and combustion processes are therefore modelled using RANS. The simulation can be solved for steady-state conditions that reduce the solving time of the simulation. RANS simulations do not require very fine meshes because the sub scale processes are modelled using turbulence and turbulence-chemistry interaction models, rather than having to refine the mesh to resolve the turbulent sub scales completely for DNS or to a certain scale level for LES. There are four main approaches to model turbulent-chemistry interaction for RANS simulations. The drawbacks of each of the models were discussed in the literature review section; below is a little bit more detail of each modelling approach.

1. **Species transport approach**, where the temporal and spatial evolution of the species is modelled using multiple conservation equations for each of species, mass, momentum and energy along with a turbulence model. Multiple simultaneous chemical reactions can be modelled, with reactions occurring in the bulk phase (volumetric reactions). Examples of turbulence-chemistry interaction models for the present approach are: (1) Laminar finite rate chemistry; (2) eddy-dissipation model; (3) eddy-dissipation finite-rate hybrid model; and (4) the eddy-dissipation concept model. The scope of the current research is limited to the species transport approach.
2. **Non-premixed combustion approach**, wherein the fuel and oxidiser enter the computational space in two separate streams. The model involves the solution of transport equations for one or two conserved scalars (mixture fraction and its variance). Equations for the individual species are not solved. The species mass fractions are derived from the predicted mixture fraction distribution. The thermochemistry calculations are pre-processed and then tabulated for look-up. Interaction of the turbulence and chemistry is accounted for with an assumed-shape probability density function (PDF) (ANSYS, 2016). Once mixed, the chemistry can be solved using one of the following approaches: (1) chemical equilibrium

analysis; (2) steady diffusion flamelet; unsteady diffusion flamelet; and flamelet generated manifold.

3. **Premixed combustion approach**, the oxidiser and fuel are molecularly mixed before combustion. The combustion thus takes place as a flame front propagating into the mixed fuel and oxidiser fluid mixture. The computational domain can split into un-burnt and burnt parts (ANSYS, 2016).
4. **Composition PDF Transport**, as discussed in section 4.2 in the species transport approach, the species and energy equations are Reynolds (or Favre) averaged, which creates the unknown turbulent scalar fluxes in the respective equations. The turbulent scalar fluxes are closed using the classical gradient approach, thus treating turbulent convection as enhanced diffusion. The mean reaction rates for complex chemical mechanisms are determined using the laminar finite rate approach, EDM or EDC models. An alternative approach to Reynolds-averaging, the species and energy equations, is to derive a transport equation for their single-point, joint probability function (PDF). This PDF, denoted by P , can be considered to represent the fraction of the time that the fluid spends at each species, temperature and pressure state. P has a very high dimensionality ($N + 2$ where N is the amount of species), and from P any single-point thermo-chemical moment can be calculated (ANSYS, 2016), (Pope, 1985).

For the remainder of this section, the EDM-FR hybrid model will be discussed along with its shortcomings and the EDC models inner working will be described in detail.

4.4.1 Eddy-dissipation finite-rate hybrid model

The EDM-FR hybrid model relates the mixing rate of combustion to the rate of dissipation of eddies and expresses the rate of reaction by the mean concentration of the reacting species, turbulent kinetic energy and the rate of dissipation of energy (Magnussen, 1981). The chemical reaction rate is determined using equation (4.36). The mixing rate and the chemical reaction rate are then compared to each other and the limiting value selected as the modelled combustion rate.

The EDM-FR model can be used to model two types of combustion flows, namely non-premixed or diffusion flames and premixed flames. For non-premixed combustion problems, fuel and oxygen find themselves in separate eddies, thus the mixing rate of combustion is the rate at which they are convected/mixed and is expressed as a function of the large eddy mixing time scale $\frac{k}{\epsilon}$ (Magnussen, 1981).

For premixed combustion problems, the oxygen and fuel species occur in the same eddies but at too low an energy state to ignite. The mixing rate of combustion is determined as the rate at which cold reactant eddies mix with the hot product bearing eddies into reaction zones (Magnussen, 1981). Ignition within CFD combustion models using the EDM is usually achieved by using a high initialisation temperature or using a high-temperature oxidiser stream.

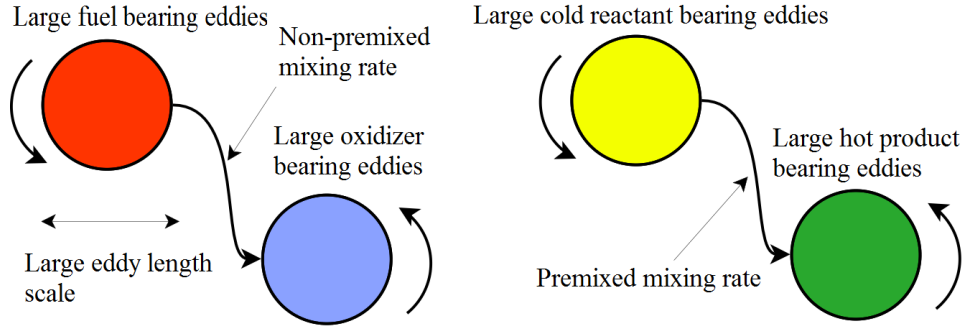


Figure 4.5: Non-premixed and premixed eddies mixing rate representation

In figure 4.5 the large eddy length scale is determined by $l = C_\mu^{3/4} \frac{k}{\epsilon} \frac{2}{\epsilon}$. The predicted mixing rate of species k for reaction j is determined using the minimum of the following two equations:

$$\dot{\omega}_{k,j} = \nu'_{k,j} M_{w,k} A_{EDM} \rho \frac{\epsilon}{k} \min_{\Re} \left(\frac{Y_{\Re}}{\nu_{\Re,j} M_{w,\Re}} \right) \quad (4.42)$$

$$\dot{\omega}_{k,j} = \nu'_{k,j} M_{w,k} A_{EDM} B \rho \frac{\epsilon}{k} \left(\frac{\sum_P Y_P}{\sum_i \nu_{i,j} M_{w,i}} \right) \quad (4.43)$$

where in equation (4.42) the subscript \Re denotes a reactant species, k the species, A_{EDM} a tuning constant and j the reaction. In equation (4.43) the subscript P denotes a product species and B the second model tuning constant. As mentioned, the EDM is used for two types of combustion; premixed and non-premixed. Equation (4.42) is used for non-premixed combustion (where the mixture of the reactants govern the mixing rate) and equation (4.43) is used for premixed combustion where the mixture of reactant and hot product species determine the rate. The A_{EDM} and B model constants were determined by Spalding as 4 and 0.5 respectively. The model accounts for chemical kinetic limited reactions by calculating the chemical reaction rate using the laminar finite-rate chemistry approach and choosing the minimum between the chemical reaction rate and the mixing reaction rate.

Scharler *et al.* (2003) used the EDM model to simulate the Sandia flame D (Barlow and Frank, 2007) (turbulent combustion flame) and found that the model constants proposed by Spalding did not yield an accurate result. They found that varying the A_{EDM} constant between 0.6 – 1.0 yielded the best

results. The motivation for their study was to use the EDM model for biomass stoker combustion simulations. The model constants' validity to a specific problem should first be investigated before use. Another shortcoming of the EDM model is the way in which it accounts for chemical kinetics. It takes the minimum reaction rate in a mechanism as the limiting chemical kinetics rate; therefore, in general, the EDM is not able to account for the effects of detailed chemical mechanisms with more than two reactions, since it does not analyse the entire reaction path (ANSYS, 2016).

The EDC model can be used to incorporate detailed chemical mechanisms with two or more reactions. The EDC differs from the EDM in the manner in which it models the turbulence-chemistry interaction. Rather than using the large eddy lifetime approach, it utilises the turbulent energy cascade to determine the mass transfer rate of reactants into the reaction zones where the mixing scale is small enough that the reactants are molecularly mixed.

4.4.2 Eddy-dissipation concept model

The use of the EDC with reduced or complex multi-step quasi-global mechanisms is the newest trend in combustion modelling and enables engineers to solve mechanisms and model the turbulence-chemistry interaction at a more fundamentally correct level (Scharler, 2013), and removes the need to correct the mixing model with empirical constants as is required with the EDM model. The EDC model does not only use the large energy-bearing eddies to model the mixing of fuel and oxidiser species as the EDM, but models the eddies for varying energy levels all the way to a level where the eddies' length and time scales are comparative to the Kolmogorov scales. At these fine scales, the assumption is made that the fuel and oxidiser eddies are molecularly mixed and react. This section delves into the inner workings of the EDC model in great detail. The theory, assumptions and modelling approaches discussed in this section form the basis on which the new EDC-ANN model was developed.

Chemical reactions in turbulent flames take place when the reactants are mixed at a molecular scale at sufficiently high temperatures. The micro-scale processes required for molecular mixing and turbulent energy dissipation are very sparse in time and space and are confined to isolated regions whose entire volume is only a small fraction of the fluid volume. These isolated regions are in turn comprised of the fine structure regions, as shown in the figure below.

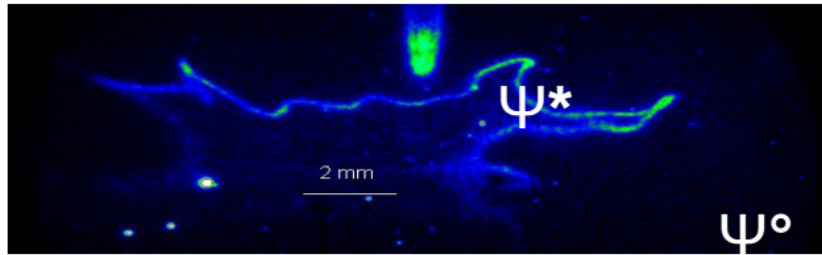


Figure 4.6: Fine structure regions in a fluid volume - Magnussen (2009)

The luminescent regions, measured by planar laser-induced fluorescence, in figure 4.6, shows the fine structure regions measuring in only a couple of millimetres. The fine structures are responsible for the dissipation of turbulent kinetic energy into heat and have the same characteristic dimensions as the Kolmogorov microscales. Within these regions, we can assume that the reactants are molecularly mixed according to Magnussen (1981), due to the small length and velocity scales. Similar characteristics of turbulence for inertial systems are observed at every level in the space-time continuum. Thus it may be assumed that the interaction between turbulence in the flow field at different scales can be modelled by a similar concept at every structural level. This assumption has formed the basis for using the energy cascade concept to model the fine structures of the EDC.

To calculate the reaction rates within these structures, it is necessary to know the fine scales volume fraction and the mass transfer rate into the regions from the surrounding fluid. These properties must therefore be calculated from known flow properties. The energy cascade model is used to relate the velocity and length scales of the fine structure regions to the turbulence properties of the bulk fluid (ν , ϵ and k).

Energy cascade model

Averaged over time, the turbulent scales are distributed over a wide spectrum of turbulent length and time scales, and the energy cascade refers to the distribution of energy through the various levels of the turbulence structure. The kinetic energy is transferred from the mean flow to the large energy-bearing eddies, and then energy is passed from the larger eddies to the smaller ones, which has higher rotational speed but contains less energy in total. These smaller eddies have much larger viscous stresses acting on them due to the high rotational speed; the viscous stresses transforms the kinetic energy of the eddies into heat (viscous dissipation). The kinetic energy dissipation occurs at all scale levels in the turbulence energy cascade, but the majority of the dissipation occurs at the smaller scales.

To model this stochastic structure of various levels of turbulence, Onsager suggested a stepwise cascade model for the turbulence spectrum (energy distribution over different frequencies). Each step is represented by a frequency that was twice the value of the preceding level's value (Ertesvag and Magnussen,

1999). The inputs to the cascade model are the turbulent kinetic energy k and the turbulent dissipation rate ϵ from the mean flow transport equations (4.27) and (4.28).

Figure 4.7 displays the model for the transfer of kinetic energy from the mean flow schematically, through the turbulence energy levels, to heat. Where $w' = u'/L'$ is the kinetic energy input to the 1st level of the cascade model and $u' = [\frac{2k}{3}]^{0.5}$ is the turbulence velocity and L' is the turbulence length scale. For RANS flow w' is the turbulence production term for kinetic energy. The sum of $q' + q'' + \dots + q^*$ is the dissipation of turbulent kinetic energy into heat. Thus, as mentioned, the first layer is the level at which one will find the large energy-bearing eddies. The first level can be used to represent the entire energy spectrum because it contains all the energy of the subsequent levels. The second level represents the part of the spectrum where the characteristic frequency is $w'' = 2w'$, the velocity is u'' and length L'' . In the same way, it was assumed that the first level contains all the energy of the subsequent levels; the second level contains all the energy of its smaller levels. In the smallest eddies (w^* , u^* and L^*) the length, velocity and time scales are of the same order of magnitude as the Kolmogorov scales. The Kolmogorov length, velocity and time scales are defined by (Ertesvag and Magnussen, 1999):

$$\begin{aligned}\eta &= \left(\frac{\nu^3}{\epsilon}\right)^{0.25} \\ u_K &= (\nu\epsilon)^{0.25} \\ t_K &= \left(\frac{\nu}{\epsilon}\right)^{0.5}\end{aligned}\tag{4.44}$$

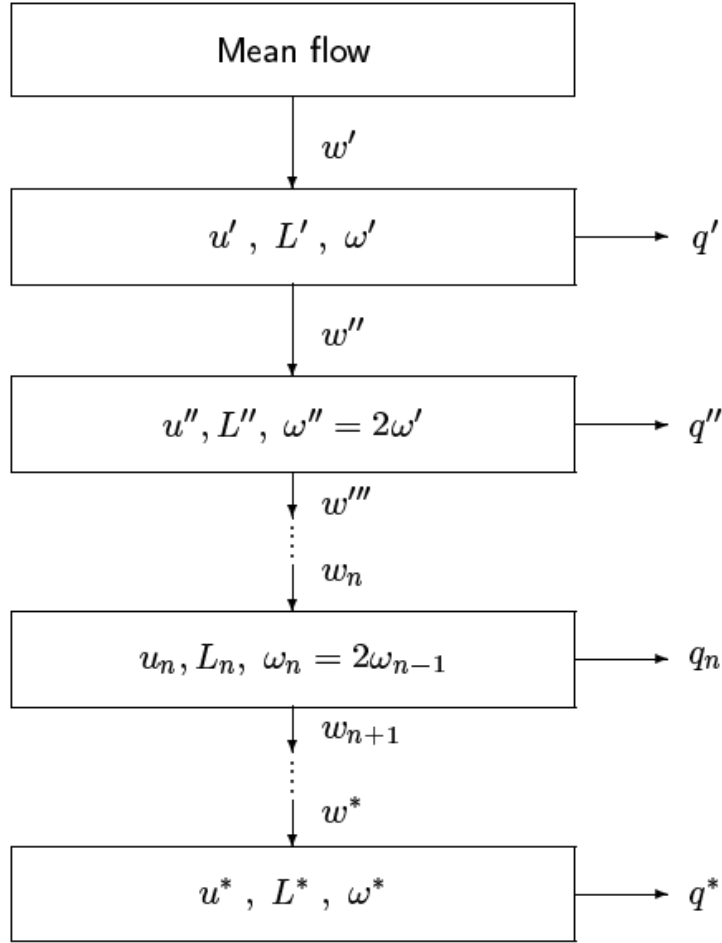


Figure 4.7: Energy cascade model for the transfer of mechanical energy from the mean flow, through turbulence energy to heat Ertesvag and Magnussen (1999)

The transfer of the energy from the first level to the second level, w'' , is equal to the sum of the dissipation from all subsequent levels, thus the total dissipation is:

$$\epsilon = q' + w'' \quad (4.45)$$

Next, the derivation of these two terms will be discussed. The production that feeds the kinetic energy to the top level of the cascade model, is a product of the turbulent stress and the mean-flow strain rate:

$$w_n = \frac{3}{2} C_{D1} u_n u_n w_n \quad (4.46)$$

Thus the kinetic energy transferred to the second level is:

$$w' = \frac{3}{2} C_{D1} u''^2 w'' \quad (4.47)$$

As mentioned, the lower level is assumed to have twice the wave number of the previous layer; equation (4.47) can be written as:

$$w'' = \frac{3}{2}C_{D1}u''^2 2w' \quad (4.48)$$

For moderate to high Reynolds number flows ($Re_t \leq 64$), the dissipation energy transfer is very small in the upper levels of the cascade model (Ertesvag and Magnussen, 1999), thus can be neglected and the following assumed:

$$w_n \approx w_{n+1} \quad (4.49)$$

The assumption of moderate to high turbulence results in that the EDC model can only be used for turbulent flows and will produce high reaction rates in low turbulence zones (this will be further discussed later in the present research). Using equation (4.46), the kinetic energy production of the first two levels can be equated:

$$\frac{3}{2}C_{D1}w'u'^2 = \frac{3}{2}C_{D1}w''u''^2 \quad (4.50)$$

Thus by inserting $w'' = 2w'$ and rearranging the equation above yields:

$$u''^2 = \frac{1}{2}u'^2 \quad (4.51)$$

Finally, inserting equation (4.51) into equation (4.48) yields:

$$w'' = \frac{3}{2}C_{D1}u'^2w' = \frac{3}{2}C_{D1}\frac{u'^3}{L'} = C_{D1}w'k \quad (4.52)$$

The turbulent viscosity can be written as $\nu_t = u'L'$; therefore equation (4.52) can be expressed as follows as a function of the turbulent viscosity:

$$L' = \frac{3}{2}C_{D1}\frac{u'^3}{\epsilon} \quad (4.53)$$

The equation above can be inserted into the turbulent viscosity definition to give:

$$\nu_t = \frac{3}{2}C_{D1}\frac{u'^4}{\epsilon} = \frac{2}{3}C_{D1}\frac{k^2}{\epsilon} \quad (4.54)$$

In equation (4.54) $\frac{2}{3}C_{D1}$ corresponds to the constant C_μ , which is a model constant in the turbulence model and set as 0.09. Therefore, $C_{D1} = 0.135$. The energy dissipation from the first level can be modelled using:

$$q' = C_{D2}\nu w'^2 = C_{D2}\nu\frac{u'^2}{L'} \quad (4.55)$$

where in equation (4.55) $C_{D2} = 0.5$. Since the model is only an approximation, numerical values for the constants were fitted to experimental data of

different types of flows by Magnussen (2005). Therefore, equation (4.45) can be approximated by:

$$\epsilon = \frac{3}{2}C_{D1}\frac{u'^3}{L'} + C_{D2}\nu\frac{u'^2}{L'} = 0.2025\frac{u'^3}{L'} + 0.5\nu\frac{u'^2}{L'} \quad (4.56)$$

The EDC model assumes no viscous dissipation occurs at the highest level of the cascade model and, due to the high viscous stresses of the smallest scales, it is assumed that the smallest scales dissipates 75% of the total dissipation. Thus it can be shown that the fine scales heat dissipation is equal to:

$$\frac{3}{4}\epsilon \rightarrow \epsilon = 0.67\nu\left(\frac{u^*}{L^*}\right)^2 \quad (4.57)$$

To ensure an energy balance in the smallest scales, the incoming kinetic energy must equal the energy being dissipated into heat, thus:

$$\epsilon = \frac{3}{4}w^* = 0.27\frac{u^*3}{L^*} \quad (4.58)$$

Therefore, using (4.57) and (4.58), the characteristic length and velocity scale for the fine structure regions can be formulated as:

$$u^* = 1.74(\nu\epsilon)^{0.25} \quad L^* = 1.43\frac{\nu^{0.75}}{\epsilon^{0.25}} \quad (4.59)$$

Now that the fine structure length and velocity scales have been formulated as a function of the mean flow quantities, the volume fraction of the fine scales and the mass transfer rate into these regions can be derived.

Fine structures

The fluid volume in the EDC model is divided into two sections, namely: (1) the fine scales fraction (where the reactions take place) defined by $\gamma^* = \gamma^3$; and (2) the surrounding fluid volume fraction (inert volume) defined by $\gamma^0 = (1 - \gamma^3)$ (Shiehnejadhesar *et al.*, 2014). The expression for the length fraction of the fine structures γ is derived by dividing the fine structure length scale by the mean fluid turbulence length scale ($\frac{L^*}{L'}$). The turbulence length scale is determined by $\frac{u'^3}{\epsilon}$ and the fine structure length scale for the derivation is $\frac{\nu^{1/2}u^*}{\epsilon^{0.5}}$ (De *et al.*, 2011). The length fraction γ of the fine scales is modelled as:

$$\gamma = C_\gamma\left(\frac{\nu\epsilon}{k^2}\right)^{0.25} \quad (4.60)$$

where in equation (4.60) the model constant is defined as $C_\gamma = \left(\frac{3C_{D2}}{4C_{D1}}\right)^{0.25} = 2.1377$. The transfer rate of the surrounding fluid into the fine structure regions can be defined as the inverse of the fine structure time scale τ^* , and in turn the fine structure time scale can be defined as:

$$\tau^* = C_\tau\left(\frac{\nu}{\epsilon}\right)^{0.25} = \frac{1}{\dot{m}^*} \quad (4.61)$$

where in equation (4.61) the model constant is defined as $C_\tau = (\frac{C_{D2}}{3})^{0.5} = 0.4082$. Figure 4.8 shows a graphic representation of the fine structure regions and the transfer rate into these minuscule volumes from the higher (or larger) cascade model levels (modelled as the surrounding fluid). In figure 4.3 the control volume is split into the fine structure regions and surrounding fluid based on the positions of the fine scale eddies in figure 4.8. In figure 4.3 the variable Y_k^* represents the mass fraction of species k after it reacted in the fine structures over time τ^* . The variable γ^* is the fine structure volume fraction of the fine structure region and \bar{Y}_k is the mass fraction of species k averaged over the control volume.

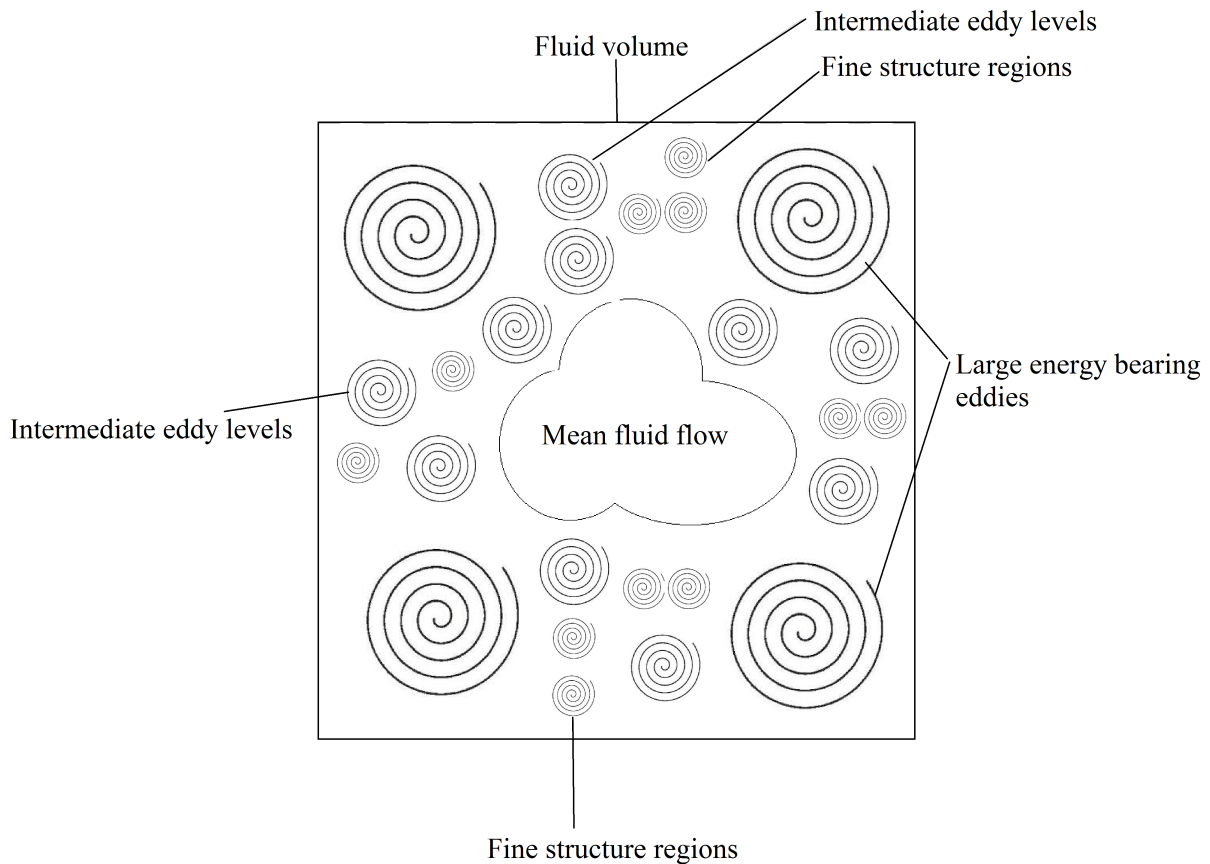


Figure 4.8: Graphic representation of the energy cascade model, with larger eddies feeding the smaller ones.

As seen in figure 4.3 and mentioned previously, the fine structure reactors are sparse in the domain. As a simplification, all the fine scale regions in a control volume can be lumped together and their volume fractions summed ($\gamma^* = \sum_N \gamma_i^*$ where N is the amount of fine structure regions in the fluid volume and i designates the specific region), to yield a volumetric averaged fine structure region per cell in the CFD computational domain. The fine

structures' volume and transfer rate are defined as functions of the mean flow quantities, and the next and final step is to determine the reaction rates of the chemical reactions in the fine structure regions.

Reaction rate

The reaction rate is expressed as

$$\dot{\omega}_k = \frac{\bar{\rho}\gamma^2}{\tau^*(1-\gamma^3)}(Y_k^* - \bar{Y}_k) \quad (4.62)$$

The reaction rate equation (4.62) is then inserted into equations (4.23) and (4.16) to model the species and energy temporal change due to chemical reactions. In equation (4.62) the fine structure species mass fraction, Y_k^* is calculated using the PFR model equations for species and energy. The initial conditions for the fine structure reactor per global iteration are the species mass fractions in the cell. The differential equation (4.63) is then integrated for the duration the species spend in the fine structures, which is the fine structure region time scale τ^* . The term \bar{Y}_k is the cell averaged species mass fraction of species k and is determined as a function of the fine scale and surrounding fluid mass fractions as $\bar{Y}_k = \gamma^2 Y_k^* + (1 - \gamma^2) Y_k^0$. In the original formulation of the EDC, Magnussen stated that the fine scales should be resolved using the WSR approach, but Jessee *et al.* (1993) showed the PFR approach could be used to simplify the model's solution and increase numerical stability. The PFR and the WSR fine structure chemistry modelling methods yield similar results for conditions of slight exothermicity and small conversion rates; otherwise, the two approaches do not correspond to each other, according to De *et al.* (2011). Fluent[®] 17.0 uses the PFR approach to model the fine structure regions as reactors. The PFR species equation for the fine structure regions is defined by:

$$\frac{dY_k^*}{d\tau^*} = \frac{\dot{\omega}_k}{\rho} \quad (4.63)$$

In equation (4.63) the reaction rate term $\dot{\omega}_k$ is determined using equation (4.36).

This concludes the discussion of turbulent combustion modelling theory. In the next sections of the theory, the focus is shifted to a special branch of artificial intelligence namely artificial neural networks. The theory governing this modelling technique will be discussed.

4.5 Artificial neural networks

Neural networks is a technique utilised in computer- and cognitive science and machine learning to approximate complex functions with a high dimensional feature and response space. Artificial neural networks are inspired by the biological neural networks of animals; in particular, the neural networks in the brain. The human brain consists of approximately 10^{11} connected neurons where each neuron can have in order of 10^4 connections (Hagan *et al.*, 1996). Each neuron is comprised out of four parts, namely: axon, dendrites, cell body, and nucleus, as shown in the figure below.

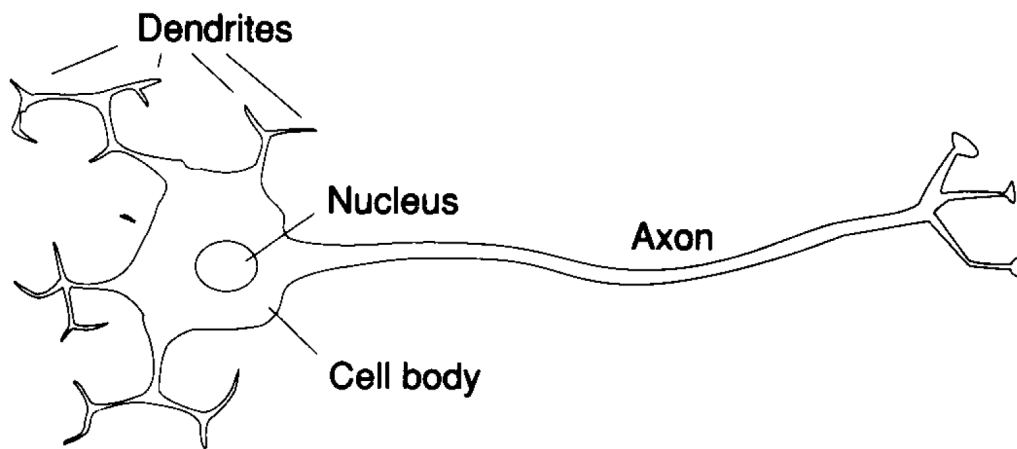


Figure 4.9: Biological representation of a single neuron - Winston (1993)

The axon is an obstruction that delivers the output signal to other neurons connected to the axon, through their dendrites. The dendrites provide a large surface area that is used by the neuron to connect to other axons. The output signal from a neuron's cell body is only generated when the collective influence of all the input signals (delivered by the axons of other neurons to the current neuron's dendrites) reaches a specific threshold level (Winston, 1993). Axons influence dendrites over narrow spaces between them called synapses, and the synapses determine the strength of the incoming signal. Stimulus from certain synapses encourages a cell body to generate an output signal where others inhibit the generation of the outputs signal.

Artificial neural networks are simplified mathematical approximations of their biological counterparts. The traits of neurons discussed above are modelled using basic mathematical building blocks such as multiplication, summation and function insertion. However, by combining multitudes of artificial neurons, very complex models capable of approximating extremely complex relationships can be created. Figure 4.10 shows a schematic of a single artificial neuron.

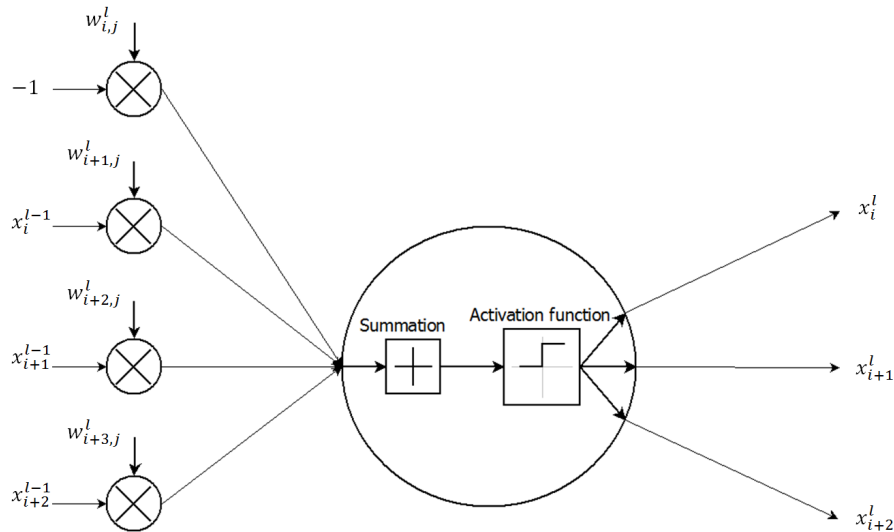


Figure 4.10: Schematic of a single artificial neuron

To model the behaviour of synapses, which influences the incoming signal (x_i^{l-1} in figure 4.10) strength, the input signal to a neuron is multiplied by a weighting variable ($w_{i,j}^l$ in figure 4.10). A large positive weight corresponds to a strong excitation, and a slight negative weight corresponds to a strong inhibition (Winston, 1993). As seen in the figure 4.10, one of the input signals is fixed to a value of -1 ; this is called the neuron bias. Using a bias is a trick that enables us to treat activation functions as though they were weights, eliminating the need for adjusting threshold functions along with the weights (Winston, 1993). Similar to biological neurons the weighted signals are summed together before being passed onto the threshold or activation function generating the output signal. There are numerous activation functions that can be used with neural networks [see Hagan *et al.* (1996)]; the ones that are utilised in the present research are the linear, hyperbolic tangent sigmoid and log-sigmoid functions. These functions are used because they are differentiable, which simplifies the implementation of the learning algorithm. Figure 4.11 shows the three different activation functions' output signals plotted as a function of the summed input signals (or cumulative input signal). The figure shows that the log-sigmoid and hyperbolic tangent sigmoid functions asymptote to values of $0, 1$ and $-1, 1$ respectively, whereas the linear function is free of restrictions.

Neural networks, as mentioned, are multiple neurons interconnected with each other, where the central idea is to extract linear combinations of the inputs as derived features, and then model the target function as a non-linear function of the features space (Hastie *et al.*, 2009). Neural networks are comprised of multiple layers of multiple neurons. The first layer of the network is called the input layer, the last layer is the output layer and the intermediate layers are the hidden layers. Figure 4.12 shows a schematic of a network with 2 hidden

layers where P_1, P_2 and P_3 designates the number of neurons in the hidden layers and the output layer respectively.

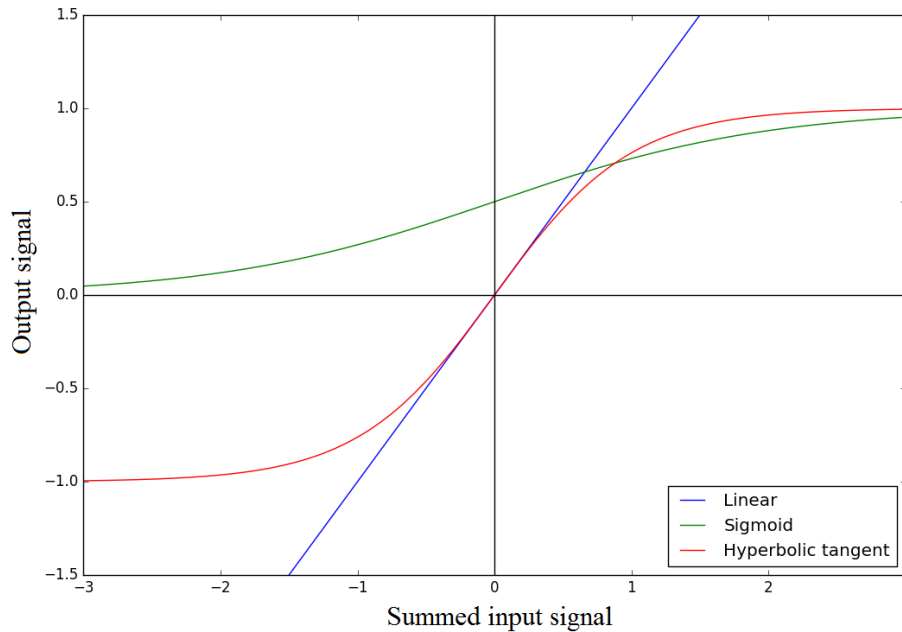


Figure 4.11: Activation functions used in artificial neural networks

This type of neural network is called a multi-layer perceptron neural network. The designation "Weights 0–1" corresponds to all the weights multiplied to all the input signals to neuron 1 in layer 0. The individual weight terms are not shown in the figure, but each signal line connecting an input node to a neuron is multiplied by a weight. In the present research, the entire neural network model was developed from the ground up. This was done to develop a full understanding of the inner workings of the model and enabled the author flexibility in modelling the training phases with various activation functions at different layers in the network. There are two modes of operation in neural networks, namely feedforward propagation and backward propagation. The former is used to calculate the network's output signal from the feature space variables for a single dataset observation; the latter is a technique utilised to adjust weights to accurately map the input features to the corresponding response features for the entire dataset, also known as training/learning the relationship between the input- and output variables. It must be stated that there are numerous types of neural networks and different methods to train datasets. The two operations discussed in this section are the methods used in the present research.

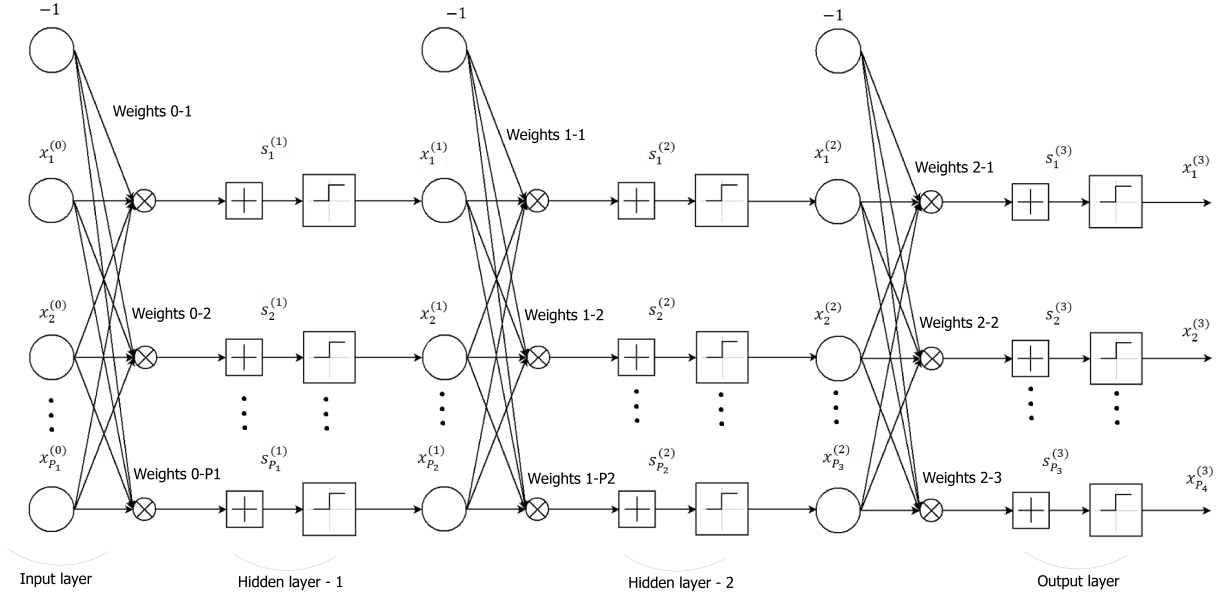


Figure 4.12: Schematic of a multi-layer regression neural network

4.5.1 Feedforward propagation

For multi-layer perceptron networks, the inputs propagate through the network. The neural network is parameterised by $w_{ij}^{(l)}$ with:

$$\begin{aligned} 1 &\leq l \leq L \\ 0 &\leq i \leq d^{(l-1)} \\ 1 &\leq j \leq d^{(l)} \end{aligned} \quad (4.64)$$

where d is the amount of neurons in layer l . Figure 4.10 will be used to explain feedforward propagation. The cumulative signal for the neuron is determined by:

$$s_1^l = -1 \times w_{i,j^l} + x_i^{l-1} \times w_{i+1,j}^l + x_{i+1}^{l-1} \times w_{i+2,j}^l + x_{i+2}^{l-1} \times w_{i+2,j}^l \quad (4.65)$$

where in equation (4.65) the variable x_i^{l-1} refers either to the input signal from the previous layer in the network, or the actual input layer values for the dataset observation. The general formulation for the cumulative signal is:

$$s_j^{(l)} = \sum_{i=0}^{d^{(l-1)}} w_{i,j}^{(l)} x_i^{(l-1)} \quad (4.66)$$

The weight variable subscripts i, j correspond to the synapse between the i^{th} neuron in the $(l-1)^{th}$ layer and the j^{th} neuron in the l^{th} layer. The next step

is to pass the cumulative signal into the selected activation function.

$$\begin{aligned}\theta(s_j^{(l)}) &= s_j^{(l)} && \text{Linear function} \\ \theta(s_j^{(l)}) &= \frac{1}{1 + e^{-s_j^{(l)}}} && \text{Log-sigmoid function} \\ \theta(s_j^{(l)}) &= \frac{e^{s_j^{(l)}} - e^{-s_j^{(l)}}}{e^{s_j^{(l)}} + e^{-s_j^{(l)}}} = \tanh(s_j^{(l)}) && \text{Hyperbolic-tangent sigmoid function}\end{aligned}$$

The variable $\theta(s_j^{(l)})$ is the output signal from the j^{th} neuron in the l^{th} layer. From here on the process above is repeated for each neuron in the l^{th} layer before moving on to the next layer or output layer, depending on the current position in the network. For the last layer, the output signal is the network approximation for observation n in dataset size N . The network output signal is written as $x_i^{(L)} = h(\mathbf{x})$ where \mathbf{x} is the input dataset vector. To develop more efficient code, the feedforward propagation equations were vectorised, which reduces runtime of the program.

$$\mathbf{x}^{(l)} = \theta(\mathbf{W}^{(l)} \mathbf{x}^{(l-1)}) \quad (4.67)$$

where the weights matrix is defined for $1 \leq l \leq L$ by:

$$\mathbf{W}^{(l)} = \begin{bmatrix} w_{0,1}^{(l)} & w_{1,1}^{(l)} & \cdots & w_{d^{(l-1)},1}^{(l)} \\ w_{0,2}^{(l)} & w_{1,2}^{(l)} & \cdots & w_{d^{(l-1)},2}^{(l)} \\ \vdots & \vdots & \vdots & \vdots \\ w_{0,d^{(l)}}^{(l)} & w_{1,d^{(l)}}^{(l)} & \cdots & w_{d^{(l-1)},d^{(l)}}^{(l)} \end{bmatrix} \quad (4.68)$$

Input and output signals are vectorised as:

$$\mathbf{x}^{(l)} \quad \text{or} \quad \mathbf{x}^{(l-1)} = \begin{bmatrix} 1 \\ x_1^{(l)} \\ x_2^{(l)} \\ \vdots \\ x_{d^{(l)}}^{(l)} \end{bmatrix} \quad (4.69)$$

We have an input \mathbf{X} and output \mathbf{Y} dataset with N observations and the dimensionality of the input/output observations are d_{in} and d_{out} respectively. The input data is thus a vector of size N : $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_N$; similarly for the output data: $\mathbf{Y} = \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \dots, \mathbf{y}_N$. Each input and output observation is a vector of length d_{in} and d_{out} respectively. A performance function is used to gauge how well the neural network is predicting the outputs from the input dataset. The mean-squared error is a popular choice for these regression-type problems where the stochastic gradient ascent algorithm is used. The

in-sample error for the n^{th} observation is calculated as:

$$e(\mathbf{w}) = -\frac{1}{2} \sum_{j=1}^{d_{out}} (y_j - x_j^{(L)})^2 \quad (4.70)$$

Equation (4.70) shows that the in-sample error is a function of the weights, therefore the weights need to be optimised to minimise the in-sample error, thus maximising equation (4.70). This process is called the learning procedure of the neural network.

4.5.2 Backward propagation

The purpose of the learning procedure is to train the network to perform some task by modifying the weights and biases. There are many types of neural network learning methods that fall into three broad categories: supervised learning, unsupervised learning and reinforcement learning (Hagan *et al.*, 1996). The category used in the present research is supervised learning. In supervised learning, the network is provided with training sets of the desired network behaviour. The weights and biases are optimised using back-propagation- and stochastic gradient ascent algorithms. Due to the high dimensionality of the feature and response variable space, the neural network model requires the necessary flexibility to capture the highly non-linear relationships between the input and output sets. The amount of fitting variables (weights and biases) thus increases with model flexibility, creating a high dimensional plane of which the peak, corresponding to the lowest error, must be found. Gradient ascent is a method in which the weights are adjusted in the direction of the steepest upward gradient, which maximises equation (4.70). The weights are adjusted using the following equation:

$$w_{i,j}^{(l),k+1} \leftarrow w_{i,j}^{(l),k} + \eta \nabla e(\mathbf{w}) \quad (4.71)$$

where in equation (4.71), k is the iteration counter, η is the learning rate parameter which relaxes the solution (ensures that overshoot is inhibited), and the gradient term is defined as:

$$\nabla e(\mathbf{w}) = \frac{\partial e(\mathbf{w})}{\partial w_{i,j}^{(l)}} \quad \text{for all } i, j, l \quad (4.72)$$

The back-propagation procedure is utilised to formulate the expression for the $\partial e(\mathbf{w})/\partial w_{i,j}^{(l)}$ term in equation (4.72); it is thus used to express the relationship between a specific weight in the network and the performance function. In back-propagation the idea is to make a large change to a particular weight if the change leads to a large reduction in the observed error (Winston, 1993). The procedure is called back-propagation because it computes changes to the weights in the final layer first, reuses the same computation to compute changes

to the weights in the penultimate layer, and ultimately proceeds to the weights at the first hidden layer.

$$\frac{\partial e(\mathbf{w})}{\partial w_{i,j}^{(l)}} = \delta_j^{(l)} \times x_i^{(l-1)} \quad (4.73)$$

where $\delta_j^{(l)} = \partial e(\mathbf{w}) / \partial s_j^{(l)}$ and varies between network layers. Assuming the activation function is the hyperbolic-tangent sigmoid function, the $\delta_j^{(L)}$ is expressed as:

$$\delta_j^{(L)} = 2(\tanh(s_j^{(L)}) - y_j)(1 - \tanh^2(s_j^{(L)})) \quad (4.74)$$

For any other layer in the neural network:

$$\delta_i^{(l-1)} = [1 - (x_i^{(l-1)})^2] \sum_{j=1}^{d^{(l)}} w_{i,j}^{(l)} \delta_j^{(l)} \quad (4.75)$$

The derivation of equations (4.73), (4.74) and (4.75) are shown in Winston (1993) and Hagan *et al.* (1996). Similar to the feedforward propagation, the backward propagation equations are vectorised for efficient computation.

$$\bar{\delta}^{(l-1)} = \mathbf{U}^{(l-1)} \cdot (\Gamma^{(l)} \bar{\delta}^{(l)}) \quad (4.76)$$

where in equation (4.76) the following variables are defined as:

$$\bar{\delta}^{(l)} = \begin{bmatrix} \delta_1^{(l)} \\ \delta_2^{(l)} \\ \delta_3^{(l)} \\ \vdots \\ \delta_{d^{(l)}}^{(l)} \end{bmatrix} \quad \mathbf{U}^{(l-1)} = \begin{bmatrix} 1 - (x_1^{(l)})^2 \\ 1 - (x_2^{(l)})^2 \\ 1 - (x_2^{(l)})^2 \\ \vdots \\ 1 - (x_{d^{(l)}}^{(l)})^2 \end{bmatrix}$$

$$\Gamma^{(l)} = \begin{bmatrix} w_{1,1}^{(l)} & w_{1,2}^{(l)} & \cdots & w_{1,d^{(l-1)}}^{(l)} \\ w_{2,1}^{(l)} & w_{2,2}^{(l)} & \cdots & w_{2,d^{(l-1)}}^{(l)} \\ \vdots & \vdots & \vdots & \vdots \\ w_{d^{(l-1),1}}^{(l)} & w_{d^{(l-1),2}}^{(l)} & \cdots & w_{d^{(l-1),d^{(l)}}}^{(l)} \end{bmatrix}$$

The backward-propagation algorithm works as follows:

1. Initialise all weights (in all layers) $w_{i,j}^{(l)}$ as random variables
2. For iterations $t = 1, 2, \dots, T_{specified}$, where t is the iteration counter and $T_{specified}$ is the user specified amount of iterations the backward-propagation algorithm will run
 - a) Pick a random observation out of the dataset (\mathbf{x}, \mathbf{y})

- b) Perform forward propagation calculation through network, equation (4.67)
- c) Perform backward-propagation calculation through network, calculating all the $\delta_j^{(l)}$ variables, use equation (4.76)
- d) Perform stochastic gradient ascent calculation which updates the weights,

$$w_{i,j}^{(l),k+1} \leftarrow w_{i,j}^{(l),k} + \eta \nabla e(\mathbf{w})$$
- e) Determine the error for all the final layer output signals compared to the data in the \mathbf{y} vector, equation (4.70)
- f) Plot error every 1000 iterations to keep visual track of the error convergence.
- g) If not converged, repeat steps: $a \rightarrow f$

3. Return final converged weight's matrix, $\mathbf{W}^{(l)}$

It must be mentioned that there two main variations to the gradient ascent algorithm, namely batch gradient ascent and stochastic gradient ascent. The former loops through the entire dataset one by one and calculates the mean-squared-error based on all the dataset observations' predictions, whereas the latter picks random samples and then passes it into the network and the error is only calculated between the current observation's prediction signal and the actual dataset response variables. Stochastic gradient ascent has a few benefits that make it a more efficient option over the batch variation: (1) cheaper computation; (2) randomisation helps the algorithm escape local minima; and (3) it is simpler to implement.

A modification that can be made to the current backward-propagation algorithm that decreases the chance of getting stuck in a local minima in the high dimensional solution space of the weights, is to use the momentum modification on the stochastic gradient ascent formulation. Adding momentum, changes equation (4.71) as follows (Hagan *et al.*, 1996):

$$w_{i,j}^{(l),k+1} \leftarrow w_{i,j}^{(l),k} - \gamma \Delta w_{i,j,(t-1)}^{(l)} + (1 - \gamma) \eta \delta_j^{(l)} x_i^{(l-1)} \quad (4.77)$$

where in the equation above the term $\Delta w_{i,j,(t-1)}^{(l)}$ is the change of the weights from the previous iteration ($t - 1$) to the current iteration t . In addition to momentum modification, regularisation is added to the stochastic gradient ascent equation. Seeing as the training datasets used in the present research is generated from solving the actual reactor species and energy equations for various initial species compositions, there is no noise in the data and thus no need for regularisation.

Chapter 5

Experimental setup CFD modelling using traditional methods

To validate the combustion model using the ANN chemistry integrator against the experimental data, a well published experimental setup was selected, namely the Sandia piloted CH_4/Air jet flame (Barlow and Frank, 2007). Four piloted flames (C, D, E and F) were tested by the group. These flames have increasing velocity in the main jet and pilot, therefore increasing turbulent conditions. The flame selected for comparison is flame D, which is a fully turbulent flame with a Reynolds number of 22400. The reason for selecting the fully turbulent flame is that the EDC model is only valid for fully turbulent combustion conditions. The Sandia experimentally measured scalars included temperature, mixture fraction, N_2 , O_2 , H_2O , H_2 , CH_4 , CO , CO_2 , OH and CO mass fractions. SANDIA National Laboratories used laser induced fluorescence, Raman scattering and laser-Doppler anemometry to measure these quantities. These measurement techniques are very expensive to perform and therefore were not practical for the author to perform validation. The dataset includes axial and radial profiles of Reynolds-, Favre-average mass fractions and RMS fluctuations. The scalar quantities for temperature, mixture fraction and N_2 , O_2 , H_2 , CH_4 , CO_2 , H_2O and CO (OH not compared, because the chemical mechanism selected does not contain it) of the experimental data will be compared to the solutions generated by the CFD models; one using Fluent[®] 17.0's EDC formulation and the other using the novel EDC-ANN chemistry integrator. In this section the experimental setup will be modelled using ANSYS Fluent[®] 17.0 software package and its standard combustion modelling procedures (DI and ISAT). This section is comprised of the following sub-sections: (1) modelling geometry; (2) boundary conditions; (3) flow solution configuration; (4) mesh independence; and (5) results, post-processing and discussion.

5.1 Geometry

The burner is located in a wind tunnel, where the walls of the wind tunnel are far enough from the flame so that the wall effects are negligible. The burner consists of a small round jet feeding the fuel stream, an annulus around the main jet feeding the pilot gas and air being fed outside the annulus. The geometry was modelled as a Two-dimensional axisymmetric domain to simplify the model geometry and reduce mesh size. Figure 5.1 below shows the computational domain (upward flowing jet flame showed in figure 4.1 turned horizontally) and table 5.1 contains the geometry dimensions.

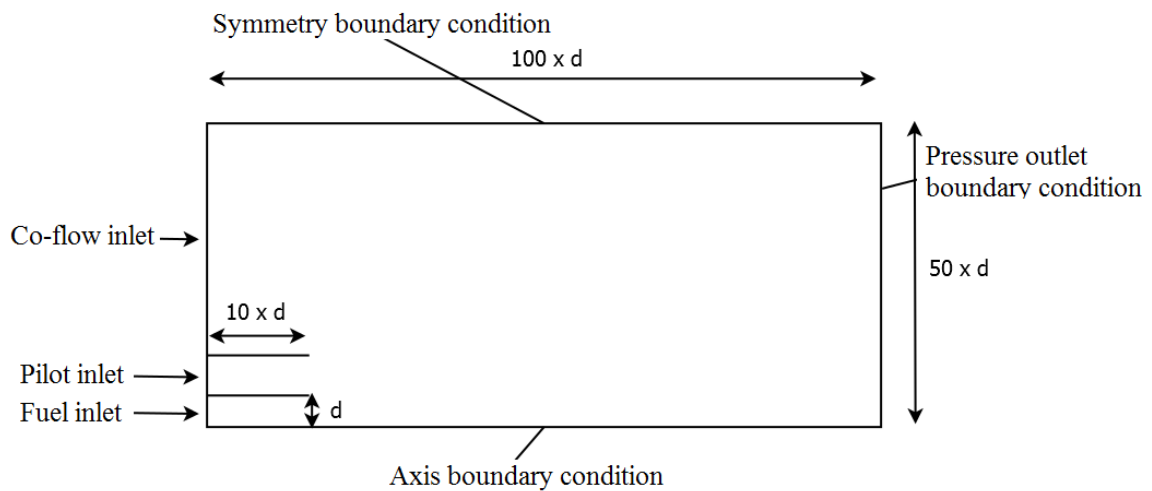


Figure 5.1: Two-dimensional axisymmetric computational domain used to model the piloted jet flame

Table 5.1: Burner dimensions

Main jet inner diameter, d	7.2 mm
Pilot annulus inner diameter	7.7 mm (wall thickness = 0.25mm)
Pilot annulus outer diameter	18.2 mm
Burner outer wall diameter	18.9 mm (wall thickness = 0.35mm)

5.2 Boundary conditions

The boundary conditions applied to the geometry are contained in the table 5.2. The jet fluid is a mixture of three parts air and one part CH_4 by volume. This was done to limit soot production, which reduces fluorescence

interference, which in turn is used to measure the CO content at various locations in the flame. Partial premixing with air also shortens the flame length and produces a more robust flame than pure methane injection or a nitrogen-diluted methane injection. This also reduces local extinction probability at high Reynolds numbers. The jet and pilot nozzle walls are assumed to be perfectly insulated and the heat flux set to zero. The axis boundary condition shown in figure 5.1 corresponds to the centreline of the jet nozzle (and therefore the computational domain). Seeing as the flame is located inside a large controlled area where the surrounding atmosphere is stationary, the computational domain's outer boundary can be selected as a symmetry-type boundary condition. The symmetry condition has zero normal velocity and zero normal gradients of all fluid variables. From the results seen later on in the present section the boundary has no effect on the flame.

Table 5.2: Boundary conditions

Boundary condition name	Boundary condition type	Assigned values
Co-flow inlet	Velocity inlet	Velocity magnitude = 0.9 m/s Temperature = 300 K $Y_{O_2} = 0.233, Y_{N_2} = 0.767$
Fuel inlet	Velocity inlet	Velocity magnitude = 49.6 m/s Temperature = 300 K $Y_{O_2} = 0.19664, Y_{N_2} = 0.6473$ $Y_{CH_4} = 0.15607$
Pilot inlet	Velocity inlet	Velocity magnitude = 11.4 m/s Temperature = 1880 K $Y_{O_2} = 0.056, Y_{N_2} = 0.742$ $Y_{CO_2} = 0.11, Y_{H_2O} = 0.092$
Jet wall	Wall	Heat flux = $0 W/m^2$
Pilot wall	Wall	Heat flux = $0 W/m^2$
Outlet	Pressure outlet	Backflow temperature = 300 K Backflow $Y_{O_2} = 0.233$

5.3 Flow simulation setup

In this section the principles covered in the theory section is applied to the numerical model. The reader is encouraged to see the theory section for the motivation behind the modelling methodologies.

The turbulence was modelled using the realizable $k - \epsilon$ model with standard model constants, see ANSYS (2016) for model constants. The model was also solved using the RSM and the $k - \omega$ SST models to determine the effect of chosen turbulence model. The near wall effects was modelled using the standard wall functions (ANSYS, 2016). Additionally, the compressibility and viscous heating effects were taken into account. The species transport model was used to simulate the flow and interaction between the various species in the domain. Reactions were assumed to be restricted to the fluid domain (only volumetric reactions enabled) and the EDC turbulence-chemistry interaction model used to calculate the individual species reaction rates. The chemistry was updated after each iteration and the aggressiveness factor (relaxation term for combustion) set to 0.5. The chemistry integration was performed using the ISAT and DI methods. The mixture species and reaction mechanism used is shown in table 4.1, the density of the mixture was resolved using the ideal gas law, and the thermal conductivity was calculated using the polynomials specified in the VDI Heat Atlas (Kleiber and Joh, 2010). The viscosity was calculated using kinetic theory of gasses. The individual species specific heat, molecular weight, standard state enthalpy and standard state entropy were imported from the GRI 3.0 mechanism's thermodynamic properties database, which uses the NASA polynomials. The pressure-based solver was utilised and the pressure-velocity coupling was resolved using the SIMPLE algorithm. The computational domain was discretised as follows: gradients by least squared cell based, pressure using the standard scheme and density, momentum, k , ϵ , species and energy were all discretised using the second order upwind technique.

5.4 Mesh independence investigation

To ensure the fluid quantities such as temperature, species mass fractions and mixture fraction are independent of the mesh resolution, the mesh counts were incrementally increased. The centreline vertex average CO , H_2 , CO_2 , O_2 , CH_4 , H_2O fractions and temperature was recorded. These quantities for each mesh refinement stage was compared to the previous mesh's quantities, to determine if the simulation is mesh independent. Table 5.3 shows the percentage change for each refinement stage. The table shows that the changes from the original mesh size of 2352 to 5823 is relatively small, except for Y_{H_2} , and that the change from a mesh of 5823 cells to 10617 cells is even smaller, showing mesh independence. The average hydrogen mass fraction for

the centreline vertex is roughly $2.8E - 4$; in reality this is an extremely small amount of gas. Taking into account that turbulent combustion is inherently transient due to species, temperature and velocity fluctuations (as seen in the results later on in the present section), and hydrogen is such a small amount, the error made by choosing a mesh size of 2352 is acceptable. To show that the model is grid independent for the mesh size of 2352; Richardson extrapolation is performed on the data for the first refinement stage. In table 5.4 the following variables are: discretisation error on coarse mesh $E_{U,1}$ and the discretisation error on the fine mesh $E_{U,2}$. Further p is the discretisation order which is 2 for second-order upwind discretisation and r is the refinement ratio which is 2.48.

Table 5.3: Mesh refinement data

Mesh size	2352	5823	10617
T	-	0.75%	0.27%
Y_{CH_4}	-	3.64%	0.41%
Y_{CO}	-	3.19%	0.01%
Y_{CO_2}	-	1.13%	0.47%
Y_{H_2}	-	7.28%	6.69%
Y_{H_2O}	-	1.25%	0.1%
Y_{O_2}	-	0.36%	0.14%

Table 5.4: Richardson extrapolation calculation for first refinement stage

Parameter	$E_{U,1}$	$E_{U,2}$	$E_{U,1}/E_{U,2}$	$\ln(\frac{E_{U,2}}{E_{U,1}})/\ln(r)$
T	-0.178	-1.09	0.163	2
Y_{CH_4}	-0.0013	-0.00845	0.163	2
Y_{CO}	$-3.61E - 5$	-0.00221	0.163	2
Y_{CO_2}	-0.000282	-0.00172	0.163	2
Y_{H_2}	$-3.99E - 6$	$-2.44E - 5$	0.163	2
Y_{H_2O}	$-8.21E - 5$	$-5E - 4$	0.163	2
Y_{O_2}	$-1.5E - 4$	$-9.1E - 4$	0.163	2

5.5 Results and post-processing

In this section, the results of the standard Fluent[®] 17.0 combustion model (EDC using DI/ISAT) are compared to the experimental results at different axial and radial locations. The axial locations are normalised by dividing by

the jet diameter (x/d). The normalised axial locations where measurements were taken are $x/d = 1, 2, 3, 7.5, 15, 30, 45, 60$ and 75 . Additional measurements were taken at multiple radial locations, but for the sake of simplicity, the results at the various normalised radial locations will be plotted for $x/d = 0.15$ and 0.3 . The standard deviation of the measurements was calculated along with the mean value and plotted along with the results from the CFD simulation, as seen below.

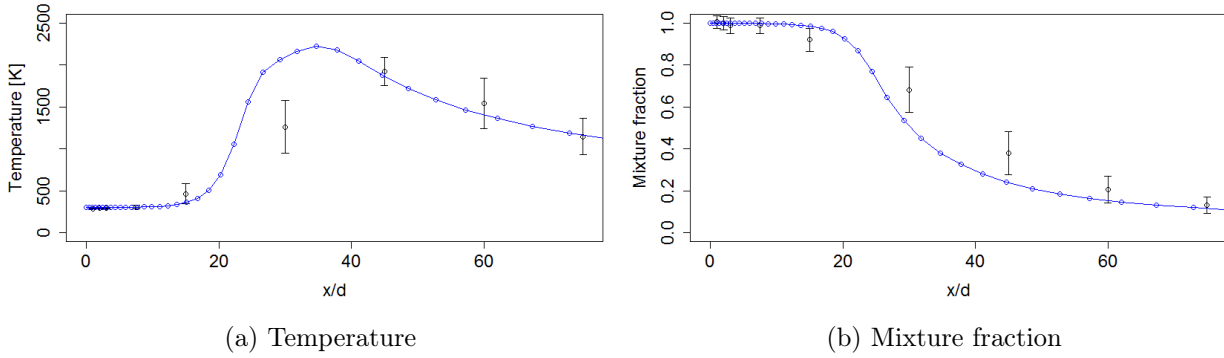


Figure 5.2: Temperature and mixture fraction distribution for the centreline normalised axial location. Blue solid line: CFD, Black dots: Experimental

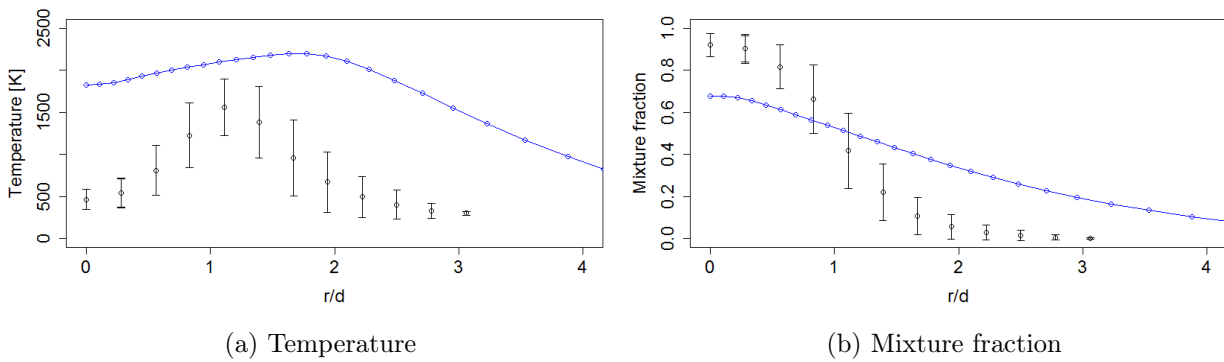


Figure 5.3: Temperature and mixture fraction distribution for normalised radial locations at $x/d = 0.15$ m. Blue solid line: CFD, Black dots: Experimental

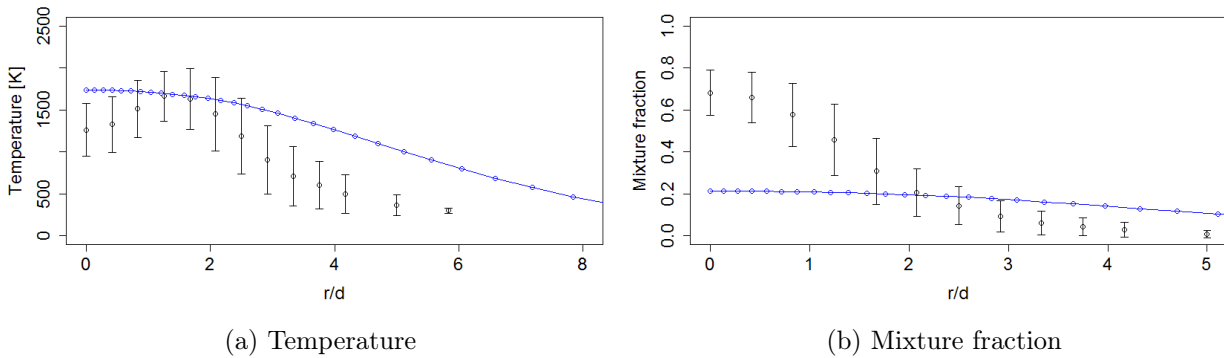


Figure 5.4: Temperature and mixture fraction distribution for normalised radial locations at $x/d = 0.30$ m. Blue solid line: CFD, Black dots: Experimental

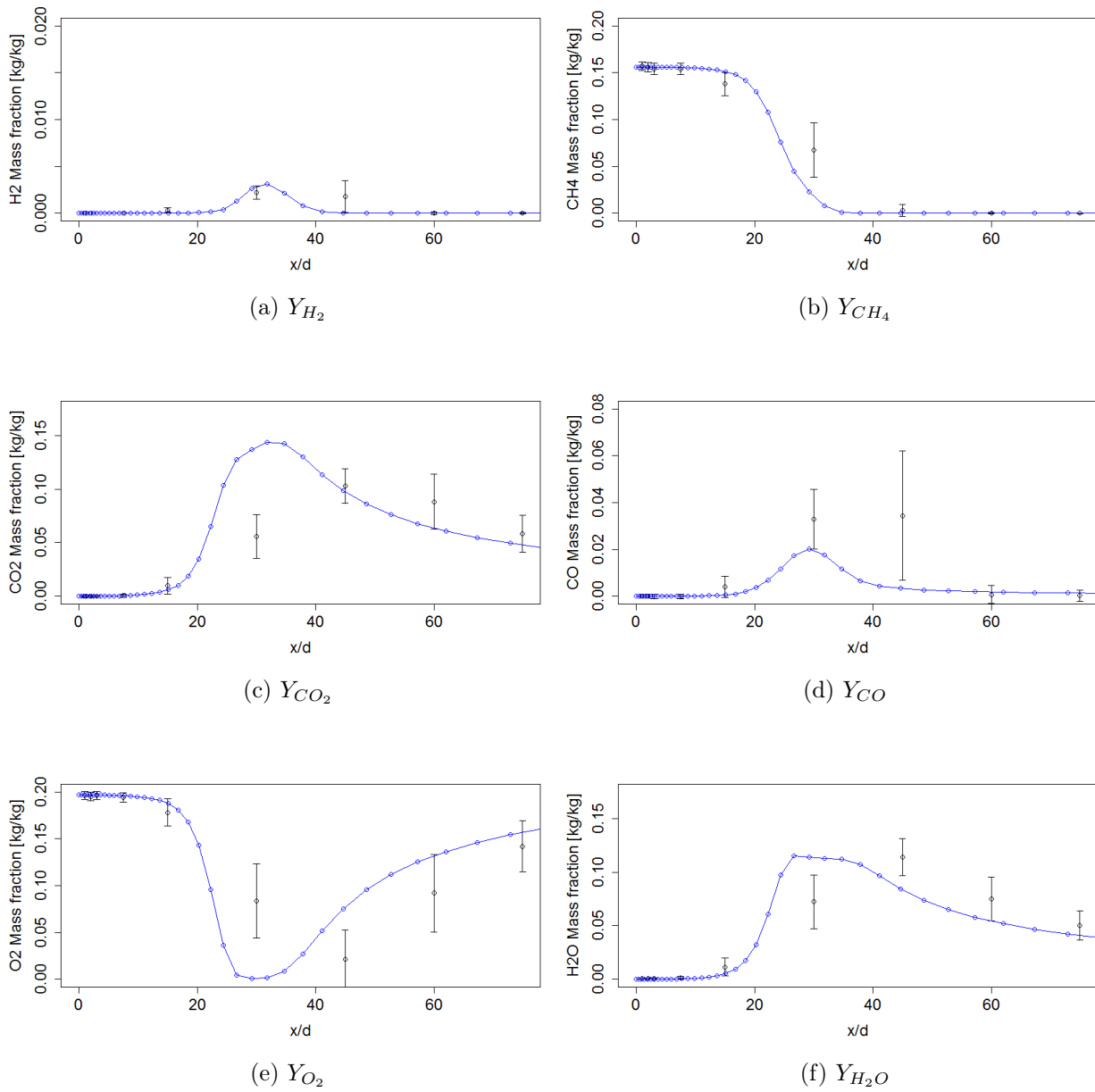


Figure 5.5: Species mass fractions for centreline normalised axial locations. Blue solid line: CFD, Black dots: Experimental

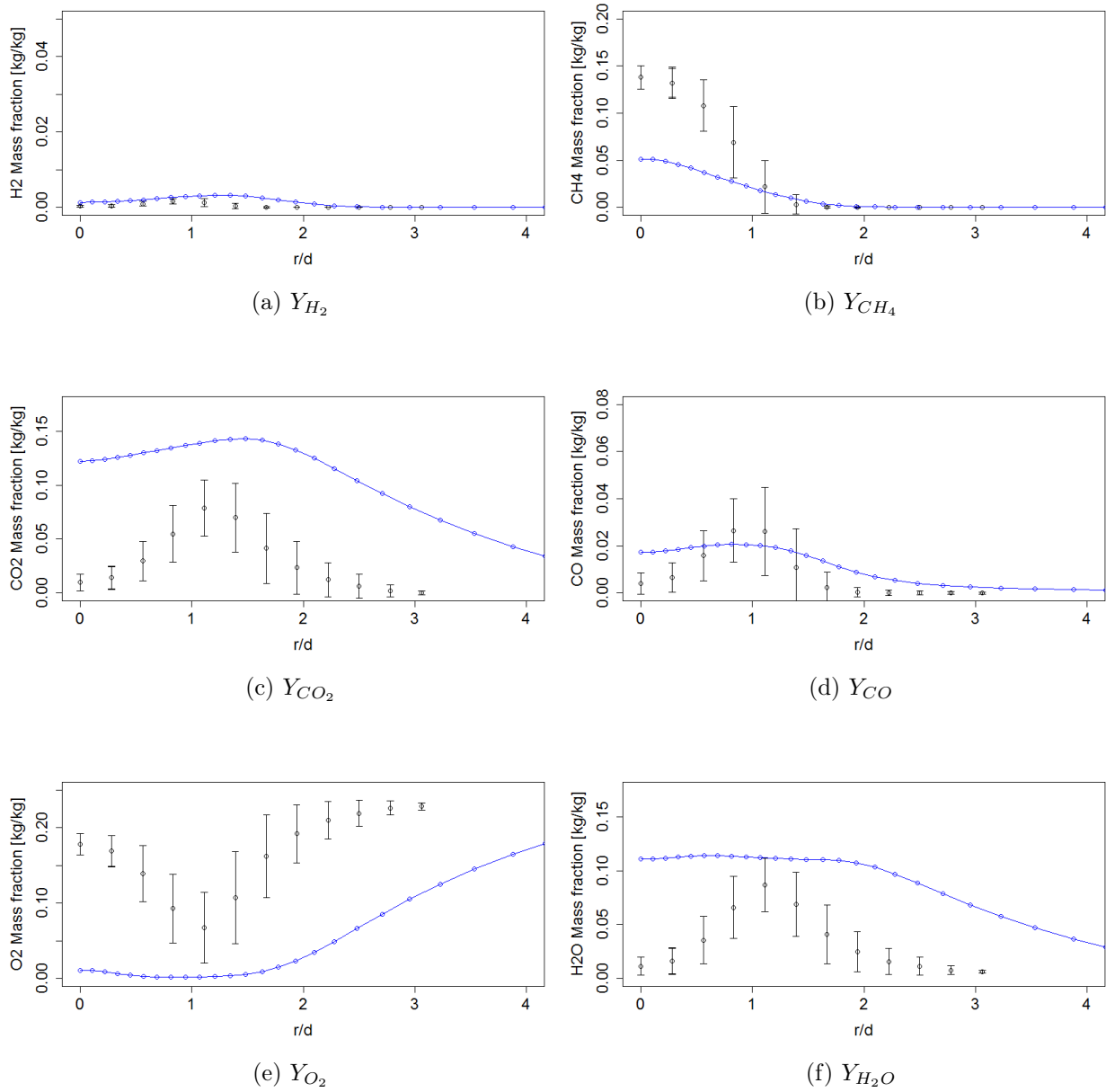


Figure 5.6: Species mass fractions for normalised radial locations at $x/d = 0.15$ m. Blue solid line: CFD, Black dots: Experimental

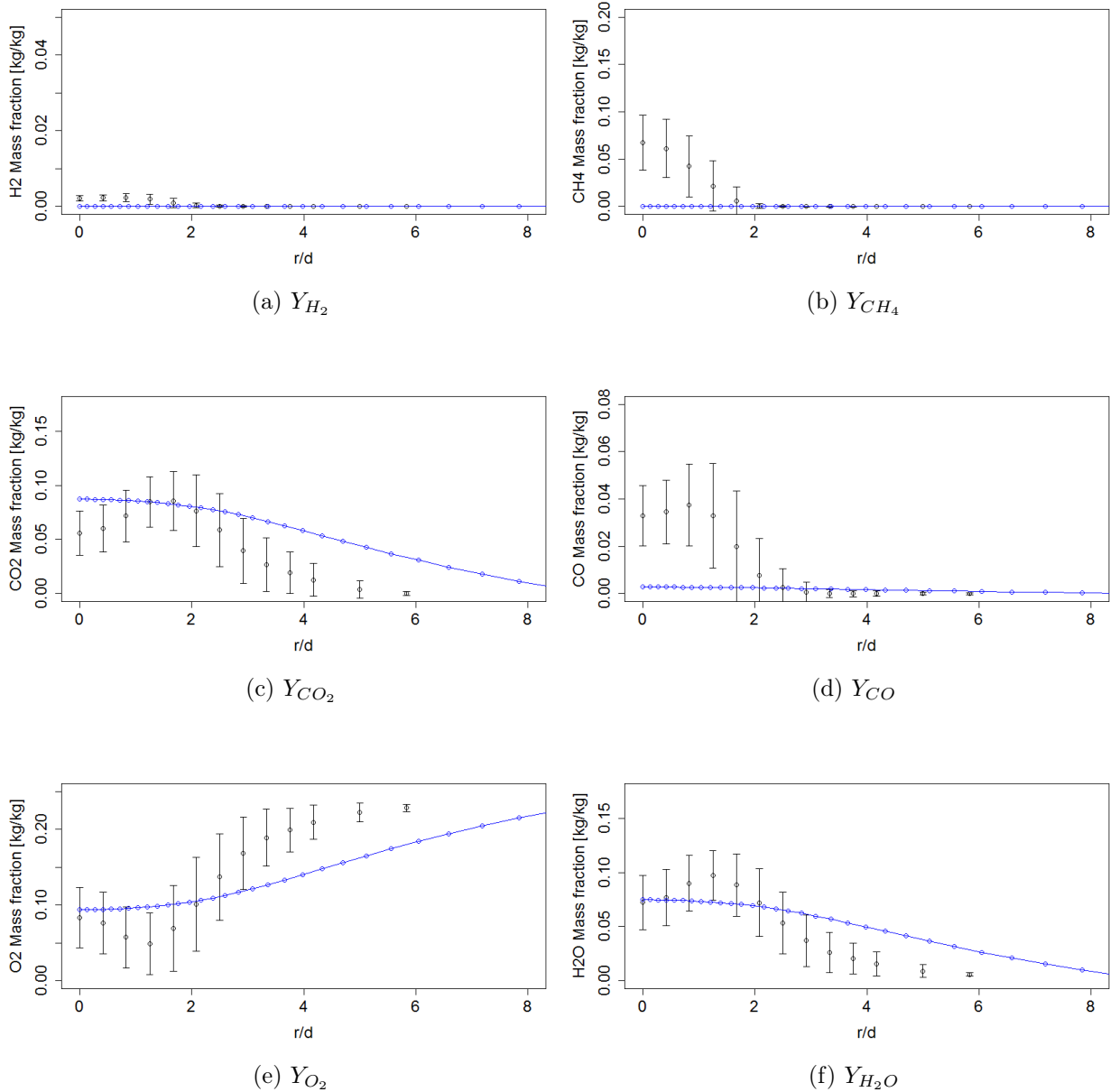


Figure 5.7: Species mass fractions for normalised radial locations at $x/d = 0.3$ m. Blue solid line: CFD, Black dots: Experimental

Figures 5.5b, 5.6b and 5.7b shows that the CFD model using the WD1 mechanism over-predicts the methane reaction rate. As the methane flows further away from the outlet of the jet nozzle, the difference between the experimental data and the simulation results becomes larger. The rapid oxidation of CH_4 leads to a temperature peak occurring closer to the jet outlet than the experimental data shows. The high temperature zone and the high

CO concentration then accelerates the combustion rate of the CO released by the CH_4 oxidation reaction, as seen in figures 5.5d, 5.6d and 5.7d. This further leads to faster formation of CO_2 compared to the experimental results. The high CH_4 and CO reaction rates are a result of either a high pre-exponential factor or low activation energy of reactions 1 and 2 in table 4.1 or a error in predicted turbulence production by the turbulence model used. To determine if the turbulence model is the culprit; the problem was simulated using the $k - \omega$ SST and RSM (quadratic pressure-strain) turbulence models to determine if the results are model independent. The results are seen in figures 5.8 and 5.9.

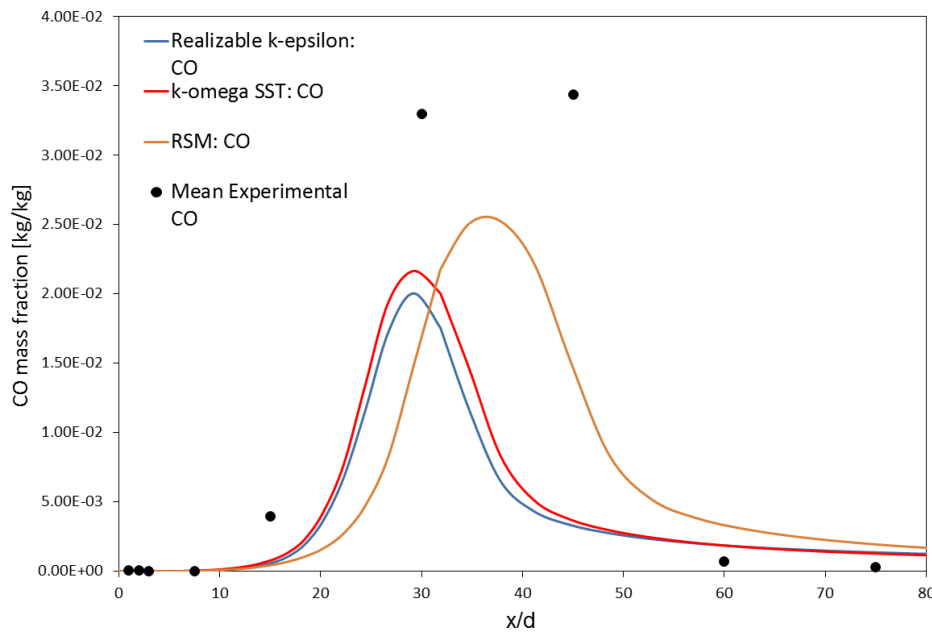


Figure 5.8: CO mass fraction results on centre line for various turbulence models

From figures 5.8 and 5.9 we see that the $k - \omega$ SST and realizable $k - \epsilon$ turbulence models produces similar results. The RSM model on the other hand predicts that the temperature peak occurs further downstream (which aligns better with the experimental data) than what the other models do, this is do to the lower (more accurate) predicted turbulence production by the more advanced RSM model. The temperature peak predicted by the RSM and the $k - \omega$ SST models are both higher than the experimental data's peak value and the CO mass fraction predicted by the various models all show that the CO is consumed too rapidly resulting in lower mass fraction peaks and in turn higher fluid temperatures. The high consumption rate of the CO is due to the chemical mechanism's empirical parameters. Two strategies to alleviate the problems with the reaction mechanism parameters are to either

change the activation energy or pre-exponential factor to fit the data more accurately, or to use a more complex mechanism. Many studies [Scharler *et al.* (2003), Yin (2011), Kempf *et al.* (2005) and Lei and Ghoniem (2014)] have shown that using complex mechanisms ensures a more accurate solution than less complex mechanisms. Laubscher (2015) compared a two-step EDM combustion model of the Sandia Flame D (Barlow and Frank, 2007) to a EDC simulation using a 41 step reduced mechanism using the realizable $k - \epsilon$ model. The results showed that the EDC model with the larger mechanism accurately predicted the temperature and species peaks on the centre line of the flame, much better than the results presented above thus enforcing the statement that more detailed mechanisms will yield more accurate results. As discussed, the purposes of the present research is to develop a chemistry integration method that reduces the computational resources of the simulation compared to the current EDC integration methods. The goal is therefore to compare the new chemistry integration method to experimental data and the traditional method's results, and benchmark its performance based on accuracy and computational resource reduction. The accuracy of the WD1 mechanism is therefore adequate for the purpose of the present work and the realizable $k - \epsilon$ model chosen as the model used for implementation of the ANN chemistry integrator with the EDC model, seeing as it is the model of choice for industrial biomass boiler simulations. The RSM turbulence model generates results with peaks better aligned with the experimental data, therefore the RSM will also be used along with the EDC-ANN combustion model for sake of thoroughness.

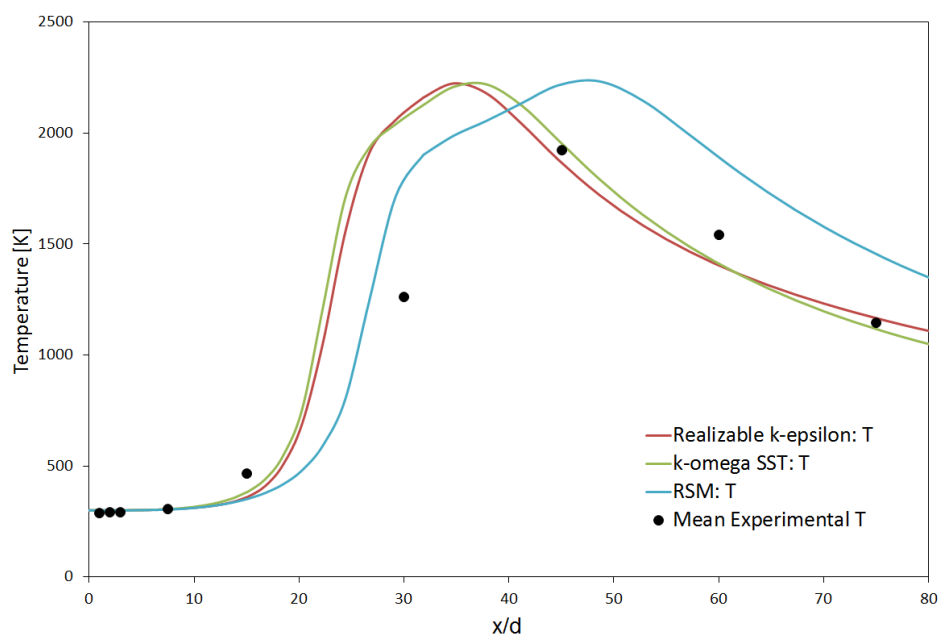


Figure 5.9: Temperature results on centre line for various turbulence models

Below is a contour plot of the temperature distribution in the computational domain. From figure 5.10 it is seen that there is a sheet of high-temperature fluid concentrated around the fuel jet inlet, which is due to the turbulent diffusion of the oxidiser and fuel towards each other causing and sustaining a thin reaction zone. Plotting the heat of reaction contour shows this thin reaction zone, seen in figure 5.11.

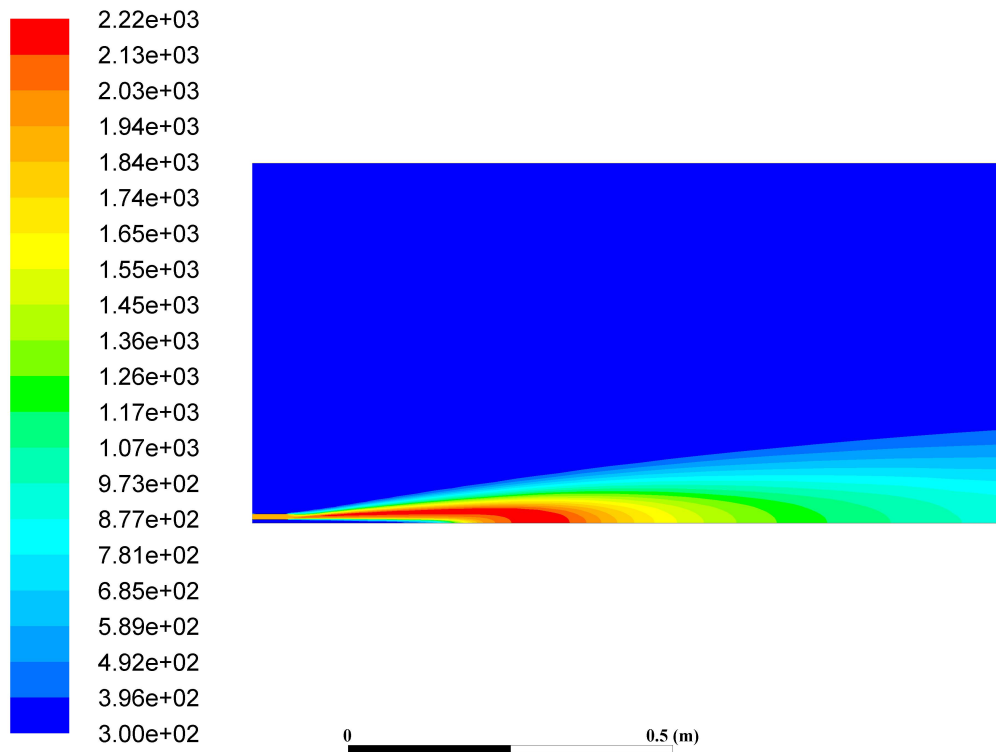


Figure 5.10: Temperature contour of the methane/air piloted jet flame solved using traditional EDC model, [K]

Figure 5.11 shows that only a fraction of the fluid volume is reacting. The reacting zone in the compositional space is therefore said to be narrow. If a random data set is to be created for training of the neural network, the unfiltered samples will bias towards steady-state situations (non-reacting compositions). The range of allowable values for each variable is determined using an upper and lower limit of the mixture fraction, and the reactions are only to occur within these bounds. The mixture fraction f represents values between $0 \rightarrow 1$ and varies throughout the computational domain depending on the amount of products, oxidiser and fuel species. The mixture fraction is set to 1 for the fuel stream and 0 for the oxidiser stream. The stoichiometric mixture fraction for the Sandia flame D experimental setup is 0.351, according to Barlow and Frank (2007) and we can assume that values far from the stoichiometric value

is non-reacting (Blasco *et al.*, 1998) because of the mixture being too fuel rich or too fuel lean.

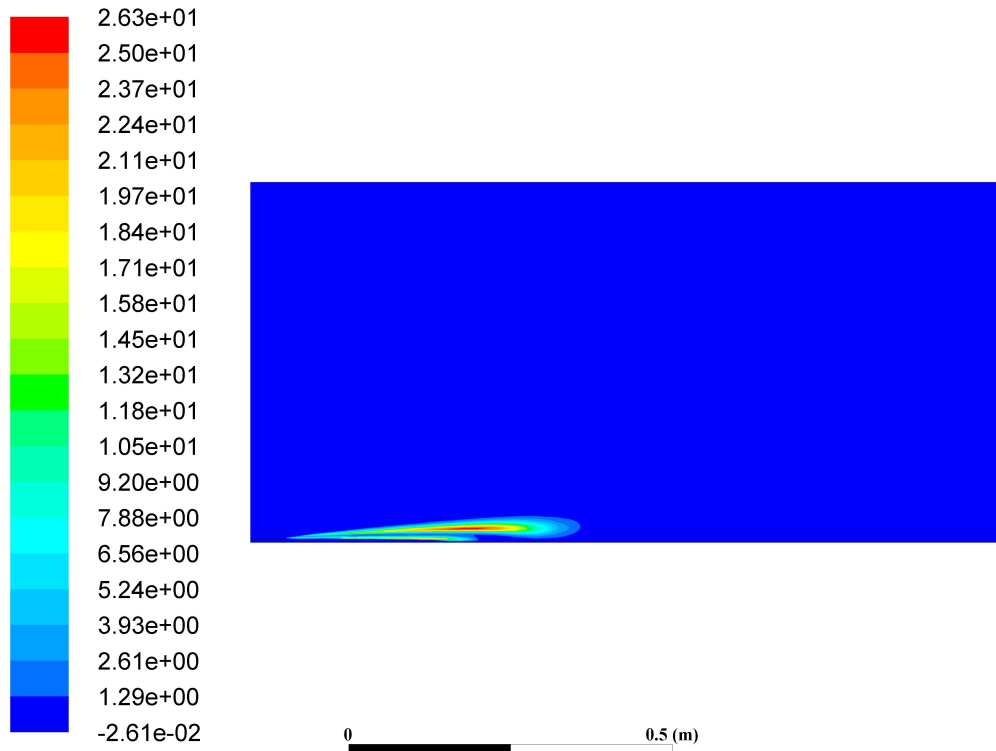


Figure 5.11: Heat of reaction contour of the methane/air piloted jet flame solved using traditional EDC model, [W]

The upper and lower bounds of the active reaction zone can be determined by utilising the mixture fraction concept and heat of reaction shown in figure 5.11. The species transport model in Fluent[®] 17.0 does not have the facility to determine the mixture fraction, therefore a user-defined function (UDF) was developed that hooked into Fluent[®] 17.0 and calculated the mixture fraction at the end of each global iteration; see Appendix C for the UDF code. The mixture fraction equation used was the Bilger equation (Versteeg and Malalasekera, 2007):

$$f = \frac{2(Z_C - Z_{C,OX})/M_C + (Z_H - Z_{H,OX})/M_H - 2(Z_O - Z_{O,OX})/W_O}{2(Z_{C,FUEL} - Z_{C,OX})/M_C + (Z_{H,FUEL} - Z_{H,OX})/M_H - 2(Z_{O,FUEL} - Z_{O,OX})/W_O} \quad (5.1)$$

In equation (5.1), Z_H , Z_C and Z_O are the elemental mass fractions of hydrogen, carbon and oxygen; and subscripts *FUEL* and *OX* designate the streams origin (jet fuel boundary or co-flow boundary). Figure 5.12 below shows the mixture fraction contour plot.

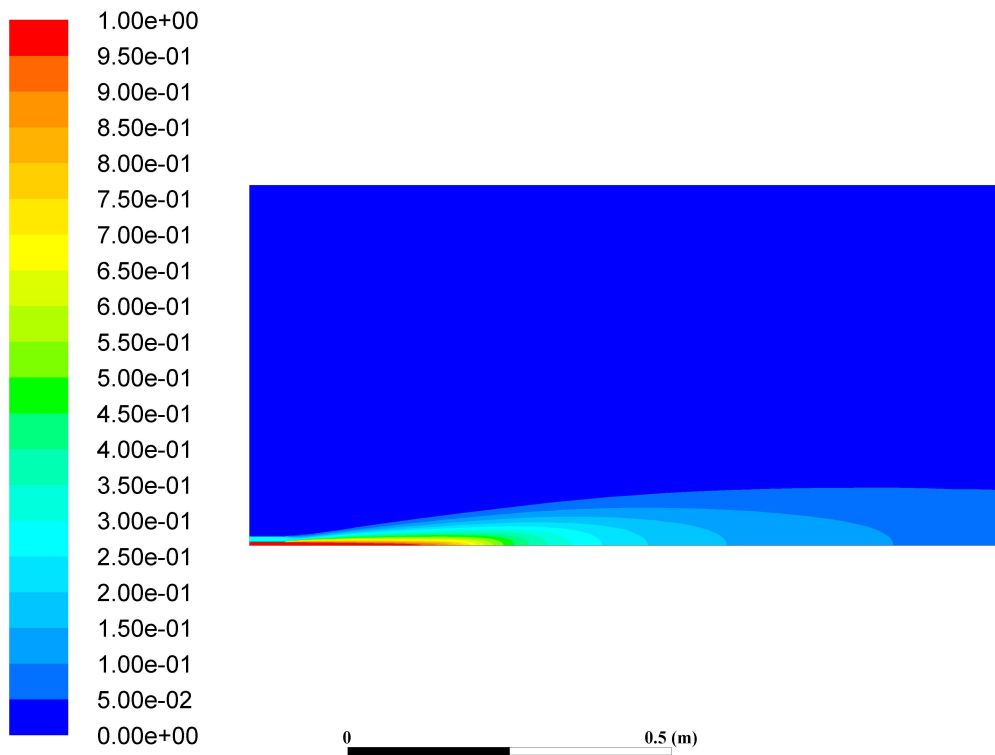


Figure 5.12: Mixture fraction contour plot

We can see that the mixture fraction is 1 at the jet fuel boundary and 0 for the co-flow boundary, as expected. The mixture fractions range found to best capture the reaction zone is $f = 0.1 \rightarrow 0.6$. This is shown in the figure 5.13, where the mixture fraction upper and lower bounds are superimposed on the heat of reactions contour plot.

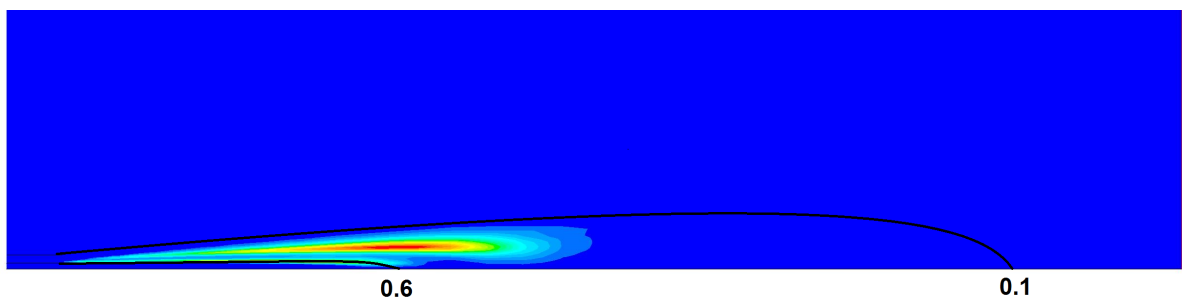


Figure 5.13: Mixture fraction upper and lower bound superimposed on the heat of reaction contour plot

In the next section the technique utilised to generate training data for the neural network will be explained.

Chapter 6

Plug-flow reactor simulation and data generation

The ANN chemistry integrator is configured to receive the current cell's species mass fractions, temperature and reaction time (τ^* , see equation (4.61)) and output the species mass fractions after reacting over time τ^* . The weights and bias values of the neural networks must be changed to minimise the in-sample error of its predictions. During the back-propagation algorithm, the routine uses a training dataset (input values and corresponding output values known) to quantify the relationship between the inputs and outputs and change the weights accordingly. This chapter, therefore, discusses the creation of the training datasets. The data generation occurs before the CFD simulation. The training data is generated by creating random samples of initial species mass fractions and temperatures and integrating the system to a steady state with a separate computer program outside Fluent[®] 17.0. The program created to generate the training data set was developed in *Python* 3.5.2, with the following imported packages: *Cantera* 2.2.1 chemical properties and kinetics database, *sciPy* ordinary differential equation solver routines, and *numPy* numerical calculation routines (matrix calculations and random number generation). This section is comprised of two parts, namely random sample generation and reaction solver.

6.1 Random sample generation

In the previous chapter, it was shown that reactions only occur in the mixture fraction range of 0.1 – 0.6, where the mixture fraction was calculated using Bilger's equation. The same equation will be used to generate random species mass fractions that could occur in the simulation. Setting the mixture fraction range will create bounds for the species mass fraction distribution in the compositional space. To ensure the random samples created falls within the compositional space of the CFD simulation, upper and lower bounds for

the mixture fraction and temperature were set to 0.1 – 0.6 and 300 – 3000 K (the temperature limit was also imposed on the CFD simulation) respectively. Next, random samples of initial mass fractions and temperatures could be created within the limits. The equation used to create a random temperature sample is:

$$T_{rand} = T_{min} + \epsilon_i(T_{max} - T_{min}) \quad (6.1)$$

where in equation (6.1) ϵ_i is a random number between 0 – 1. The T_{min} , T_{max} bounds correspond to the maximum and minimum temperature mentioned above. Next, a random mixture fraction value needs to be created, using the same approach.

$$f_{rand} = f_{min} + \epsilon_i(f_{max} - f_{min}) \quad (6.2)$$

Now that a random temperature and mixture fraction sample set is created, the accompanying random species sample is required. Using Bilger's equation, the following equations are used to generate a species composition corresponding to the random mixture fraction value f_{rand} . First, the local elemental mass fractions based on the random mixture fraction sample needs to be calculated as:

$$\begin{aligned} Z_{C,local} &= f_{rand}Z_{C,fuel} + (1 - f_{rand})Z_{C,ox} \\ Z_{O,local} &= f_{rand}Z_{O,fuel} + (1 - f_{rand})Z_{O,ox} \\ Z_{H,local} &= f_{rand}Z_{H,fuel} + (1 - f_{rand})Z_{H,ox} \end{aligned}$$

Next, the species mass fraction is written as a function of the random mixture fraction and its local elemental mass fraction.

$$\begin{aligned} Y_{CH_4,rand} &= \epsilon_i(Z_{C,local}M_{CH_4}/W_C) \\ Y_{CO,rand} &= \epsilon_i(Z_{C,local} - (W_C Y_{CH_4,rand}/M_{CH_4}) \times M_{CO}/W_C) \\ Y_{CO_2,rand} &= [Z_{C,local} - (W_C Y_{CH_4,rand}/M_{CH_4}) - (W_C Y_{CO,rand}/M_{CO})](M_{CO_2}/W_C) \\ Y_{H_2O,rand} &= \epsilon_i(Z_{H,local} - [4W_H Y_{CH_4,rand}/M_{CH_4}][M_{H_2O}/2W_H]) \\ Y_{H_2,rand} &= (Z_{H,local} - [4W_H Y_{CH_4,rand}/M_{CH_4}] - [2W_H Y_{H_2O,rand}/M_{H_2O}])(M_{H_2}/2W_H) \\ Y_{O_2,rand} &= (Z_{O,local} - [W_O Y_{CO,rand}/M_{CO}] - [2W_O Y_{CO_2,rand}/M_{CO_2}] \\ &\quad - [W_O Y_{H_2O,rand}/M_{H_2O}])(M_{O_2}/2W_O) \\ Y_{N_2,rand} &= f_{rand}Y_{N_2,fuel} + (1 - f_{rand})Y_{N_2,ox} \end{aligned}$$

Now that the initial conditions have been created, it is fed into the solver routine which calculates the temporal evolution of the species and the temperature of the reactor.

6.2 Reactor solver

The numerical solution of the thermochemical system involves five chemical reaction steps shown in table 4.1 and the seven participating species. The

random sample's species composition and temperature are set as the initial reactor condition. Thereafter the reactor is integrated through a number of intermediate time-steps to nearly steady state. Equation (4.40) is used to solve the temperature change in the reactor, and equation (4.39) is integrated for each of the seven species to calculate the change in concentrations due to chemical reactions. All the equations are discretised using the backward-differencing formula and the system of equations is solved using the VODE solver (see Appendix A). The time-step size is fixed for each reactor model, thus multiple neural networks were trained to cover the range of time scales encountered in the methane/air piloted jet flame. The reactors were solved for time-step intervals of $\Delta t = 10^{-7}, 10^{-6}, 10^{-5}$ and 10^{-4} . For each of these time-steps, 20000 random samples were generated and solved to steady state. The convergence criteria for the reactor solver was set to 1×10^{-9} , which is the same as the default value Fluent[®] 17.0 uses. From the previous chapter it is seen that the smallest quantity in the results of simulation was the mass fraction of H_2 which was roughly $2.5E - 4$, thus the convergence criteria is several orders in magnitude smaller, thus ensuring the dataset has the required accuracy to model the actual incremental species changes. The convergence was calculated using the least squares error between the current time-step and the previous one.

$$error^t = (T^t - T^{t-1})^2 + \sum_{k=1}^K (Y_k^t - Y_k^{t-1})^2 \quad (6.3)$$

where in the equation above the superscript t refers to the current time-step temperature and species mass fractions and $t - 1$ to the previous time-steps temperature and mass fractions. Once $error^t$ equals or is smaller than the set convergence criteria, the reactor model exits the solver for the current sample and then a new random sample is generated and set as the PFR's initial conditions, and the process is repeated. All data points created by the reactor solver are then used as input and output data for the neural network to train on.

It is good practice to normalise all the input variables to assist the gradient ascent algorithm in converging more rapidly. There is no need for this with the species mass fractions, seeing as they are all values between 0 – 1. The temperature, on the other hand, is three orders of magnitude larger than the species mass fractions and therefore could lead to problems with finding the optimised weights (the one large variable causes the gradient ascent solution to become stiff), thus the temperature is normalised by dividing by the maximum temperature value (3000 K). The reactor data generation procedure is, therefore:

1. Set upper and lower bounds for mixture fraction and temperature, random samples

2. Using the equations in the previous subsection, random species mass fractions are generated based on the mixture fraction sample value
3. Plug-flow reactor initial conditions set to sample fractions and temperature
4. Plug-flow reactor species and energy equations solved until convergence criteria are met
5. Temperature data points are normalised by dividing by maximum temperature
6. All species and temperature data points are saved
7. If j (amount of reactor simulations) is not equal to J (amount of user specified training scenarios), $j = j + 1$ and loop from (2) till $j = J$

Appendix D has the complete program listing for this algorithm and a sample calculation. Figure 6.1 below shows an example plot of one random training scenario solved using the PFR solver to nearly steady -state. A flowchart for the procedure listed above is seen in figure 6.2 on the next page.

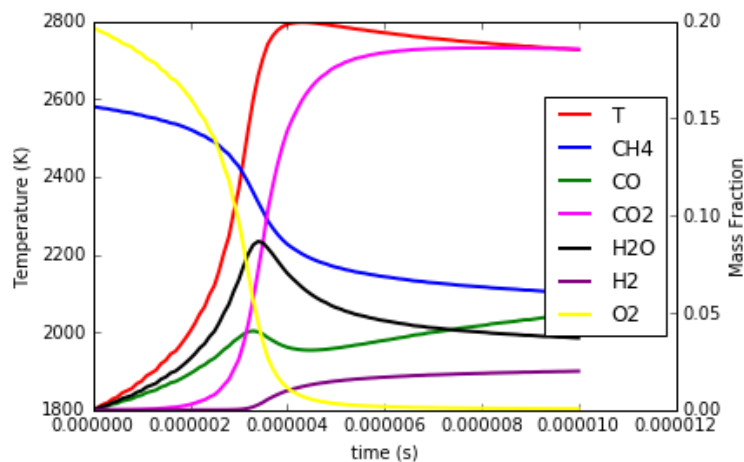


Figure 6.1: Plug-flow reactor simulation results for single scenario

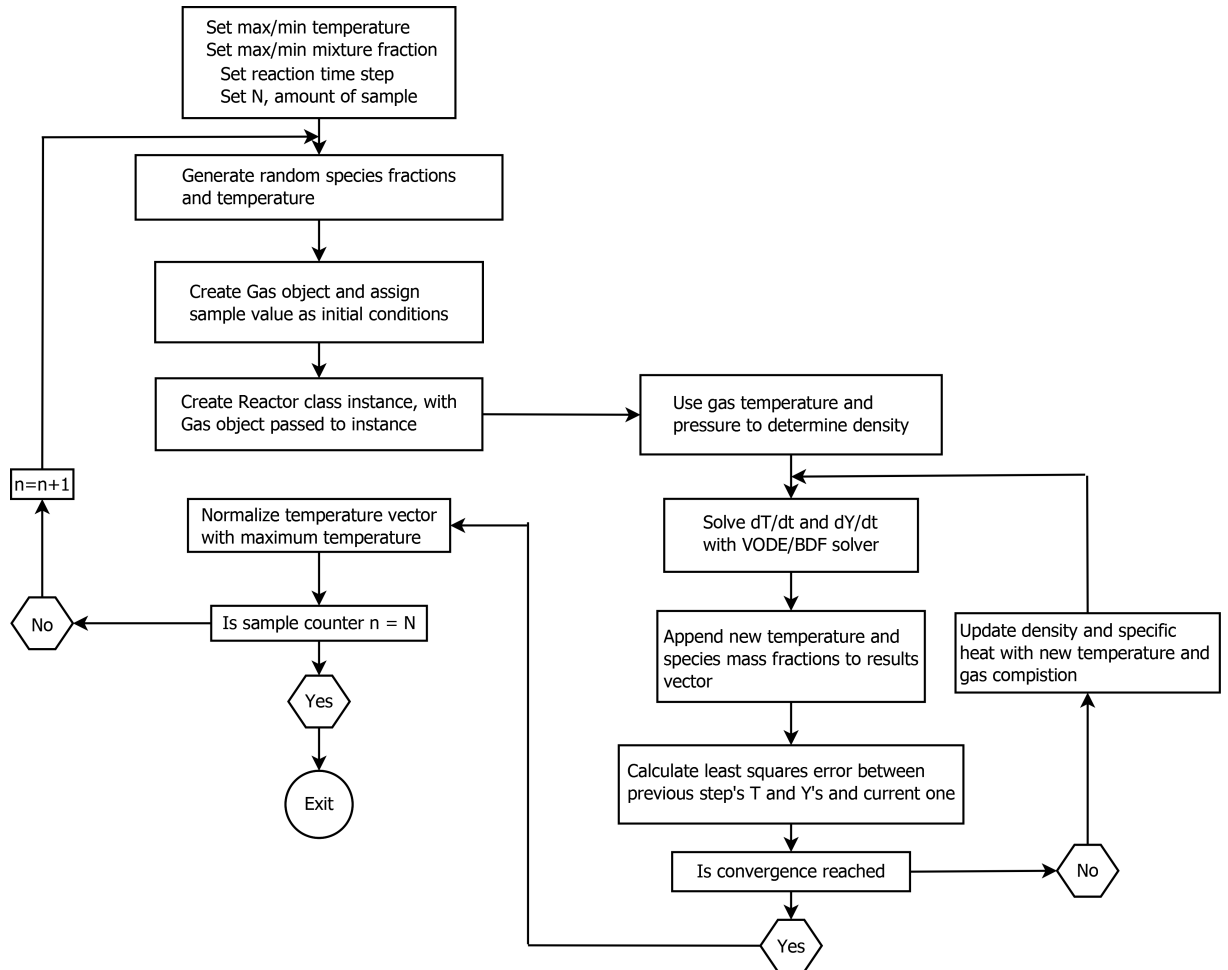


Figure 6.2: Plug-flow reactor simulation flowchart

Chapter 7

Neural network architecture selection and training

The author developed the neural network code used in the present research from the ground up, building the forward- and backward-propagation algorithms, as mentioned in the theory section. The neural network program is completely generic with the functionality of having as many hidden layers and neurons as the user specifies. The amount of input and output layer neurons can also be adjusted, along with the activation functions at each layer individually.

This section is comprised of two parts, namely architecture selection, where the amount of hidden layers and neurons selected for best performance is discussed; and the learning section, where the neural network training procedure and results are discussed.

7.1 Architecture selection

Multi-layer neural networks are more powerful than single-layer networks and enable the network to approximate complex functions, whereas single-layer networks have limited applications to simple functions. Research has shown that two-layer networks (single hidden layer), with a log-sigmoid or hyperbolic tangent sigmoid activation function in the first layer and a linear activation function in the output layer, can approximate virtually any function; whereas single-layer networks cannot do this (Hagan *et al.*, 1996). The accuracy of the approximation is a function of the amount of neurons set in the hidden layer. The problem with multi-layer networks is that there are few problems with which one can predict, beforehand, the best number of neurons needed in the hidden layers (Hagan *et al.*, 1996). Therefore, the creator of the network uses trial and error to find the best performing architecture to model his/her dataset. This section is dedicated to a sensitivity analysis within self-imposed limits to find the best performing network architecture for approximating the

incremental species changes due to chemical reactions in a PFR. The architecture elements that will be addressed are the use of different activation functions and their locations (layers), and the effect of the amount of hidden layers and neurons used on the in-sample error.

Before the influence of activation functions and amount of neurons and layer are discussed, we will first have a look at the requirements of the network. The inputs to the network should define the reactor initial conditions; therefore the species mass fractions and the temperature are required. The input vector to the input layer for observation i will take the following form:

$$\text{Input}_i = \begin{bmatrix} T \\ Y_{H_2} \\ Y_{O_2} \\ Y_{H_2O} \\ Y_{CH_4} \\ Y_{CO} \\ Y_{CO_2} \\ Y_{N_2} \end{bmatrix} \quad (7.1)$$

The response vector, for the i observation, should be the species mass fractions after reacting over the specified reaction time-step (as mentioned in the previous section, each reaction time-step has a neural network); therefore the vector will look like:

$$\text{Output}_i = \begin{bmatrix} Y_{H_2}^* \\ Y_{O_2}^* \\ Y_{H_2O}^* \\ Y_{CH_4}^* \\ Y_{CO}^* \\ Y_{CO_2}^* \\ Y_{N_2}^* \end{bmatrix} \quad (7.2)$$

where Y^* is the fine structure regions mass fractions after reacting over time τ^* . The input layer of the neural network will therefore be fixed to 8 neurons and the output layer fixed to 7 neurons. The output variables do not include the temperature as for the input layer, because as seen in equation (4.62), the reaction rate equation of the EDC model only requires the fine scales species mass fractions for it to be evaluated.

7.1.1 Performance of different activation functions

A single-layer neural network with 7 neurons in the hidden layer was used to simplify the process to determine which configuration of activation functions will work best in capturing the incremental species changes. The following combinations of activation functions were investigated:

1. Log-Sigmoid \rightarrow Log-Sigmoid \rightarrow Linear

2. Log-Sigmoid \rightarrow Log-Sigmoid \rightarrow Log-Sigmoid
3. Tanh \rightarrow Tanh \rightarrow Linear
4. Tanh \rightarrow Tanh \rightarrow Tanh

The PFR initial conditions were set to the same values for each one of the four cases to adequately compare the performance of each configuration with one another. The initial conditions are $Y_{H_2} = 0.0$, $Y_{H_2O} = 0.0$, $Y_{CH_4} = 0.15607$, $Y_{CO} = 0.0$, $Y_{CO_2} = 0.0$, $Y_{O_2} = 0.19664$ and $Y_{N_2} = 0.64729$ with a temperature value of $1880K$. The reactor time-step was set to $1 \times 10^{-6} s$ and the number of back-propagation steps or learning iterations were set to 150000 for all cases. The learning rate parameter η was set to 0.35 and the momentum parameter γ set to 0.75. These relaxation parameters were determined by trial and error. After each 100 iterations, the mean-squared error is calculated using equation (4.70) and plotted on a convergence graph to assess the performance of the network. Below are the results of the four cases; each result set has a species fraction prediction and the squared error convergence graph. In the figures below for the species, mass fractions which are the solid lines, are the PFR solutions using the standard ODE solver and the markers with matching colours to the solid lines are the neural network predictions of those species.

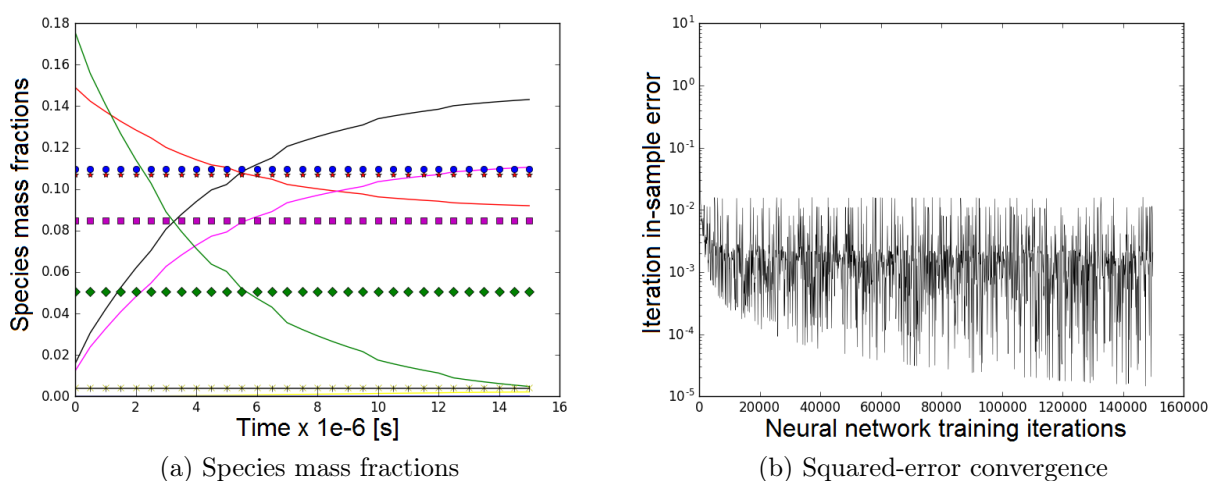


Figure 7.1: (Input) Sigmoid \rightarrow (Hidden) Sigmoid \rightarrow (Output) Sigmoid: configured network

In figure 7.1a it is seen that the configuration with only log-sigmoid activation functions is unable to capture the temporal species changes. This is because the log-sigmoid activation functions are usually used for classification problems where the probability of an occurrence is predicted. The response of the log-sigmoid function varies between 0 – 1, as shown in figure 4.11;

therefore the range of the function is only a single unit with a small gradient for the majority of cumulative signal value range. The small gradient leads to the issue that the log-sigmoid function produces output signals of similar magnitude for input signals with large relative variation in value. The result of this inadequacy leads to a mean error of ± 0.013 on species mass fraction, where the error mentioned is the square root of the mean-squared error ($\sqrt{MSE} = \sqrt{\frac{1}{d_{out}} \sum_{j=1}^{d_{out}} (y_j - x_j^{(L)})^2}$) between the network's output variables' values and the desired values for the specific random sample and training iteration. These errors can be viewed as the mass fraction error the network makes in the prediction. The low accuracy and the inability to capture changes in the input signal of the network can be improved by replacing the output layer log-sigmoid function by a linear function, as seen in figure 7.2a.

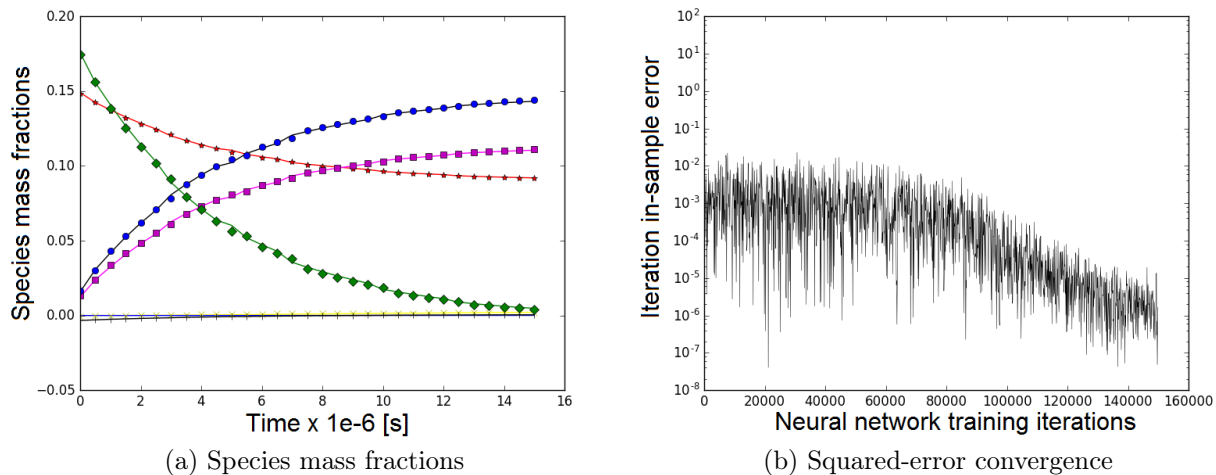


Figure 7.2: (Input) Sigmoid \rightarrow (Hidden) Sigmoid \rightarrow (Output) Linear: configured network

The log-sigmoid activation functions used in the input and hidden layers capture the non-linearity of the data, whereas linear function in the output layer gives the network the ability to scale the signal linearly to the desired response values. The mean-error for the Log-Sigmoid \rightarrow Log-Sigmoid \rightarrow Linear network is $\pm 0.4 \times 10^{-3}$. The network with only hyperbolic tangent sigmoid functions has a mean error of $\pm 0.9 \times 10^{-3}$, which is somewhat lower than the network with only log-sigmoid functions, as seen in figure 7.3a. This is due to two reasons: firstly, the output signal range for the hyperbolic tangent sigmoid function is between $-1 \rightarrow 1$, giving it a larger range for the output signal magnitude; and secondly, the function has a large output signal gradient for input signal values centred around the origin. The former reason gives the function the ability to almost act as a linear function for certain input

signal values, and therefore can capture linear and non-linear behaviour in the dataset.

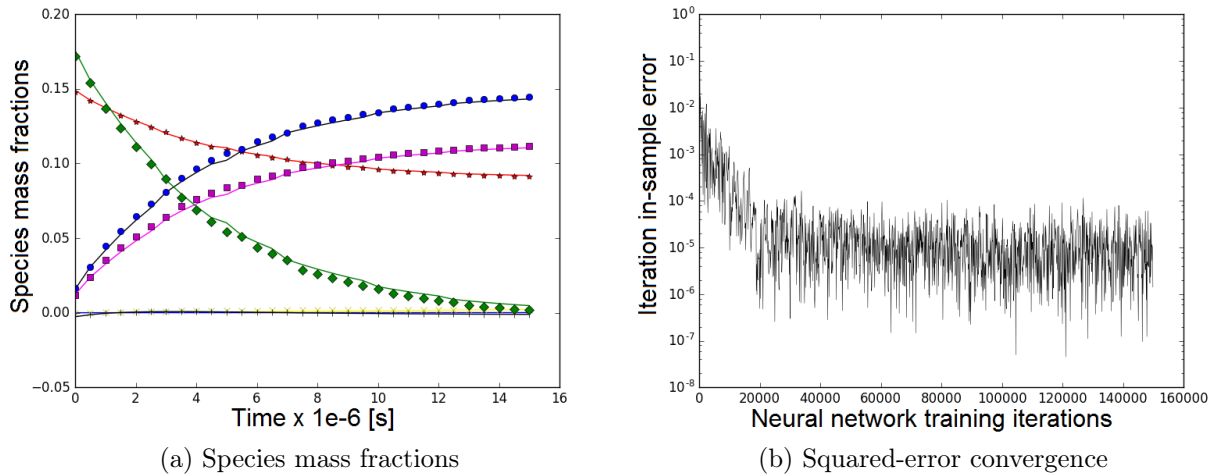


Figure 7.3: (Input) Tanh \rightarrow (Hidden) Tanh \rightarrow (Output) Tanh: configured network

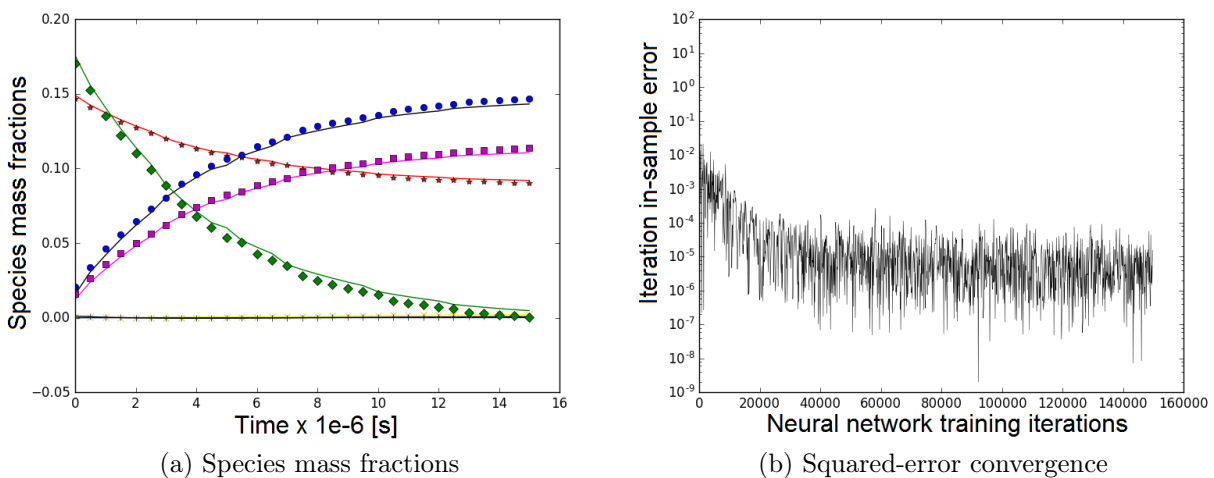


Figure 7.4: (Input) Tanh \rightarrow (Hidden) Tanh \rightarrow (Output) Linear: configured network

The Tanh \rightarrow Tanh \rightarrow Linear (figure 7.4a) network has a mean error of $\pm 0.59 \times 10^{-3}$, similar to the Log-Sigmoid \rightarrow Log-Sigmoid \rightarrow Linear network. According to LeCun *et al.* (1998), hyperbolic tangent sigmoid functions converge faster than log-sigmoid functions as seen in figures above, and have better

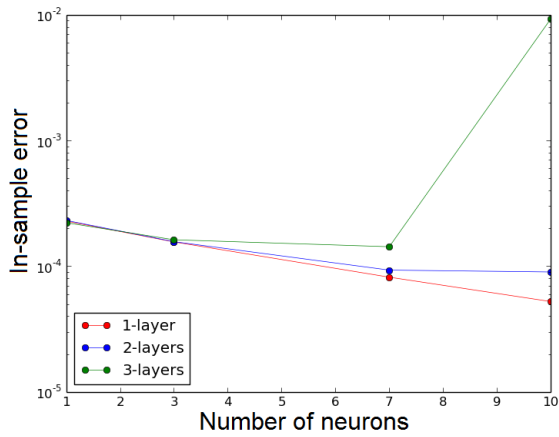
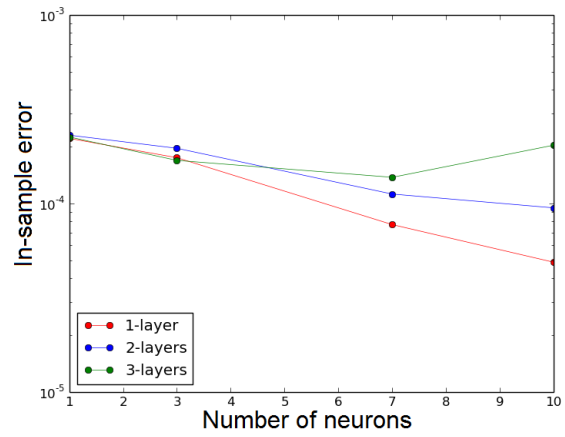
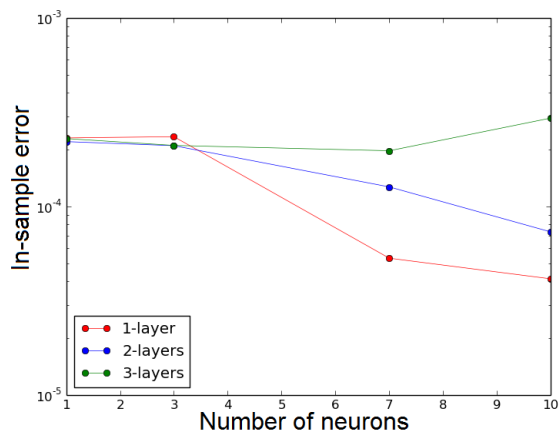
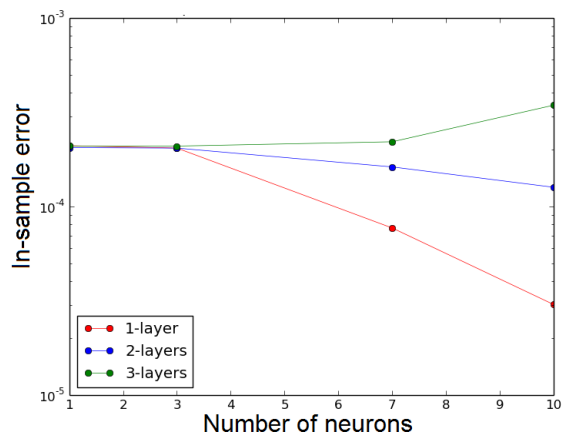
non-linear function capturing abilities as mentioned. Therefore, the activation functions used for the remainder of the present research is comprised of the hyperbolic tangent activation function with a linear function at the final layer to give the network additional linear scaling ability.

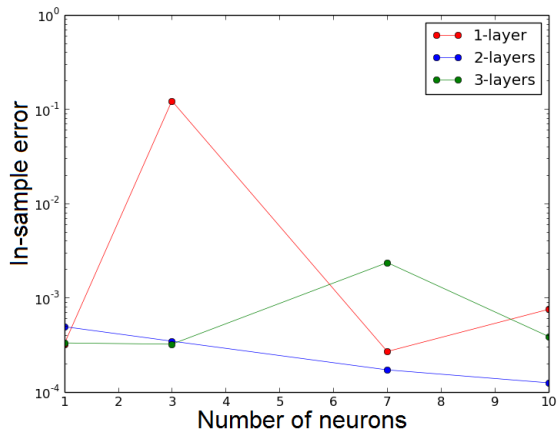
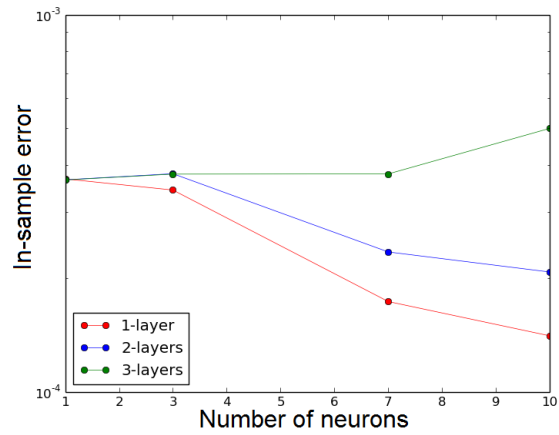
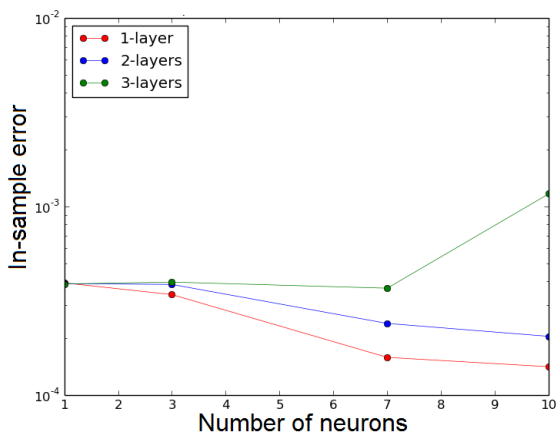
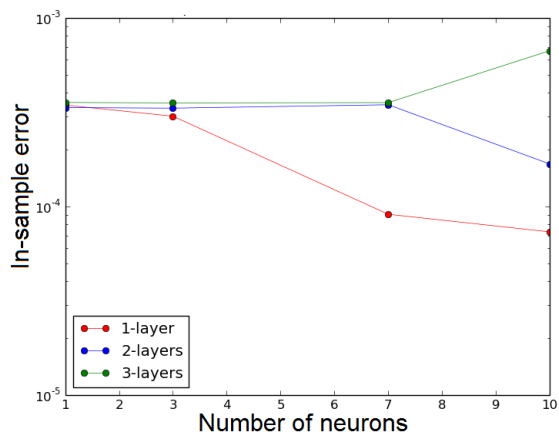
7.1.2 Effect of amount of hidden layers and neurons on performance of network

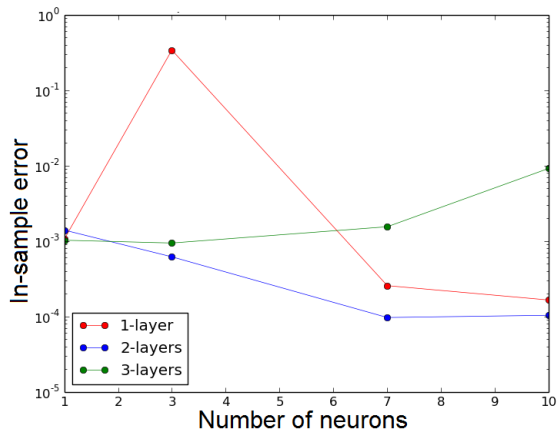
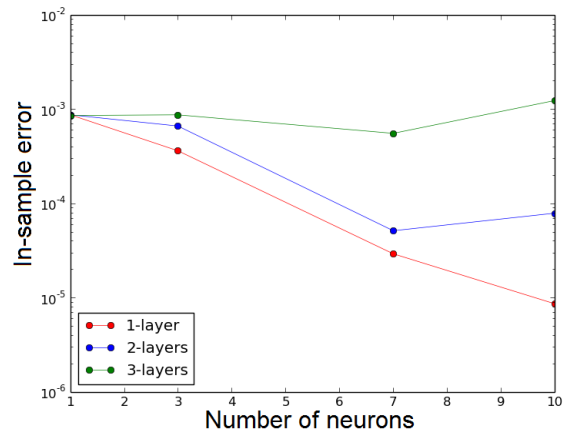
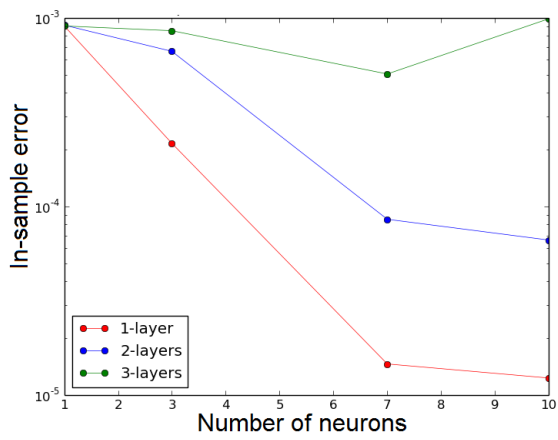
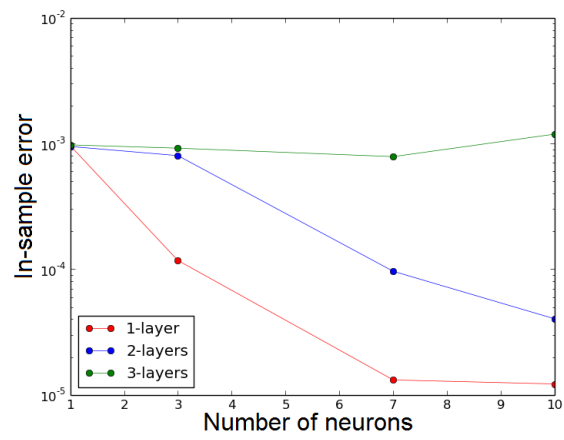
In this section, the amount of neurons and hidden layers of the neural network was varied and the in-sample prediction accuracy calculated to determine the best-performing architecture. For each configuration, a network was trained for the following reaction time-steps: $\Delta 0.5 \times 10^{-6}$, $\Delta 0.5 \times 10^{-5}$, $\Delta 0.5 \times 10^{-4}$ and $\Delta 0.5 \times 10^{-3}$ s. The number of layers was varied between 1 \rightarrow 3, where the number of neurons was varied 1, 3, 7 and 10 for each layer variation. The momentum and learning rate parameters were set to 0.75 and 0.35 respectively. A training dataset was generated by solving 4000 random initial samples with the VODE chemistry integrator and the learning iterations for the neural network training phase was set to 8×10^5 . Due to randomisation introduced into the model by random initial samples and random weights initialisation, each network was trained 20 times on different training datasets; therefore 20×4000 random initial samples. The mean-squared error of the 20 training phases was averaged to determine a global mean error (*MSE*) value with its accompanying standard deviations (σ) for each of the species. The figures on the next page show the global mean error plus the standard deviation ($E = MSE + \sigma$) for the different configurations.

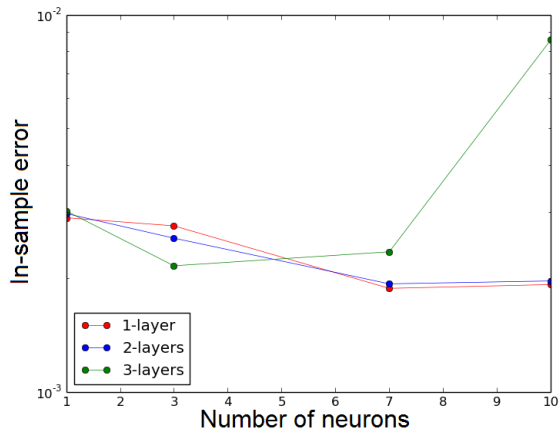
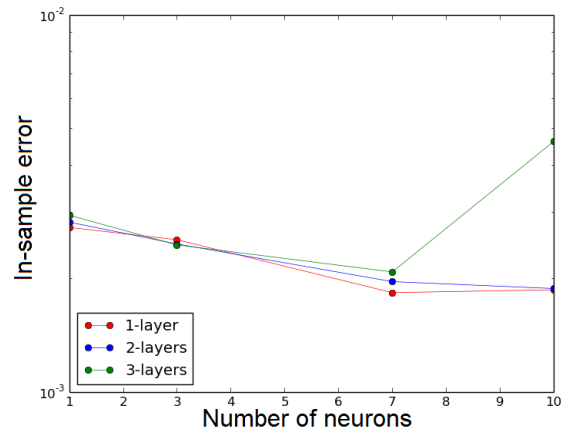
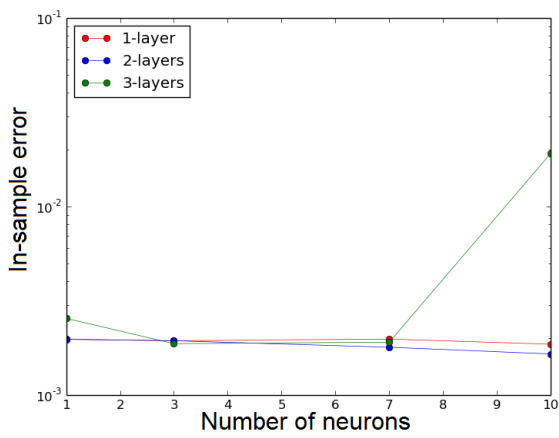
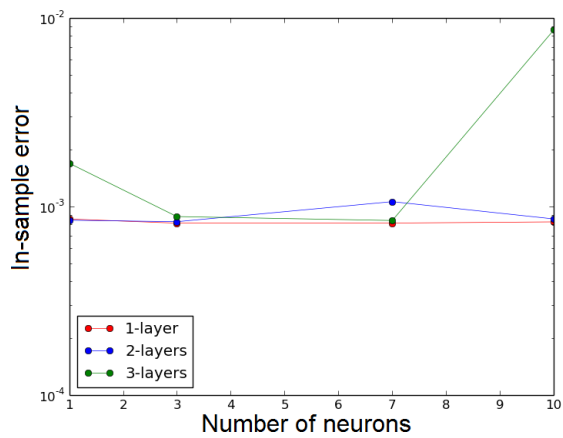
By studying figures 7.5a \rightarrow 7.10d, the following conclusions are made:

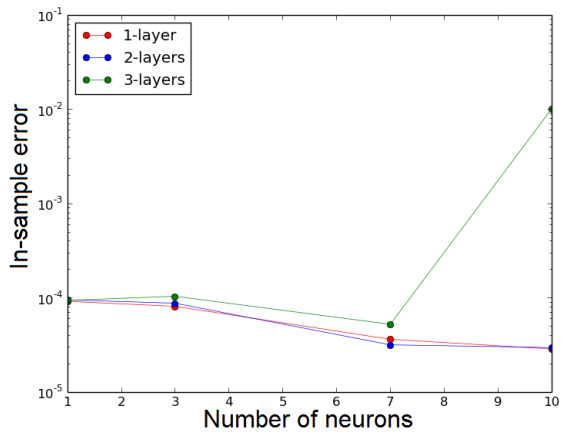
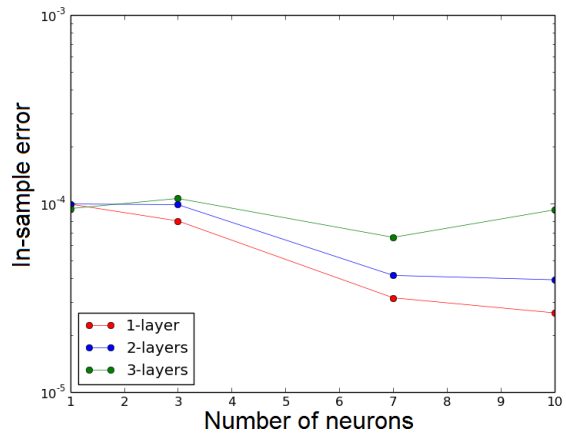
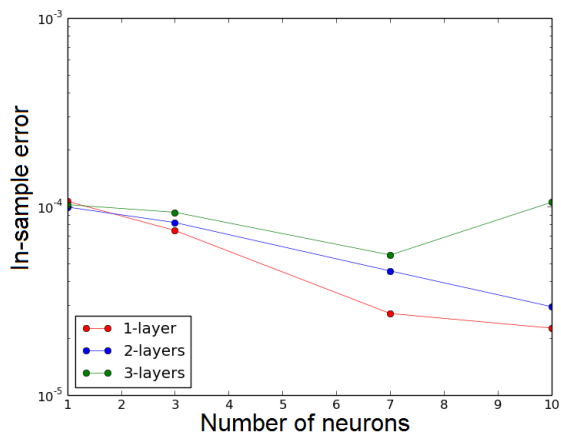
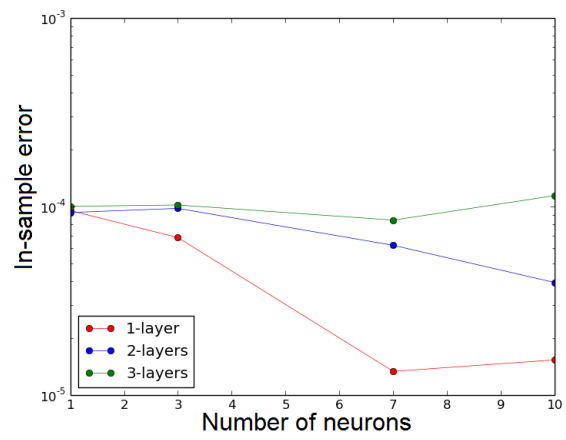
- as the number of neurons increases, the error decreases due to the increased flexibility of model to fit the sample data
- in general, as the number of hidden layers increases, the error increases, this was also observed by Blasco *et al.* (1998); they postulated that the error surface becomes more complex as the number of tuning parameters (weights) increases, which implies that the minimisation algorithm is more likely to get stuck in local minima. The local minima problem is quite general in deep networks (more than one hidden layer) and the in-sample error can be reduced by using an advanced optimization algorithm such as the Adagrad (Duchi *et al.*, 2011) routine as opposed to the standard stochastic gradient ascent/decent algorithm (with momentum).
- as the reaction time-step increases, a more complex network is required to capture the behaviour accurately

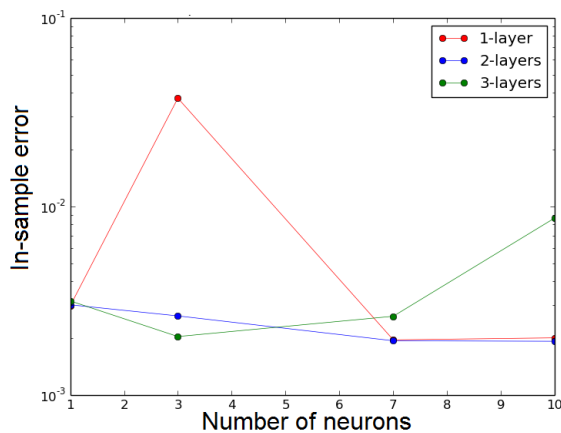
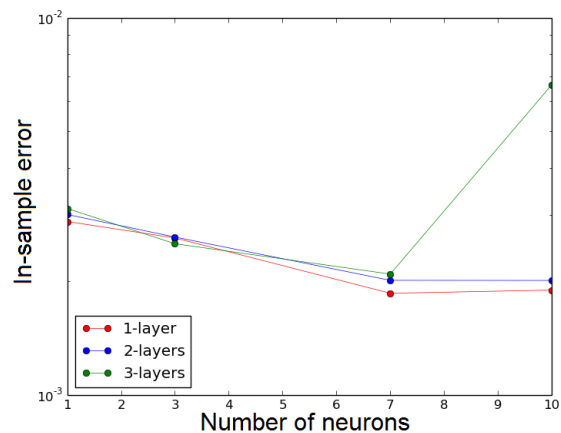
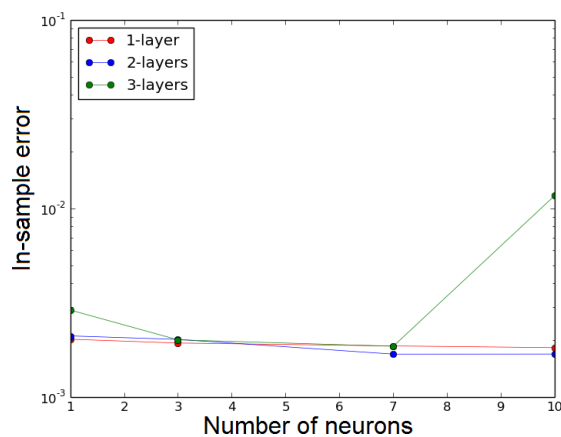
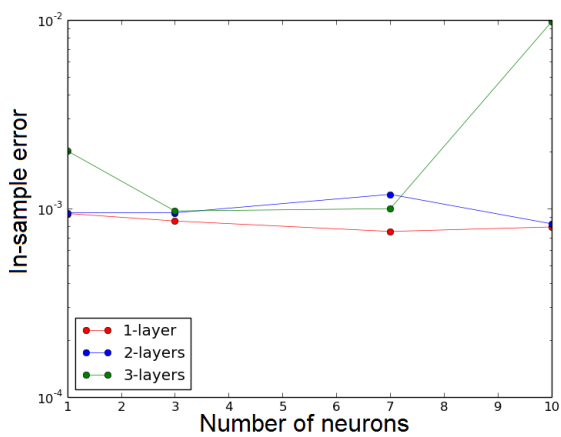
(a) CH_4 mass fractions for 0.5×10^{-3} (b) CH_4 mass fractions for 0.5×10^{-4} (c) CH_4 mass fractions for 0.5×10^{-5} (d) CH_4 mass fractions for 0.5×10^{-6} Figure 7.5: CH_4 mass fractions for varying amount of neurons and hidden layers

(a) *CO* mass fractions for 0.5×10^{-3} (b) *CO* mass fractions for 0.5×10^{-4} (c) *CO* mass fractions for 0.5×10^{-5} (d) *CO* mass fractions for 0.5×10^{-6} Figure 7.6: *CO* mass fractions for varying amount of neurons and hidden layers

(a) CO_2 mass fractions for 0.5×10^{-3} (b) CO_2 mass fractions for 0.5×10^{-4} (c) CO_2 mass fractions for 0.5×10^{-5} (d) CO_2 mass fractions for 0.5×10^{-6} Figure 7.7: CO_2 mass fractions for varying amount of neurons and hidden layers

(a) H_2O mass fractions for 0.5×10^{-3} (b) H_2O mass fractions for 0.5×10^{-4} (c) H_2O mass fractions for 0.5×10^{-5} (d) H_2O mass fractions for 0.5×10^{-6} Figure 7.8: H_2O mass fractions for varying amount of neurons and hidden layers

(a) H_2 mass fractions for 0.5×10^{-3} (b) H_2 mass fractions for 0.5×10^{-4} (c) H_2 mass fractions for 0.5×10^{-5} (d) H_2 mass fractions for 0.5×10^{-6} Figure 7.9: H_2 mass fractions for varying amount of neurons and hidden layers

(a) O_2 mass fractions for 0.5×10^{-3} (b) O_2 mass fractions for 0.5×10^{-4} (c) O_2 mass fractions for 0.5×10^{-5} (d) O_2 mass fractions for 0.5×10^{-6} Figure 7.10: O_2 mass fractions for varying amount of neurons and hidden layers

For the second observation, the problem can be alleviated by increasing the number of training iterations and adjusting the momentum parameter and the learning rate parameter. Studying the in-sample errors of the figures above, the single hidden layer model with 7 neurons was selected to use for the ANN chemistry integrator model to be implemented in the CFD problem. The increase in performance between the 7 and 10 neuron model (single hidden layer) was not substantial; therefore the former was selected to simplify further implementation and reduce training iterations. The appropriate neural network complexity (amount of hidden layers and neurons) is determined by the sample size, the level of noise and the target function complexity. For the current problem the sample size can be set to any size because the training dataset is generated by the user specifying the amount of data points. The training

dataset will have no statistical noise, seeing as the data is generated from the conservation equations for the governing physics. In the next section, it will be shown that the chosen network complexity ($8 \rightarrow 7 \rightarrow 7$) has the necessary degrees of freedom to adequately predict the incremental species changes for samples that were not included in the training set. Figure 7.11 below shows the in-sample error as a function of the reaction time-step. As mentioned, this shows that the relationship between the input variables and the output variables is simpler for shorter time-steps due to smaller changes in species mass fractions.

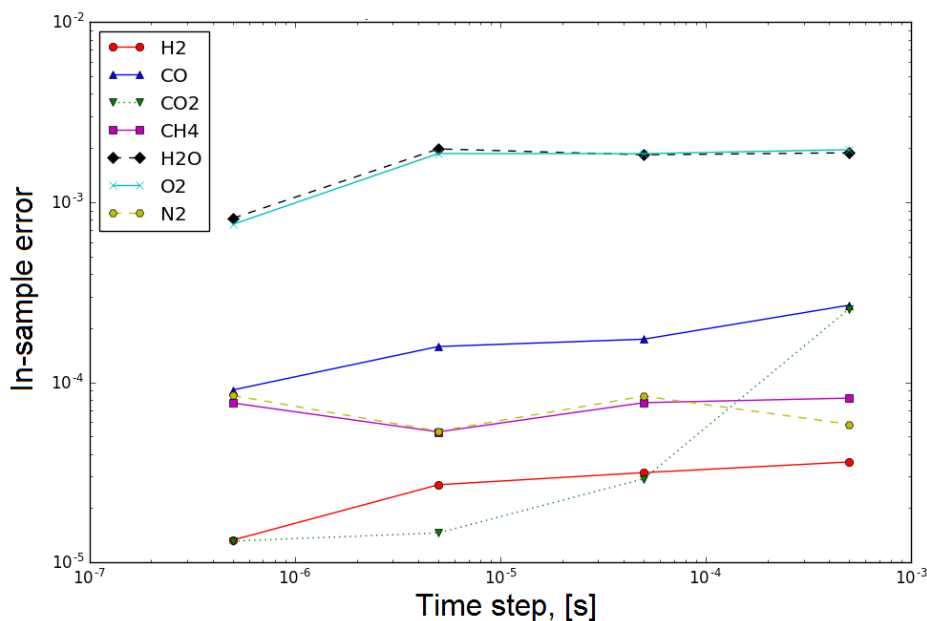


Figure 7.11: In-sample error as a function of reaction time-step for single hidden layer (7 neurons) neural network

Now that the best performing neural network configuration is selected, the networks for the different reaction time-steps must be trained on the PFR simulation data of the WD1 chemical mechanism. The next section discusses this procedure.

7.2 Neural network training of plug-flow reactor generated data

The goal of neural network training is to map the input data sets to their respective response values with relative accuracy. In doing so, the network has adjusted the weights, which minimises the in-sample error and in turn gives the

ANN the ability to capture the linear or non-linear relationship between the training data's input and output observations. In understanding the relationship between the input and output training data, the network then can predict output values for input data values that it has not trained on with relative accuracy. The error between the predicted output values and the actual values are called the out-of-sample error. To ensure the neural networks trained on the PFR ODE solver's results can be used with confidence in the EDC turbulent combustion model of the Sandia flame (Barlow and Frank, 2007), the estimated out-of-sample error should be minimised. The generalisation error is the difference between the in-sample and the out-of-sample error. Therefore, if the generalisation error is minimised, the out-of-sample error is said to track or follow the in-sample error. This means the magnitude of the out-of-sample error will be very similar to the in-sample error calculated during training of the networks. Increasing the amount of training data observations increases the network's ability to capture the relationships between the input and output data; this in turn decreases the generalisation error. For models with high complexity (amount of tuning variables such as the weights), the amount of data points required to ensure the out-of-sample error tracks the in-sample error increases drastically. A theoretical representation of this phenomena is seen in figure 7.12 below.

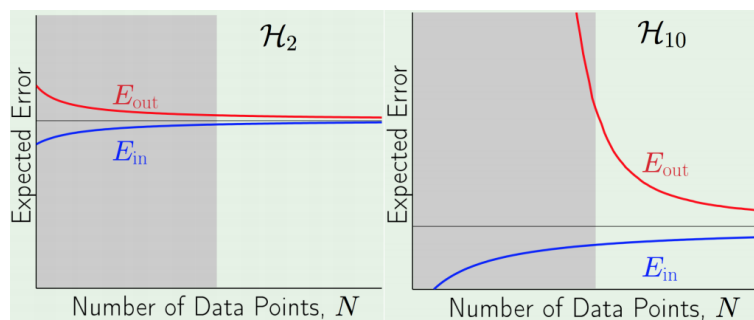


Figure 7.12: Theoretical plot of in-sample error and out-of-sample error for simple and complex statistical models, Abu-Mostafa *et al.* (2012)

In figure 7.12 the graph on the left is for a simple model and the graph on the right is for a complex model. Seeing as training data is generated by the methods discussed in Chapter 6, any size training dataset can be generated. To determine the required size of the training dataset to ensure the generalisation error is minimised, the selected architecture is trained on data sets of size $N = 10, 100, 500, 1000, 2000, 4000, 10000$ and 20000 each. For each dataset, the training iterations were set at 1×10^6 to ensure the model trains on all the samples within the training set. Once each network is finished training, the weights are saved. The network weights are then used in the forward propagation routine to predict the response values of the test dataset. The

test dataset is generated using the same technique as used in training dataset creation and is comprised of all the data points for 4000 random initial samples. The in-sample and out-of-sample errors for the increasing dataset sizes were calculated for a small and large reaction time-step neural network model of 0.5×10^{-6} and 0.5×10^{-3} respectively. This was done to ensure the generalisation error is minimised over the entire range of reaction time-steps used in the CFD simulation. Figure 7.13 and 7.14 below shows the in-sample and out-of-sample errors for these exercises.

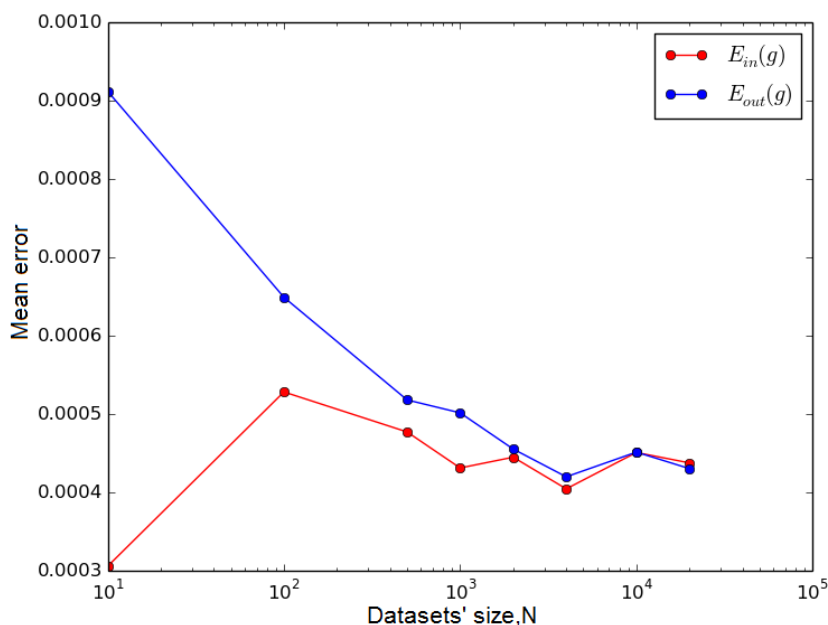


Figure 7.13: In-sample and out-of-sample error for 0.5×10^{-6} network model of increasing dataset sizes

The data above shows that the generalisation error is minimised for training dataset sizes above 10000. The error for the small reaction time-step, as expected, is lower than the error for the large reaction time-step model. We see a strange phenomenon in that the out-of-sample error almost perfectly tracks the in-sample error, but in figure 7.12 theory shows that this is not possible. The perfect tracking is due to the fact that the training data generated by the PFR program has no noise, whereas the theory assumes there is noise in the dataset. The theoretical formulation of the expected out-of-sample error is:

$$\mathbb{E}[E_{out}(g^{(\mathbb{D})})] = \mathbb{E}[(g^{(\mathbb{D})}(\mathbf{x}) - \mu_g(\mathbf{x}))^2] + \mathbb{E}[(\mu_g(\mathbf{x}) - f(\mathbf{x}))^2] + \mathbb{E}[\epsilon(\mathbf{x})^2] \quad (7.3)$$

where in equation (7.3), $g^{(\mathbb{D})}(\mathbf{x})$ is the model expected value; $\mu_g(\mathbf{x})$ is the average output value of the model; $f(\mathbf{x})$ is the target function; and $\epsilon(\mathbf{x})$ is the

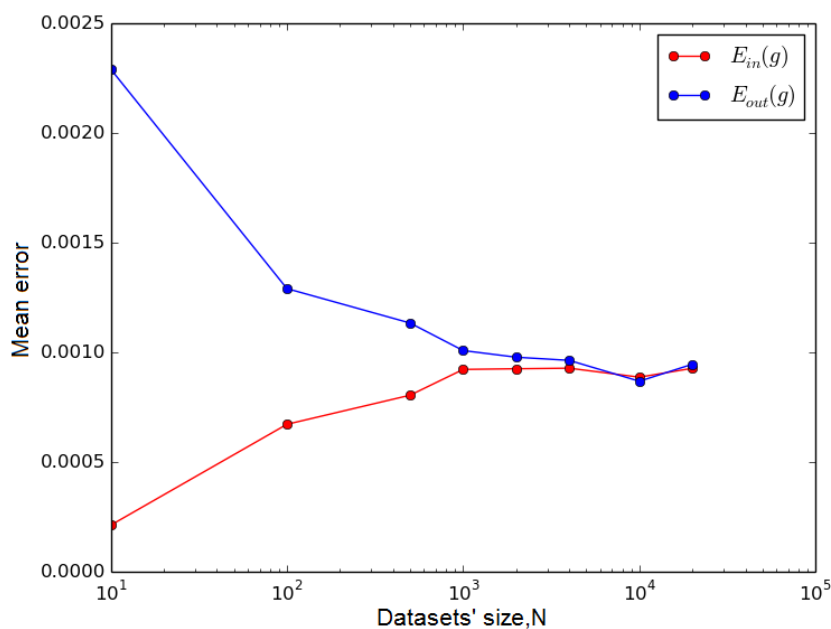


Figure 7.14: In-sample and out-of-sample error for 0.5×10^{-3} network model of increasing dataset sizes

noise in the data. The first term is the variance of the model and is lowered by increasing the training dataset size N and the second term is the bias error of the model. The variance error is the error due to the variability of the model's predictions. The bias error is due to the difference between the predicted value and the actual value. Bias error can be reduced by increasing the model complexity. However, this leads us to a well-known problem called the bias-variance trade-off where, as the bias error is reduced, the variance error increases, and vice versa. High bias error leads to under-fitting, and high variance error leads to over-fitting, but for the data generated in the present research where there is no noise, a large dataset will reduce the variance error and the correct model complexity will reduce the bias error. As mentioned, care must be taken when the model complexity is set too high, as this can lead to problems with the gradient ascent algorithm getting stuck in local minima of the error surface. Based on these findings, the final neural networks will be trained on the data generated by the PFR program using 20000 random initial samples.

It has been shown that the neural networks can predict, with relative accuracy, the incremental mass fractions changes of species during combustion in a PFR. The computational speed difference between the neural network chemistry integrator and the BDF VODE solver was also investigated. This was investigated by solving a user-specified number of random samples (species and temperature) to nearly steady -state conditions. The sample sizes were

set $N = 10, 100, 500, 1000, 2000, 4000, 10000$ and 20000 , where for each sample size the speed of the BDF ODE solver and the ANN chemistry integrator was recorded. The results are shown in the figure below.

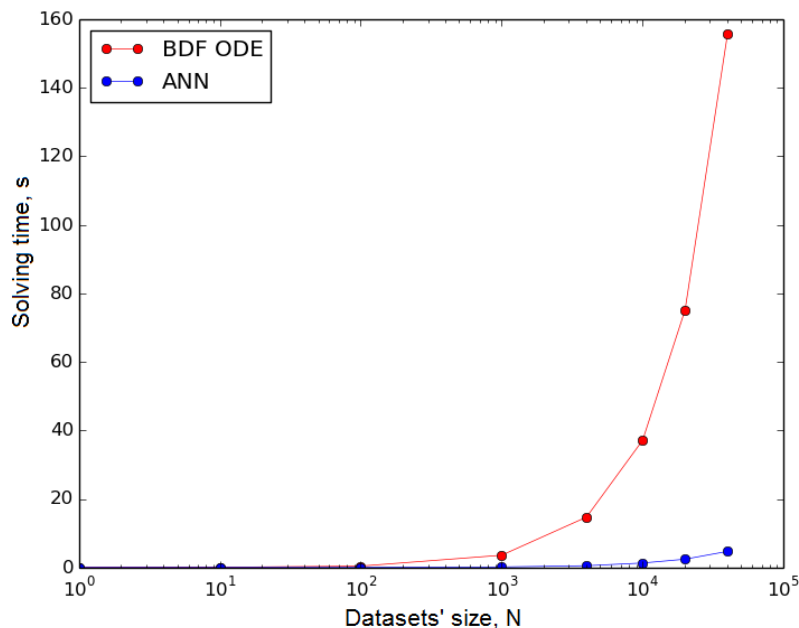


Figure 7.15: Solving time as a function of amount of species compositions solved to nearly steady -state

Figure 7.15, shows a large reduction in solving time if the ANN chemistry integrator is used. The difference in solving time between the traditional ODE solver and ANN integrator increases exponentially as the number of reactions increases. This is because the number of calculations performed by the BDF ODE solver over the ANN integrator increases as the number of reactions increases. The ODE solver integrates the species and energy equation by time-stepping, therefore solving 7 species and 1 energy equation for each time-step, whereas the ANN integrator only performs two matrix multiplications for each time-step. The reduced amount of calculations performed by the CPU for the ANN integrator gives this method the solving time reduction over the traditional method, as seen in figure 7.15.

The computer program codes used to perform all the calculations seen in this chapter can be found in Appendix E along with sample calculations and program flowcharts for the forward- and backward-propagation programs.

Chapter 8

Implementation of artificial neural network in Fluent[®] 17.0

In Chapter 5 the Sandia flame D (Barlow and Frank, 2007) was modelled with Fluent[®] 17.0's standard EDC turbulence-chemistry interaction (DI and ISAT) methods for a methane chemical mechanism. A UDF that calculates the mixture fraction for Fluent[®] 17.0's species transport model was developed. The mixture fraction was used to determine the upper and lower bounds of the mixture fluid volume that is chemically active.

These bounds ensure that the random initial samples generated do not bias towards the chemically non-reactive section of the compositional space and were used in Chapter 6 to generate the training set for the ANN. The dataset was generated using the PFR's species and energy equations for a set amount of random samples and specified reaction time-steps. The equations for each time-step were solved using a BDF VODE solver. Each time-step iteration's initial condition is recorded as the observation's input variable, and the species fractions after reacting over the time-steps was set as the observation's output variables and the next iteration's initial condition. This process was continued until steady state was reached; at that point a new random sample was generated and the process repeated.

In Chapter 7, the author explained methods of training and finding the best performing neural networks, within the self-specified limits, that are capable of predicting incremental species changes. It was shown that the best performing activation functions are a combination of log-sigmoid and linear functions or hyperbolic tangent sigmoid and linear activation functions. The networks require the ability to scale up linearly due to the large formation/-consumption gradients of the species fractions during combustion; therefore requiring hyperbolic-tangent sigmoid and linear activation functions as shown in figures 7.3a and 7.4a. Further, it was shown that a network with a single hidden layer and 7 hidden neurons has an adequate model complexity to fit the training and out-of-sample data reasonably accurately with a large computational time reduction over the traditional BDF VODE solver. In this chapter,

all these aspects of the research will be brought together, and the artificial neural network chemistry integrator will be implemented in the CFD model of the Sandia flame D.

Previous research where ANNs were used to predict chemistry were mostly related to non-premixed combustion where flamelet library entries were used as the training datasets for the neural networks. The combustion was resolved using the steady flamelet model (incorporated in Fluent[®] 17.0). The flamelet model only moderately incorporates chemical non-equilibrium and cannot realistically model combustion that depends on detailed kinetics. The unsteady laminar flamelet model, on the other hand, is able to resolve combustion with a low Damkohler number, but requires the model to be solved transiently, which is not always a viable industrial approach. The computer RAM requirement of the flamelet library is also very large because it needs to store the flamelet library in the computer's memory for fast access during the simulation. The EDC combustion model, as mentioned, is a generic model that can incorporate detailed chemistry and resolve the turbulence-chemistry interaction more fundamentally. The fact that the EDC model is a species transport model and solves a transport equation for each of the species and needs to resolve the stiff chemical kinetics integration, makes it a very computational resource-intensive model. To alleviate the heavy computational resources of the EDC model's integration procedures, the present research implements a unique procedure where the ANN chemistry integrator is programmed to resolve the chemical kinetics integration in Fluent[®] 17.0 without the need for the DI/ISAT methods. The ANN integrator will enable practising engineers to use large chemical mechanisms with the EDC model on an industrial scale and solve the problems within a reasonable time and with reasonable computer resources.

The variables passed to the ANN chemistry integrator are the species mass fractions of the cell, the fine structure time scales τ^* and the cell temperature. To output the incremental species changes in the fine scales over τ^* , the implemented code should calculate the mixture fraction of the cell and determine if the cell is reacting based on the limits specified in Chapter 5. If the cell is reacting, the code should perform a forward propagation routine calling the correct weights and bias matrices based on the cell's chemical reaction time. Once the fine scales mass fractions are calculated, these values are utilised to determine the reaction rates for all the species transport equations and the energy release rate in the energy equation. The reaction rates and energy release rate should then be hooked into Fluent[®] 17.0's conservation equations.

This section will explain in short the implementation of the ANN integrator in Fluent[®] 17.0. The solution procedure for the ANN integrator in Fluent[®] 17.0 is:

1. Prior to Fluent[®] 17.0 global iterations, the mixture fraction is calculated by running the DEFINE_ADJUST(mf_BILGER,d) UDF (see Appendix C), which loops through all the cells in the computational domain.

This routine calculates the mixture fraction and sets it equal to a specified user-defined memory location (C_UDMI(c,t,i) where i designates the location of the variable).

2. The fine structure regions' volume fraction and time scale are calculated for each cell, by the
DEFINE_ADJUST(NN_fineScaleReaction,d) (see Appendix F) UDF, where this UDF is the main function, responsible for getting values from the cells and passing values between the relevant functions such as the neural network function and the source functions.
3. If the mixture fraction is within the pre-specified bounds, the cell's mass fractions and temperatures are called by the main function and cast to a species and temperature vector. This vector will be the input to neural network routine. The main function
[DEFINE_ADJUST(NN_fineScaleReaction,d)] then passes this vector to the neural network function.
4. Based on the reaction time scale τ^* , the neural network function then returns the fine scales species mass fractions for the given initial species mass fraction values and reaction time of τ^* for the cell. The function, therefore, performs the forward propagation calculation of the neural networks. The output vector is then returned to the main function.
5. The individual fine scales species mass fractions of each of the species in the mechanism are then passed to their individual source-term functions by the main function (see Appendix F).
6. The source term functions then calculates the net reaction rate of each species and the energy source due to chemical reactions. These reaction rates and the energy source value for the cell are then hooked to Fluent[®] 17.0's conservation equations.
7. The main function steps to the next cell and the process above is repeated from step (2) until the function has looped through all the cells in the computational domain.
8. Fluent[®] 17.0 performs a global iteration, and the process is repeated from step (1).

To solve the above-mentioned procedure the CFD model is first initialised and solved for 10 – 50 iterations with the traditional EDC model, before the ANN chemistry integrator is activated and Fluent[®] 17.0's reactions deactivated. This is done to ensure that all the fluid variables are initialised and contain values before they are passed to the user-defined functions. The solution methods, pressure-velocity coupling and spatial discretisation, are the

same for the traditional Fluent[®] 17.0 models and the novel ANN-EDC chemistry integrator. The under-relaxation factors for both approaches are seen in the table below.

Table 8.1: Under-relaxation factors for ANN chemistry integrator and standard EDC simulations with DI/ISAT activated

Pressure	0.15
Density	0.6
Body forces	1
Momentum	0.6
Turbulence kinetic energy	0.7
Turbulence dissipation rate	0.7
Turbulence viscosity	0.8
Species	0.5
Energy	0.5

The DI, ISAT and ANN chemistry integration models were solved until the residuals and custom monitors reached convergence. The monitors configured are: (1) the average vertex CO mass fraction of the centreline axis; (2) maximum vertex H_2 mass fraction of the centreline axis; (3) average vertex O_2 mass fraction of the centreline axis; (4) maximum vertex CH_4 mass fraction at a radius of 0.02 m and the maximum vertex temperature at the centreline, 0.01 m radius and 0.02 m radius.

An alternative approach to model detailed chemistry with the EDC combustion model was proposed and implemented by Schmidt *et al.* (2004) and Anderson *et al.* (2015). They used the EDC as a chemistry solver applied to a developed flow, turbulence and temperature field. This entails solving the combustion with EDM for a two-step mechanism for the given fuel. Once the simulation is converged, the flow, turbulence and energy equations are disabled. The EDC is then activated, and a detailed chemical mechanism is loaded into the simulation and solved; thus the EDC is used as a chemistry post-processor to predict intermediate and pollutant species. Anderson *et al.* (2015) found that the EDC model resulted in slow fuel conversion rates for a 50 kW non-swirling natural gas combustor, but once the EDC calculations were superimposed on the flow-field from the EDM model, superior results were achieved. Schmidt *et al.* (2004) and Anderson *et al.* (2015) showed that reasonable results could be achieved with this approach. This alternative approach was also modelled in the present research and compared to the experimental data, EDC method with traditional chemistry integration, EDC with ANN chemistry integration and some results from literature. Scharler *et al.* (2003) showed that the original Magnussen constants A_{EDM} and B of 4 and

0.5 did not yield good prediction accuracy for a $CO/H_2/N_2$ flame, in that it over-predicted the reaction rate of the CO oxidation reaction. Scharler *et al.* (2003) proposed that the EDM constants A_{EDM} and B should be 0.6 and 0.5 respectively. These constants were used in the EDM model for reaction 2 in table 4.1. The simulation was also configured only to solve the first two reactions of the WD1 mechanism.

Using a similar approach as Schmidt *et al.* (2004) and Anderson *et al.* (2015), a second alternative is also proposed. Rather than using the EDM and a two-step mechanism to model the temperature, flow and turbulence field, a two-step EDC model is used with the DI/ISAT integration procedure. Once the solution has converged, the full mechanism is loaded into the model and the chemical kinetics resolved using the ANN chemistry integrator while the flow, turbulence and energy fields are frozen. This approach gives the added benefit of not having to adjust the Magnussen constants for the specific problem being solved. The amount of equations needed to be integrated is also lower, thus reducing the computational load of the simulation.

The methods mentioned above enables a practising engineer to predict temperature, flow and species quantities on industrial scale combustion problems using the EDC with the ANN chemistry integrator. Two levels of approaches were mentioned: first; using the ANN chemistry integrator and EDC model as a full combustion modelling approach and the second level model uses the ANN-EDC combustion model as a chemical post processor which can be used on new or already existing and validated CFD combustion models of boilers or internal combustion engines to predict formation of pollutants and unburned hydrocarbons.

Chapter 9

Results and computational performance

The performance of the novel ANN chemistry integrator-EDC model will be assessed in two parts, namely accuracy and computational performance. For the accuracy assessment the results generated solving the CFD model of the Sandia flame with the new approaches (full EDC solution and the superimposed EDC chemistry solutions) are compared to the experimental data, the results of the EDC model using the DI and ISAT chemistry integration procedures and Kempf *et al.* (2005)'s results for their LES steady laminar flamelet-ANN model. In the second part, the utilised computational resources of the EDC-ANN model will be compared to the method of using the traditional ISAT and DI integration techniques. All the models will use the realizable $k - \epsilon$ turbulence model and at the end of this section two of the chemistry integration models will be simulated using the RSM model. The goal is to determine if the new approach will yield reasonably accurate results compared to the experimental data and results generated using the traditional integration techniques, while offering a substantial computational resource reduction.

9.1 Results comparison

In this section, the predicted species mass fractions, temperatures and mixture fractions of the different modelling approaches are compared to the experimental data and results of Kempf *et al.* (2005)'s LES steady laminar flamelet-ANN model. The same data sampling locations in the computational domain used in Chapter 5 will be utilised here again. The species and temperature results of the EDC model using the ISAT or DI procedures are similar; thus only the DI results will be shown in the following figures.

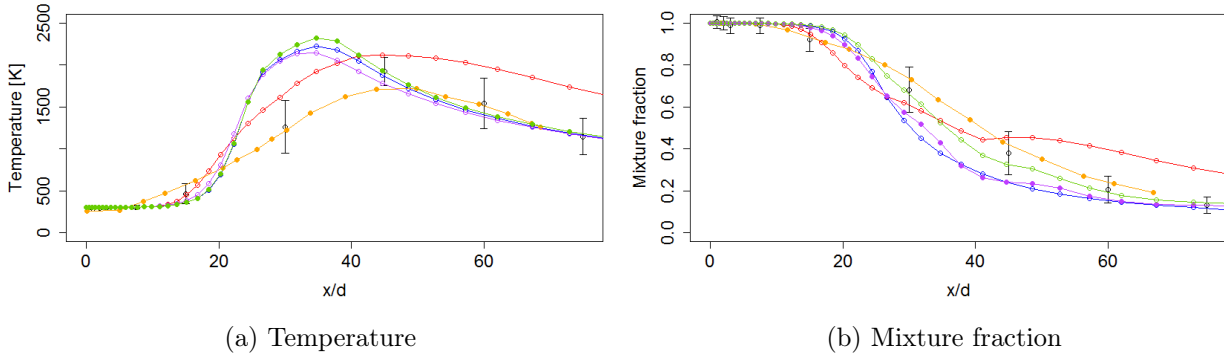


Figure 9.1: Temperature and mixture fraction distribution at axial locations on the centreline of the flame. \ominus (black) - Exp, \ominus (blue) - DI/ISAT EDC, \ominus (red) - EDC/ANN, \ominus (green) - EDC/ANN on EDM temperature and flow field, \ominus (magenta) - EDC/ANN on EDC two-step temperature and flow field and \ominus (orange)- Kempf *et al.* (2005)

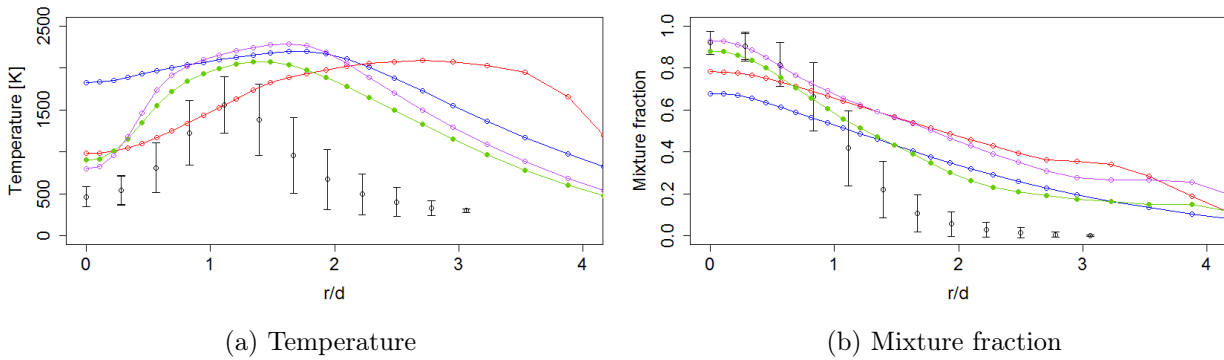


Figure 9.2: Temperature and mixture fraction distribution at radial locations at $x/d = 0.15$ m

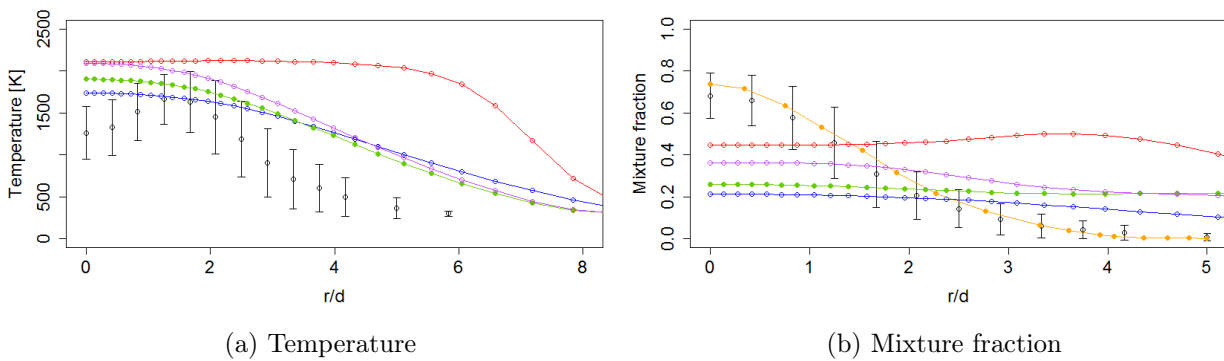


Figure 9.3: Temperature and mixture fraction distribution at radial locations at $x/d = 0.3$ m

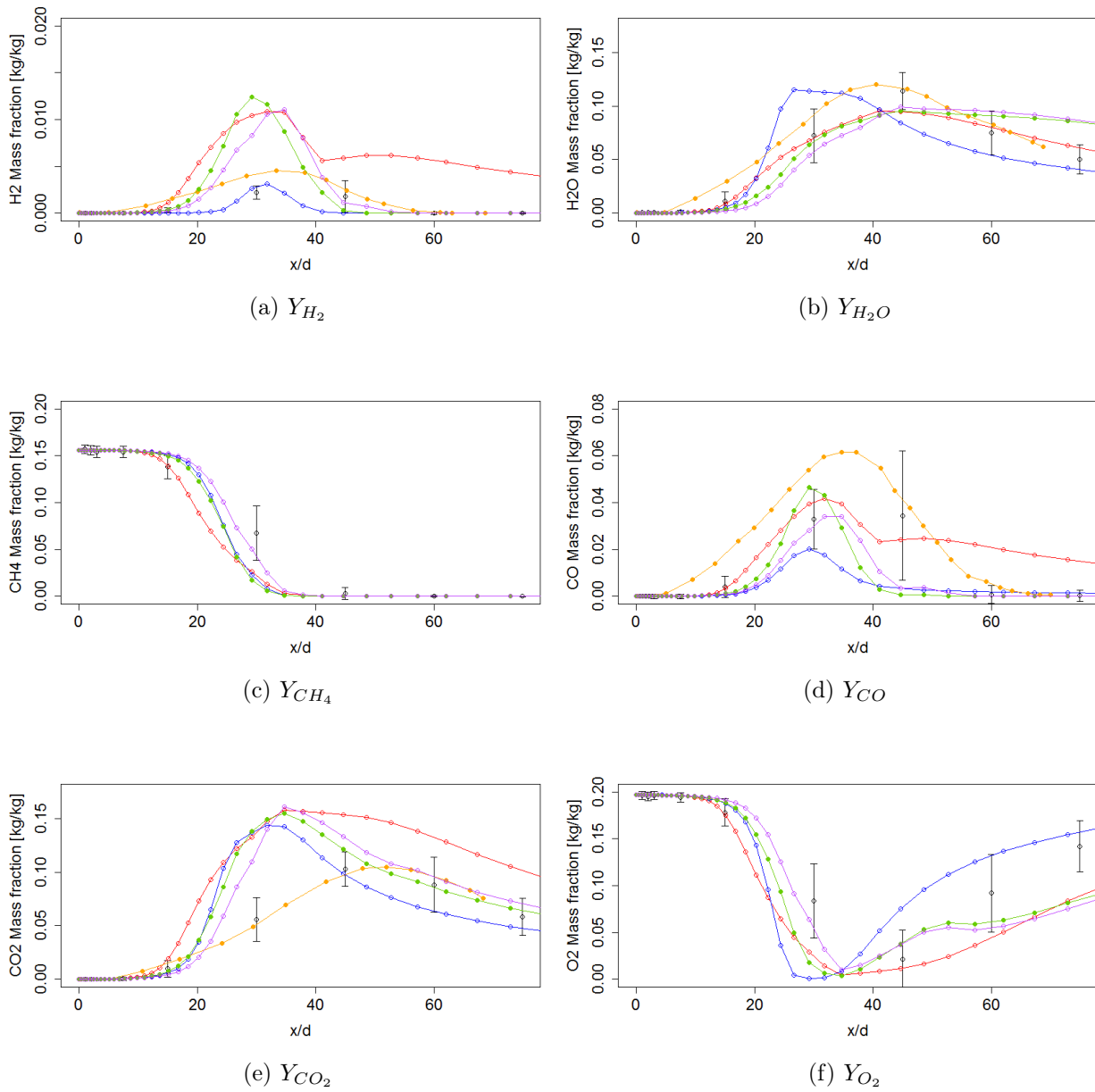


Figure 9.4: Species mass fractions at axial locations along centreline of flame. \ominus (black) - Exp, \ominus (blue) - DI/ISAT EDC, \ominus (red) - EDC/ANN, \ominus (green) - EDC/ANN on EDM temperature and flow field, \ominus (magenta) - EDC/ANN on EDC two-step temperature and flow field and \ominus (orange) - Kempf *et al.* (2005)

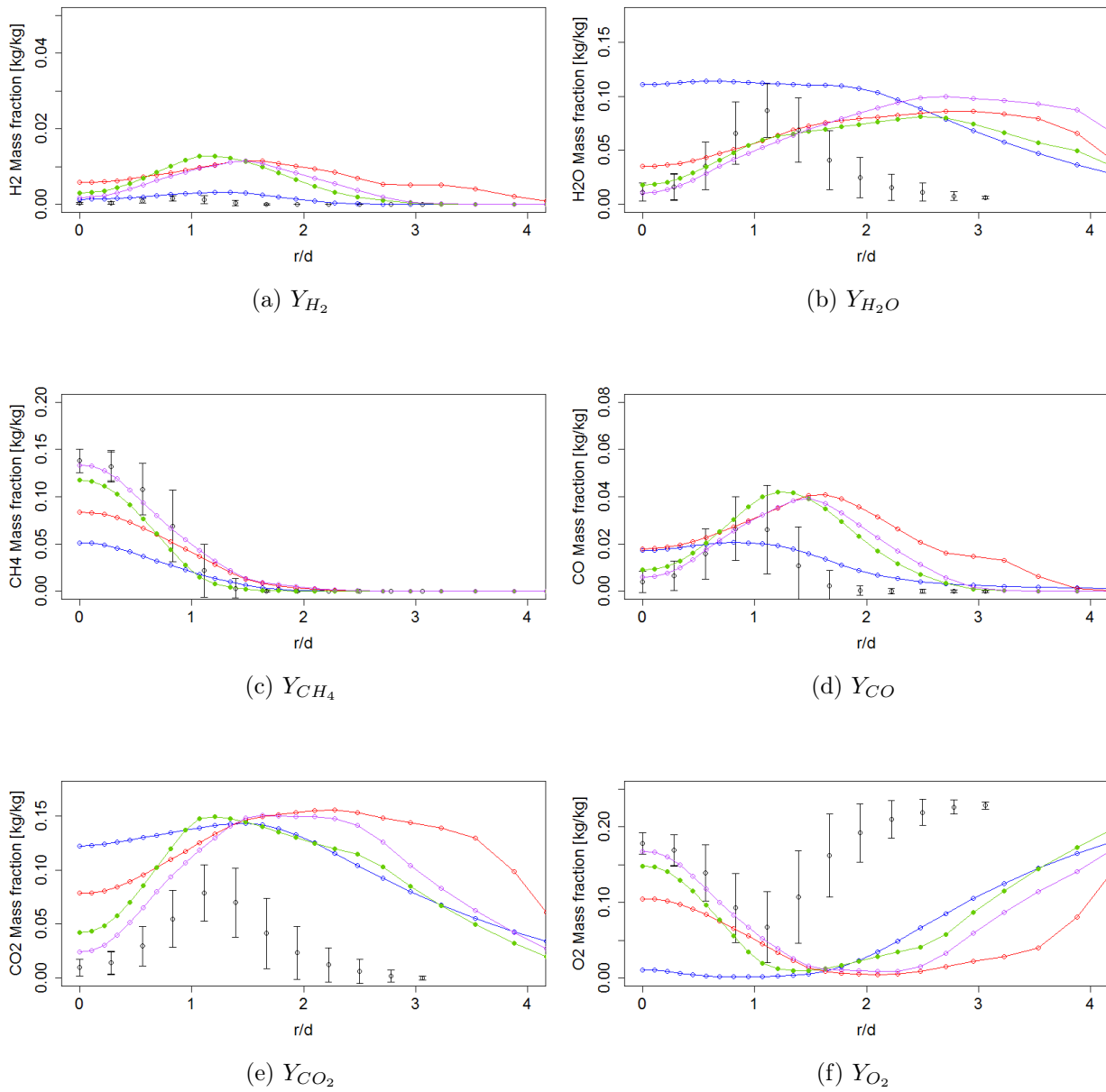


Figure 9.5: Species mass fractions at radial locations at $x/d = 0.15$ m. Θ (black) - Exp, \ominus (blue) - DI/ISAT EDC, $\omin�$ (red) - EDC/ANN, $\omin�$ (green) - EDC/ANN on EDM temperature and flow field, $\omin�$ (magenta)- EDC/ANN on EDC two-step temperature and flow field and $\omin�$ (orange) - Kempf *et al.* (2005)

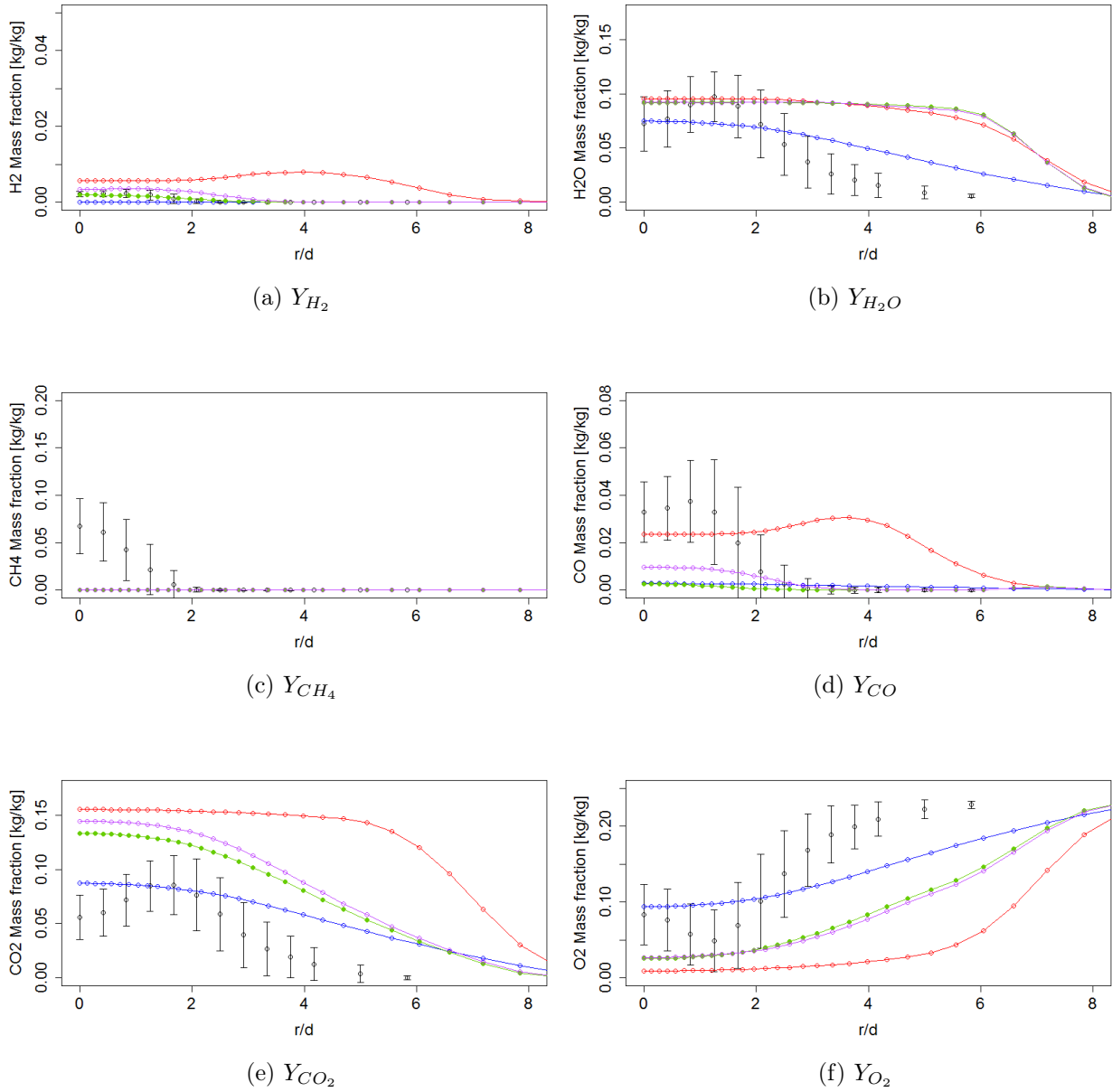


Figure 9.6: Species mass fractions at radial locations at $x/d = 0.3$ m. Θ (black) - Exp, \ominus (blue)- DI/ISAT EDC, $\omin�$ (red)- EDC/ANN, $\omin�$ (green) - EDC/ANN on EDM temperature and flow field, $\omin�$ (magenta) - EDC/ANN on EDC two-step temperature and flow field and $\omin�$ (orange) - Kempf *et al.* (2005)

For the remainder of this discussion, the different methods applied will be named:

- EDC model solved using traditional ISAT/DI techniques - Model 1

- EDC model solved using the ANN chemistry integrator - Model 2
- EDC model solved with ANN chemistry integrator while energy, flow and turbulence fields are frozen. The flow, turbulence and energy fields are calculated using the EDM model with a two-step reaction mechanism - Model 3
- EDC model solved with ANN chemistry integrator while energy, flow and turbulence fields are frozen. The frozen fields are resolved using an EDC two-step chemical mechanism combustion model solving major species - Model 4

Figures 9.7a and 9.7b below show the convergence of the species fractions' residuals for simulations using the ANN and DI/ISAT integration techniques. The residuals show that both integration techniques yield good convergence, whereas the traditional procedures outperform the ANN technique.

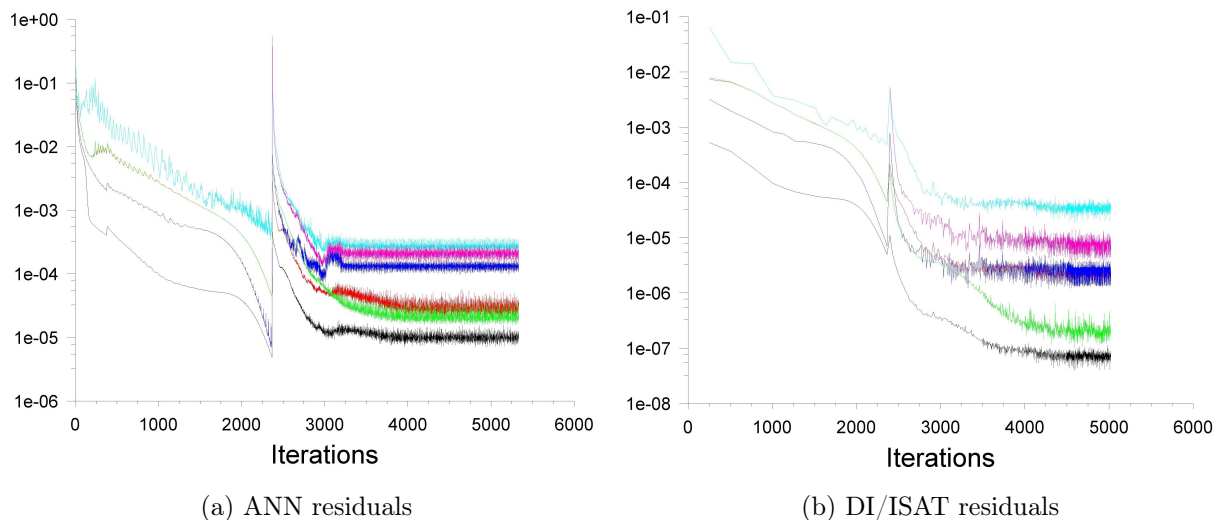


Figure 9.7: Residuals of species fractions for Sandia flame D simulation using ANN chemistry integrator and ISAT/DI integrators

Figure 9.1a shows that model 1, 3 and 4 all predict an early temperature peak occurring closer to the burner than the experimental data shows. The temperature peaks predicted by the models are also higher than the experimental data. The oxidation rate of CH_4 predicted by these models are higher than the rate observed during the experiment. This is due to reaction 1 in table 4.1 that has either too high a pre-exponential factor or too low activation energy and the over production of turbulence close to the nozzle by the realizable $k - \epsilon$ model. The rapid oxidation of the CH_4 into CO , high turbulence production and high-temperature zone located close to the burner causes

the CO oxidation rate predicted by the models to be higher than the reaction rates of the experiment. This is seen in figures 9.4d, 9.5d and 9.6d, where the CO mass fraction peaks predicted by models 1, 3 and 4 occur closer to the jet outlet and is consumed more rapidly than the experimental results show. This is further illustrated by figure 9.6d, where it is seen that the predicted CO is almost completely consumed, whereas the experimental data shows that there is still some CO left in the mixture close to the centreline of the flame at an axial location of $x/d = 0.3m$. The difference in the height of the CO peaks between model 1 and models 3,4 in figure 9.4d is due to the out-of-sample error of the neural network's prediction. In figure 9.1a, it is seen that the EDM model predicts a higher temperature than the EDC model with a two-step reaction mechanism. The lower CH_4 oxidation rate of the EDC model leads to a slower CO oxidation rate, which means that the CO is consumed at a slower rate for model 3, and this is seen in figure 9.4d. The high temperature zone and high intermediate species concentrations cause accelerated oxidation of the fuel and unburned species. Evidence of this is seen in figures 9.4c, 9.5c and 9.6c where high concentrations of CO_2 are observed closer to the jet outlet than the experimental data shows; and further downstream, the CO_2 mass fractions predicted by models 1, 3 and 4 are higher than the experimental values off from the centre line. Figure 9.4a shows that the ANN integrator (models 3,4) over-predicts the amount H_2 in the gas mixture compared to the experimental data and model 1; this is most likely due to the neural network's out-of-sample error. From the figure, it is seen that the amount of H_2 in the gas mixture is very small, thus the effect of the neural network's out-of-sample error is more substantial and therefore the magnitude of over-prediction more severe.

The mixture fraction bounds implemented also have an effect on the results predicted by the neural networks. The lower bound does not have a major effect on the results if the value is lower than 0.15. Any specified bound higher than this will cause the ANN active zone to shrink to a smaller volume than what the actual reacting zone is (see figure 5.13). The results are more sensitive to the upper bound (0.6), because this value moves the ANN active zone closer and further away from the jet outlet. Decreasing the upper mixture fraction bound will move the active centreline further away from the jet outlet and thus delay the oxidation of CH_4 and subsequent reactions. Increasing the upper mixture fraction bound will move the active ANN zone closer to the jet outlet and cause reactions to take place closer to the jet outlet. Figure 9.8 below shows the CO predictions by the ANN chemistry integrator for different upper bound mixture fractions values on the centreline of the flame. It is clearly seen that as the mixture fraction increases, the reaction zone moves closer to the jet outlet.

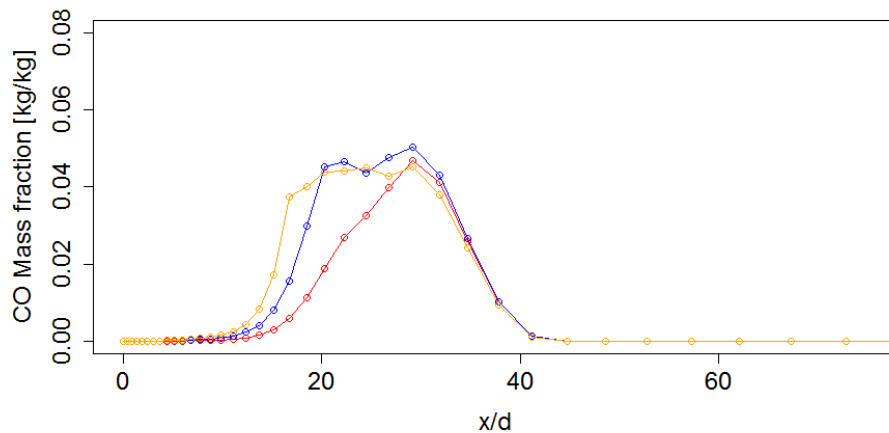


Figure 9.8: Predicted CO mass fractions by ANN chemistry integrator for different upper bound mixture fraction limits with models 3,4. \ominus - $f = 0.9$, $\omin�$ - $f = 0.8$ and $\omin�$ - $f = 0.6$

In conclusion, models 3,4 are in good agreement with the EDC model using the traditional integration procedures, but still have the same drawbacks as the traditional modelling approach because of the mechanism and turbulence model used.

Next, model 2 will be compared to models 1,3,4 and experimental results. Model 2 enables the chemistry solution to change the energy, flow and turbulence fields of the CFD simulation. It is seen in figure 9.1a that the temperature peak predicted by model 2 occurs further downstream from the jet outlet. A larger temperature difference is seen between model 2 and models 1,3,4 than between the latter models (1,3,4). This result is due to the ANN chemistry integrator's predicted reaction rates that can change the temperature of the mixture through the chemical energy source term that is included in the energy equation of Fluent[®] 17.0. The high predicted H_2 and CO mass fractions of model 2 indicates that the chemical energy contained in the hydrogen and carbon-monoxide bonds is yet to be released, thus the temperature of the gas mixture is lower. The mixture fraction limits therefore delay the reaction and heat release, causing all subsequent reactions to take place further downstream than predicted by models 1,3,4. Model 2 suffers from the same problem as models 3,4 in that the quantities of hydrogen in the mixture is extremely small; thus the out-of-sample error of the neural network's prediction has a substantial effect on the prediction accuracy; this is seen in figure 9.4a. Based on the results of model 2, it can be concluded the temperature and species mass fractions are much more sensitive to the mixture fraction limits than the other models, as seen in figure 9.9.

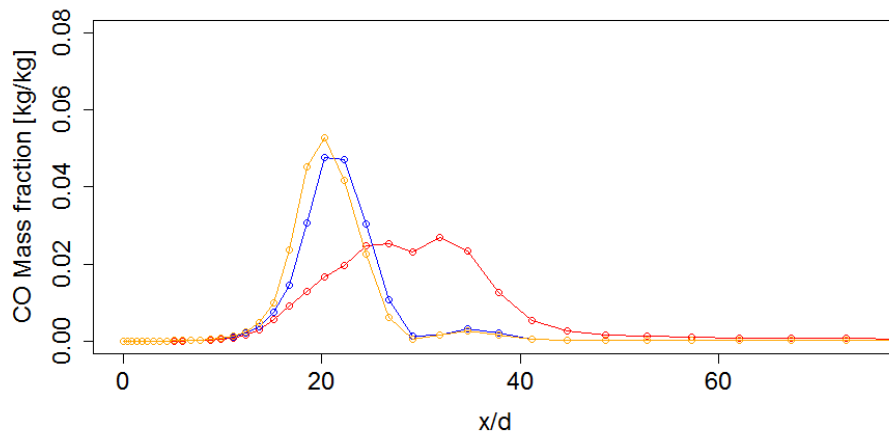


Figure 9.9: Predicted CO mass fractions by ANN chemistry integrator for different upper bound mixture fraction limits with model 2. \ominus - $f = 0.9$, $\omin�$ - $f = 0.8$ and $\omin�$ - $f = 0.6$

The three modelling approaches used (models 2,3,4) all show relatively good agreement with each other and the CFD simulation using the ISAT/DI integration procedures. This was expected, given that the new mathematical approach to handling the chemistry integration was developed to be a low-resource alternative chemistry integration technique that can be used with the EDC model on an industrial scale, and not an entirely new combustion model.

The results show that the model of Kempf *et al.* (2005) predicts the experimental results more accurately than models 1, 2, 3 and 4. This would be expected because Kempf *et al.* (2005) used a superior turbulence model along with a detailed chemical mechanism (GRI 3.0). The combustion model used by Kempf *et al.* (2005) was the steady laminar flamelet model. As mentioned, the model is only able to moderately account for chemical non-equilibrium ($Da \gg 1$). The Damkohler number range for the Sandia flame D model is between $1 \times 10^{-12} \rightarrow 6$, thus the combustion conditions of the experimental setup is on the limit of what the model can accurately predict; nonetheless, the more detailed chemical mechanism used by Kempf *et al.* (2005) accounts for intermediate reactions that cause the temperature, CO , H_2 and H_2O peaks to align closer to the experimental results. Figure 9.4d shows that the steady laminar flamelet model over predicts the CO mass fractions close to the burner outlet. This is due to the model's inability to account for detailed chemical kinetics (only applicable to fast chemistry). Close to the burner the composition of the mixture is fuel rich and the flow highly turbulent. The fine structure time scales therefore becomes smaller than the time required to reach chemical equilibrium, and the fast chemistry assumption becomes invalid in that area leading to the over prediction of the CO . LES turbulence models require that

the mesh is extremely fine to resolve some of the subscale turbulence effects and to be solved transiently, making it not industrially viable to solve large problems such as combustion in boiler furnaces.

The ANN chemistry integrator will in future research be configured to work with chemical mechanisms such as the one used by Kempf *et al.* (2005) that enable the prediction of fouling constituents (KCl) and pollutants (NO_x , SO_x and unburned hydrocarbons). A larger chemical mechanism will naturally comprise of more species. The input layers of the networks will therefore be larger, and the hidden layers will also have more neurons. If the relationship between the compositions and their incremental changes are more complex, additional layers can be used if necessary. Even with the additional layers and neurons, the neural network approach will still outperform the traditional methods in terms of computational speed. For larger networks, the forward propagation routine will still only be a single-digit number of matrix multiplications. The matrices will be larger due to the increase in the number of neurons, but this will have a very small effect on solving time. The DI number of calculations performed by the CPU, on the other hand, will increase substantially due to the increase in the number of reactions and species. The addition of radical species that have a very high reaction rate will cause the set of ODEs to become stiffer, thus increasing solving time even further. As the chemical mechanism's complexity increases, the performance benefit of the ANN over the DI will increase. Complex mechanisms also require that the available ISAT table memory increases to accommodate the additional reactions and species. Again, the ratio of RAM reduction of the ANN integrator over the ISAT table will also increase as the mechanism complexity increases.

As shown in Chapter 5, the RSM turbulence model predicted a lower turbulence production rate which resulted in the temperature peak on the centre line of the flame to occur further downstream than what the two equation models predicted. The peak also aligned closer to the experimental data. Using the modelling approaches of models 2 and 4 and the RSM turbulence model the Sandia Flame D was simulated. Figures 9.10a \rightarrow 9.10f and 9.11 shows the results of the fluid species mass fractions and temperature on the centre line of the flame. As with the results seen in Chapter 5, we see that the RSM model pushes the temperature, CO and CO_2 peaks further downstream. The realizable $k - \epsilon$ turbulence model over predicts the turbulence generated close to the burner nozzle in the active reaction zone, in turn over predicting the CH_4 and CO oxidation rates. The contour plots of Re_t for the RSM and realizable $k - \epsilon$ models can be seen in Appendix G.

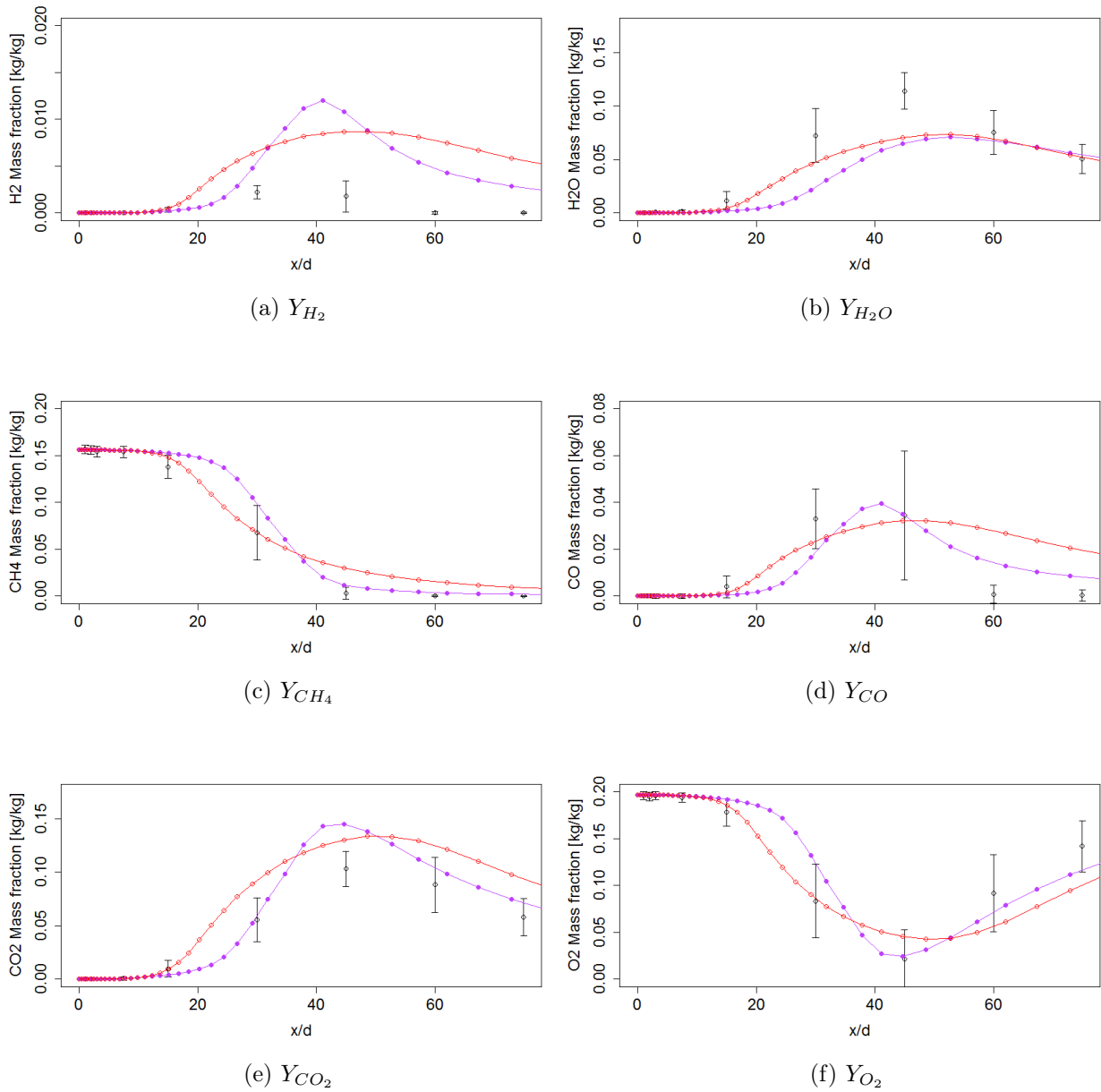


Figure 9.10: Species mass fractions at axial locations along centreline of flame predicted using the RSM turbulence model. \ominus (black) - Exp, \ominus (red) - ED-C/ANN and \ominus (magenta) - EDC/ANN on EDC two-step temperature and flow field

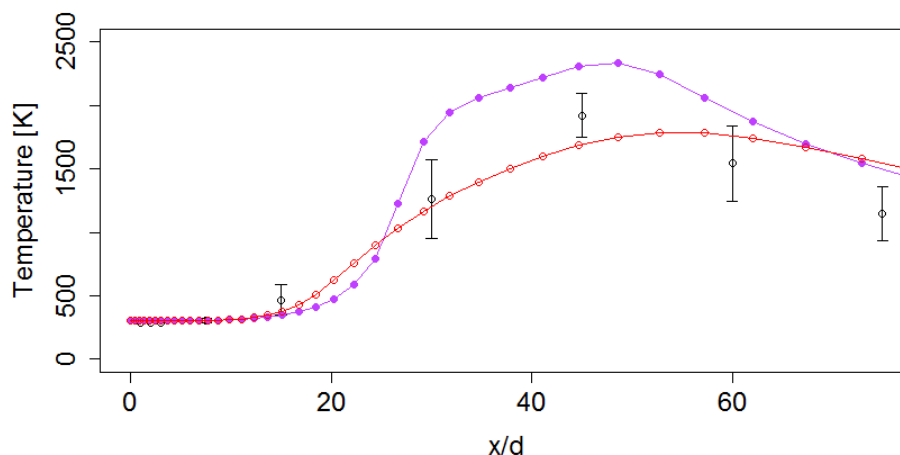


Figure 9.11: Temperatures at axial locations along centreline of flame predicted using the RSM turbulence model. Θ (black) - Exp, \ominus (red) - EDC/ANN and $\omin�$ (magenta) - EDC/ANN on EDC two-step temperature and flow field

9.2 Computational resource comparison

A brief comparative study of the computational demands by different chemistry simulations is presented in this subsection, with the aim of highlighting the advantages and drawbacks of the ANN, ISAT and DI techniques. The comparative study involved solving the Sandia flame D model for a set amount of iterations and recording the time per iteration and computer memory usage (RAM). This study was performed on a Dell Precision T5610, 2-off Intel(R) Xeon(R) CPU E5-2609 v2 @ 2.5 GHz, 48 GB memory and running Windows 7 64-bit. Tests were performed to gauge the computational resource benefits of using the ANN chemistry integrator approach based on the computer processor time and memory requirements. The Sandia flame D CFD model was solved using five different chemistry integration approaches on the mesh used in the previous simulations (cells = 2352). Each configuration was solved for 5000 iterations, where the solving time and RAM usage were to be recorded. If the simulations' residuals and monitors converged before the set amount of iterations were reached, the time to convergence was recorded and the simulations allowed to solve to 5000 iterations. A summary of the results is found in the table below.

Table 9.1: Computational performance of integration techniques: CPU time

	Iterations per second	Ratio
Direct integration	3.33	1
In-situ adaptive table	3.75	1.13
Artificial neural network	4.28	1.30
Artificial neural network chemistry solver superimposed on two-step EDC (DI/ISAT) solution	4.01	1.2
Artificial neural network chemistry solver superimposed on two-step EDM solution	4.5	1.35

Two simulations were solved using the ISAT integration procedure. The first had the ISAT error tolerance at the default value of 0.001; where the error tolerance was reduced to 1×10^{-6} for the second simulation. The tolerance is used to control the numerical error of the ISAT table and is, therefore, the maximum allowable increment of the table entries, making the tables much larger. The value of 0.001 is relatively large, which allows for faster convergence, but the magnitude should be reduced to ensure an accurate solution, according to ANSYS (2016). Decreasing the magnitude of the error tolerance resulted in the memory and time requirements to build the ISAT to increase, and the ISAT method performed slower than the DI procedure. Once the user-specified maximum storage of the ISAT is filled, Fluent[®] 17.0 will revert to DI. Table 9.1 shows that the ANN techniques outperform the DI and ISAT techniques on CPU time.

The best performing model was the one where the EDM was used to first solve the flow, turbulence and energy fields using a two-step reaction mechanism and then using ANN chemistry integrator and five-step mechanism as a post-processor to determine additional species concentrations. This is because the amount of calculations performed by the processor is minimised for the duration of the solving procedure (5000 iterations). For the "EDM" part, the model solves 3 velocity equations, 2 turbulence equations, 1 energy equation and 5 species transport equations ($3+2+1+5 = 11$ equations) without the need for integrating the chemical kinetics versus the "EDC" model, which uses the ANN integrator that solves the 3 velocity equations, 2 turbulence equations, 1 energy equation and 6 species transport equations ($3 + 2 + 1 + 6 = 12$ equations), which also requires no integration. For the second part, the "EDM" model's flow, turbulence and energy equations are frozen and thus the model only solves the 6 species transport equations and uses the ANN chemistry integrator, while the "EDC" model still solves the 12 equations. The ANN chemistry integrator also eliminates the amount of iterations performed by the CPU. For the traditional DI/ISAT methods, multiple iterations are per-

formed during the fine scales residence time, $0 \rightarrow \tau^*$, which ensures numerical stability and reduced truncation error, whereas the ANN chemistry integrator replaces these iterations by two matrix multiplications (forward propagation routine), thus reducing the amount of operations performed.

As expected, the difference between the ANN and ISAT (22% solving time reduction) is not as large as the difference between the ANN and the DI techniques (35% computational time reduction). The table below shows the RAM requirements of the ISAT (large and small tolerance settings) and ANN techniques:

Table 9.2: Computational performance of integration techniques: RAM requirements

	RAM requirements [MB]	Ratio
Direct integration	76	1
In-situ agglomeration table	152	2.0
In-situ agglomeration table (tolerance = 1×10^{-6})	820	10.78
Artificial neural network	78	1.03

Table 9.2 shows that the DI and ANN have lower RAM requirements than the ISAT technique. Additionally, if a higher degree of accuracy is required from the ISAT table, the required RAM increases drastically.

As seen in Chapter 6, figure 7.15, the real performance increase of the ANN occurs at above 10000 initial reaction samples. The mesh for the current problem, as mentioned, has only 2352 cells and the same amount of reaction sets being evaluated; therefore the current problem is quite small and the real performance benefits of the ANN not fully seen. To evaluate the performance benefits of the ANN for larger meshes, the following mesh sizes were solved over 5000 iterations: 2352, 5595, 12489 and 22500, where the RAM and CPU resource requirements were recorded for each mesh size. The figures below show the speed-up ratio over the DI technique by the ANN and ISAT methods and the RAM usage reduction over the ISAT method by the ANN and DI methods, respectively. From figures 9.12 and 9.13, it is clearly seen that the performance benefits of using the ANN are very positive, in that it has low RAM resource requirements and low CPU times. As the mesh size of the simulation was increased, the CPU time reduction ratio of the ANN over the DI increased. The same was seen for RAM usage: as the mesh size increased, the RAM usage of the ISAT method increased substantially, and the RAM reduction ratio of the ANN over the ISAT increased exponentially.

A typical industrial boiler mesh can range from $2 \times 10^6 - 40 \times 10^6$ cells, depending on the amount and velocity of the over-fire air jets, size of the furnace and whether the superheater's heat transfer is to be resolved (requiring

a fine mesh for the gas stream around the heat transfer surfaces, usually a $y^+ = 0.5$). Assuming a linear scale-up on solving time, the ANN chemistry integrator could be 180 – 3700 times faster for an industrial boiler model. In reality, the linear assumption is perhaps not that accurate since the required resources for the other computational processes (momentum, energy, radiation and species equations) also increase and become the limiting processes.

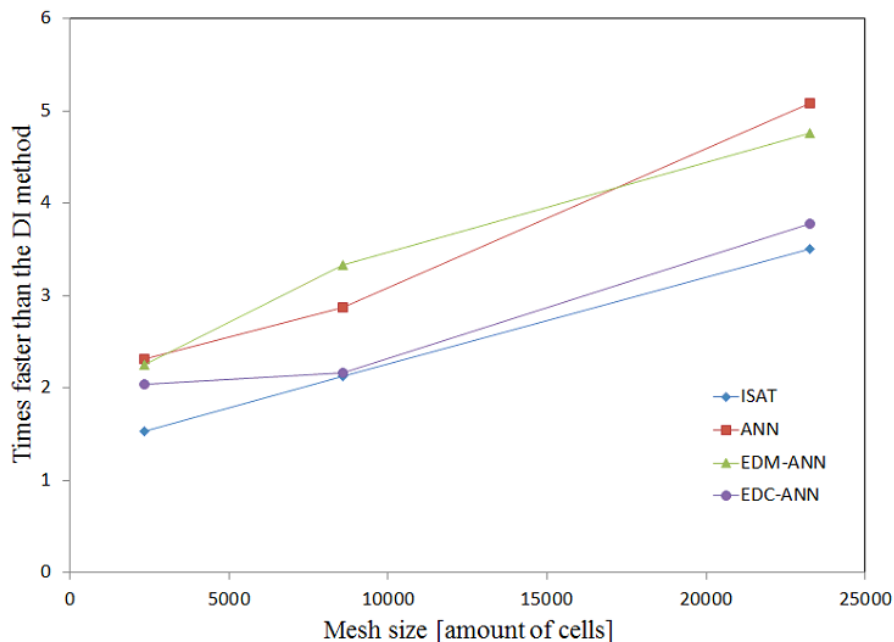


Figure 9.12: Speed-up ratio over the direct integration method. \ominus - ANN-EDC (Model 2), \ominus - EDC models using ISAT, \ominus - EDM-EDC-ANN (Model 3) and \ominus - EDC-EDC-ANN (Model 4)

Nonetheless, the large speed-up mentioned (180 – 3700) could be seen over just the DI chemistry integration procedure, which will drastically reduce solving time. With the increase in mesh size, the required computer memory also drastically increases. As an example, for an industrial biomass boiler model (12×10^6 cells) using a two-step mechanism and an available ISAT memory of 8000 MB, the ISAT table had to be cleared every iteration to achieve speed-up versus the DI method. The ANN chemistry integrator will therefore offer large memory savings compared to the ISAT technique, seeing as the computer will only have to store the network matrices in the memory of the computer. For a two-step mechanism using 6 species, this could be roughly two matrices of 7×6 and 6×6 entries.

To determine if the EDM model and the ANN integrator solves at comparable solving times, the EDM model was solved using the WD1 mechanism. Although this will not yield an accurate set of results, it will be a fair comparison between the "pure" EDM and EDC-ANN solving times. It was found that

the ANN chemistry integrator model solves at a speed comparable to the EDM model, as seen in figure 9.14. An industrial boiler model using the EDM model (two-step mechanism) solves roughly at a speed of 4.2 iterations per minute and takes about 0.83 days to converge, whereas the EDC (same two-step mechanism used in the EDM) model using ISAT solves at roughly 0.5 iterations per minute which results in about a seven day solving time before convergence is reached. Seven days is too long a period for industrial design CFD simulations where multiple design permutations must be analysed, various load cases solved and, in some cases, multiple fuels' performance determined. The EDC-ANN combustion model could offer the ability to use an advanced combustion model (EDC) with large chemical mechanisms on large scale applications and solve it with the same computational requirements as the EDM model.

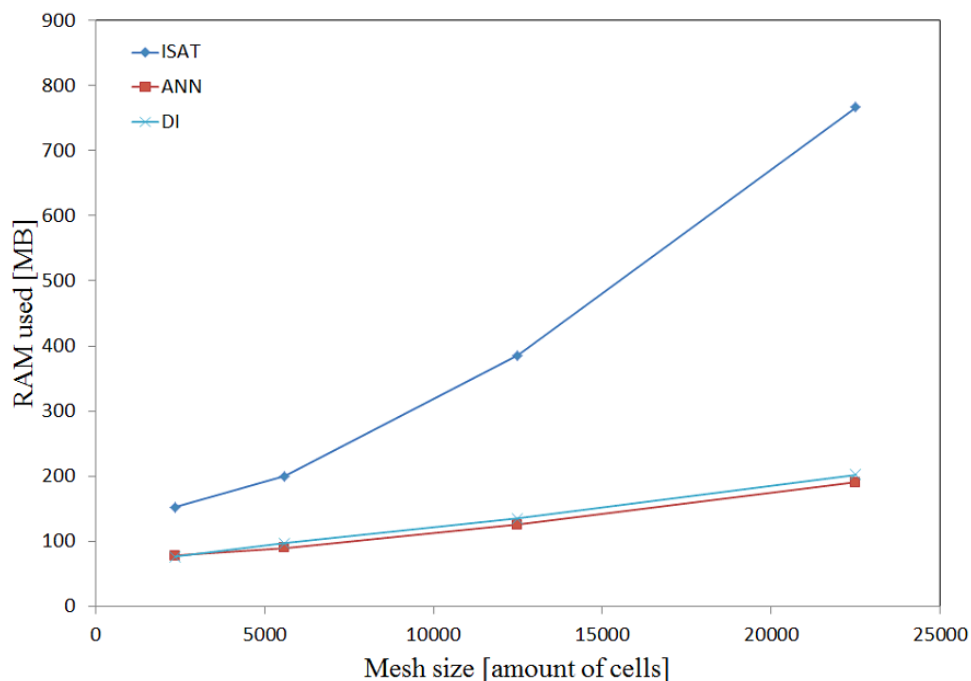


Figure 9.13: Computer memory usage for ISAT, DI and ANN models. \ominus - ANN, \oplus - DI, \ominus - ISAT

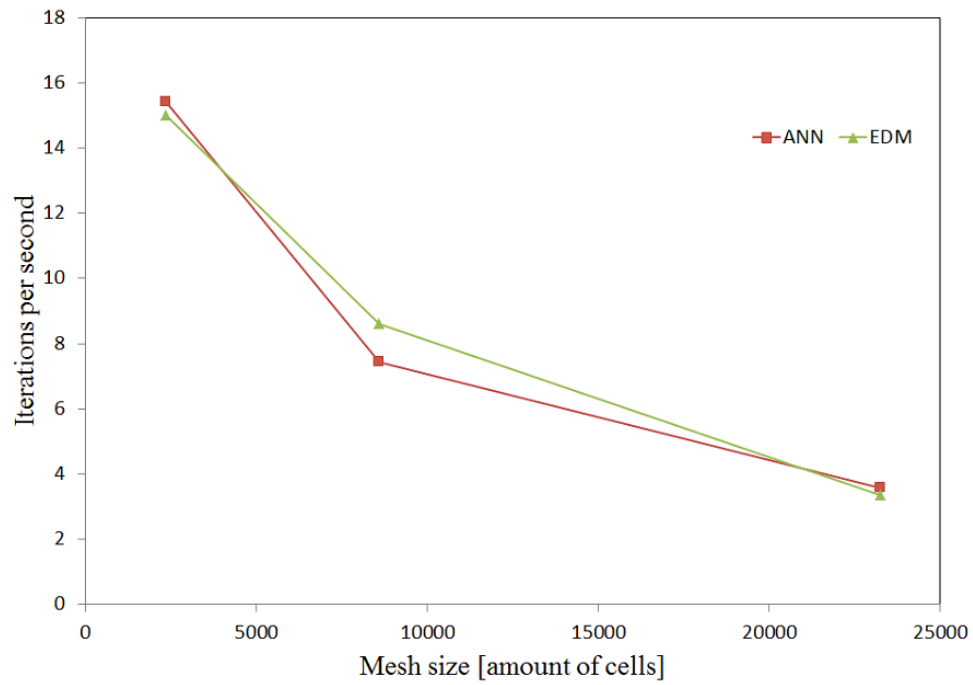


Figure 9.14: Solving time comparison between EDM and EDC-ANN combustion models for the WD1 mechanism. \ominus - ANN, \ominus - EDM

Chapter 10

Summary, conclusion and recommendations

This section will present discussion and summary of the research leading to conclusions and recommendations.

10.1 Summary and conclusions

The purpose of the project was to develop and validate a novel chemistry integration technique for the eddy dissipation concept advanced combustion model for steady-state RANS applications. The technique uses ANNs to predict the incremental changes in species mass fractions due to chemical reactions in the fine scales of the turbulent flow field. The motivation for the novel approach to solving the chemistry integration as an alternative method to what is currently employed in commercial CFD codes, was to address the high computational resource requirements of the DI and ISAT integration techniques. The new technique uses ANNs to predict the incremental species changes in the fine scale reactors, by feeding the cells' species composition and temperature to the network. By utilising a forward propagation routine, the network then calculates the species composition of the mixture that has reacted over the fine structure time scale. The new chemistry integration technique circumvents the need for high CPU time and RAM usage. The CPU time is reduced due to the elimination of time-stepping calculations for stiff ordinary differential equations. The RAM usage of the simulation is reduced because the novel approach does not require large amounts of computer memory to store the generated look-up tables. A program that simulates a plug-flow reactor was created in order to develop the new approach that models the chemical reactions in the fine scales. The program was used to generate training data from the solutions of thousands of plug-flow reactors with random initial species compositions and temperatures. The training data was in turn used to select the best -performing network architecture within the author's specified limits

and to train the networks to be able to predict the incremental species changes. A network was trained for each of the chemical time scale ranges encountered in turbulent combustion.

The ANN chemistry integrator was then implemented into commercial CFD code (Fluent[®] 17.0). The new model was validated against well-published experimental data and the results were compared to those of the traditional chemistry integration approaches.

Researching the theory governing turbulent combustion modelling and ANNs enabled the author to grasp the complexities involved and to use these methodologies to develop a novel mathematical approach. The governing transport equations (mass, momentum and energy) for combustion flows were investigated, and research showed many simplifications that could be made for subsonic combustion flow, such as negligible pressure variation, Dufour and bulk viscosity effects.

The experimental setup modelled in the present research is a methane/air piloted jet flame under fully turbulent conditions. Due to the high mean shear rate of jet flows, the standard k - ϵ turbulence model does not perform well due to over-prediction of the eddy viscosity; therefore the realizable k - ϵ turbulence approach was used to model the turbulence field.

The four main approaches to turbulence-chemistry interaction modelling were discussed, namely species transport, non-premixed combustion, premixed combustion and composition PDF transport models. The focus of the present work was on turbulent combustion modelling using the species transport approach; the eddy dissipation-finite rate hybrid model and the eddy dissipation concept model were further investigated. The major drawback of the EDM-FR model is that the tuning constants are not universally valid and the model is unable to accurately handle more than two-step chemical mechanisms, therefore limiting the applicability of the model. The EDC model allows for complex interaction of turbulence and detailed chemical kinetics. The EDC model is based on the turbulent energy cascade, therefore modelling the larger eddies breaking and forming smaller and smaller eddies. The model assumes that the species are well mixed in the smallest eddies, and that combustion is confined to these regions. These fine scale regions are then treated as small plug-flow reactors, which are integrated over the fine structure region chemical time scale to resolve the chemical mechanism's combustion reactions. For the present work, as mentioned, the purpose was to develop the new mathematical approach to solve the chemical reactions in the fine scales of the EDC model. To prove this concept, a relatively simple multi-step chemical mechanism is selected: five-step WD1 methane mechanism.

Artificial neural networks are a sub-field of machine learning and are used to map an arbitrary number of inputs to outputs, and in doing so, understand the complex relationship between the feature spaces. A neural network predicts an output from a given input observation by forward propagating the initial signal through the network. At each hidden layer, the signals are mul-

multiplied by the layer's weights and summed together before being passed into an activation or threshold function. To ensure the outputs predicted are in agreement with the training data, the back-propagation algorithm is used to tune the weights through a stochastic gradient ascent algorithm. The literature study was concluded by investigating historical work where ANNs have been used to model combustion reactions. The majority of the work was focused on using a non-premixed combustion modelling approach with ANN to predict the reaction rates. Neural networks were trained on laminar flamelet libraries and used to store and interpolate the species mass fractions, density and viscosity. The inputs to the network were the mixture fraction, mixture fraction variance and the scalar dissipation rate. The concept proposed in the current work, stores the incremental species mass fraction changes and was trained directly from the solution of the stiff ordinary differential equations used to resolve the chemical reactions in the plug-flow reactor. The proposed neural networks receive the current cell's species mass fractions, temperature and chemical reaction time and return the species mass fractions after it had reacted over the specified reaction time.

Experimental setup CFD modelling

In Chapter 5, the experimental setup was modelled using CFD. The geometry and flow simulation setup were discussed and the results presented. The CFD model results using the WD1 mechanism showed reasonable agreement to the experimental data. It was observed that the simulated CO and CH_4 oxidation reactions seemed to occur too rapidly compared to the data, due to either low activation energies or high pre-exponential factors of the reactions' Arrhenius equations and the over production of turbulence due to the two-equation models. A mixture fraction user-defined function, based on Bilger's equation, was developed and linked into Fluent[®] 17.0. The mixture fraction and heat of reaction results were used to determine the upper and lower limits of the active chemical reaction space within the complete chemical reaction compositional space. It was found that a mixture fraction range between $0.1 \rightarrow 0.6$ covers the compositional space where reactions occur.

Plug-flow reactor simulation and training data generation

Using the active mixture fraction range of $0.1 \rightarrow 0.6$ found in Chapter 5, random samples of species compositions were generated within the specified mixture fraction and temperature ranges. These samples were then solved to nearly steady state using the species and energy transport equations for a plug-flow reactor. The chemical time-step intervals of $d\tau^* = 1 \times 10^{-7}$, $d\tau^* = 1 \times 10^{-6}$, $d\tau^* = 1 \times 10^{-5}$ and $d\tau^* = 1 \times 10^{-4}$ were evaluated. The generated data was then fed into the neural networks to train on the data and adjust their weights and biases to minimise the in-sample error of the network's predictions.

Neural network architecture selection and training

In Chapter 7, the optimal architecture for the neural networks was selected by investigating the effect of activation functions, the amount of hidden layers, and the amount of neurons used. It was found that the networks using only log-sigmoid activation functions were unable to capture the incremental species changes. This is due to the small gradient of the activation function that leads to output signals of similar magnitude for input signals over a relatively large range. The research showed that using the hyperbolic-tangent sigmoid function or placing a linear activation function at the output layer increases the ability of the network to predict the incremental species changes accurately. Increasing the amount of neurons and hidden layers of the network increased the model's flexibility and thus its ability to fit the training data more accurately. Increasing the model complexity reduced the bias error of the model. One observation that was noted is that if the model flexibility is increased too much (3 hidden layers and $3 \rightarrow 7$ neurons per layer), the in-sample error of the network prediction increased. The reason for this phenomenon is that the error surface becomes more complex, and the minimisation algorithm gets stuck in local minima. The best-performing architecture selected was a single hidden layer with 7 neurons. The in-sample error for prediction of the species in the WD1 mechanism increased as the chemical reaction time-step value increased, and this showed that a more complex non-linear relationship exists between the input and output values for a larger time-step.

To minimise the generalisation error of the neural networks, it was found that the training datasets should be generated from 10000 random initial samples. The increased dataset size decreased the variance error of the prediction model. The solving time of the neural networks was also compared to that of the BDF VODE solver for a plug-flow reactor. The results showed that the neural network chemistry integrator becomes exponentially faster than the VODE solver for datasets larger than 1000 random initial samples.

CFD model results and computational performance

Comparing the artificial neural network chemistry integrator results to the experimental data and the results from Chapter 5 showed that the new approach can, with relative accuracy, predict the species mass fractions, temperatures and mixture fractions distributions through the computational domain. Along with the EDC model solved using the ANN chemistry integrator, two alternative models were utilised. One model used the flow, turbulence, species and energy field solution of the EDM combustion model with a two-step chemical mechanism. Once the EDM simulation was converged, the flow, turbulence and energy equations of the CFD model were frozen. The EDC model with the five-step mechanism was then loaded, and the EDC-ANN chemistry integrator model was solved as a post-processor. The other alternative model

worked similarly to the method just mentioned, but rather than using a two-step EDM model, a two-step EDC combustion model was used to resolve the flow, turbulence and energy fields, after which the five-step mechanism was solved using the ANN chemistry integrator. The three models compared better to the CFD model using the traditional integration approach, than to the experimental results. This was expected to be observed, as the model was developed to be an alternative chemistry integration technique to be used with the EDC model and not a completely different modelling approach; therefore the deficiencies of the chemical mechanism and turbulence model were also observed for the ANN integration models. There are relatively small differences between the results of the three modelling approaches and the CFD results from chapter 5. These small differences are due to the out-of-sample prediction error of the ANN integrator. The prediction error of the neural networks can be decreased by changing the architecture of the network accordingly. One such change is to use a network with only a single output, therefore seven neural networks each used to predict a single species with the same amount of inputs (8). This approach increases the amount of dedicated weighting variables used per predicted output variables; therefore each network is trained to handle the incremental changes of a single species (increased model flexibility per species).

The RSM turbulence model was also used in conjunction with the ANN chemistry integrator. The RSM model predicted lower turbulence in the active reaction zone around the nozzle, than the two-equation models. The lower turbulence pushed the temperature, CO and CO_2 peaks of the simulation to align closer to the experimental results.

The computational benefits of using the ANN chemistry integration approach were also evaluated. It was found that for a mesh of 2352 cells, the solving time was decreased by 35 % over the direct integration method and 22 % over the ISAT method. The reason for the decrease in solving time is that, for the direct integration approach, the information must be sent to the CPU to perform the time-stepping integration calculations for the stiff differential equations. For the neural network method, the need for time-stepping integration is eliminated, and only the forward propagation calculations are performed, which reduces the amount of calculations required to be performed by the CPU. The ISAT method uses direct integration procedures to populate the look-up table; therefore the neural network approach will also be faster. Depending on the in-situ adaptive tabulation method's error tolerance setting, the reduction in computer memory usage ranges between 100 \rightarrow 900 % if the neural network model is rather used. The ANN technique eliminates the need to store large chemistry table in the RAM of the computer and uses the trained weights and biases to interpolate the chemical kinetics equations. Thus, the only RAM requirement of the ANN is the weights and bias matrices, which are small in comparison to the ISAT tables. Increasing the mesh size further yielded larger computational performance increases as seen in figures 9.12 and

9.13. It is seen that the speed up for 22500 cells is about 200 % over the DI method.

In the present research, a model was developed that solves the combustion chemistry ODEs in the fine structure regions of the turbulence field as specified by the EDC combustion model. The novel approach has comparable accuracy in comparison to the traditional methods (DI and ISAT) used in commercial CFD codes, but at vastly reduced hardware and CPU requirements.

10.2 Recommendations and future work

As an extension to the procedures developed in the present thesis, the following research can be further investigated:

1. Model more detailed chemistry mechanisms such as the WD-mult mechanism
2. Change neural network architecture so that a network only predicts a single species. This will attempt to lower prediction error
3. Implement the novel approach on an industrial boiler problem. This work will be undertaken by the author of the current document and PC du Toit in his PhD research (DuToit, 2016).
4. Use machine learning models to develop a generic biomass devolatilisation model, based on the chemical percolation model
5. Represent a steady laminar flamelet with a neural network and use approach to model large chemical mechanisms on industrial boilers

Appendices

Appendix A

Backward-differencing variable-coefficient ordinary differential equation solver

The solution of most chemical mechanisms is quite tedious due to the stiffness of the resulting species differential equations. The stiffness is due to the large variation of the chemical time scales (inverse of the specific reaction rate constants) between the different reactions. To achieved the fastest and most stable solution the backward differentiation formula (BDF) and the variable-coefficient ordinary differential (VODE) solver was used in the solution of the species transport equations. The solution of the species equations in this section is only applicable to the plug-flow reactor equations, seeing as the plug-flow reactor was used in the artificial neural network training routines.

Consider a ordinary differential equation (ODE)

$$\frac{dy}{dt} = f(y, t), \quad y(0) = y_0 \quad (\text{A.1})$$

Where $f(y, t)$ is an n -vector function of the n -vector y where in the present research is the species mass fraction of a specific species. t is the independent variable where in the case of the present research is fine structure regions time scale τ^* , and y_0 is an n -vector of the initial condition which is the species mass fractions in the CFD cell before a chemical integration step is performed. The implicit BDF form of the ODE is

$$\frac{y^{t+1} - y^t}{\Delta t} = f(y^{t+1}) \quad (\text{A.2})$$

The formulation above is implicit because the y^{t+1} is found of both sides of the equation. The BDF formulation is said to unconditionally stable, but still suffers truncation error. For stiff problems the typical approach is to use an implicit method such as the BDF to ensure stability Law (2013). Next the Westbrook and Dryer (WD1) mechanism's species transport equations (4.39)

APPENDIX A. BACKWARD-DIFFERENCING VARIABLE-COEFFICIENT
ORDINARY DIFFERENTIAL EQUATION SOLVER 121

will be written in the BDF form (note the variables being differentiated are the species concentrations)

$$\frac{[CH_4]^{\tau+1} - [CH_4]^\tau}{\Delta\tau} = \rho(-k_1([CH_4]^{\tau+1})^{0.7}[O_2]^{0.8}) \quad (A.3)$$

$$\begin{aligned} \frac{[O_2]^{\tau+1} - [O_2]^\tau}{\Delta\tau} = \rho(-1.5k_1([O_2]^{\tau+1})^{0.8}[CH_4]^{0.7} - 0.5k_2[CO]([O_2]^{\tau+1})^{0.25}[H_2O]^{0.5} \\ + 0.5k_3[CO_2] - 0.5k_4[H_2]([O_2]^{\tau+1})^{0.5} + 0.5k_6[H_2O]) \end{aligned} \quad (A.4)$$

$$\frac{[CO]^{\tau+1} - [CO]^\tau}{\Delta\tau} = \rho(k_1[CH_4]^{0.7}[O_2]^{0.8} - k_2([CO]^{\tau+1})[O_2]^{0.25}[H_2O]^{0.5} + k_3[CO_2]) \quad (A.5)$$

$$\frac{[CO_2]^{\tau+1} - [CO_2]^\tau}{\Delta\tau} = \rho(k_2[CO][O_2]^{0.25}[H_2O]^{0.5} - k_3[CO_2]^{\tau+1}) \quad (A.6)$$

$$\frac{[H_2]^{\tau+1} - [H_2]^\tau}{\Delta\tau} = \rho(-k_4[H_2]^{\tau+1}[O_2]^{0.5}[H_2O]^{0.5} + k_5[H_2O]) \quad (A.7)$$

$$\frac{[H_2O]^{\tau+1} - [H_2O]^\tau}{\Delta\tau} = \rho(k_4[H_2][O_2]^{0.5}[H_2O]^{0.5} - k_5[H_2O]^{\tau+1}) \quad (A.8)$$

Now that the species transport equations are formulated in their respective BDF forms (A.3) - (A.8), the VODE solution technique is applied to calculate the solution over the time period $0 - \tau^*$ seconds. In the previous equations the variable $\Delta\tau$ is the step size of the numerical solution, the VODE automatically and dynamically vary the step size $\Delta\tau$ to ensure the specified error tolerance is maintained Byrne and Dean (1993). The reader is encouraged to study Byrne and Dean (1993) and Brown *et al.* (1988) for further information on the internal workings of the VODE step size adjustment routine.

Now that the species transport equations are in their discretized forms a modified Newton method along with the variable step size adjustment routine of the VODE is used to solve the system of equations and keep the error tolerance within the user specified range.

Appendix B

NASA polynomials for fluid properties

The NASA 7-coefficient polynomial parametrisation is used to calculate the species reference-state thermodynamic properties $c_p^0(T)$, $h^0(T)$ and $s^0(T)$, which is used for the CFD and plug-flow reactor simulations. The polynomial form uses 7 coefficients in two temperature regions 200 – 1000 K and 1000 – 3500 K McBride *et al.* (2002). The mentioned fluid properties can be calculated using the following equations

$$\begin{aligned}\frac{c_p^0(T)}{R} &= a_0 + a_1T + a_2T^2 + a_3T^3 + a_4T^4 \\ \frac{h^0(T)}{RT} &= a_0 + \frac{a_1}{2}T + \frac{a_2}{3}T^2 + \frac{a_3}{4}T^3 + \frac{a_4}{5}T^4 + \frac{a_5}{T} \\ \frac{s^0(T)}{R} &= a_0\ln(T) + a_1T + \frac{a_2}{2} + \frac{a_3}{3}T^3 + \frac{a_4}{4}T^4 + a_6\end{aligned}$$

The coefficient for the above equation can be found in the table below for the different species used in the methane jet combustion simulations.

Table B.1: NASA -7 polynomial coefficients

Species	a_0	a_1	a_2	a_3	a_4	a_5	a_6
H_2 [200-1000K]	2.34	7.9E-3	-1.9E-5	2.5E-8	-7.3E-12	-9.179E+2	6.830E-1
H_2 [1000-3500K]	3.3	-4.94E-5	4.9E-7	-1.79E-10	2.0E-14	-9.5E2	-3.205
O_2 [200-1000K]	3.78	-2.99E-3	9.8E-6	-9.6E-9	3.24E-12	-1.06E3	3.657
O_2 [1000-3500K]	3.28	1.48E-3	-7.57E-7	2.09E-10	-2.16E-14	-1.08E3	5.45
H_2O [200-1000K]	4.19	-2.03E-3	6.52E-6	-5.48E-9	1.771E-12	-3.02E4	-0.84
H_2O [1000-3500K]	3.03	2.1E-3	-1.64E-7	-9.7E-11	1.6E-14	-3E4	4.96
CH_4 [200-1000K]	5.14	-1.36E-2	4.91E-5	-4.84E-8	1.66E-11	-1.02E4	-4.64
CH_4 [1000-3500K]	0.074	1.33E-2	-5.73E-6	1.2E-9	-1.01E-13	-9.46E3	18.43
CO [200-1000K]	3.57	-6.10E-4	1.01E-6	9.07E-10	-9.04E-13	-1.43E4	3.508
CO [1000-3500K]	2.715	2.06E-3	-9.9E-7	2.3E-10	-2.03E-14	-1.4E4	7.81
CO_2 [200-1000K]	2.3	8.98E-3	-7.1E-6	2.45E-9	-1.4E-13	-4.8E4	9.9
CO_2 [1000-3500K]	3.8	4.4E-3	-2.2E-6	5.2E-10	-4.72E-14	-4.87E4	2.27
N_2 [200-1000K]	3.29	1.40E-3	-3.9E-6	5.64E-9	-2.44E-12	-1.02E3	3.95
N_2 [1000-3500K]	2.9	1.4E-3	-5.6E-7	1.0E-10	-6.7E-15	-9.2E2	5.98

Appendix C

Mixture fraction calculation UDF

```

1  DEFINE_ADJUST(mf_BILGER, d)
2  {
3  real Z_fu_C = 0.11684286;
4  real Z_fu_H = 0.03922422;
5  real Z_fu_O = 0.19663877;
6  real Z_ox_C = 3.19417e-5;
7  real Z_ox_H = 1.03625e-5;
8  real Z_ox_O = 0.23298543;
9  real W_C = 12.0107;
10 real W_H = 1.008;
11 real W_O = 15.994;
12 real MW_O2 = 31.999;
13 real MW_CH4 = 16.043;
14 real MW_CO2 = 44.01;
15 real MW_H2O = 18.015;
16 real MW_CO = 28.011;
17 real MW_H2 = 2.016;
18
19 real Z_C;
20 real Z_O;
21 real Z_H;
22 Thread *t;
23 cell_t c;
24
25 thread_loop_c(t, d)
26 {
27 begin_c_loop(c, t)
28 Z_C = 1.0*W_C*C_YI(c, t, 1) / MW_CO2 + 1.0*C_YI(c, t, 3)*W_C /
    MW_CH4
29 + 1.0*C_YI(c, t, 4)*W_C / MW_CO;
30 Z_H = 2.0*W_H*C_YI(c, t, 2) / MW_H2O + 4.0*C_YI(c, t, 3)*W_H /
    MW_CH4
31 + 2.0*C_YI(c, t, 5)*W_H / MW_H2;
32 Z_O = 2.0*W_O*C_YI(c, t, 0) / MW_O2 + 2.0*W_O*C_YI(c, t, 1) /
    MW_CO2 +
33 1.0*W_O*C_YI(c, t, 2) / MW_H2O + 1.0*W_O*C_YI(c, t, 4) / MW_CO;

```



```
34 C_UDMI(c, t, 0) = (2 * (Z_C - Z_ox_C) / W_C + (Z_H - Z_ox_H) / 2
    * W_H
35 + 2 * (Z_O - Z_ox_O) / W_O) / (2 * (Z_fu_C - Z_ox_C) / W_C + (
    Z_fu_H -
36 Z_ox_H) / 2 * W_H + 2 * (Z_fu_O - Z_ox_O) / W_O);
37 if(C_UDMI(c, t, 0) > 1.0){
38 C_UDMI(c, t, 0) = 1.0;
39 }
40 else if(C_UDMI(c, t, 0) < 0.0){
41 C_UDMI(c, t, 0) = 0.0;
42 }
43
44 end_c_loop(c, t)
45 }
46 }
47
```

Appendix D

Plug-flow reactor computer program

In this section the program developed to generate data by solving the ordinary differential equations of the plug-flow reactor will be discussed. In addition to solving the species and energy equation of the PFR this code also prepares the data for the neural network to train on. This section is therefore comprised of the following subsections:

1. Sample calculation
2. Code listing

D.1 Sample calculation

Generate random sample for initial condition of plug-flow reactor:

$$\begin{aligned} T &:= 1000 & YO_2 &:= 0.19664 & YN_2 &:= 0.6473 & YCH_4 &:= 0.15607 \\ YH_2 &:= 0 & YH_2O &:= 0 & YCO &:= 0 \\ YCO_2 &:= 0 \end{aligned}$$

Set reaction time step:

$$\Delta t := 1 \cdot 10^{-5}$$

Calculation constants: Molecular weights, pressure, etc.

$$MO_2 := 31.999 \quad MCO_2 := 44.01 \quad MCO := 28.011 \quad MN_2 := 28.013 \quad MH_2O := 18.015$$

$$MCH_4 := 16.043 \quad MH_2 := 2.016 \quad P := 101325 \quad R := 8314$$

Using the backward differencing formula the species and energy equations are discretized and solved, to determine the species fractions/concentrations and temperature for the next time step. The procedure is as follows:

1) Calculate the molar concentrations for the current time step:

$$M := R \cdot \left(YO_2 \cdot \frac{T}{MO_2} + YCO_2 \cdot \frac{T}{MCO_2} + YCO \cdot \frac{T}{MCO} + YCH_4 \cdot \frac{T}{MCH_4} + 0 \dots \right. \\ \left. + YN_2 \cdot \frac{T}{MN_2} + YH_2 \cdot \frac{T}{MH_2} + YH_2O \cdot \frac{T}{MH_2O} \right)$$

$$XO_2 := P \cdot \frac{\left(\frac{YO_2}{MO_2} \right)}{(M)}$$

$$XO_2 = 1.921 \times 10^{-3}$$

$$XCO_2 := P \cdot \frac{\left(\frac{YCO_2}{MCO_2} \right)}{(M)}$$

$$XCO_2 = 0$$

$$XCO := P \cdot \frac{\left(\frac{YCO}{MCO} \right)}{(M)}$$

$$XCO = 0$$

$$XCH_4 := P \cdot \frac{\left(\frac{YCH_4}{MCH_4} \right)}{(M)}$$

$$XCH_4 = 3.042 \times 10^{-3}$$

$$XH_2 := P \cdot \frac{\left(\frac{YH_2}{MH_2} \right)}{(M)}$$

$$XH_2 = 0$$

$$XH_2O := P \cdot \frac{\left(\frac{YH_2O}{MH_2O} \right)}{(M)}$$

$$XH_2O = 0$$

$$XN_2 := P \cdot \frac{\left(\frac{YN_2}{MN_2} \right)}{(M)}$$

$$XN_2 = 7.224 \times 10^{-3}$$

2) The density is then calculated from the species molar concentrations by:

$$\rho := X_{N_2} \cdot M_{N_2} + X_{H_2O} \cdot M_{H_2O} + X_{H_2} \cdot M_{H_2} + X_{CO} \cdot M_{CO} + X_{CO_2} \cdot M_{CO_2} + X_{CH_4} \cdot M_{CH_4} + X_{O_2} \cdot M_{O_2}$$

$$\rho = 0.313$$

Define the reactions constants (pre-exponential factors and activation energies):

$$\begin{array}{ll} A_1 := 5.03 \cdot 10^{11} & E_{a1} := 2 \cdot 10^8 \\ A_2 := 2.24 \cdot 10^{12} & E_{a2} := 1.703 \cdot 10^8 \\ A_3 := 5.0 \cdot 10^8 & E_{a3} := 1.703 \cdot 10^8 \\ A_4 := 5.69 \cdot 10^{11} & E_{a4} := 1.464 \cdot 10^8 \\ A_5 := 2.41 \cdot 10^{14} & E_{a5} := 3.979 \cdot 10^8 \end{array}$$

3) Using the Arrhenius rate formulation, the next time step species concentrations are calculated.

For methane, the Newton-Raphson method is used to find the value of the molar concentration for the next time step. This is because the next time step value can't be formulated as a function of other variable only, this is a result of the BDF formulation. Therefore an initial guess is made for the next time step concentration of methane:

$$X_{CH_4_2} := 0$$

Using the methane transport equation formulation the next time step value is calculated using the Newton method as:

$$X_{CH_4_2} = X_{CH_4} + \Delta t \cdot \left(-A_1 \cdot e^{\frac{-E_{a1}}{R \cdot T}} \cdot X_{CH_4_2}^{0.7} \cdot X_{O_2}^{0.8} \right)$$

$$\text{Find}(X_{CH_4_2}) = 3.042 \times 10^{-3}$$

Carbon-monoxide:

For CO (H₂O, CO₂, H₂) the next time step concentration can be written as a function of only current time step values, thus this simplifies the solution as:

$$X_{CO_2} := \frac{\left(X_{CO} + \Delta t \cdot A_1 \cdot e^{\frac{-E_{a1}}{R \cdot T}} \cdot X_{CH_4}^{0.7} \cdot X_{O_2}^{0.8} + \Delta t \cdot A_3 \cdot e^{\frac{-E_{a3}}{R \cdot T}} \cdot X_{CO_2} \right)}{1 + A_2 \cdot e^{\frac{-E_{a2}}{R \cdot T}} \cdot X_{O_2}^{0.25} \cdot X_{H_2O}^{0.5}}$$

$$X_{CO_2} = 2.086 \times 10^{-8}$$

Water-vapor:

$$XH2O_2 := \frac{\left(XH2O + A1 \cdot e^{\frac{-Ea1}{R \cdot T}} \cdot \Delta t \cdot XCH4^{0.7} \cdot XO2^{0.8} + A4 \cdot e^{\frac{-Ea4}{R \cdot T}} \cdot XH2 \cdot XO2^{0.5} \cdot \Delta t \right)}{1 + \Delta t \cdot A5 \cdot e^{\frac{-Ea5}{R \cdot T}}}$$

$$XH2O_2 = 2.086 \times 10^{-8}$$

Carbon Dioxide:

$$XCO2_2 := \frac{\left(XCO2 + A2 \cdot e^{\frac{-Ea2}{R \cdot T}} \cdot XCO \cdot XO2^{0.25} \cdot XH2O^{0.5} \right)}{1 + \Delta t \cdot A3 \cdot e^{\frac{-Ea3}{R \cdot T}}}$$

$$XCO2_2 = 0$$

Oxygen:

Similar to methane the oxygen concentration for the next time step can't be written as a function of just the other constituents. Therefore the Newton method is again applied to solve the next time step value for the oxygen concentration. First a next time step oxygen concentration is guessed:

$$XO2_2 := 0$$

Given the initial guess value the next time step oxygen concentration can be calculated using the oxygen species equation.

$$XO2_2 = XO2 + \Delta t \cdot \left(\begin{array}{l} -1.5 \cdot A1 \cdot e^{\frac{-Ea1}{R \cdot T}} \cdot XCH4^{0.7} \cdot XO2_2^{0.8} \dots \\ + -0.5 \cdot A2 \cdot e^{\frac{-Ea2}{R \cdot T}} \cdot XCO \cdot XO2_2^{0.25} \cdot XH2O^{0.5} \dots \\ + 0 + 0.5 \cdot A3 \cdot e^{\frac{-Ea3}{R \cdot T}} \cdot XCO2 - 0.5 \cdot A4 \cdot e^{\frac{-Ea4}{R \cdot T}} \cdot XH2 \cdot XO2_2^{0.5} + 0 \dots \\ + 0.5 \cdot A5 \cdot e^{\frac{-Ea5}{R \cdot T}} \cdot XH2O \end{array} \right)$$

$$\text{Find}(XO2_2) = 1.921 \times 10^{-3}$$

Hydrogen:

$$XH2_2 := \frac{\left(XH2 + \Delta t \cdot A5 \cdot e^{\frac{-Ea5}{R \cdot T}} \cdot XH2O \right)}{1 + \Delta t \cdot A4 \cdot e^{\frac{-Ea4}{R \cdot T}} \cdot XO2^{0.5}}$$

$$X_{H2_2} = 0$$

Nitrogen:

Nitrogen remains the same seeing as it is not a reactant or product in any reaction in the WD1 mechanism.

Energy equation:

4) The reaction rates can be calculated as the change in concentration over the time step.

$$\omega_{CH4} := \frac{(X_{CH4} - X_{CH4_2})}{\Delta t} \quad \omega_{CO2} := \frac{(X_{CO2} - X_{CO2_2})}{\Delta t}$$

$$\omega_{CH4} = 304.153$$

$$\omega_{CO2} = 0$$

$$\omega_{CO} := \frac{(X_{CO_2} - X_{CO})}{\Delta t} \quad \omega_{H2} := \frac{(X_{H2} - X_{H2_2})}{\Delta t}$$

$$\omega_{CO} = 2.086 \times 10^{-3}$$

$$\omega_{H2} = 0$$

$$\omega_{H2O} := \frac{(X_{H2O_2} - X_{H2O})}{\Delta t} \quad \omega_{O2} := \frac{(X_{O2} - X_{O2_2})}{\Delta t}$$

$$\omega_{H2O} = 2.086 \times 10^{-3}$$

$$\omega_{O2} = 192.13$$

5) The partial molar enthalpy (J/kmol) @ 1000K (current time step value) for each species is defined as (data from NASA-7 Polynomials):

$$h_{CO} := -88839401.3663819$$

$$h_{H2O} := -215822091.56978986$$

$$h_{CO2} := -360110669.91860056$$

$$h_{CH4} := -35948442.42485466$$

$$h_{H2} := 20686532.61228712$$

$$h_{O2} := 22706809.504715163$$

$$h_{N2} := 21469863.861560624$$

6) The specific heat of the composition is also determined using the NASA-7 polynomials:

$$C_p := 1687.539$$

7) The next time step temperature is calculated using the following relation:

$$T_2 := T + \Delta t \cdot \left[\frac{-\left(\omega_{CH4} \cdot h_{CH4} + \omega_{CO2} \cdot h_{CO2} + \omega_{CO} \cdot h_{CO} + \omega_{H2} \cdot h_{H2} \dots\right)}{\rho \cdot C_p} \right]$$

$$T_2 = 1.125 \times 10^3$$

8) Now using the new temperature (T_2) and concentrations (X_{j_2} , where j is the various species) the new density and specific heat is calculated and the process iterated again from (1).

D.2 Code listing

```

1 #Author: R Laubscher
2 #Plug-flow reactor simulation and data generation for neural
   network training
3 import cantera
4 import numpy as np
5 from scipy import integrate
6 import matplotlib.pyplot as plt
7 import neuralnetwork_varActivationFunc
8 import pickle
9 import time
10 #####
11 def save_variable(var, filename):
12 f = open('C:/Users/rymol/My Work/PhD/Neural networks/Saved NN
   memory 2/1e-6s/dt=9/%s.pckl'%filename, 'wb')
13 pickle.dump(var, f)
14 f.close()
15 #####
16 class ReactorODE(object):
17
18 def __init__(self, gas):
19 self.gas = gas
20 self.P = gas.P
21
22 def __call__(self, t, y):
23
24 self.gas.set_unnormalized_mass_fractions(y[1:])
25 self.gas.TP = y[0], self.P
26 density = self.gas.density
27
28 wdot = self.gas.net_production_rates
29 dTdt = -(np.dot(self.gas.partial_molar_enthalpies, wdot) / (density
   * self.gas.cp))
30 dYdt = wdot*self.gas.molecular_weights / density
31
32 return np.hstack((dTdt, dYdt))
33 #####
34 def rearrange(input_set, output_set):
35 rearranged_total_inputs = []
36 rearranged_total_outputs = []
37
38 for i in range(len(input_set)):
39 for j in range(len(input_set[i])):
40 rearranged_total_inputs.append(list(input_set[i][j]))
41
42 for i in range(len(output_set)):
43 for j in range(len(output_set[i])):
44 rearranged_total_outputs.append(list(output_set[i][j]))
45
46 return rearranged_total_inputs, rearranged_total_outputs
47 #####

```

```

48 def final_input_rearrange(input_set):
49 return_set = []
50 for i in range(input_set.shape[0]):
51 int_set=[]
52 for j in range(input_set.shape[1]):
53 int_set.append(input_set[i][j])
54 return_set.append(int_set)
55 return return_set
56 #####
57 def scale(testset):
58 xi = (testset-np.average(testset))/np.std(testset)
59 xii = -1 + 2*(xi-np.min(xi))/(np.max(xi)-np.min(xi))
60 sig = xii#1/(1+np.exp(-xii))
61 return sig
62 #####
63 gas = cantera.Solution('WDL.cti')
64 dt = 1e-7
65 N = 1
66 global_counter = 0
67 total_inputs = []
68 total_outputs = []
69 temp_list=[]
70 mixture_fraction_list = []
71 set_max_temp = 3000
72 set_min_temp = 300
73 set_max_time = 0.5
74 set_min_mf = 0
75 set_max_mf = 1.0
76 Z_C_fuel = 0.11684286
77 Z_O_fuel = 0.19663877
78 Z_H_fuel = 0.03922422
79 Z_C_ox = 3.19417e-5
80 Z_O_ox = 0.23298543
81 Z_H_ox = 1.03625e-5
82 Y_N2_fuel = 0.64729
83 Y_N2_ox = 0.766970811
84 W_C = 12.0107
85 W_H = 1.008
86 W_O = 15.9994
87 MW_O2 = 31.999
88 MW_CH4 = 16.043
89 MW_N2 = 28.013
90 MW_CO2 = 44.01
91 MW_CO = 28.011
92 MW_H2O = 18.015
93 MW_H2 = 2*W_H
94 #####
95 to = time.clock()
96 while global_counter < N:
97 random_temp = set_min_temp + np.random.random()*(set_max_temp-
    set_min_temp)
98 temp_list.append(random_temp)

```



```

99 mixture_fraction_random = np.random.random(1)[0]
100 mixture_fraction = set_min_mf + mixture_fraction_random*(
    set_max_mf-set_min_mf)
101 mixture_fraction_list.append(mixture_fraction)
102
103 Z_C_local = mixture_fraction * Z_C_fuel + (1 - mixture_fraction) *
    Z_C_ox
104 Z_O_local = mixture_fraction * Z_O_fuel + (1 - mixture_fraction) *
    Z_O_ox
105 Z_H_local = mixture_fraction * Z_H_fuel + (1 - mixture_fraction) *
    Z_H_ox
106 Y_CH4 = 0 + np.random.random(1)[0] * (Z_C_local * MW_CH4 / W_C)
107 Y_CO = 0 + np.random.random(1)[0] * ((Z_C_local - (W_C * Y_CH4 /
    MW_CH4)) * (MW_CO / W_C))
108 Y_CO2 = (Z_C_local - (W_C * Y_CH4 / MW_CH4) - (W_C * Y_CO / MW_CO)
    ) * (MW_CO2 / W_C)
109 Y_H2O = 0 + np.random.random(1)[0] * ((Z_H_local - (4 * W_H *
    Y_CH4 / MW_CH4)) * (MW_H2O / (2 * W_H)))
110 Y_H2 = (Z_H_local - (4 * W_H * Y_CH4 / MW_CH4) - (2 * W_H * Y_H2O
    / MW_H2O)) * (MW_H2 / (2 * W_H))
111 Y_O2 = (Z_O_local - (1 * W_O * Y_CO / MW_CO) - (2 * W_O * Y_CO2 /
    MW_CO2) - (1 * W_O * Y_H2O / MW_H2O)) * (
112 MW_O2 / (2 * W_O))
113 Y_N2 = mixture_fraction * Y_N2_fuel + (1 - mixture_fraction) *
    Y_N2_ox
114 if Y_O2 < 0:
115 Y_O2 = 0
116 sum_massfractions = Y_N2 + Y_CO + Y_CO2 + Y_CH4 + Y_H2O + Y_H2 +
    Y_O2
117 if sum_massfractions > 1.0:
118 Y_N2 = Y_N2 / sum_massfractions
119 Y_O2 = Y_O2 / sum_massfractions
120 Y_CO2 = Y_CO2 / sum_massfractions
121 Y_CO = Y_CO / sum_massfractions
122 Y_CH4 = Y_CH4 / sum_massfractions
123 Y_H2O = Y_H2O / sum_massfractions
124 Y_H2 = Y_H2 / sum_massfractions
125
126 gas.TPY = random_temp, cantera.one_atm, [Y_H2, Y_O2, Y_H2O, Y_CH4,
    Y_CO, Y_CO2, Y_N2]
127 y_initial = np.hstack((gas.T, gas.Y))
128 model_solver = integrate.ode(ReactorODE(gas))
129 model_solver.set_integrator('vode', method='bdf', with_jacobian=
    True)
130 model_solver.set_initial_value(y_initial, 0.0)
131
132 t_final = set_max_time
133 t_out = []
134 Y_out = []
135 T_out = [gas.T]
136 t_out.append(0.0)
137 Y_out.append(gas.Y)

```

```
138 counter = 0
139 convergence_criteria = 1e-6
140 while model_solver.successful() and model_solver.t < t_final:
141     model_solver.integrate(model_solver.t + dt)
142     t_out.append(model_solver.t)
143     Y_out.append(gas.Y)
144     T_out.append(gas.T)
145     error = 0.0
146     counter += 1
147     for p in range(len(Y_out[:, counter])):
148         error += (abs(Y_out[counter][p] - Y_out[counter-1][p]))**2.0
149     if error <= convergence_criteria:
150         break
151     else:
152         continue
153 Y_out = np.array(Y_out).T
154
155 inputs = np.zeros((Y_out.shape[1], Y_out.shape[0]+1))
156 outputs = np.zeros((Y_out.shape[1], Y_out.shape[0]))
157
158 for i in range(len(Y_out[0][:])):
159     dummy_inputs = []
160     dummy_outputs = []
161     dummy_inputs.append(random_temp/set_max_temp)
162     for j in range(len(Y_out[:])):
163         dummy_inputs.append(Y_out[j][i])
164         dummy_outputs.append(Y_out[j][i])
165
166     inputs[i] = dummy_inputs
167     outputs[i] = dummy_outputs
168
169 inputs = inputs[0:-1]
170 outputs = outputs[1:]
171 global_counter += 1
172 total_inputs.append(inputs)
173 total_outputs.append(outputs)
174 print(global_counter)
175 print(time.clock()-to)
176 inputs_final, outputs_final = rearrange(total_inputs, total_outputs
    )
177 #inputs_final = scale(inputs_final)
178 #inputs_final = final_input_rearrange(inputs_final)
179 inputs_final = tuple(inputs_final)
180 outputs_final = tuple(outputs_final)
```

Appendix E

Neural network programs and sample calculations

In the current appendix the working of the forward and backward-propagation programs will be discussed at the hand of program flowcharts, code explanations and sample calculations.

E.1 Forward propagation procedure

The forward propagation procedure is used to calculate the outputs given a set of inputs vectors, weights- and bias matrices.

E.1.1 Program flowchart

See figure E.1 for the program flow chart.

E.1.2 Code explanation

This section will explain how the forward propagation algorithm works.

Step 1: Initialization

For the program to be able to execute the forward propagation procedure the following inputs are required:

1. Network constructor vector $\bar{C} = [c_1, c_2, c_3, \dots, c_L]$, this vector is used to construct the network. The length of the vector corresponds to the amount of layers the network will have, including the input and output layer. The value of each vector entry is the amount of neurons that corresponding layer has.
2. The input data $\bar{X} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_N]$, where each input data observation has the length of the feature space dimension, $\mathbf{x}_n = [x_n^1, x_n^2, x_n^3, \dots, x_n^P]$.

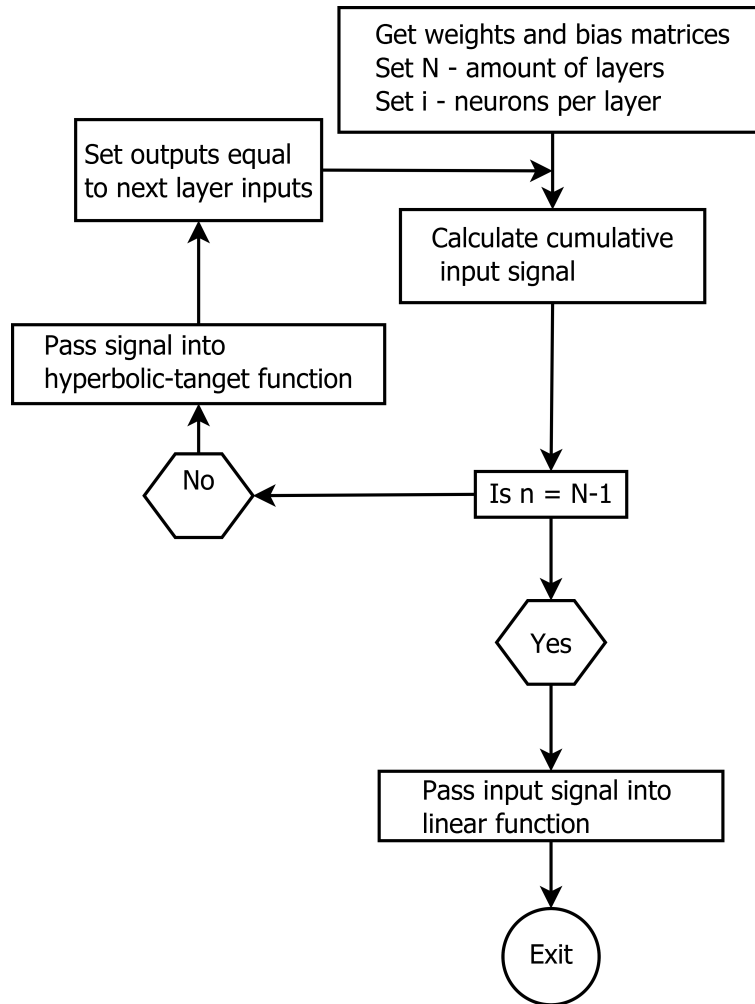


Figure E.1: Program flowchart for forward-propagation procedure

- Weights $\bar{\mathbf{W}} = [\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \mathbf{W}^{(3)}, \dots, \mathbf{W}^{(L)}]$ and bias $\bar{\mathbf{B}}$ matrices must be passed into the routine.

Step 2: Stepping through the network and calculating the output signals

Step 2a: Initialise a for loop used to step through the layers of the network. For the input layer the cumulative input signal is calculated by

$$\mathbf{s}^{(1)} = \mathbf{W}^{(1)} \cdot \mathbf{X} - \mathbf{B}^{(1)}$$

The signal is then passed into the first layer neurons' activation functions and the output signals is generated.

$$\mathbf{o}^{(1)} = \theta_{\tanh}(\mathbf{s}^{(1)})$$

Step to next layer.

Step 2b: The current layer's input signal is equated to the previous layers

output signal

$$\mathbf{x}^{(l)} = \mathbf{o}^{(l-1)}$$

If the layer stepped into is the last layer then

$$\mathbf{s}^{(L)} = \mathbf{W}^{(L)} \cdot \mathbf{x}^{(L)} - \mathbf{B}^{(L)}$$

The final layer cumulative input signal is then passed into a linear activation function (or the activation function selected for the last layer by the user),

$$\mathbf{o}^{(L)} = \theta_{lin}(\mathbf{s}^{(L)})$$

The program then writes the output to a file and exits. If the layer stepped into is not the final layer the cumulative signal is calculated by

$$s^{(l)} = \mathbf{W}^{(l)} \cdot \mathbf{x}^{(l)} - \mathbf{B}^{(l)}$$

The signal is then passed into the activation functions for the current layer.

$$\mathbf{o}^{(l)} = \theta_l(\mathbf{s}^{(l)})$$

The process then loops back to the beginning of Step 2b and repeats.

E.1.3 Sample calculation

The sample calculation is for a gas composition and temperature of: $Y_{O_2}=0.19664$, $Y_{CH_4}=0.1567$, $Y_{N_2}=0.64729$ and a temperature of 1800K.

The converged neural network weights for the 1E-7s time step network was used. For the forward propagation sample calculation the following network constructor vector, input observation weights matrices and bias matrices was used.

$$C := (8 \ 7 \ 7)$$

$$X := (0.6 \ 0 \ 0.19664 \ 0 \ 0.15607 \ 0 \ 0 \ 0.64729)$$

$$W_1 := \begin{pmatrix} 0.1277 & 0.2117 & -0.2194 & 0.0332 & 0.9569 & 0.2132 & 0.7959 & 1.050 \\ 0.037 & 0.1597 & 0.296 & 0.5822 & 1.318 & 0.597 & 0.5746 & -0.1694 \\ 0.437 & 0.292 & -0.415 & 1.004 & 0.508 & 1.328 & 0.46 & 0.252 \\ 0.485 & 0.483 & -0.1414 & 0.0562 & 0.0826 & 0.2608 & 1.55614 & 0.0399 \\ 0.8133 & 0.8961 & 0.6477 & 0.7687 & 0.9856 & 0.4517 & 0.587 & 0.47 \\ -0.06313 & -0.083578 & 0.144 & 1.468 & 0.535 & 0.05616 & 0.694 & 0.4512 \\ 0.398 & 0.511 & 0.466 & 0.5166 & 0.675 & 0.8804 & 0.757 & 1.651 \end{pmatrix}$$

$$W_2 := \begin{pmatrix} -0.0161 & 0.0388 & 0.0416 & 0.0708 & -0.0808 & -0.13 & 0.1795 \\ -0.5486 & -0.0674 & -0.5124 & 0.00287 & 0.304 & -0.0863 & 0.54813 \\ -0.3061 & -0.0596 & 0.1621 & -0.1185 & 0.0756 & 0.627 & 0.348 \\ 0.4046 & 0.7008 & -0.154 & -0.2456 & 0.097 & -0.0402 & 0.5727 \\ -0.2616 & 0.0826 & 0.5415 & -0.00997 & -0.3018 & -0.499 & 0.38 \\ -0.0419 & 0.122 & -0.167 & 0.0211 & -0.262 & 0.108 & 0.22132 \\ 0.5957 & -0.6413 & 0.032 & 0.07 - 0.2797 & 0.0732 & 0.0563 & 0.5519 \end{pmatrix}$$

$$B_1 := \begin{pmatrix} 0.62930 \\ 0.0769 \\ 0.3409 \\ 0.3857 \\ 0.8876 \\ 0.4567 \\ -0.3234 \end{pmatrix} \quad B_2 := \begin{pmatrix} 0.164 \\ 0.2343 \\ 0.2239 \\ 0.5862 \\ 0.3302 \\ 0.08311 \\ -0.01586 \end{pmatrix}$$

The first layer cumulative signal is calculated as

$$s_1 := W_1 \cdot X^T - B_1$$

$$s_1 = \begin{pmatrix} 0.233 \\ 0.1 \\ 0.082 \\ -0.084 \\ 0.186 \\ -0.091 \\ 1.828 \end{pmatrix}$$

The cumulative input signal for the first layer is then passed in the hyperbolic-tangent function:

$$o_1 := \tanh(s_1)$$

$$o_1 = \begin{pmatrix} 0.229 \\ 0.099 \\ 0.082 \\ -0.084 \\ 0.184 \\ -0.09 \\ 0.95 \end{pmatrix}$$

The output of the layer is then equated to the input of the next layer:

$$x_2 := o_1$$

Next the second layer cumulative signal is calculated as

$$s_2 := W_2 \cdot x_2 - B_2$$

$$s_2 = \begin{pmatrix} 1.026 \times 10^{-3} \\ 0.175 \\ 0.011 \\ 0.149 \\ 0.014 \\ 0.056 \\ 0.641 \end{pmatrix}$$

The cumulative signal is then passed into the linear activation function for the last layer and gives:

$$o_2 := s_2$$

$$o_2 = \begin{pmatrix} 1.026 \times 10^{-3} \\ 0.175 \\ 0.011 \\ 0.149 \\ 0.014 \\ 0.056 \\ 0.641 \end{pmatrix}$$

Therefore the species composition after reacting over a time step of 1E-7s at a initial temperature of 1800 K are:

$$Y_{H_2} := 1.026 \cdot 10^{-3} \quad Y_{O_2} := 0.175 \quad Y_{H_2O} := 0.011 \quad Y_{CH_4} := 0.149 \quad Y_{CO} := 0.014$$

$$Y_{CO_2} := 0.056 \quad Y_{N_2} := 0.641$$

E.1.4 Program listing

```

1  def predict(self, amount_testdata, test_data, neuron_constructor,
2      input_weights, input_bias):
3      self.amount_testdata = amount_testdata
4      self.test_data = test_data
5      self.neuron_constructor = neuron_constructor
6      self.input_weights = input_weights
7      self.input_bias = input_bias
8      self.test_results = []
9      if self.amount_testdata == 1:
10     for z in range(len(self.neuron_constructor)-1):
11         if z == 0:
12             input_var = np.atleast_2d(self.test_data).T
13             p = self.input_weights[z].dot(input_var) - self.input_bias[z]
14             o = self.activation(p, self.first_activation_function)
15             input_var = o
16         elif z == len(self.neuron_constructor)-2:
17             p = self.input_weights[z].dot(input_var) - self.input_bias[z]
18             o = self.activation(p, self.last_activation_function)
19             input_var = o
20         else:
21             p = self.input_weights[z].dot(input_var) - self.input_bias[z]
22             o = self.activation(p, self.act_func)
23             input_var = o
24             self.test_results.append(o)
25         else:
26             for i in range(len(self.test_data)):
27                 for z in range(len(self.neuron_constructor)-1):
28                     if z == 0:
29                         input_var = np.atleast_2d(self.test_data[i]).T
30                         p = self.input_weights[z].dot(input_var) - self.input_bias[z]
31                         o = self.activation(p, self.first_activation_function)
32                         input_var = o
33                     elif z == len(self.neuron_constructor)-2:
34                         p = self.input_weights[z].dot(input_var) - self.input_bias[z]
35                         o = self.activation(p, self.last_activation_function)
36                         input_var = o
37                     else:
38                         p = self.input_weights[z].dot(input_var) - self.input_bias[z]
39                         o = self.activation(p, self.act_func)
40                         input_var = o
41                         self.test_results.append(o)
42             return self.test_results[0]

```

E.2 Back-propagation procedure

The back-propagation procedure is used to adjust weights and biases so that a neural network can map a given set of input to its outputs, and in doing so learn the non-linear or linear relationship between the input and outputs. By

*APPENDIX E. NEURAL NETWORK PROGRAMS AND SAMPLE
CALCULATIONS*

141

understanding this relationship the network can be used to predict outputs for input data sets that it has not trained on.

E.2.1 Program flowchart

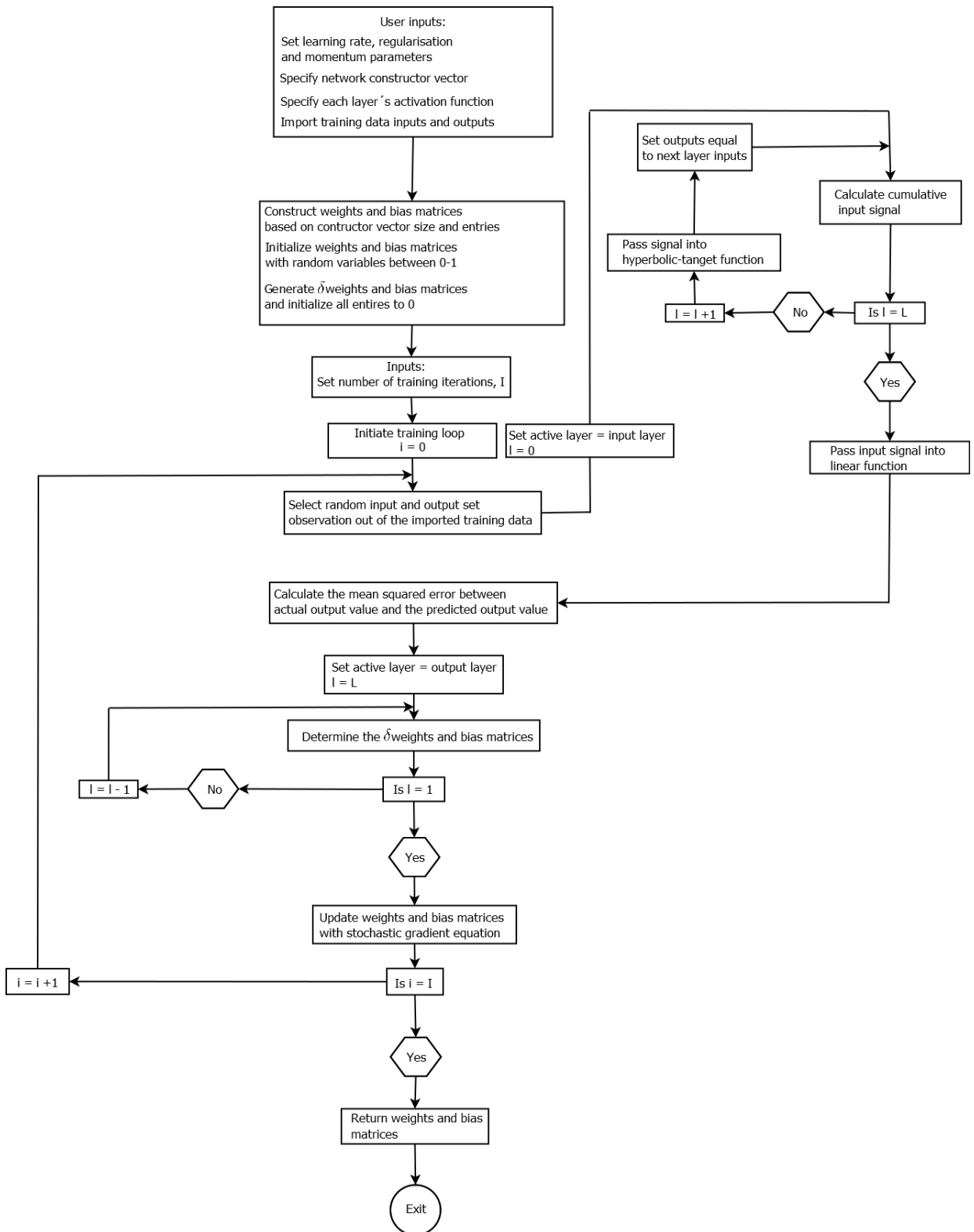


Figure E.2: Program flowchart for back-propagation procedure

E.2.2 Code explanation

Similar to the forward propagation section the procedure to calculate the updated weights with the back-propagation algorithm will be discussed in the present section.

Step 1: Required inputs

The user should specify the following inputs:

1. The learning rate parameter, η
2. Momentum parameter, γ
3. Specify network constructor vector, $\bar{\mathbf{C}}$
4. Specify which activation functions are used at different layers in the network
5. Import input and output training data sets, where the input data is defined similar to the forward propagation procedure discussed above, the output data is defined as $\bar{\mathbf{Y}} = [\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \dots, \mathbf{y}_N]$
6. Specify the amount of training iterations, I

Step 2: Select random data observation

A random observation is selected out of the training dataset. The random number is between $0 \rightarrow N$, where N is the amount of observations in the dataset.

Step 3: Perform a forward-propagation calculation

See the previous section for the code explanation.

Step 4: Calculation of the error

The performance of the current weights are gauged by calculating the mean-squared error between the predicted network outputs and the actual outputs of the dataset. The error for the i^{th} iteration and n^{th} random sample in the dataset is calculated by

$$e^i = \frac{1}{2}(\mathbf{y}_n - \mathbf{o}^{(L)})^2$$

Where in the equation above the superscript (L) designates the final layer of the network.

Step 5: Calculate of δ matrices

The program loops through each layer of the network, starting from the output layer working its way to the input layer, therefore if $l = L$ the δ matrix is calculated using the following equation

$$\delta^{(L)} = 2(\mathbf{o}^{(L)} - \mathbf{y}_n)(1 - \tanh^2(\mathbf{s}^{(L)}))$$

If the the active layer l is not equal the last layer L , the δ matrix is calculated by the following equation

$$\delta^{(l-1)} = (1 - (\mathbf{x}^{(l-1)})^2) \cdot (\mathbf{W}^{(l)}\delta^{(l)})$$

The active layer is update by $l = l - 1$ and step 5 repeated.

Step 6: Update weights and bias matrices with gradient ascent equation

Now that the δ matrices have been generated the actual weights and bias matrices can be update by the following equations

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \gamma \Delta \mathbf{W}_{i-1}^{(l)} + (1 - \gamma) \eta \delta^{(l)} \mathbf{x}^{(l-1)}$$

$$\mathbf{B}^{(l)} \leftarrow \mathbf{B}^{(l)} - \gamma \Delta \mathbf{B}_{i-1}^{(l)} + (1 - \gamma) \eta \delta^{(l)} \mathbf{x}^{(l-1)}$$

Where in the equations above the subscript $i - 1$ refers to the previous training iteration values.

Step 7: Saving the weights and bias matrices

If the number of specified iterations is reached the weights and bias matrices are saved and the program exited. If the specified iterations is not reached the procedure is repeated from step 2.

E.2.3 Sample calculation

For the back-propagation sample calculation the random input and output data sample is assumed to be

$$X := (0.6 \ 0 \ 0.19664 \ 0 \ 0.15607 \ 0 \ 0 \ 0.64729)$$

$$Y := (0.01 \ 0.165 \ 0.015 \ 0.149 \ 0.02 \ 0.075 \ 0.64729)$$

The network constructor vector is taken as:

$$C := (8 \ 7 \ 7)$$

The gradient ascent algorithm parameters (learning rate, momentum and regularisation) were set to:

$$\eta := 1.0 \quad \Upsilon := 0.0 \quad \mu := 0.0$$

Next the random weights and bias matrices are initialized for the network, specified by the network constructor vector

$$W_1 := \begin{pmatrix} 0.1277 & 0.2117 & -0.2194 & 0.0332 & 0.9569 & 0.2132 & 0.7959 & 1.050 \\ 0.037 & 0.1597 & 0.296 & 0.5822 & 1.318 & 0.597 & 0.5746 & -0.1694 \\ 0.437 & 0.292 & -0.415 & 1.004 & 0.508 & 1.328 & 0.46 & 0.252 \\ 0.485 & 0.483 & -0.1414 & 0.0562 & 0.0826 & 0.2608 & 1.55614 & 0.0399 \\ 0.8133 & 0.8961 & 0.6477 & 0.7687 & 0.9856 & 0.4517 & 0.587 & 0.47 \\ -0.06313 & -0.083578 & 0.144 & 1.468 & 0.535 & 0.05616 & 0.694 & 0.4512 \\ 0.398 & 0.511 & 0.466 & 0.5166 & 0.675 & 0.8804 & 0.757 & 1.651 \end{pmatrix}$$

$$W_2 := \begin{pmatrix} -0.0161 & 0.0388 & 0.0416 & 0.0708 & -0.0808 & -0.13 & 0.1795 \\ -0.5486 & -0.0674 & -0.5124 & 0.00287 & 0.304 & -0.0863 & 0.54813 \\ -0.3061 & -0.0596 & 0.1621 & -0.1185 & 0.0756 & 0.627 & 0.348 \\ 0.4046 & 0.7008 & -0.154 & -0.2456 & 0.097 & -0.0402 & 0.5727 \\ -0.2616 & 0.0826 & 0.5415 & -0.00997 & -0.3018 & -0.499 & 0.38 \\ -0.0419 & 0.122 & -0.167 & 0.0211 & -0.262 & 0.108 & 0.22132 \\ 0.5957 & -0.6413 & 0.032 & 0.07 & -0.2797 & 0.0732 & 0.0563 & 0.5519 \end{pmatrix}$$

$$B_1 := \begin{pmatrix} 0.62930 \\ 0.0769 \\ 0.3409 \\ 0.3857 \\ 0.8876 \\ 0.4567 \\ -0.3234 \end{pmatrix} \quad B_2 := \begin{pmatrix} 0.164 \\ 0.2343 \\ 0.2239 \\ 0.5862 \\ 0.3302 \\ 0.08311 \\ -0.01586 \end{pmatrix}$$

A forward propagation procedure is performed to determine the inputs for all the network neurons and to determine the error between the actual output data and the predicted outputs. Below is the forward propagation procedure without its descriptions, please see the previous section for more detail.

$$s_1 := W_1 \cdot X^T - B_1$$

$$s_1 = \begin{pmatrix} 0.233 \\ 0.1 \\ 0.082 \\ -0.084 \\ 0.186 \\ -0.091 \\ 1.828 \end{pmatrix}$$

$$o_1 := \tanh(s_1)$$

$$o_1 = \begin{pmatrix} 0.229 \\ 0.099 \\ 0.082 \\ -0.084 \\ 0.184 \\ -0.09 \\ 0.95 \end{pmatrix}$$

$$x_2 := o_1$$

$$s_2 := W_2 \cdot x_2 - B_2$$

$$s_2 = \begin{pmatrix} 1.026 \times 10^{-3} \\ 0.175 \\ 0.011 \\ 0.149 \\ 0.014 \\ 0.056 \\ 0.641 \end{pmatrix}$$

$$o_2 := s_2$$

$$o_2 = \begin{pmatrix} 1.026 \times 10^{-3} \\ 0.175 \\ 0.011 \\ 0.149 \\ 0.014 \\ 0.056 \\ 0.641 \end{pmatrix}$$

Next the error is calculated between the dataset values and the predicted values:

$$\text{Difference} := o_2^T - Y$$

$$e_i := 0.5 \overrightarrow{(\text{Difference} \cdot \text{Difference})}$$

$$e_i = (4.027 \times 10^{-5} \quad 5.314 \times 10^{-5} \quad 8.423 \times 10^{-6} \quad 2.468 \times 10^{-8} \quad 1.906 \times 10^{-5} \quad 1.762 \times 10^{-4} \quad 1.811 \times 10^{-5})$$

$$e_sum := 0.315 \cdot 10^{-3}$$

The δ matrix for the output layer is then calculated as

$$\delta_L := (Y^T - o_2) \cdot 1$$

$$\delta_L = \begin{pmatrix} 8.974 \times 10^{-3} \\ -0.01 \\ 4.104 \times 10^{-3} \\ -2.222 \times 10^{-4} \\ 6.175 \times 10^{-3} \\ 0.019 \\ 6.03 \times 10^{-3} \end{pmatrix}$$

The δ matrix for the L-1 layer is calculated as

$$a := W_2 \cdot \delta_L$$

$$b := 1 - o_1$$

$$\delta_1 := \overset{\longrightarrow}{(b \cdot a)}$$

$$\delta_1 = \begin{pmatrix} -1.732 \times 10^{-3} \\ -2.495 \times 10^{-3} \\ 0.012 \\ -9.47 \times 10^{-4} \\ -8.092 \times 10^{-3} \\ -6.326 \times 10^{-4} \\ 8.551 \times 10^{-4} \end{pmatrix}$$

Now that the δ matrices have been determined the weights and biases can be updated. One of the variables required for the modified weights update equation is the difference between the previous weights and bias matrices and the current weights and bias matrices, for the purpose of the sample calculation the previous weights and bias matrices are also assumed to be:

$$W_1_0 := \begin{pmatrix} 0.1277 & 0.2117 & -0.2194 & 0.0332 & 0.9569 & 0.2132 & 0.7959 & 1.050 \\ 0.037 & 0.1597 & 0.296 & 0.5822 & 1.318 & 0.597 & 0.5746 & -0.1694 \\ 0.437 & 0.292 & -0.415 & 1.004 & 0.508 & 1.328 & 0.46 & 0.252 \\ 0.485 & 0.483 & -0.1414 & 0.0562 & 0.0826 & 0.2608 & 1.55614 & 0.0399 \\ 0.8133 & 0.8961 & 0.6477 & 0.7687 & 0.9856 & 0.4517 & 0.587 & 0.47 \\ -0.06313 & -0.083578 & 0.144 & 1.468 & 0.535 & 0.05616 & 0.694 & 0.4512 \\ 0.398 & 0.511 & 0.466 & 0.5166 & 0.675 & 0.8804 & 0.757 & 1.651 \end{pmatrix}$$

$$W_{2_0} := \begin{pmatrix} -0.0161 & 0.0388 & 0.0416 & 0.0708 & -0.0808 & -0.13 & 0.1795 \\ -0.5486 & -0.0674 & -0.5124 & 0.00287 & 0.304 & -0.0863 & 0.54813 \\ -0.3061 & -0.0596 & 0.1621 & -0.1185 & 0.0756 & 0.627 & 0.348 \\ 0.4046 & 0.7008 & -0.154 & -0.2456 & 0.097 & -0.0402 & 0.5727 \\ -0.2616 & 0.0826 & 0.5415 & -0.00997 & -0.3018 & -0.499 & 0.38 \\ -0.0419 & 0.122 & -0.167 & 0.0211 & -0.262 & 0.108 & 0.22132 \\ 0.5957 & -0.6413 & 0.032 & 0.07 - 0.2797 & 0.0732 & 0.0563 & 0.5519 \end{pmatrix}$$

$$B_{1_0} := \begin{pmatrix} 0.62930 \\ 0.0769 \\ 0.3409 \\ 0.3857 \\ 0.8876 \\ 0.4567 \\ -0.3234 \end{pmatrix} \quad B_{2_0} := \begin{pmatrix} 0.164 \\ 0.2343 \\ 0.2239 \\ 0.5862 \\ 0.3302 \\ 0.08311 \\ -0.01586 \end{pmatrix}$$

Then the updated weights (W_{1_2} and W_{2_2}) and bias (B_{1_2} and B_{2_2}) matrices can be calculated as

$$W_{1_2} := W_1 + (1 - \Upsilon) \cdot \eta \cdot (\delta_{1_1} \cdot X) - \Upsilon \cdot (W_1 - W_{1_0})$$

$$W_{1_2} = \begin{pmatrix} 0.127 & 0.212 & -0.22 & 0.033 & 0.957 & 0.213 & 0.796 & 1.049 \\ 0.036 & 0.16 & 0.296 & 0.582 & 1.318 & 0.597 & 0.575 & -0.171 \\ 0.444 & 0.292 & -0.413 & 1.004 & 0.51 & 1.328 & 0.46 & 0.26 \\ 0.484 & 0.483 & -0.142 & 0.056 & 0.082 & 0.261 & 1.556 & 0.039 \\ 0.808 & 0.896 & 0.646 & 0.769 & 0.984 & 0.452 & 0.587 & 0.465 \\ -0.064 & -0.084 & 0.144 & 1.468 & 0.535 & 0.056 & 0.694 & 0.451 \\ 0.399 & 0.511 & 0.466 & 0.517 & 0.675 & 0.88 & 0.757 & 1.652 \end{pmatrix}$$

$$B_{1_2} := B_1 - (1 - \Upsilon) \cdot \eta \cdot (\delta_{1_1}) + \Upsilon \cdot (B_1 - B_{1_0})$$

$$B_{1_2} = \begin{pmatrix} 0.631 \\ 0.079 \\ 0.329 \\ 0.387 \\ 0.896 \\ 0.457 \\ -0.324 \end{pmatrix}$$

$$W_{2_2} := W_2 + (1 - \Upsilon) \cdot \eta \cdot (\delta_{L_0} \cdot o_{1_1}) - \Upsilon \cdot (W_2 - W_{2_0})$$

$$W_{2_2} = \begin{pmatrix} -9.551 \times 10^{-3} & 0.045 & 0.048 & 0.077 & -0.074 & -0.123 & 0.186 \\ -0.542 & -0.061 & -0.506 & 9.419 \times 10^{-3} & 0.311 & -0.08 & 0.555 \\ -0.3 & -0.053 & 0.169 & -0.112 & 0.082 & 0.634 & 0.355 \\ 0.411 & 0.707 & -0.147 & -0.239 & 0.104 & -0.034 & 0.579 \\ -0.255 & 0.089 & 0.548 & -3.421 \times 10^{-3} & -0.295 & -0.492 & 0.387 \\ -0.035 & 0.129 & -0.16 & 0.028 & -0.255 & 0.115 & 0.228 \\ 0.602 & -0.635 & 0.039 & -0.203 & 0.08 & 0.063 & 0.558 \end{pmatrix}$$

$$B_{2_2} := B_2 - (1 - \Upsilon) \cdot \eta \cdot (\delta_L) + \Upsilon \cdot (B_2 - B_{2_0})$$

$$B_{2_2} = \begin{pmatrix} 0.155 \\ 0.245 \\ 0.22 \\ 0.586 \\ 0.324 \\ 0.064 \\ -0.022 \end{pmatrix}$$

E.2.4 Program listing

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  class Sequential:
5  def activation(self ,x, act):
6  if act == 'tanh':
7  return np.tanh(x)
8  elif act == 'purelin':
9  return x
10 else:
11 return 1/(1+np.exp(-x))
12
13 def activation_prime(self ,x, act):
14 if act == 'tanh':
15 return 1.0 - np.power(self.activation(x, act) ,2.0)
16 elif act == 'purelin':
17 return np.multiply(x,0.0) + 1.0
18 else:
19 return self.activation(x, act)*(1-self.activation(x, act))
20
21 def performance_func(self ,d,x):
22 self.x = x
23 self.d = d
24 return -0.5*np.power((self.d-self.x) ,2)
25
26 def __init__(self ,constructor ,X,y, act_func = 'sigmoid'):
27 self.construct = constructor
28 self.X = X
29 self.y = y
30 self.weights=[]
31 self.bias=[]
32 self.delta_weights=[]
33 self.delta_bias=[]
34 self.previous_delta_weights =[]
35 self.previous_delta_bias = []
36 self.error_data=[]
37 self.error_total=[]
38 self.error_total_array=[]
39 self.iterations_data=[]
40 self.userinputs=[]
41 self.userdesired=[]
42 self.act_func = act_func
43 self.first_activation_function = 'tanh'
44 self.last_activation_function = 'purelin'
45
46
47 for i in range(1, len(self.construct)):
48 w = np.random.random((self.construct[i] ,self.construct[i-1]))
49 b = np.random.random((self.construct[i] ,1))
50 w_update = np.zeros((self.construct[i] ,self.construct[i-1]))

```

```

51  b_update = np.zeros((self.construct[i],1))
52  self.weights.append(w)
53  self.bias.append(b)
54  self.delta_weights.append(w_update)
55  self.delta_bias.append(b_update)
56  self.previous_delta_weights.append(w_update)
57  self.previous_delta_bias.append(b_update)
58
59  for i in range(len(X)):
60  self.userinputs.append(self.X[i])
61  self.userdesired.append(self.y[i])
62
63  def fit(self, alpha, beta, momentum, iterations):
64  print('Initiating training of neural network using sequential
        training algotihm...\n')
65  print('Itr\tIteration\tError\tSummed Error\n')
66  self.alpha = alpha
67  self.beta = beta
68  self.momentum = momentum
69  self.iterations = iterations
70
71  for l in range(self.iterations):
72  self.error_track=[]
73
74  for q in range(len(self.userinputs)):
75  self.inputs = []
76  self.desired = []
77  self.products = []
78  self.outputs = []
79  self.delta = []
80  self.inputs.append(self.userinputs[q])
81  self.desired.append(self.userdesired[q])
82  self.inputs[0] = np.atleast_2d(self.inputs[0])
83  self.inputs[0] = self.inputs[0].T
84
85  for i in range(len(self.construct)-1):
86
87  p = self.weights[i].dot(self.inputs[i]) - self.bias[i]
88  self.products.append(p)
89  if i == 0:
90  o = self.activation(p, self.first_activation_function)
91  if i == len(self.construct)-2:
92  o = self.activation(p, self.last_activation_function)
93  else:
94  o = self.activation(p, self.act_func)
95  self.outputs.append(o)
96  if i < len(self.construct)-2:
97  self.inputs.append(o)
98  else:
99  break
100
101  P = 0

```

```

102 error = 0
103 for i in range(len(self.outputs[-1])):
104     P += self.performance_func(np.atleast_2d(self.desired[0]).T[i],
105                               self.outputs[-1][i])
106
107 error = np.abs(P)/(len(self.outputs[-1]))
108
109 #self.delta.append((np.atleast_2d(self.desired[0]).T - self.
110                    #outputs[-1])*self.activation_prime(self.products[-1],self.
111                    #last_activation_function))
112 self.delta.append((np.atleast_2d(self.desired[0]).T - self.
113                   outputs[-1])*self.activation_prime(self.products[-1],self.
114                   last_activation_function))
115
116 for i in range(len(self.construct)-2,0,-1):
117     if i == 1:
118         self.delta.append(self.activation_prime(self.products[i-1],
119         self.first_activation_function) * np.dot(self.weights[i].T, self.
120         .delta[-1]))
121     else:
122         self.delta.append(self.activation_prime(self.products[i-1], self.
123         act_func)*np.dot(self.weights[i].T, self.delta[-1]))
124
125 self.delta.reverse()
126
127 counter = 0
128 for p in range(len(self.weights)):
129     counter += self.weights[p].shape[0]*self.weights[p].shape[1]
130
131 if q==0:
132     dummy_previous_weights = self.weights
133     previous_weights = np.zeros(len(self.construct)-1)
134 else:
135     previous_weights = dummy_previous_weights
136     dummy_previous_weights = self.weights
137
138 for i in range(len(self.construct)-1):
139     self.delta_weights[i] = (1.0 - self.momentum) * self.alpha * (
140     self.delta[i].dot(self.inputs[i].T)) + (self.beta * self.alpha)
141     * self.weights[i] - self.momentum * \
142     self.previous_delta_weights[
143     i]
144     self.delta_bias[i] = -(1.0 - self.momentum) * self.alpha * (self.
145     .delta[i]) - self.momentum * \
146     self.previous_delta_bias[
147     i]
148     self.weights[i] += (1.0 - self.momentum) * self.alpha * (self.
149     delta[i].dot(self.inputs[i].T)) + (
150     self.beta * self.alpha) * \
151     self.weights[
152     i] - self.momentum * \
153     self.previous_delta_weights[

```

```

143     i]
144     self.bias[i] += -(1.0 - self.momentum) * self.alpha * (self.
145         delta[i]) - self.momentum * \
146     self.previous_delta_bias[i]
147     self.previous_delta_weights[i] = self.delta_weights[i]
148     self.previous_delta_bias[i] = self.delta_bias[i]
149
150     self.error_track.append(error)
151
152     self.error_total = sum(self.error_track)
153
154     if l % 10 == 0:
155         print('%r\t%E'%(l, error))
156
157     if l % 10 == 0:
158         self.error_data.append(error)
159         self.error_total_array.append(self.error_total)
160         self.iterations_data.append(l)
161
162     print('Checking final results...')
163
164     self.return_outputs=[]
165     for i in range(len(self.X)):
166         for z in range(len(self.construct)-1):
167             if z == 0:
168                 input_var = np.atleast_2d(self.X[i]).T
169                 p = self.weights[z].dot(input_var) - self.bias[z]
170                 o = self.activation(p, self.first_activation_function)
171                 input_var = o
172             elif z == len(self.construct)-2:
173                 p = self.weights[z].dot(input_var) - self.bias[z]
174                 o = self.activation(p, self.last_activation_function)
175                 input_var = o
176             else:
177                 p = self.weights[z].dot(input_var) - self.bias[z]
178                 o = self.activation(p, self.act_func)
179                 input_var = o
180                 self.return_outputs.append(o)
181
182     print(self.return_outputs)
183     print('Procedures successfully completed')
184
185     return self.weights, self.bias
186
187     def plot_error_convergence(self):
188         plt.figure(1)
189         plt.plot(self.iterations_data, self.error_data, c='black', lw=0.25,
190             label='Iteration error')
191         #plt.plot(self.iterations_data, self.error_total_array, c='blue',
192             label='Summed error')
193         #plt.axis([0, self.iterations, 0, 0.1])
194         plt.yscale('log')

```

```

192 plt.title('Least squares error convergence of each iteration')
193 plt.xlabel('Weight changing cycles')
194 plt.ylabel('Iteration Error')
195 plt.legend()
196 plt.grid()
197 plt.show()
198 return self.iterations_data, self.error_data
199
200 def predict(self, amount_testdata, test_data, neuron_constructor,
201            input_weights, input_bias):
202     self.amount_testdata = amount_testdata
203     self.test_data = test_data
204     self.neuron_constructor = neuron_constructor
205     self.input_weights = input_weights
206     self.input_bias = input_bias
207     self.test_results = []
208     if self.amount_testdata == 1:
209         for z in range(len(self.neuron_constructor)-1):
210             if z == 0:
211                 input_var = np.atleast_2d(self.test_data).T
212                 p = self.input_weights[z].dot(input_var) - self.input_bias[z]
213                 o = self.activation(p, self.first_activation_function)
214                 input_var = o
215             elif z == len(self.neuron_constructor)-2:
216                 p = self.input_weights[z].dot(input_var) - self.input_bias[z]
217                 o = self.activation(p, self.last_activation_function)
218                 input_var = o
219             else:
220                 p = self.input_weights[z].dot(input_var) - self.input_bias[z]
221                 o = self.activation(p, self.act_func)
222                 input_var = o
223             self.test_results.append(o)
224         else:
225             for i in range(len(self.test_data)):
226                 for z in range(len(self.neuron_constructor)-1):
227                     if z == 0:
228                         input_var = np.atleast_2d(self.test_data[i]).T
229                         p = self.input_weights[z].dot(input_var) - self.input_bias[z]
230                         o = self.activation(p, self.first_activation_function)
231                         input_var = o
232                     elif z == len(self.neuron_constructor)-2:
233                         p = self.input_weights[z].dot(input_var) - self.input_bias[z]
234                         o = self.activation(p, self.last_activation_function)
235                         input_var = o
236                     else:
237                         p = self.input_weights[z].dot(input_var) - self.input_bias[z]
238                         o = self.activation(p, self.act_func)
239                         input_var = o
240                     self.test_results.append(o)
241                 return self.test_results[0]
242     class Random(Sequential):

```

```

243
244 def fit(self, alpha, beta, momentum, iterations):
245     print('Initiating training of neural network using sequential
246           training algorithm...\n')
247     print('Itr\tIteration Error\tSummed Error\n')
248     self.alpha = alpha
249     self.beta = beta
250     self.momentum = momentum
251     self.iterations = iterations
252
253     for l in range(self.iterations):
254         self.error_track = []
255         self.inputs = []
256         self.desired = []
257         self.products = []
258         self.outputs = []
259         self.delta = []
260         randomnumber = np.random.randint(len(self.X))
261         self.inputs.append(self.userinputs[randomnumber])
262         self.desired.append(self.userdesired[randomnumber])
263         self.inputs[0] = np.atleast_2d(self.inputs[0])
264         self.inputs[0] = self.inputs[0].T
265
266         for i in range(len(self.construct)-1):
267             p = self.weights[i].dot(self.inputs[i]) - self.bias[i]
268             self.products.append(p)
269             if i == 0:
270                 o = self.activation(p, self.first_activation_function)
271             if i == len(self.construct)-2:
272                 o = self.activation(p, self.last_activation_function)
273             else:
274                 o = self.activation(p, self.act_func)
275             self.outputs.append(o)
276             if i < len(self.construct)-2:
277                 self.inputs.append(o)
278             else:
279                 break
280
281         P = 0
282         error = 0
283         for i in range(len(self.outputs[-1])):
284             P += self.performance_func(np.atleast_2d(self.desired[0]).T[i],
285                                     self.outputs[-1][i])
286
287         error = np.abs(P)
288
289         self.delta.append((np.atleast_2d(self.desired[0]).T - self.
290                           outputs[-1])*self.activation_prime(self.products[-1], self.
291                           last_activation_function))
292         for i in range(len(self.construct)-2,0,-1):
293             if i ==1:

```

```

291 self.delta.append(self.activation_prime(self.products[i - 1],
    self.first_activation_function) * np.dot(self.weights[i].T, self
    .delta[-1]))
292 else:
293 self.delta.append(self.activation_prime(self.products[i - 1], self.
    act_func)*np.dot(self.weights[i].T, self.delta[-1]))
294
295 self.delta.reverse()
296
297 counter = 0
298 for p in range(len(self.weights)):
299 counter += self.weights[p].shape[0]*self.weights[p].shape[1]
300
301 for i in range(len(self.construct)-1):
302 self.delta_weights[i] = (1.0-self.momentum)*self.alpha * (self.
    delta[i].dot(self.inputs[i].T)) + (self.beta * self.alpha) *
    self.weights[i] - self.momentum*self.previous_delta_weights[i]
303 self.delta_bias[i] = -(1.0-self.momentum)*self.alpha * (self.
    delta[i]) - self.momentum*self.previous_delta_bias[i]
304 self.weights[i] += (1.0-self.momentum)*self.alpha*(self.delta[i]
    .dot(self.inputs[i].T)) + (self.beta*self.alpha)*self.weights[i]
    - self.momentum*self.previous_delta_weights[i]
305 self.bias[i] += -(1.0-self.momentum)*self.alpha*(self.delta[i])
    - self.momentum*self.previous_delta_bias[i]
306 self.previous_delta_weights[i] = self.delta_weights[i]
307 self.previous_delta_bias[i] = self.delta_bias[i]
308
309 self.error_track.append(error)
310
311 self.error_total = sum(self.error_track)
312
313 if l % 100 == 0:
314 print('%r\t%E'%(l, error))
315
316 if l % 100 == 0:
317 self.error_data.append(error)
318 self.error_total_array.append(self.error_total)
319 self.iterations_data.append(l)
320
321 print('Checking final results...')
322
323 self.return_outputs=[]
324 for i in range(len(self.X)):
325 for z in range(len(self.construct)-1):
326 if z == 0:
327 input_var = np.atleast_2d(self.X[i]).T
328 p = self.weights[z].dot(input_var) - self.bias[z]
329 o = self.activation(p, self.first_activation_function)
330 input_var = o
331 else:
332 if z == len(self.construct)-2:
333 p = self.weights[z].dot(input_var) - self.bias[z]

```


APPENDIX E. NEURAL NETWORK PROGRAMS AND SAMPLE
CALCULATIONS

157

```
334 o = self.activation(p, self.last_activation_function)
335 else:
336 p = self.weights[z].dot(input_var) - self.bias[z]
337 o = self.activation(p, self.act_func)
338 input_var = o
339 self.return_outputs.append(o)
340
341 print(self.return_outputs)
342 print('Procedures successfully completed')
343 return self.weights, self.bias
344
```

Appendix F

Artificial neural network implementation code

In this appendix only some of the ANN-chemistry integrator functions which is used to implement the techniques in Fluent[®] 17.0 will be shown due to space constraints.

F.1 Neural network main function

```

1 DEFINE_ADJUST(NN_fineScaleReaction ,d)
2 {
3   Thread *t;
4   cell_t c;
5
6   double speciesTempArray [8] = { 0.0 };
7   double chemTime = 0.0;
8   double finescaleSpeciesArray [7] = { 0.0 };
9   int outputSize = 7;
10  int i = 0;
11  double psi = 0.0;
12  double speciesSource [7] = { 0.0 };
13  double speciesCellAverage = 0.0;
14  double kinv=0.0;
15  double Re_t = 0.0;
16  double a = 0.0;
17  double b = 0.0;
18  double cc = 0.0;
19
20  thread_loop_c(t , d)
21  {
22    begin_c_loop(c , t)
23    {
24      //ASSIGN CELL SPECIES TO ARRAY AND WEIGHT TEMPERATURE WITH MAX
        TEMP USED IN NN TRAINING
25    speciesTempArray [0] = C_T(c , t) / setMaxTemp;
26    speciesTempArray [1] = C_YI(c , t , 5); //H2

```

APPENDIX F. ARTIFICIAL NEURAL NETWORK IMPLEMENTATION
CODE

159

```

27 speciesTempArray [2] = C_YI(c, t, 0); //O2
28 speciesTempArray [3] = C_YI(c, t, 2); //H2O
29 speciesTempArray [4] = C_YI(c, t, 3); //CH4
30 speciesTempArray [5] = C_YI(c, t, 4); //CO
31 speciesTempArray [6] = C_YI(c, t, 1); //CO2
32 speciesTempArray [7] = C_YI(c, t, 6); //N2
33
34 //CHEMICAL TIME SCALE CALCULATION
35 if (C_D(c, t) < 0.001) {
36 chemTime = 0.4082*pow((C_MU_L(c, t) / C_R(c, t)) / 0.001, 0.5);
37 }
38 else {
39 chemTime = 0.4082*pow((C_MU_L(c, t) / C_R(c, t)) / C_D(c, t), 0.5)
40 ;
41 }
42 //FINE SCALE VOLUME CALCULATION
43 kinv = (C_MU_L(c, t) / C_R(c, t));
44 Re_t = (pow(C_K(c, t), 2.0) / (kinv*C_D(c, t)));
45 if (Re_t <= 64.0) {
46 Re_t = 64.0;
47 }
48 else {
49 Re_t = (pow(C_K(c, t), 2.0) / (kinv*C_D(c, t)));
50 }
51 psi = c_xi*pow(Re_t, -0.25);
52 //NEURAL NETWORK FINE SCALES CALCULATION:
53 if ((C_UDMI(c, t, 0) < MF) && (C_UDMI(c, t, 0) > MF_min)) {
54 finescaleSpeciesArray [0] = neuralnetwork(speciesTempArray,
55 chemTime) [0];
56 finescaleSpeciesArray [1] = neuralnetwork(speciesTempArray,
57 chemTime) [1];
58 finescaleSpeciesArray [2] = neuralnetwork(speciesTempArray,
59 chemTime) [2];
60 finescaleSpeciesArray [3] = neuralnetwork(speciesTempArray,
61 chemTime) [3];
62 finescaleSpeciesArray [4] = neuralnetwork(speciesTempArray,
63 chemTime) [4];
64 finescaleSpeciesArray [5] = neuralnetwork(speciesTempArray,
65 chemTime) [5];
66 finescaleSpeciesArray [6] = neuralnetwork(speciesTempArray,
67 chemTime) [6];
68 }
69 else {
70 finescaleSpeciesArray [0] = speciesTempArray [1];
71 finescaleSpeciesArray [1] = speciesTempArray [2];
72 finescaleSpeciesArray [2] = speciesTempArray [3];
73 finescaleSpeciesArray [3] = speciesTempArray [4];
74 finescaleSpeciesArray [4] = speciesTempArray [5];
75 finescaleSpeciesArray [5] = speciesTempArray [6];
76 finescaleSpeciesArray [6] = speciesTempArray [7];
77 }

```

```

71
72 C_UDMI(c, t, 1) = finescaleSpeciesArray [0]; //H2
73 C_UDMI(c, t, 2) = finescaleSpeciesArray [1]; //O2
74 C_UDMI(c, t, 3) = finescaleSpeciesArray [2]; //H2O
75 C_UDMI(c, t, 4) = finescaleSpeciesArray [3]; //CH4
76 C_UDMI(c, t, 5) = finescaleSpeciesArray [4]; //CO
77 C_UDMI(c, t, 6) = finescaleSpeciesArray [5]; //CO2
78 C_UDMI(c, t, 7) = finescaleSpeciesArray [6]; //N2
79
80 }
81 end_c_loop(c, t)
82 }
83 }

```

F.2 Species source macros

```

1 DEFINE_SOURCE(h2_source, c, t, dS, eqn)
2 {
3 real xc[ND_ND];
4 double psi = 0.0;
5 double chemTime = 0.0;
6 double speciesSource = 0.0;
7 double speciesCellAverage = 0.0;
8 double Re_t = 0.0;
9 double kinv = 0.0;
10
11 if (C_D(c, t) < 0.001) {
12 chemTime = 0.4082*pow((C_MU_L(c, t) / C_R(c, t)) / 0.001, 0.5);
13 }
14 else {
15 chemTime = 0.4082*pow((C_MU_L(c, t) / C_R(c, t)) / C_D(c, t), 0.5)
16 ;
17 }
18 //FINE SCALE VOLUME CALCULATION
19 kinv = (C_MU_L(c, t) / C_R(c, t));
20 Re_t = (pow(C_K(c, t), 2.0) / (kinv*C_D(c, t)));
21 if (Re_t <= 64.0) {
22 Re_t = 64.0;
23 }
24 else {
25 Re_t = (pow(C_K(c, t), 2.0) / (kinv*C_D(c, t)));
26 }
27 psi = c_xi*pow(Re_t, -0.25);
28 C_CENTROID(xc, c, t);
29 if (((C_UDMI(c, t, 0) < MF) && (C_UDMI(c, t, 0) > MF_min) && xc[0]
30 <= x_max && xc[0] >= x_min )) /*|| ((xc[0] < x_min) && (xc[1] >
31 y_max) && (xc[1] <=0.009))*/ {
32
33 speciesCellAverage = pow(psi, 3.0)*C_UDMI(c, t, 1) + (1 - pow(psi,
34 3.0)) * C_YI(c,t,5);
35 speciesSource = ((C_R(c, t)*pow(psi, 2.0)) / (chemTime*(1 - pow(
36 psi, 3.0))))*(C_UDMI(c, t, 1) - C_YI(c,t,5))/*speciesCellAverage

```

APPENDIX F. ARTIFICIAL NEURAL NETWORK IMPLEMENTATION
CODE

161

```

    */);
32 }
33 else {
34 speciesSource = 0.0;
35 }
36
37 C_UDMI(c, t, 8) = speciesSource;
38 dS[eqn] = -((C_R(c, t)*pow(psi, 2.0)) / (chemTime*(1 - pow(psi,
    3.0)))));
39 return speciesSource;
40 }
41
42 DEFINE_SOURCE(o2_source, c, t, dS, eqn)
43 {
44 real xc[ND_ND];
45 double psi = 0.0;
46 double chemTime = 0.0;
47 double speciesSource = 0.0;
48 double speciesCellAverage = 0.0;
49 double Re_t = 0.0;
50 double kinv = 0.0;
51
52 if (C_D(c, t) < 0.001) {
53 chemTime = 0.4082*pow((C_MU_L(c, t) / C_R(c, t)) / 0.001, 0.5);
54 }
55 else {
56 chemTime = 0.4082*pow((C_MU_L(c, t) / C_R(c, t)) / C_D(c, t), 0.5)
    ;
57 }
58 //FINE SCALE VOLUME CALCULATION
59 kinv = (C_MU_L(c, t) / C_R(c, t));
60 Re_t = (pow(C_K(c, t), 2.0) / (kinv*C_D(c, t)));
61 if (Re_t <= 64.0) {
62 Re_t = 64.0;
63 }
64 else {
65 Re_t = (pow(C_K(c, t), 2.0) / (kinv*C_D(c, t)));
66 }
67 psi = c_xi*pow(Re_t, -0.25);
68 C_CENTROID(xc, c, t);
69 if (((C_UDMI(c, t, 0) < MF) && (C_UDMI(c, t, 0) > MF_min) && xc[0]
    <= x_max && xc[0] >= x_min )) /*|| ((xc[0] < x_min) && (xc[1] >
    y_max) && (xc[1] <=0.009))*/ {
70
71 speciesCellAverage = pow(psi, 3.0)*C_UDMI(c, t, 2) + (1 - pow(psi,
    3.0)) * C_YI(c, t, 0);
72 speciesSource = ((C_R(c, t)*pow(psi, 2.0)) / (chemTime*(1 - pow(
    psi, 3.0))))*(C_UDMI(c, t, 2) - C_YI(c, t, 0)/*
    speciesCellAverage*/);
73 if (speciesSource > 0.0) {
74 speciesSource = 0.0;
75 }

```

APPENDIX F. ARTIFICIAL NEURAL NETWORK IMPLEMENTATION
CODE

162

```

76 }
77 else {
78 speciesSource = 0.0;
79 }
80
81 C_UDMI(c, t, 9) = speciesSource;
82 dS[eqn] = -((C_R(c, t)*pow(psi, 2.0)) / (chemTime*(1 - pow(psi,
      3.0)))));
83 return speciesSource;
84 }
85
86 DEFINE_SOURCE(h2o_source, c, t, dS, eqn)
87 {
88 real xc[ND_ND];
89 double psi = 0.0;
90 double chemTime = 0.0;
91 double speciesSource = 0.0;
92 double speciesCellAverage = 0.0;
93 double Re_t = 0.0;
94 double kinv = 0.0;
95
96 if (C_D(c, t) < 0.001) {
97 chemTime = 0.4082*pow((C_MU_L(c, t) / C_R(c, t)) / 0.001, 0.5);
98 }
99 else {
100 chemTime = 0.4082*pow((C_MU_L(c, t) / C_R(c, t)) / C_D(c, t), 0.5)
      ;
101 }
102 //FINE SCALE VOLUME CALCULATION
103 kinv = (C_MU_L(c, t) / C_R(c, t));
104 Re_t = (pow(C_K(c, t), 2.0) / (kinv*C_D(c, t)));
105 if (Re_t <= 64.0) {
106 Re_t = 64.0;
107 }
108 else {
109 Re_t = (pow(C_K(c, t), 2.0) / (kinv*C_D(c, t)));
110 }
111 psi = c_xi*pow(Re_t, -0.25);
112 C_CENTROID(xc, c, t);
113 if (((C_UDMI(c, t, 0) < MF) && (C_UDMI(c, t, 0) > MF_min) && xc[0]
      <= x_max && xc[0] >= x_min )) /* || ((xc[0] < x_min) && (xc[1] >
      y_max) && (xc[1] <=0.009))*/ {
114
115 speciesCellAverage = pow(psi, 3.0)*C_UDMI(c, t, 3) + (1 - pow(psi,
      3.0)) * C_YI(c, t, 2);
116 speciesSource = ((C_R(c, t)*pow(psi, 2.0)) / (chemTime*(1 - pow(
      psi, 3.0))))*(C_UDMI(c, t, 3) - C_YI(c, t, 2)/*
      speciesCellAverage*/);
117 if (speciesSource < 0.0) {
118 speciesSource = 0.0;
119 }
120 }

```

APPENDIX F. ARTIFICIAL NEURAL NETWORK IMPLEMENTATION
CODE

163

```

121 else {
122 speciesSource = 0.0;
123 }
124
125 C_UDMI(c, t, 10) = speciesSource;
126 dS[eqn] = -((C_R(c, t)*pow(psi, 2.0)) / (chemTime*(1 - pow(psi,
127 3.0)))));
128 return speciesSource;
129 }
130 DEFINE_SOURCE(ch4_source, c, t, dS, eqn)
131 {
132 real xc[ND_ND];
133 double psi = 0.0;
134 double chemTime = 0.0;
135 int i = 0;
136 double speciesSource = 0.0;
137 double speciesCellAverage = 0.0;
138 double Re_t = 0.0;
139 double kinv = 0.0;
140
141 if (C_D(c, t) < 0.001) {
142 chemTime = 0.4082*pow((C_MU_L(c, t) / C_R(c, t)) / 0.001, 0.5);
143 }
144 else {
145 chemTime = 0.4082*pow((C_MU_L(c, t) / C_R(c, t)) / C_D(c, t), 0.5)
146 ;
147 }
148 //FINE SCALE VOLUME CALCULATION
149 kinv = (C_MU_L(c, t) / C_R(c, t));
150 Re_t = (pow(C_K(c, t), 2.0) / (kinv*C_D(c, t)));
151 if (Re_t <= 64.0) {
152 Re_t = 64.0;
153 }
154 else {
155 Re_t = (pow(C_K(c, t), 2.0) / (kinv*C_D(c, t)));
156 }
157 psi = c_xi*pow(Re_t, -0.25);
158 C_CENTROID(xc, c, t);
159 if (((C_UDMI(c, t, 0) < MF) && (C_UDMI(c, t, 0) > MF_min) && xc[0]
160 <= x_max && xc[0] >= x_min )) /*|| ((xc[0] < x_min) && (xc[1] >
161 y_max) && (xc[1] <=0.009))*/ {
162 speciesCellAverage = pow(psi, 3.0)*C_UDMI(c, t, 4) + (1 - pow(psi,
163 3.0)) * C_YI(c, t, 3);
164 speciesSource = ((C_R(c, t)*pow(psi, 2.0)) / (chemTime*(1 - pow(
165 psi, 3.0))))*(C_UDMI(c, t, 4) - C_YI(c, t, 3)/*
166 speciesCellAverage*/);
167 if (speciesSource > 0.0) {
168 speciesSource = 0.0;
169 }
170 }

```

APPENDIX F. ARTIFICIAL NEURAL NETWORK IMPLEMENTATION
CODE

164

```

166 else {
167 speciesSource = 0.0;
168 }
169
170 C_UDMI(c, t, 11) = speciesSource;
171 dS[eqn] = -((C_R(c, t)*pow(psi, 2.0)) / (chemTime*(1 - pow(psi,
    3.0)))));
172 return speciesSource;
173 }
174
175 DEFINE_SOURCE(co_source, c, t, dS, eqn)
176 {
177 real xc[ND_ND];
178 double psi = 0.0;
179 double chemTime = 0.0;
180 int i = 0;
181 double speciesSource = 0.0;
182 double speciesCellAverage = 0.0;
183 double Re_t = 0.0;
184 double kinv = 0.0;
185
186 if (C_D(c, t) < 0.001) {
187 chemTime = 0.4082*pow((C_MU_L(c, t) / C_R(c, t)) / 0.001, 0.5);
188 }
189 else {
190 chemTime = 0.4082*pow((C_MU_L(c, t) / C_R(c, t)) / C_D(c, t), 0.5)
    ;
191 }
192 //FINE SCALE VOLUME CALCULATION
193 kinv = (C_MU_L(c, t) / C_R(c, t));
194 Re_t = (pow(C_K(c, t), 2.0) / (kinv*C_D(c, t)));
195 if (Re_t <= 64.0) {
196 Re_t = 64.0;
197 }
198 else {
199 Re_t = (pow(C_K(c, t), 2.0) / (kinv*C_D(c, t)));
200 }
201 psi = c_xi*pow(Re_t, -0.25);
202 C_CENTROID(xc, c, t);
203 if (((C_UDMI(c, t, 0) < MF) && (C_UDMI(c, t, 0) > MF_min) && xc[0]
    <= x_max && xc[0] >= x_min )) /*|| ((xc[0] < x_min) && (xc[1] >
    y_max) && (xc[1] <=0.009))*/ {
204
205 speciesCellAverage = pow(psi, 3.0)*C_UDMI(c, t, 5) + (1 - pow(psi,
    3.0)) * C_YI(c, t, 4);
206 speciesSource = ((C_R(c, t)*pow(psi, 2.0)) / (chemTime*(1 - pow(
    psi, 3.0))))*(C_UDMI(c, t, 5) - C_YI(c, t, 4)/*
    speciesCellAverage*/);
207
208 }
209 else {
210 speciesSource = 0.0;

```


APPENDIX F. ARTIFICIAL NEURAL NETWORK IMPLEMENTATION
CODE

165

```

211 }
212
213 C_UDMI(c, t, 12) = speciesSource;
214 dS[eqn] = -((C_R(c, t)*pow(psi, 2.0)) / (chemTime*(1 - pow(psi,
    3.0)))));
215 return speciesSource;
216 }
217
218 DEFINE_SOURCE(co2_source, c, t, dS, eqn)
219 {
220 real xc[ND_ND];
221 double psi = 0.0;
222 double chemTime = 0.0;
223 int i = 0;
224 double speciesSource = 0.0;
225 double speciesCellAverage = 0.0;
226 double Re_t = 0.0;
227 double kinv = 0.0;
228
229 if (C_D(c, t) < 0.001) {
230 chemTime = 0.4082*pow((C_MU_L(c, t) / C_R(c, t)) / 0.001, 0.5);
231 }
232 else {
233 chemTime = 0.4082*pow((C_MU_L(c, t) / C_R(c, t)) / C_D(c, t), 0.5)
    ;
234 }
235 //FINE SCALE VOLUME CALCULATION
236 kinv = (C_MU_L(c, t) / C_R(c, t));
237 Re_t = (pow(C_K(c, t), 2.0) / (kinv*C_D(c, t)));
238 if (Re_t <= 64.0) {
239 Re_t = 64.0;
240 }
241 else {
242 Re_t = (pow(C_K(c, t), 2.0) / (kinv*C_D(c, t)));
243 }
244 psi = c_xi*pow(Re_t, -0.25);
245 C_CENTROID(xc, c, t);
246 if (((C_UDMI(c, t, 0) < MF) && (C_UDMI(c, t, 0) > MF_min) && xc[0]
    <= x_max && xc[0] >= x_min )) /* || ((xc[0] < x_min) && (xc[1] >
    y_max) && (xc[1] <=0.009))*/ {
247
248 speciesCellAverage = pow(psi, 3.0)*C_UDMI(c, t, 6) + (1 - pow(psi,
    3.0)) * C_YI(c, t, 1);
249 speciesSource = ((C_R(c, t)*pow(psi, 2.0)) / (chemTime*(1 - pow(
    psi, 3.0))))*(C_UDMI(c, t, 6) - C_YI(c, t, 1)/
    speciesCellAverage*/);
250 if (speciesSource < 0.0) {
251 speciesSource = 0.0;
252 }
253 }
254 else {
255 speciesSource = 0.0;

```

```

256 }
257
258 C_UDMI(c, t, 13) = speciesSource;
259 dS[eqn] = -((C_R(c, t)*pow(psi, 2.0)) / (chemTime*(1 - pow(psi,
    3.0)))));
260 return speciesSource;
261 }

```

F.3 Energy source macro

```

1 DEFINE_SOURCE(energy, c, t, dS, eqn)
2 {
3   real xc[ND_ND];
4   double h_o2 = 0.01634309;
5   double h_h2 = 0.01328158;
6   double h_co2 = -3.93502*pow(10.0, 8.0);
7   double h_h2o = -2.418211*pow(10.0, 8.0);
8   double h_co = -1.105277*pow(10.0, 8.0);
9   double h_ch4 = -7.459847*pow(10.0, 7.0);
10  double h_n2 = 1429.881;
11
12  double M_o2 = 31.9988;
13  double M_h2 = 2.01594;
14  double M_co2 = 44.00995;
15  double M_h2o = 18.01528;
16  double M_co = 28.0104;
17  double M_ch4 = 16.04276;
18  double M_n2 = 28.0134;
19
20  C_CENTROID(xc, c, t);
21  if (((C_UDMI(c, t, 0) < MF) && (C_UDMI(c, t, 0) > MF_min) && xc[0]
    <= x_max && xc[0] >= x_min )) /* || ((xc[0] < x_min) && (xc[1] >
    y_max) && (xc[1] <= 0.009)) */ {
22  C_UDMI(c, t, 14) = -(C_UDMI(c, t, 11) * h_ch4 / M_ch4 + C_UDMI(c,
    t, 9) * h_o2 / M_o2 + C_UDMI(c, t, 13) * h_co2 / M_co2 + C_UDMI
    (c, t, 10) * h_h2o / M_h2o + C_UDMI(c, t, 12) * h_co / M_co +
    C_UDMI(c, t, 8) * h_h2 / M_h2 );
23  }
24  else {
25  C_UDMI(c, t, 14) = 0.0;
26  }
27  dS[eqn] = 0.0;
28  return C_UDMI(c, t, 14);
29  }

```

Appendix G

Contour plots of turbulent Reynolds numbers for RSM and realizable k-epsilon models

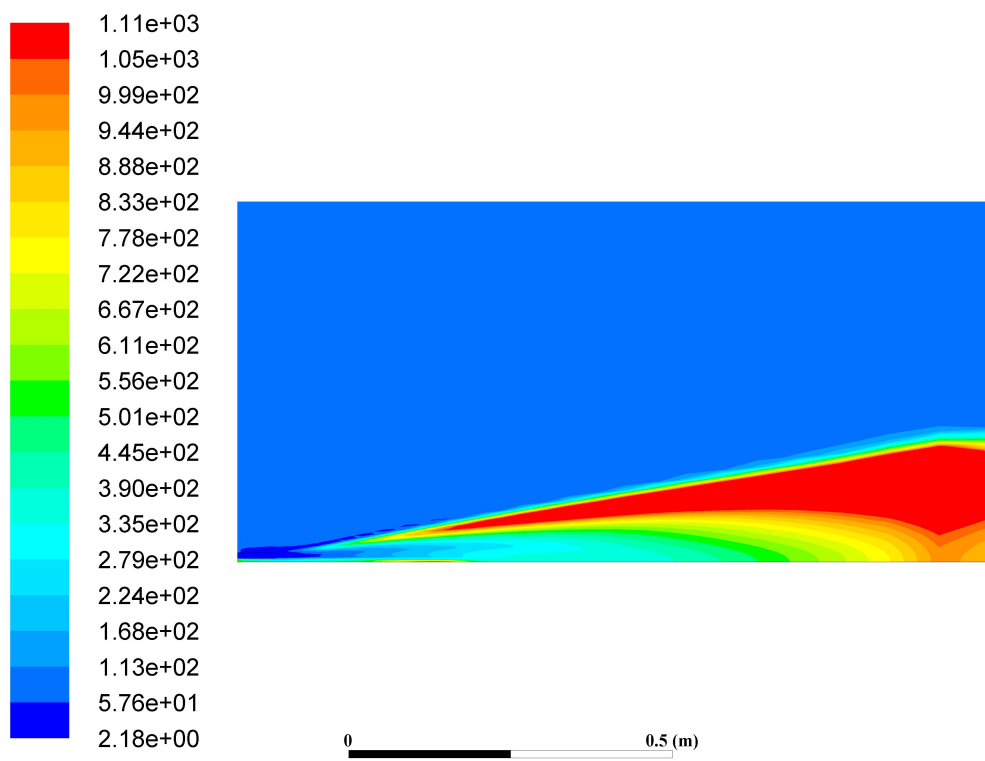


Figure G.1: Contour plot of turbulent Re number for RSM turbulence model

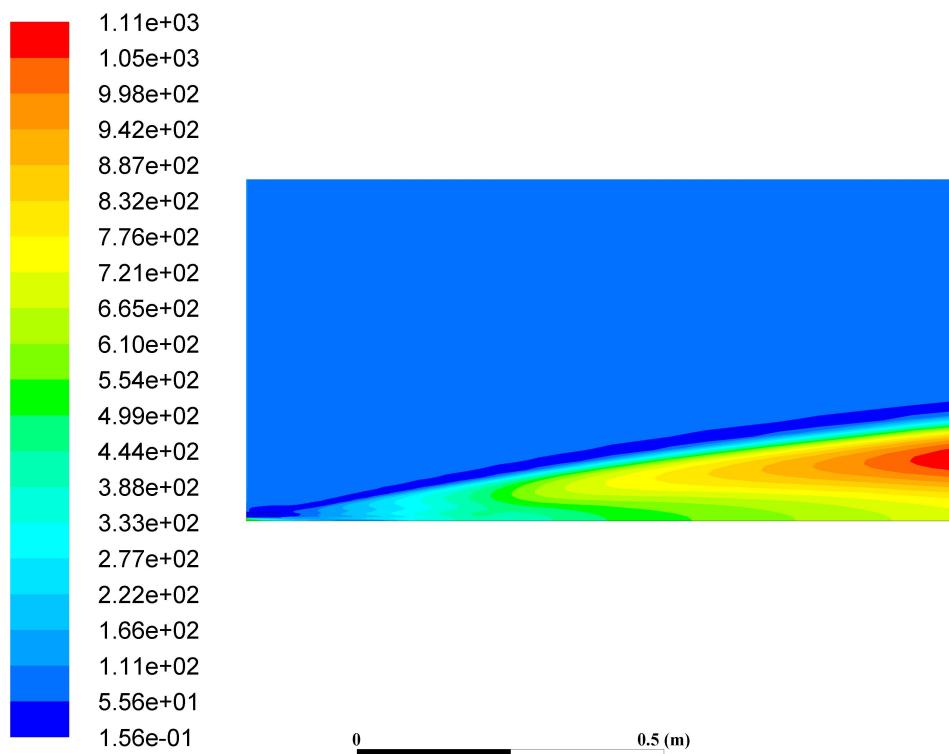


Figure G.2: Contour plot of turbulent Re number for realizable k - ϵ turbulence model

List of References

- Abu-Mostafa, Y.S., Magdon-Ismael, M. and Lin, H. (2012). *Learning from data*. Caltech.
- Anderson, J., Glarborg, P., Jensen, P., Arendt, L. and Hvid, S. (2015). *Experimental and CFD investigation of gas phase freeboard combustion*. Ph.D. thesis, Technical University of Denmark.
- ANSYS (2016). *ANSYS Fluent Theory Guide*. ANSYS Inc., 17th edn.
- Barlow, R. and Frank, J. (2007). Piloted CH₄/Air Flames C, D, E and F - Release 2.1. Tech. Rep., Sandia National Laboratories.
- Bilger, R.W. (1976). The structure of diffusion flames. *Combustion Science and Technology*, vol. 13, no. 1-6, pp. 155–170.
- Blasco, J.A., Fueyo, N., Dopazo, C. and Ballester, J. (1998). Modelling the temporal evolution of a reduced combustion system with an artificial neural network. *Combustion and Flame*, vol. 113, no. 1-2, pp. 38–52.
- Brown, P.N., Byrne, G.D. and Hindmarch, A.C. (1988). VODE, a variable-coefficient ODE solver. *SIAM Journal on scientific and statistical computing*.
- Byrne, G.D. and Dean, A.M. (1993). The numerical solution of some kinetics models with VODE and Chemkin ii. *Computers and Chemistry*, vol. 17, no. 3, pp. 297–302.
- Cengel, Y.A. and Boles, M.A. (2015). *Thermodynamics: An Engineering Approach*. McGraw-Hill.
- Christo, F.C., Masri, A.R. and Nebot, E.M. (1996a). Artificial neural network implementation of chemistry with PDF simulation of H₂/CO₂ flames. *Combustion and Flame*, vol. 106, pp. 406–427.
- Christo, F.C., Masri, A.R. and Nebot, E.M. (1996b). An integrated PDF/neural network approach for simulation turbulent reacting systems. *International Symposium on Combustion*, vol. 26, no. 1, pp. 43–48.
- De, A., Oldenhof, E. and Sathiah, P. (2011). Numerical simulation of Delft-Jet-in-Hot-Coflow (DJHC) flames using the eddy dissipation concept model for turbulence-chemistry interaction. *Flow, Turbulence and Combustion*, vol. 87, no. 4, pp. 537–567.

- Duchi, J., Hazan, E. and Singer, Y. (2011). Adaptive sub gradient methods for on-line learning and stochastic optimization. *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159.
- DuToit, P.C. (2016). Numerical modelling of an industrial boiler using advanced turbulence-chemistry interaction and particle dynamics models. Tech. Rep., University of Stellenbosch.
- Enami, M.D. and Fard, A.E. (2012). Laminar flamelet modelling of a turbulent CH₄/H₂/N₂ jet diffusion flame using artificial neural networks. *Applied Mathematical Modelling*, vol. 36, pp. 2082–2093.
- Ertesvag, I.S. and Magnussen, B.F. (1999). The eddy dissipation turbulence energy cascade model. *Combustion Science and Technology*, vol. 159, pp. 213–235.
- Glassman, I. and Yetter, R.A. (1987). *Combustion*. 2nd edn. Academic Press, New York.
- Gutmark, E. and Wygnanski, I. (1976). The planar turbulent jet. *Journal of Fluid Mechanics*, vol. 73, pp. 465–495.
- Hagan, M.T., Demuth, H.B., Beale, M.H. and Jesus, O.D. (1996). *Neural network design*. Hagan and Demuth.
- Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.
- Hindmarch, A.C. (1983). ODEPACK: A systematized collection of ODE solvers. *Scientific computing- applications of mathematics and computing to the physical sciences*.
- Hinze, J.O. (1975). *Turbulence*. 2nd edn. McGraw-Hill.
- IEA (2013). 2013 key world energy statistics. Tech. Rep., International Energy Agency.
- Jessee, J.P., Gansman, R.F. and Fiveland, W.A. (1993). Calculation of chemical reacting flows using finite rate kinetics. *Heat Transfer and Combustion System (ASME ed.)*, vol. 250, pp. 43–53.
- Jones, W.P. and Priddin, C.H. (1982). Calculation methods for reacting turbulent flows: A review. *Combustion and Flame*, vol. 48, pp. 1–26.
- Kays, W., Crawford, M. and Weigand, B. (2005). *Convective Heat and Mass Transfer*. 4th edn. McGraw-Hill.
- Kee, R., Rupley, F.M., Meeks, E. and Miller, J.A. (1991). Chemkin-III: A fortran chemical kinetics package for the analysis of gas-phase chemical kinetics. Tech. Rep., Sandia National laboratories, Livermore, CA, USA.

- Kempf, A., Flemming, F. and Janicka, J. (2005). Investigation of length-scales, scalar dissipation and flame orientation in a piloted diffusion flame by LES. *Proceedings of the Combustion Institute*, vol. 30, no. 1, pp. 557–565.
- Kent, J.H. and Bilger, R.W. (1973). Turbulent diffusion flames. *Proceedings of the Symposium of Combustion*.
- Kleiber, M. and Joh, R. (2010). *VDI Heat atlas*. Springer. Chapter D3.1.
- Kuo, K.K. (2005). *Principles of Combustion*. 2nd edn. Wiley.
- Laubscher, R. (2015 August). Advanced modelling of homogeneous volatile combustion through the use of reduced chemical mechanisms. *Proceedings of South African Sugar Technologists Association*, vol. 88, no. 126, pp. 138–150.
- Law, V.J. (2013). *Numerical methods for chemical engineers: Using Excel, VBA and Matlab*. CRC Press.
- LeCun, Y., Bottou, L., Orr, G.B. and Muller, K.R. (1998). *Neural networks: Tricks of the Trade*. Springer. Chapter: Efficient backprop.
- Lei, C. and Ghoniem, A.F. (2014). Modelling CO₂ chemical effects on CO formation in oxy-fuel diffusion flames using detailed, quasi-global and global reaction mechanisms. *Combustion Science and Technology*, vol. 186, no. 7, pp. 829–848.
- Magnussen, B.F. (1981 January). On the structure of turbulence and a generalized eddy dissipation concept for chemical reaction in turbulent flow. In: *19th American Institute of Aeronautics and Astronautics Aerospace Science Meeting*. St Louis, Missouri, USA.
- Magnussen, B.F. (2005 June). The eddy dissipation concept: A bridge between science and technology. In: *ECCOMAS Thematic Conference on Computational Combustion*. Lisbon, Portugal.
- Magnussen, B.F. and Hjertager, B.H. (1977). On mathematical modelling of turbulent combustion with special emphasis on soot formation and combustion. *International Symposium on Combustion*, vol. 16, no. 1, pp. 719–729.
- Magnussen, P. (2009). *Investigation into structure and behaviour of laminar and turbulent by planar laser-induced fluorescence measurements*. Ph.D. thesis, Norwegian University of Science and Technology.
- Mcbride, B.J., Zehe, M.J. and Gordon, S. (2002). NASA Glenn coefficients for calculating thermodynamic properties of individual species. *NASA/TP-2002-211556*.
- Perry (2008). *Perry's Chemical Engineers' Handbook*. 8th edn. McGraw-Hill.
- Peters, N. (1984). Laminar diffusion flamelet models in non-premixed turbulent combustion. *Progress in Energy and Combustion Sciences*, vol. 10, no. 3, pp. 319–339.

- Poinsot, T. and Veynante, D. (2005). *Theoretical and Numerical Combustion*. 2nd edn. Edwards.
- Pope, S.B. (1985). PDF methods for turbulent reactive flows. *Progress in Energy and Combustion Sciences*, vol. 11, no. 2, pp. 119–192.
- Scharler, R. (2013 June). CFD simulation of biomass combustion plants - new developments. Bio Energy 2020+, BIOENERGIESYSTEME GmbH.
- Scharler, R., Fleckl, T. and Obernberger, I. (2003). Modification of a Magnussen constant of the Eddy Dissipation Model for biomass grate furnaces by means of hot gas in-situ FT-IR absorption spectroscopy. *Progress in Computational Fluid Dynamics*, vol. 3, no. 2-4.
- Schmidt, U., Rexroth, C.H., Scharler, R. and Cremer, P.I. (2004). New trends in combustion simulation. Tech. Rep., Graz university of technology.
- Sen, B.A. and Menon, S. (2009). Turbulent premixed flame modelling using artificial neural network based chemical kinetics. *Proceedings of the Combustion Institute*, vol. 32, no. 1, pp. 1605–1611.
- Shiehnejadhesar, A., Mehrabian, R., Scharler, R., Goldin, G.M. and Obernberger, I. (2014 March). Development of a gas phase combustion model suitable for low and high turbulence conditions. *Fuel*, vol. 126, pp. 177–187.
- Shih, T.H., Liou, W.W., Shabbir, A., Yang, Z. and Zhu, J. (1995). A new k-epsilon eddy viscosity model for high reynolds number turbulent flows. *Computers and Fluids*, vol. 24, no. 3.
- Spalding, D.B. (1971). Mixing and chemical reaction in steady confined turbulent flames. *Symposium (International) on combustion*, vol. 13, no. 1, pp. 649–657.
- Turns, S.R. (2000). *An Introduction to Combustion: Concepts and Applications*. McGraw-Hill.
- Versteeg, H.K. and Malalasekera, W. (2007). *An Introduction to Computational Fluid Dynamics*. Pearson Prentice Hall.
- Wang, L., Liu, Z., Chen, S. and Zheng, C. (2012). Comparison of different global combustion mechanisms under hot and diluted oxidation conditions. *Combustion Science Technology*, vol. 184, no. 2.
- Warnatz, J., Maas, U. and Dibble, R.W. (2006). *Combustion: Physical and Chemical Fundamentals, Modelling and Simulation, Experiments, Pollutant Formation*. 4th edn. Springer.
- Westbrook, C.K. and Dryer, F.L. (1984). Chemical kinetic modelling of hydrocarbon combustion. *Progress in Energy and Combustion Sciences*, vol. 10, no. 1, pp. 1–57.
- Winston, P.H. (1993). *Artificial intelligence*. Addison-Wesley Publishing company.

- Yin, C. (2011). Advanced modelling of oxy-fuel combustion of natural gas. Tech. Rep., Department of Energy Technology, University of Aalborg.
- Zahirovic, S., Scharler, R. and Obernberger, I. (2006). Advanced gas phase combustion models: validation for biogases by means of LES and experiments as well as application to biomass furnaces. In: *7th European Conference on Industrial Furnaces and Boilers*.