Implementation of Improved EDC Combustion Model in the Open LES Code FDS

David R.U. Johansen

Master’s thesis in Process Safety
University of Bergen
Department of Physics and Technology
Bergen, Norway

August 2011
Preface

This thesis is submitted as the final part of the degree Master of Science in Process Safety Engineering at the University of Bergen. The work has been carried out at Stord/Haugesund University College throughout the whole period. To work with this thesis has been both challenging and time consuming. Still, it has been a valuable experience and really exciting.

The major challenge of this thesis has not been to implement the code in FDS, but to adapt it to the rest of the code and to orientate in the jungle of modules and subroutines. To handle the large amount of numbers requires systematic work and has also been challenging. Unfortunately, the experiments at Lund University gave unsuccessful PIV data. However, the main goal with the experiments was to gain more knowledge to experimental work and not validate the implemented code.

To be able to quantify such complex physical and chemical phenomenon as a fire, and also fluid flow in general with CFD, is truly fascinating me. Since the use of CFD requires a broad knowledge in scientific topics makes it extra interesting. Through this project I have linked loose ends of knowledge together from earlier courses and learned how to apply it in the context of CFD. I have also gained more knowledge to numerical methods and experimental work. However, it is still much more to learn. To work with this thesis has encouraged me to continue to work in the field of fire research.
Acknowledgments

I would like to express my sincere gratitude to my supervisor Associate Professor Bjarne P. Husted at Stord/Haugesund University College. He has been a great inspiration and encouraging to work with. Throughout the process of my work, Bjarne P. Husted has been providing me with expert advice on CFD, fire dynamics, computer science, numerical methods and experimental work. I really appreciate that he linked me up to the pool fire project and even joined the experiments at Lund University.

I appreciate that the staff at Lund University let me take part of the pool fire project, and at the same time let me to do experiments relating my Master’s thesis. I would like to particularly thank Mr. Per Petersson for operating PIV measurements, Professor Patrick van Hees for leading the experiments and PhD student Jonathan Wahlquist who helped with set up.

Other people I would like to thank are:
- Dr. Balram Panjwani (Norwegian University of Science and Technology) who e-mailed me his PhD thesis and answered questions regarding LES-EDC
- Ms. Mari Ueland and Ms. Siri H. Walaker for feedback on my report
- my older brother Mr. Remi Johansen who helped me with Linux, MatLab and LATEX
- Dr. Monika Metallinou who arranged an office for me at Stord/Haugesund University College
- Professor Bjorn Arntzen at the University of Bergen and fellow Master student Ms. Natalja Pedersen for discussion of EDC
- PhD student Bjarne Chr. Hagen (Stord/Haugesund University College) and fellow Master student Mr. Einar Kolstad for general discussion and advice
- Dr. Alf Reidar Nilsen (Stord/Haugesund University College and Statoil) for letting me use his computers for simulations
- Mr. Arjen Kraaijeveld (Stord/Haugesund University College) for advices regarding experimental work
Summary

In this thesis, Magnussen’s and Hjertager’s Eddy Dissipation Concept (EDC) Combustion Model [1][2][3] has been implemented in the CFD code Fire Dynamics Simulator (FDS) [4]. FDS is an open source non-commercial Large Eddy Simulation (LES) code mainly developed by the National Institute of Standards and Technology (NIST). EDC is developed for Reynolds Average Navier Stokes (RANS) equations where fluctuating values caused by turbulence are modeled. The reaction rate in EDC is predicted by RANS quantities such as Turbulent Kinetic Energy (TKE) and Dissipation of Turbulent Kinetic Energy, which is not solved explicitly in LES codes. A promising extension of EDC to LES was proposed Panjwani et al. treating eddy viscosity and strain rate instead [5][6]. In their validation work a model constant was established for jet flames for the Smagorinsky turbulence model. However, the main purpose of FDS is studying smoke spread, fire detection and smoke ventilation in building fires. Such fires involves low Mach number flows driven by buoyancy, in contrast to jet fire that are strongly influenced by the momentum fuel release and are highly turbulent.

The first motivation of this thesis has been to implement LES-EDC in FDS, and second to establish a model constant for buoyancy driven fires and evaluate whether a static constant is sufficient or a dynamic constant is necessary. This is solved by validate the code against velocity profiles in Sandia plume experiments, Heskestad flame height correlation, McCaffery centerline temperature and velocity correlation. Results are also compared with the existing combustion model in FDS for the default turbulence model, Deardorff, in the unofficial version 6.

Experiments with square pipes inserted in the persistent flame region were performed at Lund University. Particle Image Velocimetry (PIV) technique was applied to measure the velocity vector field above the pipes. The goal was to study the affected of generated turbulence from the pipes and perhaps be able to investigate how the turbulence effected the LES-EDC model constant. Unfortunately, the experiments gave unsatisfactory results for CFD validation. During the experiments it was some technical problems with the shutter on the camera occurred as well as the seeding of particles turned out to be quite challenging. Therefore, this part of the thesis must be regarded as a contribution to the project Prediction and validation of pool fire developed in enclosures by means of CFD models for risk assessment of nuclear power plants which the experiments were linked up to, and not validation of FDS-EDC.

A model constant of 0.015 gave satisfactory results for all the chosen validation cases. But a somewhat smaller constant is preferable for the centerline temperature and velocity profiles in the McCaffery simulations. With $C_{LES} = 0.015$ the maximum temperature i over estimated. The slope change in flame height around $Q^* \approx 1$ was not captured by the Vreman and Deardorff turbulence model. The already existing combustion had the same difficulties. LES-EDC with the Smagorinsky model in addition to the existing combustion did also capture the dip for vertical velocity profile in the Sandia plume experiment test 17. But the disadvantage with the Smagorinsky model is the CPU clock times compared with the two other turbulence models.
The errors of the implemented code (referred as FDS-EDC) are in most cases less or the same as for the existing model. The models are also about the same computational expensive. In contrast to the existing model, the implemented code is strongly grid dependent. So before FDS-EDC can be applied in fire analysis the model must be modified to be grid independent. A dynamic constant is not necessary for buoyancy-driven fires in fire engineering application but a more accurate constant is recommended to be established. Temperature and velocity should in further work be validated in a wider range of $Q^*$ for practical fire sizes than in this thesis.
## Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>number of collisions</td>
<td>[-]</td>
</tr>
<tr>
<td>$a$</td>
<td>number of nitrogen atoms</td>
<td>[-]</td>
</tr>
<tr>
<td>$C$</td>
<td>model constant</td>
<td>[-]</td>
</tr>
<tr>
<td>$C$</td>
<td>concentration</td>
<td>mol/m$^3$</td>
</tr>
<tr>
<td>$C_D$</td>
<td>Deardorff model constant</td>
<td>[-]</td>
</tr>
<tr>
<td>$C_{LES}$</td>
<td>EDC-LES model constant</td>
<td>[-]</td>
</tr>
<tr>
<td>$C_S$</td>
<td>Smagorinsky model constant</td>
<td>[-]</td>
</tr>
<tr>
<td>$C_V$</td>
<td>Vreman model constant</td>
<td>[-]</td>
</tr>
<tr>
<td>$c_p$</td>
<td>heat capacity</td>
<td>J/kg·K</td>
</tr>
<tr>
<td>$c$</td>
<td>speed of light</td>
<td>m/s</td>
</tr>
<tr>
<td>$D$</td>
<td>mass diffusivity</td>
<td>m$^2$/s</td>
</tr>
<tr>
<td>$D$</td>
<td>diameter</td>
<td>m</td>
</tr>
<tr>
<td>$D^*$</td>
<td>non-dimensional fire diameter</td>
<td>[-]</td>
</tr>
<tr>
<td>$D_{a}$</td>
<td>the turbulent Damkohler number</td>
<td>[-]</td>
</tr>
<tr>
<td>$E_a$</td>
<td>activation energy</td>
<td>J</td>
</tr>
<tr>
<td>$E$</td>
<td>thermal radiative energy</td>
<td>W/m$^2$</td>
</tr>
<tr>
<td>$f$</td>
<td>acceleration</td>
<td>m/s$^2$</td>
</tr>
<tr>
<td>$Fr$</td>
<td>Froude number</td>
<td>[-]</td>
</tr>
<tr>
<td>$g$</td>
<td>gravitational constant</td>
<td>m/s$^2$</td>
</tr>
<tr>
<td>$H$</td>
<td>heat</td>
<td>kJ</td>
</tr>
<tr>
<td>$\Delta H_c$</td>
<td>heat of combustion</td>
<td>kJ/kg</td>
</tr>
<tr>
<td>$h$</td>
<td>specific enthalpy</td>
<td>kJ/kg</td>
</tr>
<tr>
<td>$h$</td>
<td>convective heat transfer coefficient</td>
<td>W/m$^2$K</td>
</tr>
<tr>
<td>$h$</td>
<td>Planck’s constant</td>
<td>Js</td>
</tr>
<tr>
<td>$j$</td>
<td>diffusiv mass flux</td>
<td>kg/m$^2$s</td>
</tr>
<tr>
<td>$k$</td>
<td>number of reactions each second</td>
<td>s$^{-1}$</td>
</tr>
<tr>
<td>$k$</td>
<td>model constant</td>
<td>[-]</td>
</tr>
<tr>
<td>$k$</td>
<td>turbulent kinetic energy</td>
<td>kJ/kg</td>
</tr>
<tr>
<td>$k$</td>
<td>thermal conductivity</td>
<td>W/m·K</td>
</tr>
<tr>
<td>$k$</td>
<td>Boltzman constant</td>
<td>J/K</td>
</tr>
<tr>
<td>$K_{a}$</td>
<td>the turbulent Karlovitz number</td>
<td>[-]</td>
</tr>
<tr>
<td>$L$</td>
<td>characteristic length scale</td>
<td>m</td>
</tr>
<tr>
<td>$L$</td>
<td>length</td>
<td>m</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
<td>Unit</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>$L_v$</td>
<td>heat required to produce volatiles</td>
<td>[kJ/g]</td>
</tr>
<tr>
<td>$l$</td>
<td>length scale</td>
<td>[m]</td>
</tr>
<tr>
<td>$M$</td>
<td>mass</td>
<td>[kg]</td>
</tr>
<tr>
<td>$M$</td>
<td>molar mass</td>
<td>[g/mol]</td>
</tr>
<tr>
<td>$Ma$</td>
<td>Mack number</td>
<td>[-]</td>
</tr>
<tr>
<td>$m$</td>
<td>mass</td>
<td>[kg]</td>
</tr>
<tr>
<td>$N$</td>
<td>number of species</td>
<td>[-]</td>
</tr>
<tr>
<td>$Nu$</td>
<td>Nusselt number</td>
<td>[-]</td>
</tr>
<tr>
<td>$p$</td>
<td>pressure</td>
<td>[N/m²]</td>
</tr>
<tr>
<td>$\dot{Q}_{c}$</td>
<td>heat release rate (HRR)</td>
<td>[W]</td>
</tr>
<tr>
<td>$\dot{Q}_r$</td>
<td>radiative heat transfer</td>
<td>[W]</td>
</tr>
<tr>
<td>$Q^*$</td>
<td>dimensionless heat release rate</td>
<td>[-]</td>
</tr>
<tr>
<td>$q$</td>
<td>heat</td>
<td>[kJ/kg]</td>
</tr>
<tr>
<td>$R$</td>
<td>universal gas constant</td>
<td>[J/mol·K]</td>
</tr>
<tr>
<td>$R$</td>
<td>reaction rate</td>
<td>[kg/s]</td>
</tr>
<tr>
<td>$Re$</td>
<td>Reynolds number</td>
<td>[-]</td>
</tr>
<tr>
<td>$Rtl$</td>
<td>turbulent Reynolds number</td>
<td>[-]</td>
</tr>
<tr>
<td>$S$</td>
<td>strain rate</td>
<td>[s⁻¹]</td>
</tr>
<tr>
<td>$S$</td>
<td>source term</td>
<td>[-]</td>
</tr>
<tr>
<td>$Sc$</td>
<td>Schmidt number</td>
<td>[-]</td>
</tr>
<tr>
<td>$s$</td>
<td>stoichiometric coefficient</td>
<td>[-]</td>
</tr>
<tr>
<td>$T$</td>
<td>temperature</td>
<td>[K]</td>
</tr>
<tr>
<td>$t$</td>
<td>time</td>
<td>[s]</td>
</tr>
<tr>
<td>$U$</td>
<td>internal energy</td>
<td>[J]</td>
</tr>
<tr>
<td>$u$</td>
<td>velocity</td>
<td>[m/s]</td>
</tr>
<tr>
<td>$W$</td>
<td>molecular weight</td>
<td>[kg/kmol]</td>
</tr>
<tr>
<td>$w$</td>
<td>mechanical work</td>
<td>[kJ/kg]</td>
</tr>
<tr>
<td>$x$</td>
<td>direction $x$</td>
<td>[-]</td>
</tr>
<tr>
<td>$x$</td>
<td>number of carbon atoms</td>
<td>[-]</td>
</tr>
<tr>
<td>$Y$</td>
<td>mass fraction</td>
<td>[kg/kg]</td>
</tr>
<tr>
<td>$y$</td>
<td>direction $y$</td>
<td>[-]</td>
</tr>
<tr>
<td>$y$</td>
<td>number of hydrogen atoms</td>
<td>[-]</td>
</tr>
<tr>
<td>$Z$</td>
<td>mixture fraction</td>
<td>[kg/kg]</td>
</tr>
<tr>
<td>$z$</td>
<td>direction $z$</td>
<td>[-]</td>
</tr>
<tr>
<td>$z$</td>
<td>number of oxygen atoms</td>
<td>[-]</td>
</tr>
<tr>
<td>$z$</td>
<td>height</td>
<td>[m]</td>
</tr>
</tbody>
</table>
Greek symbols

\( \alpha \) thermal diffusivity \([\text{m}^2/\text{s}]\)
\( \beta \) model constant \([-]\)
\( \gamma \) ratio of fine structure mass between large eddies to the total mass \([\text{kg/kg}]\)
\( \gamma^* \) ratio of fine structure mass to the total mass \([\text{kg/kg}]\)
\( \Delta \) grid cell size \([\text{m}]\)
\( \delta_{ij} \) Kronecker-delta \((i = j \rightarrow \delta_{ij} = 1 \text{ and } i \neq j \rightarrow \delta_{ij} = 0)\) \([-]\)
\( \epsilon \) dissipation rate of turbulent kinetic energy \(k\) \([\text{W/kg}]\)
\( \eta \) model constant \([-]\)
\( \lambda \) wave length \([\text{m}]\)
\( \mu \) dynamic molecular viscosity \([\text{kg/m} \cdot \text{s}]\)
\( \nu \) kinematic molecular viscosity \([\text{m}^2/\text{s}]\)
\( \nu \) yield \([\text{kg/kg}]\)
\( \rho \) density \([\text{kg/kg}]\)
\( \sigma_k \) Schmidt number for turbulent kinetic energy \(k\) \([-]\)
\( \sigma_\epsilon \) Schmidt number for dissipation rate of turbulent kinetic energy \(\epsilon\) \([-]\)
\( \sigma \) Stefan-Boltzman constant \([\text{W/m}^2\text{K}^4]\)
\( \tau \) viscous shear tensor \([\text{N/m}^2]\)
\( \tau \) time scale \([\text{s}]\)
\( \varphi \) variable in transport equation or filtered value \([-]\)
\( \chi \) efficient coefficient \([-]\)
\( \dot{\omega} \) reaction rate \([\text{kg/s}]\)
\( \omega \) strain \([\text{s}^{-1}]\)

Superscripts

' fluctuating value
' \( , \) characteristic turbulent scale
'' per \( m^2 \)
- Favre average value
- filtered value
' per second
* fine structure
0 surrounding
Subscripts

0  integral length scale
0  centerline
∞  ambient
a  number of nitrogen atoms
d  diffusion
F  fuel
F  flame
f  formation
f  flame
f  fluid
g  gravitational
i  component in x-direction
j  species j
j  component in y-direction
k  component in z-direction
k  species k
k  turbulent kinetic energy
K  Kolmogorov (length scale)
LFT  lower flame temperature
L  laminar
L  loss
n  number
P  products
r  radiation
SGS  subgrid scale
S  soot
s  surface
T  Taylor (length scale)
T  Turbulent
t  turbulent
u  turbulent
v  vapourizing
\( x \) number of carbon atoms
\( y \) number of hydrogen atoms
\( z \) number of oxygen atoms
\( \alpha \) number
\( \epsilon \) dissipation rate of turbulent kinetic energy
## Contents

1 Introduction ................................. 5

2 Theory ........................................ 7
   2.1 Computational Fluid Dynamics ................. 7
      2.1.1 Governing Equations ................. 7
      2.1.2 Reacting Flows ...................... 8
      2.1.3 Turbulence and Turbulence Modeling .... 9
   2.2 Combustion .................................. 13
      2.2.1 Chemical Kinetics ...................... 15
      2.2.2 Heat Release Rate (HRR) and Reaction Rate 15
   2.3 Flame Characteristics and Fire Plumes ............ 17
      2.3.1 Borghi Diagram ...................... 18
      2.3.2 Flame Height ...................... 19
      2.3.3 Centerline Flame Temperature and Velocity 20
   2.4 Heat Transfer .................................. 21
      2.4.1 Conduction ...................... 22
      2.4.2 Convection ...................... 22
      2.4.3 Radiation ...................... 23
   2.5 Eddy Dissipation Concept (EDC) ............... 23
      2.5.1 Energy Cascade ...................... 23
      2.5.2 Fine Structures and Mass Exchange .... 25
      2.5.3 Perfectly Stirred Reactor (PSR) .... 26

3 Fire Dynamics Simulator (FDS) ................ 28
   3.1 Local Heat Release Rate (HRR) and Reaction Rate 28
   3.2 Mulit-Parameter Mixture Fraction ............... 30
      3.2.1 Two parameter single-step reaction .... 31
      3.2.2 Three parameter two-step reaction .... 32
   3.3 Extinction Criteria ........................... 34
   3.4 Grid Resolution ................................ 34
## CONTENTS

### 4 Implementation of LES-EDC in FDS

4.1 LES-EDC .................................................. 35
4.2 Numerical Procedure .................................... 36
4.3 The Implemented Code ................................... 37

### 5 Validation of FDS-EDC

5.1 McCaffery’s Plume Correlation, Velocity and Temperature Profiles .......... 42
5.2 Heskestad Flame Height Correlation ................................ 60
5.3 Sandia Plume ............................................. 65
  5.3.1 Methane Fire, Test 14 ............................................. 66
  5.3.2 Methane Fire, Test 17 ............................................. 76
  5.3.3 Methane Fire, Test 24 ............................................. 86
  5.3.4 CPU clock time ............................................... 96
5.4 Discussion .................................................. 100
  5.4.1 McCaffery’s Plume Correlation, Velocity and Temperature Profiles 100
  5.4.2 Heskestad Flame Height Correlation ................................ 101
  5.4.3 Sandia Plume ............................................. 104

### 6 Flow Field Above Obstacle Inserted in Fire Plume

6.1 Designing Experiments for CFD Validation .................................. 108
6.2 Measurement Techniques ........................................ 109
  6.2.1 Particle Image Velocimetry (PIV) ................................ 109
  6.2.2 Thermocouples .............................................. 110
6.3 Experiments .................................................. 111
  6.3.1 Summary of the experiments .................................. 113

### 7 Conclusion

7.1 Further Work ............................................... 117

### A FDS input files

A.1 Sandia test 14 ............................................. 121
A.2 Sandia test 17 ............................................. 126
A.3 Sandia test 24 ............................................. 132
A.4 Heskestad ................................................. 138
A.5 McCaffery .................................................. 138

### B Implemented code

B.1 fire.f90 ...................................................... 140
B.2 velo.f90 ...................................................... 165
B.3 mesh.f90 ...................................................... 174
B.4 init.f90 ...................................................... 181

### C Flame Height Calculation, post-prosessing

210
# List of Figures

2.1 Vortex Streching .................................................. 10
2.2 Energy Cascade of TKE ........................................... 10
2.3 Physical processes involved in a fire .......................... 14
2.4 Potential Energy .................................................. 16
2.5 Borghi diagram ................................................... 19
2.6 Flame Length .................................................... 20
2.7 Fire Plume Regimes .............................................. 21
2.8 Energy Cascade .................................................. 24
2.9 Reactor model .................................................... 26
3.1 Finite Rate Reaction and Infinitely fast Reaction .......... 29
3.2 The threshold temperature .................................... 30
4.1 Numerical Procedure in FDS Combustion Model ............ 36
5.1 McCaffery 14 kW Deardorff .................................... 43
5.2 McCaffery 22 kW Deardorff .................................... 44
5.3 McCaffery 33 kW Deardorff .................................... 45
5.4 McCaffery 45 kW Deardorff .................................... 46
5.5 McCaffery 57 kW Deardorff .................................... 47
5.6 McCaffery 14 kW Smagorinsky ................................ 48
5.7 McCaffery 22 kW Smagorinsky ................................ 49
5.8 McCaffery 33 kW Smagorinsky ................................ 50
5.9 McCaffery 45 kW Smagorinsky ................................ 51
5.10 McCaffery 57 kW Smagorinsky ................................ 52
5.11 McCaffery 14 kW Vreman ...................................... 53
5.12 McCaffery 22 kW Vreman ...................................... 54
5.13 McCaffery 33 kW Vreman ...................................... 55
5.14 McCaffery 45 kW Vreman ...................................... 56
5.15 McCaffery 57 kW Vreman ...................................... 57
5.16 McCaffery FDS6 .................................................. 58
5.17 McCaffery FDS6 .................................................. 59
5.18 Flame Height Heskestad ....................................... 61
LIST OF FIGURES

5.19 Flame Height Smagorinsky ........................................ 62
5.20 Flame Height Vreman ............................................. 63
5.21 Flame Height comparison, FDS6 ................................. 64
5.22 Sandia test 14 Deardorff $z = 0.3$ m ..................... 66
5.23 Sandia test 14 Deardorff $z = 0.5$ m ..................... 67
5.24 Sandia test 14 Deardorff $z = 0.9$ m ..................... 68
5.25 Sandia test 14 Smagorinsky $z = 0.3$ m ................. 69
5.26 Sandia test 14 Smagorinsky $z = 0.5$ m ................ 70
5.27 Sandia test 14 Smagorinsky $z = 0.9$ m ................ 71
5.28 Sandia test 14 Vreman $z = 0.3$ m ....................... 72
5.29 Sandia test 14 Vreman $z = 0.5$ m ....................... 73
5.30 Sandia test 14 Vreman $z = 0.9$ m ....................... 74
5.31 Sandia test 14 FDS6 ........................................... 75
5.32 Sandia test 17 Deardorff $z = 0.3$ m ..................... 76
5.33 Sandia test 17 Deardorff $z = 0.5$ m ..................... 77
5.34 Sandia test 17 Deardorff $z = 0.9$ m ..................... 78
5.35 Sandia test 17 Smagorinsky $z = 0.3$ m ................ 79
5.36 Sandia test 17 Smagorinsky $z = 0.5$ m ................ 80
5.37 Sandia test 17 Smagorinsky $z = 0.9$ m ................ 81
5.38 Sandia test 17 Vreman $z = 0.3$ m ....................... 82
5.39 Sandia test 17 Vreman $z = 0.5$ m ....................... 83
5.40 Sandia test 17 Vreman $z = 0.9$ m ....................... 84
5.41 Sandia test 17 FDS6 ........................................... 85
5.42 Sandia test 24 Deardorff $z = 0.3$ m ..................... 86
5.43 Sandia test 24 Deardorff $z = 0.5$ m ..................... 87
5.44 Sandia test 24 Deardorff $z = 0.9$ m ..................... 88
5.45 Sandia test 24 Smagorinsky $z = 0.3$ m ................ 89
5.46 Sandia test 24 Smagorinsky $z = 0.5$ m ................ 90
5.47 Sandia test 24 Smagorinsky $z = 0.9$ m ................ 91
5.48 Sandia test 24 Vreman $z = 0.3$ m ....................... 92
5.49 Sandia test 24 Vreman $z = 0.5$ m ....................... 93
5.50 Sandia test 24 Vreman $z = 0.9$ m ....................... 94
5.51 Sandia test 24 FDS6 ........................................... 95
5.52 Sandia plume CPU clock time, $dx = 1.5$ cm .......... 96
5.53 Sandia plume CPU clock time, $dx = 3$ cm ............. 97
5.54 Sandia plume CPU clock time, $dx = 6$ cm ............. 98
5.55 Sandia plume CPU clock time, comparison between FDS6 and FDS-EDC 99
5.56 Temperature and velocity contours for McCaffery 57 kW ............... 101
5.57 Heskestad Percentage Error, Smagorinsky .................. 102
5.58 HRRPUV Slice File Smagorinsky ............................. 103
5.59 Temperature contours for FDS6 test 14 .................... 105

6.1 CFD road map ...................................................... 108
LIST OF FIGURES

6.2 Numbering of thermal couples in the pipes. 109
6.3 Particle Image Velocimetry (PIV) 110
6.4 Thermocouple measurement circuit 111
6.5 Experimental setup 112
6.6 Calibration of the camera 114
List of Tables

2.1 Chemical Reactions ........................................ 15
2.2 Flame types ........................................ 17
2.3 McCaffery Empirical Constants ................................. 22
5.1 Heskestad Flame Height Simulations .......................... 60
5.2 Sandia plume experiment simulations ...................... 65
6.1 Experimental scenarios ...................................... 112
Chapter 1

Introduction

Accidental fires and explosions are a severe threat to human life and expensive process equipment in the process industries. To obtain an acceptable risk when handling flammable gas and liquid, knowledge to the physical processes involved in a fire or explosion is required for safety designing. Computational Fluid Dynamics (CFD) is a useful engineering tool to evaluate potential consequences, such as heat loads on process equipment or gas dispersion from accidental leaks, detector and mitigation optimization. Over the last decades, the use of CFD has increased as a result of increased computational power. Increasing computational power is also allowing continuously development of CFD.

Several commercial and non-commercial CFD codes exist for different use. Mainly the codes are divided in which way the turbulence is modeled. Fire Dynamic Simulator (FDS) is a non-commercial Large Eddy Simulation (LES) code developed by NIST (National Institute of Standard and Technology) appropriate for indoor building fire analysis [4]. FDS is limited to simulations of thermal driven flows as buoyancy-driven fires, i.e. FDS is unsuitable for jet fire and explosion simulations.

Correct modeling of the combustion process in CFD is crucial to achieve reliable results. Combustion models are based on certain assumptions related to the simulated flame type. The Eddy Dissipation Concept (EDC) proposed by Magnussen and Hjertager, is based on the turbulent mixing to model the reaction rate [1][2][3]. The model assumes that the chemical reaction occurs where reactants and hot products are molecularly mixed. The mixing process is located where the kinetic energy is dissipated into heat. These regions, referred as fine structure, are treated as a perfectly stirred reactor.

EDC in its original form assumes full turbulence cascading in each numerical cell, as the way turbulence is modeled in the Reynolds Average Navier Stokes (RANS). Extension of EDC to LES has been performed by Hu et al. and Zhou et al., both with unsatisfactory predictions [7][8]. The reason for the unsatisfactory predictions was that RANS model constants were used. Panjwani et al. proposed two approaches to LES-EDC, where the fine structure regions are based on subgrid viscosity instead of turbulent kinetic energy (TKE) and dissipation, because they are usually not solved explicitly in
LES codes [5][6]. A parameter study in their validation showed that the model constant $C_{LES} = 0.25$ was satisfactory for the Smagorinsky turbulence model. However, this was for a jet fire strongly influenced by momentum. The Froude number for the jet flames Panjwani et al. studied were in order of $10^4$ while fires interested in context of buildings are in order of less than $10^{-3}$.

The objective of this thesis has been to implement LES-EDC in FDS and validate the implemented code (hereafter referred as FDS-EDC) to the application area of FDS. More precisely the objective has been to:

- establish a model constant for thermal buoyancy-driven fires and evaluate whether a static constant is sufficient with respect to different fire sizes or a dynamic constant is necessary
- establish a model constant for all supported turbulence models in FDS6; Vreman, Smagorinsky and Deardorff
- evaluate if LES-EDC can replace the already existing combustion model in FDS (hereafter referred as FDS6)

The simulations were limited to two parameter mixture fraction with a infinitely fast single-step reactions for non-premixed flames, i.e detailed kinetics are outside the scope of this thesis.
Chapter 2

Theory

2.1 Computational Fluid Dynamics

Computational fluid dynamics (CFD) is a powerful engineering tool, which is used to analyze fluid flow problems. CFD is used in a wide range of engineering disciplines, from designing airplanes, studying pipeline flows to fire engineering applications. Ever-increasing computational power leads to continuously development of CFD, which over time allows implementing of more complex algorithms. Because of inaccuracy in numerical techniques and limited precision in representation of numbers in a computer, CFD should always be considered as a supplement for experimental investigation and not a substitute [9]. In general, CFD is deterministic and is the reason why results are totally dependent on input values as boundary conditions and specified transient behavior in the computational domain. This leads to correct assumptions has to be taken into account. Therefore, CFD requires expert knowledge in physics, chemistry, computer science and numerical methods [10].

2.1.1 Governing Equations

In CFD, the finite volume method is applied by dividing a computational domain in to several sub volumes, a grid, where partial differential equations are solved in center of each sub volume or grid cell. The partial differential equations (PDE) are based on conservation of mass, momentum, energy and other static variables, \( \varphi \). A general flow equation is given by

\[
\frac{\partial}{\partial t} (\rho \varphi) + \frac{\partial}{\partial x_j} (\rho \varphi u_j) = \frac{\partial}{\partial x_j} (-j_{\varphi,j}) + S_{\varphi}
\]  

(2.1)

The first term is accumulation of \( \varphi \), the second is convective transport of \( \varphi \), the third term is diffusive transport of \( \varphi \) and the last is the source term. Since an analytical solution on these equations does not exist, the equations are solved numerically. Taylor
expansions series is the most common technique to solve these partial differential equations [9]. The governing equations are:

Conservation of mass (Continuity Equation)
\[
\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_j) = 0 \tag{2.2}
\]

Conservation of momentum (Newton’s Second Law)
\[
\frac{\partial}{\partial t}(\rho u_j) + \frac{\partial}{\partial x_j}(\rho u_i u_j) = -\frac{\partial p}{\partial x_i} + \partial r_{ij} \frac{\partial}{\partial x_j} + \rho f_i \tag{2.3}
\]

Conservation energy (First Law of Thermodynamics)
\[
\frac{\partial}{\partial t}(\rho h) + \frac{\partial}{\partial x_j}(\rho u_i h) = \frac{\partial}{\partial x_j}\left(\rho \frac{\partial h}{\partial x_j}\right) + \dot{Q}_r + \rho \omega Y_f \Delta H_c \tag{2.4}
\]

These five partial differential equation consist six solution vectors: density \(\rho\), temperature \(T\), pressure \(p\), three velocity components \(u_i\), \(u_j\) and \(u_k\). Note that it is three equations for conservation of momentum, one in each direction. The enthalpy is given as a function of temperature; \(h = \int_{T_0}^{T} c_p T dT\). Pressure is calculated from the equation of state
\[
p = \frac{\rho RT}{M} \tag{2.5}
\]

Additional sub models are needed for closing the source terms, e.g. radiation, Reynolds Stresses and combustion.

2.1.2 Reacting Flows

For reacting flows or multi-species flows, mass fractions for each species, \(k\), is conserved
\[
\frac{\partial}{\partial t}(\rho Y_k) + \frac{\partial}{\partial x_j}(\rho u_i Y_k) = \frac{\partial}{\partial x_j}\left(\rho D \frac{\partial Y_k}{\partial x_j}\right) + R_k \tag{2.6}
\]

From the relation \(\sum_k Y_k = 1\) the last species can be calculated, hence \(N - 1\) partial differential equations are necessary for a total number \(N\) species.

PDE are expensive to compute. So to reduce the number of PDE when computing reacting flow, the mixture fraction can be computed instead of mass fractions:
\[
\frac{\partial}{\partial t}(\rho Z_k) + \frac{\partial}{\partial x_j}(\rho u_i Z_k) = \frac{\partial}{\partial x_j}\left(\rho D \frac{\partial Z_k}{\partial x_j}\right) + R_k \tag{2.7}
\]

Considering two streams mixing together, \(Z\) kg with property \(\varphi_1\) in stream 1 and \((1 - Z)\) kg with property \(\varphi_2\) in stream 2 giving 1 kg mixture \(\varphi_{mix} = Z \varphi_1 + (1 - Z) \varphi_2\). The mixture fraction is then
\[
Z = \frac{\varphi_{mix} - \varphi_2}{\varphi_1 - \varphi_2} \tag{2.8}
\]
Assuming no sink or source, this expression can be used for all kinds of scalars [11]. In combustion modeling, stream 1 could be fuel and stream 2 air with property mass fractions \((Y_k)_\text{mix} = Z(Y_k)_1 + (Z - 1)(Y_k)_2\). Assuming stoichiometric infinitely fast chemistry of
\[
1 \text{ kg fuel} + s \text{ kg air} \rightarrow (1 + s) \text{ products}
\] (2.9)

"mixed is burned", i.e oxygen and fuel cannot coexist, the scalar \(Y_F - \frac{1}{4}Y_{O_2}\) is conserved for a complete reaction. \(Y_F\) and \(Y_{O_2}\) are mass fractions for fuel and oxygen respectively, and \(s\) is the stoichiometric coefficient. At stoichiometric condition, the mixture fraction is
\[
Z_{\text{stoich}} = \frac{(Y_F - \frac{1}{4}Y_{O_2})_{\text{stoic}} - (Y_F - \frac{1}{4}Y_{O_2})^2}{(Y_F - \frac{1}{4}Y_{O_2})_1 - (Y_F - \frac{1}{4}Y_{O_2})_2} = \frac{1}{s + 1}
\] (2.10)

Then \(Y_{\text{products}} = 1\) and there is no fuel or oxygen left. In the fuel-rich region \((Z > Z_{\text{stoich}})\) no oxygen is left, and in the fuel-lean region \((Z < Z_{\text{stoich}})\) no fuel is left. Now, all mass fractions can simply be calculated.

2.1.3 Turbulence and Turbulence Modeling

To give a precise definition of turbulence is very difficult, but in fluid dynamics may turbulent flows be characterized as irregular motion of fluid where velocity is rapidly fluctuating. The underlying physical mechanism in turbulence is complex and is yet not fully understood. Turbulence remains as one of the most important unsolved physical problem, because of the fact that almost all natural flows are turbulent [12]. To evaluate whether a flow is turbulent or laminar, the dimensionless Reynolds number is calculated. The Reynolds number is the ratio of inertial forces to viscous forces. In turbulent flows inertial forces are dominating, hence the Reynolds number is large.

In turbulent flows, rotational flow structures called eddies are formed, each with different characteristic length and velocity scale \((l', n', m', ..., *)\). Eddies are formed as a consequence of shear stress (friction) generated between fluid sheets of different velocities. Turbulent flows are always dissipative [12]. Eddies are dissipating through vortex stretching caused by viscous shear stress in the flow and the kinetic energy is converted to internal energy (see Figure 2.1). Mechanic energy is transferred from the main stream to large eddies and further down to smaller eddies [13]. The eddy dissipation continues down to a level where the diffusion time across the diameter equals the time for an eddy to rotate 1/2 revolution [14]. At this level, the length scale of an eddy equals Kolmogorov length scale and is where molecular mixing dominating (see Figure 2.2). This breakdown of eddies is called the energy cascade model and is seen in Figure 2.8.

The major part of the kinetic energy is in the large eddies [14]. In turbulent flows for high turbulence Reynolds number large eddies are much larger than small eddies. According to Kolmogorov the small eddies are then not influenced by large eddies and the main stream. A typical distribution of turbulent kinetic energy is given in Figure 2.2.

Integral length scale, \(l_0\), is the largest length scale and corresponds to geometrical dimensions [14]. Large eddies has characteristic scales as the mean flow and are therefore
CHAPTER 2. THEORY

Figure 2.1: (a) Vortex stretching in wind tunnel [12]. Angular momentum is conserved. (b) Principle of vortex stretching [13].

Figure 2.2: Energy cascade: Distribution of turbulent kinetic energy at different length scales [14]. Integral length scale, \( l_0 \), Taylor length scale, \( l_T \), and Kolmogorov length scale, \( l_K \).
dominated by inertia effects [15]. Large eddies are inviscid in contrast to small eddies which are influenced by viscosity. For this reason, high viscosity flows are associated with few large eddies while low viscosity flows are associated with less larger eddies but a larger amount of smaller eddies. The largest length scale where viscosity affects the dynamics of turbulent eddies is called Taylor length scale, \( l_T \).

Depending on how the turbulence is treated, CFD models are divided in

- Reynolds Average Navier-Stokes (RANS)
- Large Eddy Simulation (LES)
- Direct Numerical Simulation (DNS)

In RANS, all turbulent structures are modeled in a sub grid scale model (SGS) , in contrast to DNS where fluctuations are captured on the grid points and calculated directly by the governing equations. Between DNS and RANS, LES are capturing large eddies on the grid points while small eddies are modeled.

In the momentum equation (eq. (2.3)), the viscous stress tensor is divided in \( \tau_{eff} = \tau_{mol} + \tau_{turb} \), where \( \tau_{turb} \) can be regarded as an extra stress (or viscosity) caused by increased transport in turbulent flows. This extra stress is closed by a turbulence model. According to Boussinesq hypothesis, Reynolds stresses \( (-\bar{\rho}u_i'u_j', -\bar{\rho}u_i'\rho' \) and \( -\bar{\rho}u_i'Y_i' \) ) are proportional to the mean rate of strain:

\[
-\bar{\rho}u_i'u_j' + \frac{2}{3} \bar{\rho}k \delta_{ij} = 2\mu_t \ddot{S}_{ij} - \frac{2}{3} \mu_t \frac{\partial \ddot{u}_k}{\partial x_k} \delta_{ij} \tag{2.11}
\]

where \( \mu_t \) is the turbulent viscosity, \( \delta_{ij} \) is the Kronecker delta \( (i = j \rightarrow \delta_{ij} = 1 \) and \( i \neq j \rightarrow \delta_{ij} = 0 \) ) and turbulent kinetic energy \( \bar{k} = \frac{1}{2} \bar{u}_i' \bar{u}_i' \). Bars over the variables represents favre filtered quantities (for LES) or average quantities (for RANS). The strain rate tensor is expressed as

\[
\ddot{S}_{ij} = \frac{1}{2} \left( \frac{\partial \ddot{u}_i}{\partial x_j} + \frac{\partial \ddot{u}_j}{\partial x_i} \right) \tag{2.12}
\]

Reynold’s Average Navier-Stokes (RANS)

RANS is applied when only average values are interesting in fluid flow analysis. RANS is the least computational expensive model and is used in most commercial CFD codes. In RANS, the solution vector is split in a fluctuating term and an average term, e.g for velocity \( u = \bar{u} + u' \). The most common model to close RANS equations is \( k-\epsilon \) two-equation model, where two additional partial differential equations for transport of turbulent kinetic energy, \( k \), and dissipation rate of kinetic energy, \( \epsilon \), are solved:

\[
\frac{\partial (\bar{\rho}k)}{\partial t} + \frac{\partial (\bar{\rho} \bar{u}_i k)}{\partial x_i} = \frac{\partial}{\partial x} \left[ \frac{\mu_t}{\sigma_k} \frac{\partial k}{\partial x_j} \right] - \bar{\rho}u_i' \bar{u}_j' \frac{\partial \bar{u}_i}{\partial x_j} - \bar{\rho} \epsilon \tag{2.13}
\]

\[
\frac{\partial (\bar{\rho} \epsilon)}{\partial t} + (\bar{\rho} \frac{\partial \bar{u}_i}{\partial x_i}) = \frac{\partial}{\partial x} \left[ \frac{\mu_t}{\sigma_\epsilon} \frac{\partial \epsilon}{\partial x_j} \right] - C_{\epsilon 1} \frac{\epsilon}{k} \left( \bar{\rho}u_i' \bar{u}_j' \frac{\partial \bar{u}_i}{\partial x_j} \right) - C_{\epsilon 2} \frac{\epsilon^2}{k} \tag{2.14}
\]
For the two-equation model the turbulent viscosity is
\[
\mu_t = \bar{\rho} C_\mu \frac{k^2}{\epsilon},
\]
(2.15)
where \( k = \frac{1}{2} \bar{u}_i' \bar{u}_i' \) and \( \epsilon = \frac{\mu_T}{\bar{\rho} \left( \frac{\partial \bar{u}_i'}{\partial x_j} \right) \left( \frac{\partial \bar{u}_i'}{\partial x_j} \right)} \). The constants for eq. (2.13), (2.14) and (2.15) have been established by data fittings from experiments of a broad range of turbulent flows; \( C_\mu = 0.09, \sigma_k = 1.0, \sigma_\epsilon = 1.3, C_{\epsilon1} = 1.44, C_{\epsilon2} = 1.92 \) [16]. However, the one-equation model only consist one additional partial differential equation. This equation is for conservation of turbulent kinetic energy. In the zero-equation model, the turbulent viscosity is calculated directly from an empirical correlation.

Other turbulence models for RANS equations may be \( k-\omega \), Menter model, and the more sophisticated Reynolds Stress Model which involves calculation of individual Reynolds stresses and gives seven additional partial differential equations.

**Large Eddy Simulation (LES)**

The computational time of Large Eddy Simulation is less than DNS and more than RANS. LES compute explicitly the largest structures in the flow by filtered Navier-Stokes equation, while small structures, typically smaller than the grid cell size, are modeled. The grid cell size must be in order that a sufficient amount of the turbulent energy is solved on the grid points (see Section 3.4). In 1D, large structure (low wave numbers) are filtered by
\[
\tilde{\varphi}(x,t) = \int_{x-\frac{\Delta}{2}}^{x+\frac{\Delta}{2}} \varphi(r,t) dr
\]
(2.16)
where \( \tilde{\varphi} \) is the filtered quantity and \( \Delta \) is the local cell size.

The most popular approach for the unresolved fluxes in LES is the Smagorinsky model:
\[
\mu_t = \rho C_S^2 \Delta^{4/3} l_i^{2/3} |\vec{S}|\]
(2.17)
where \( \Delta = (\delta x \delta y \delta z) \) and the strain rate, \( \vec{S} \), is given in eq. (2.12). The empirical constant, \( C_S \), is usually 0.1 - 0.2. Eq. (2.17) is simplified assuming that the turbulence integral length scale is in the same order as the grid cell size, \( l_i \approx \Delta \):
\[
\mu_t = \rho (C_S \Delta)^2 |\vec{S}|
\]
(2.18)
Germano et al. attempted to formulate a more universal approach, and proposed a dynamic Smagorinsky constant [17]. A detailed description of the dynamic constant is found in various literature (e.g Theoretical and Numerical Combustion, 2005 [18]).

Another model turbulence model is the Deardorff eddy viscosity model. The eddy viscosity is expressed as
\[
\mu_T = \rho C_D \Delta \sqrt{k_{SGS}},
\]
(2.19)
CHAPTER 2. THEORY

where $C_D = 0.1$ (in FDS v.6) and $k_{SGS}$ is the sub grid scale kinetic energy.

Recently, Vreman proposed a simple turbulence model that similar to the Smagorinsky model, only needs the first-order derivatives of the velocity and the local filter width to compute the viscosity [19]:

$$\mu_T = \rho C_V \sqrt{\frac{B_\beta}{\alpha_{i,j} \alpha_{i,j}}}$$

(2.20)

$C_V \approx 2.5C_S$. For homogenous isotropic turbulence, the Smagorinsky constant equals 0.17 and gives $C_V = 0.07$ (default in FDS v.6) [19]. $\alpha$ is a $3 \times 3$ matrix with derivatives of the filtered velocity:

$$\alpha = \begin{bmatrix} \frac{\partial \tilde{u}}{\partial x} & \frac{\partial \tilde{u}}{\partial y} & \frac{\partial \tilde{u}}{\partial z} \\ \frac{\partial \tilde{v}}{\partial x} & \frac{\partial \tilde{v}}{\partial y} & \frac{\partial \tilde{v}}{\partial z} \\ \frac{\partial \tilde{w}}{\partial x} & \frac{\partial \tilde{w}}{\partial y} & \frac{\partial \tilde{w}}{\partial z} \end{bmatrix}$$

(2.21)

By definition, $\mu_T$ is set to zero if $\alpha_{i,j} \alpha_{i,j} = 0$. Further,

$$B_\beta = \beta_{11}\beta_{22} - \beta_{12}^2 + \beta_{11}\beta_{33} - \beta_{13}^2 + \beta_{22}\beta_{33} - \beta_{23}^2$$

(2.22)

where $\beta_{i,j} = \Delta_m\alpha_{m,i}\alpha_{m,j}$.

Direct Numerical Simulation (DNS)

DNS is the most computational expensive approach where the transport equations are solved numerically without any turbulence model. To capture all the spatial scales of turbulence on the numerical grid, from Kolmogorov length scale, $l_k$ (smallest dissipative scales), to integral length scale, $l_0$, a sufficient number of grid points are needed. According to J. Warnatz et al. [14] about 1000 grid points in each direction are necessary to resolve the smallest turbulent structures for a typical flow with turbulence Reynolds number $R_l = 500$ and ratio $l_0/l_k \approx 100$. This equals for a three-dimensional simulation a total of $10^9$ grid points. For RANS and LES a typical range in order of $10^5$-$10^6$ grid points are sufficient. Therefore, DNS is limited to small-scale laminar bench scale flames in the matter of fire research.

2.2 Combustion

Combustion is an exothermic process where fuel and oxidant are reacting chemically involving rather complex physics and chemistry. The combustion can be seen as light, either as a flame or a glow, caused by the heat released in the reactions. Simplified, fuel and an oxidant are reacting forming carbon dioxide, carbon monoxide, soot and water vapor for combustion of hydrocarbons:

$$fuel + oxidant \rightarrow H_2O + CO + CO_2 + soot$$

(2.23)
In natural fires, oxygen in the air is the oxidant in the chemical reactions. The fuel can be solid, gas or liquid usually carbon-based. However, the flame is a gas phase phenomenon. This means that in a combustion process involving a solid or liquid, a conversion to gaseous form is necessary. In a liquid fire the radiation from the flame is evaporating the liquid, but for most solids, a chemical decomposition or pyrolysis takes place instead of evaporation. Pyrolysis yields products light enough to volatilize from the surface before entering the combustion zone. A general mass burning rate be expressed as

$$\dot{m}' = \frac{\dot{Q}_F'}{L_v} - \dot{Q}_L'$$

(2.24)

where $L_v$ is the heat required for evaporation or pyrolysis, $\dot{Q}_F'$ the heat supplied by the flame to the fire area and $\dot{Q}_L'$ heat loss from the fuel surface. A visual description of physical processes in a fire is shown in Figure 2.3.
2.2.1 Chemical Kinetics

A combustion process consists of several chemical reactions, called *elementary reaction*. Even a simple combustion of methane in air consists 400 reactions [21]. The reactions of interest, *radical chain reactions*, are divided in *chain propagation*, *chain branching* and *chain termination*. Warnatz et al. considered a hydrogen combustion where the most important reactions are given in Table 2.1 [14]. An overall stoichiometric reaction for combustion of hydrogen in oxygen is:

\[ 2H_2 + O_2 \rightarrow 2H_2O \]  

(2.25)

Table 2.1: Chemical reactions with respect to hydrogen ignition (adapted from Warnatz et al., 2006 [14])

<table>
<thead>
<tr>
<th>Reaction number</th>
<th>Chemical reaction</th>
<th>Reaction Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( H_2 + O_2 \rightarrow 2OH \bullet )</td>
<td>chain initiation</td>
</tr>
<tr>
<td>1</td>
<td>( OH \bullet + H_2 \rightarrow H_2O + H \bullet )</td>
<td>chain propagation</td>
</tr>
<tr>
<td>2</td>
<td>( H \bullet + O_2 \rightarrow OH \bullet + O \bullet )</td>
<td>chain branching</td>
</tr>
<tr>
<td>3</td>
<td>( O \bullet + H_2 \rightarrow OH \bullet + H \bullet )</td>
<td>chain branching</td>
</tr>
<tr>
<td>4</td>
<td>( H \bullet \rightarrow 1/2H_2 )</td>
<td>chain termination</td>
</tr>
<tr>
<td>5</td>
<td>( H \bullet + O_2 + M \rightarrow HO_2 + M )</td>
<td>chain termination</td>
</tr>
</tbody>
</table>

The dots in the reactions represents that the species are a free radical. Free radicals are highly reactive because of its charge and are essential to obtain the chemical reactions in a combustion process. In the first reaction, *chain initiation*; a stable species are forming one reacting species. The *chain initiation* leads to *chain propagation* (reaction 2 and 3) where a reactive species react with stable species forming another reactive species. In *chain branching steps* two reactive species are formed from a stable species and a reactive species. The last steps (reaction 4 and 5), *chain termination*; no reactive species are formed.

2.2.2 Heat Release Rate (HRR) and Reaction Rate

The heat release rate (HRR) is the most important factor to characterize a fire. Mathematically HRR can be expressed as

\[ \dot{Q}_c = \chi \dot{m} \Delta H_c \]  

(2.26)

where \( \chi \) is an efficiency coefficient which takes into account that the combustion is incomplete, and heat is lost to surroundings e.g. through radiation and heat transfer to the air entrained (see figure 2.3). \( \dot{m} \) is the fuel consumption rate. The reaction rate is depended on a characteristic time scale limiting the reaction. A simple assumption
would be finite chemical reaction rate, hence the chemical time scale limits the reaction rate. Collisions between molecules is the underlying phenomena in chemical reactions, i.e the reaction rate is a function of temperature, pressure and concentration. The reaction rate may be given as

$$R_k = -k_p^2 C_{\text{fuel}}C_{\text{oxygen}}$$  \hspace{1cm} (2.27)

where $C_{\text{fuel}}$ and $C_{\text{oxygen}}$ are concentrations of fuel and oxygen respectively. $k$ is the rate coefficient are given by Arrhenius law for temperature depended reactions:

$$k = Ae^{-E_a/RT}$$  \hspace{1cm} (2.28)

$A$ is the total number of molecular collisions and $e^{-E_a/RT}$ the fraction of collisions that leads to reactions. Figure 2.4 shows that $E_a$, the activation energy, is the energy barrier to start the reaction. $R$ is the universal gas constant and $T$ absolute temperature in Kelvin. A more sophisticated method assumes infinitely fast chemistry and would (also) involve other limiting time scale e.g turbulent time scale, diffusive timescale or acceleration time scale. See section 2.5 and 3.1.

Figure 2.4: A graphical description of energy in an exothermic reaction [14]. $E_a^{(f)}$ is the energy barrier to start the reaction. Considering a combustion $\Delta H_c \approx U_{\text{reactants}} - U_{\text{products}}$.

$\Delta H_c$ is the heat of combustion, the energy released during combustion per unit mass or mole. Heat of combustion equals the difference in energy of species before the combustion and after the combustion. Assuming adiabatic conditions (no heat loss to surroundings), the heat of combustion can be determined experimentally in calorimeter bomb at constant volume, where temperature rise is measured. The change in enthalpy according to the first law of thermo dynamics is then

$$\Delta H = \Delta U + p\Delta V$$  \hspace{1cm} (2.29)
CHAPTER 2. THEORY

From the ideal gas law (eq. (2.5)) the work done \( p\Delta V \) can be calculated. But since the work done is small compared to the increase of internal energy \( \Delta U \), it may be neglected, leaving \( \Delta H_c \approx U_{\text{products}} - U_{\text{reactants}} \), as shown in Figure 2.4.

When a compound is formed under standard state (1 atm. pressure and 298 K), the change of enthalpy is by definition, the same as heat of formation \( \Delta H_f \) [20]. Considering stoichiometric combustion of methane, the balanced reaction is

\[
CH_4 + 2O_2 \rightarrow 2H_2O + CO_2
\]

The heat of combustion may be calculated by definition of heat of formation

\[
\Delta H_c(CH_4) = (2\Delta H_f(H_2O) + \Delta H_f(CO_2)) - (2\Delta H_f(O_2) + \Delta H_f(CH_4)) \tag{2.31}
\]

\( \Delta H_f \) for reactants and products are found in various literature.

2.3 Flame Characteristics and Fire Plumes

Flames are categorized in four different types given in Table 2.2. When fuel and the oxidant are mixed and burned simultaneously, the flame is non-premixed. If the fuel and oxidant are mixed before the combustion it is called a premixed flame. Non-premixed and premixed flames are either turbulent or laminar. As flows in general, most flames are also turbulent. Rapid mixing of reactants in the combustion zone for turbulent flames increases the reaction rate and is characterized by an irregular flame sheet.

Table 2.2: Flame types (adapted from Warnatz et al., 2006 [14]).

<table>
<thead>
<tr>
<th>Fuel/Oxidizer Mixing</th>
<th>Fluid Motion</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>premixed</td>
<td>laminar</td>
<td>flat flame Bunsen burner</td>
</tr>
<tr>
<td></td>
<td>turbulent</td>
<td>gasoline engine and gas turbines</td>
</tr>
<tr>
<td>nonpremixed</td>
<td>laminar</td>
<td>wood fire and candles</td>
</tr>
<tr>
<td></td>
<td>turbulent</td>
<td>aircraft turbine, diesel engine and coal combustion</td>
</tr>
</tbody>
</table>

Froude number is a way to characterize a fire and is expressed as

\[
Fr = \frac{U^2}{gD} \tag{2.32}
\]

and is the ratio of momentum forces and buoyancy forces. \( U \) is the gas velocity, \( D \) fire diameter or a characteristic length and \( g \) gravitational constant. The velocity is estimated by

\[
U = \frac{\dot{Q}_c}{\Delta H_c \rho (\pi D^2 / 4)} \tag{2.33}
\]
where \((\pi D^2/4)\) is the diameter, \(\dot{Q}_c\) the HRR, \(\Delta H_c\) the heat of combustion and \(D\) the fire diameter. Dimensionless HRR is a function of the Froude number and is in fire dynamics often replaced by the Froude number. The dimensionless heat release rate was first introduced by Zukoski and others and is expressed as

\[
Q^*_c = \frac{\dot{Q}_c}{\rho_\infty c_p T_\infty \sqrt{gDD^2}}
\]

where \(\rho\) is the density, \(c_p\) the heat capacity and \(T\) the temperature. The indexes, \(\infty\), represent the ambient value.

Flames dominated by high momentum (high Froude numbers or \(Q^*_c > 10^5\), region \(V\) in Figure 2.6) fuel release and in practice highly turbulent are called jet flames. Such flames can occur from accidental leaks in pressurized pipelines or vessels at process plants, where hydrocarbons are processed.

In natural fires, buoyancy caused by density difference between rising hot gases in the fire plume and ambient air is the dominant driving force. The buoyancy force, in contrast to the momentum from the fuel flow in jet flames, is resisted by viscous drag working in opposite direction, leads to shear stress in the flame sheet. As a consequence eddies are formed if the buoyancy force is large enough. In this process air is entrained in the fire plume and mixes fuel with the air. The flame itself is not influenced by the momentum from the fuel release as for jet flames. Natural fires are associated with relatively low velocity since the radiation from the flame is controlling the fuel release (evaporation or pyrolysis) [22]. As the fire diameter increase the radiation also increases, generally resulting in a larger and more turbulent flame.

### 2.3.1 Borghi Diagram

The Borghi diagram is a diagram for characterizing of flame regimes, seen in Figure 2.5. \(u'/u_L\) versus \(l_0/l_L\) is plotted with log-log axis where \(u'\) is the velocity fluctuation, \(u_L\) the laminar burning velocity, \(l_0\) the integral length scale and \(l_L\) the laminar flame thickness.

The diagram is divided by three diagonal lines which represents where the dimensionless numbers \(Da\), \(Ka\) and \(Re_T\) are unity. \(Re_l = Re_T^2\) where the turbulent Reynolds number is

\[
Re_l = \frac{u' l_0}{\nu}
\]

\(\nu\) is the kinematic viscosity \(\nu = \mu/\rho\). The turbulent Damköhler number, \(Da\), is large for a fast reaction and small for slow reaction. The number denotes the ratio between macroscopic time scale and the chemical times scale given by

\[
Da = \frac{t_0}{t_L} = \frac{l_0 u_L}{l_L u'}
\]

The turbulent Karlovitz number is expressed as

\[
Ka = \frac{t_L}{t_K}
\]
where $t_l = l_L/v_L$ and $t_K = \sqrt{\nu/\epsilon}$. $t_l$ and $t_K$ is the time scales of laminar flame and Kolomogorov, respectively.

In the laminar regime $Re_T > 1$, the flame sheet has a thin and a flat reaction zone. Moving in positive x-direction in the Borghi diagram, the turbulence is increasing forming a wrinkled flame front. In the turbulent regime where $Ka < 1$ is laminar burning fuel in the and air in the flame sheet is observed. Between $Da = 1$ and $Ka = 1$ the chemical time is larger than change of fluid motion [14]. Still, the chemical time is not so large compared to the fluid motion that island formations occur. Above $Da = 1$ the turbulence is so intense compared to the chemical time, allowing a perfect mix fuel and oxidant before the reaction occurs. This regime is referred as perfectly stirred reactor as in Section 2.5.3 in the EDC combustion model.

### 2.3.2 Flame Height

Flame height is an important parameter in fire safety engineering, for example to determine the distance where the radiant heat is sufficient to ignite combustible items. The
correlation by Heskestad for buoyancy flames is given by [23]

\[
\frac{L_f}{D} = 3.7(q_c^*)^{2/5} - 1.02
\]  

(2.38)

where \(L_f\) is the flame height and \(D\) is fire area diameter. The dimensionless heat release rate is given in eq. (2.34). The dimensionless flame length, \(l_f/D\), as a function of the dimensionless heat release rate is seen in Figure 2.6. Fire types are divided in five regions. Turbulent buoyancy driven diffusion flames are in regions I and II while jet flame is in region V. For natural fires (i.e not jet fires), \(Q^*\) is less than 10 in most situations [10].

Figure 2.6: The figure is showing the dimensionless flame length, \(l_f/D\), as a function of the dimensionless heat release rate and the Froude number [20].

### 2.3.3 Centerline Flame Temperature and Velocity

In 1979, McCaffery measured centerline temperature and velocity profiles above 0.3 m square burner [24]. The fuel was methane corresponding to five HRR ranging from 14 kW - 57 kW. McCaffery divided the fire plume in three regimes (in Figure 2.7):

- Persistent flame: accelerating flow
- Intermittent flame: nearly constant flow velocity
- Buoyant plume: decreasing velocity and temperature with respect to height
CHAPTER 2. THEORY

2.4 Heat Transfer

Heat can be transferred in three different ways: conduction, convection and radiation - all involved in a fire.
2.4.1 Conduction
Heat transfer through a solid or non-moving fluid, due to a temperature gradient, is called conduction. The physical nature of heat is vibrations of molecules, and spreads from regions of higher temperatures to regions of lower temperature, by collisions and diffusion of molecules. According to Fourier’s law of conduction, the steady state one-dimensional heat rate transfer through an area is proportional to the negative temperature gradient

\[ \dot{q}_x'' = -k \frac{dT}{dx} \]  \hspace{1cm} (2.41)

The thermal conductivity, \( k \), is a measure of how well a material can transfer heat. In transient phenomenas such as fires, the non-steady state conduction is expressed as

\[ \rho c_v \frac{\partial T}{\partial t} = \frac{\partial}{\partial x} \left( k \frac{\partial T}{\partial x} \right) \]  \hspace{1cm} (2.42)

where \( c_v \) is the heat capacity. Eq. (2.41) and (2.42) assumes constant thermal conductivity, heat capacity and density that in reality is a function of temperature.

2.4.2 Convection
Convection is heat exchange between a moving liquid or gas and a solid and is given by Newton’s law of cooling

\[ \dot{q}'' = h(T_s - T_\infty) \]  \hspace{1cm} (2.43)

where \( h \) is the convective heat transfer, \( T_s \) the temperature at the surface and \( T_\infty \) the surrounding temperature. From the ratio of convective heat transfer coefficient to conductive heat transfer, also known as the dimensionless Nusselt number, the convective heat transfer is determined

\[ Nu = \frac{hL}{k_f} \]  \hspace{1cm} (2.44)

\( L \) is the characteristic length and \( k_f \) conductivity of the fluid. The Nusselt number is depended on the fluid property, the thickness of boundary layer created by the shear stress and the flow property. Hence convection involving buoyancy-driven flows as a consequence of temperature gradients are referred as natural convection is distinguished from forced convection involving external forces.

Table 2.3: Empirical constants in McCaffery’s centerline temperature and velocity profile

<table>
<thead>
<tr>
<th>Region</th>
<th>( k )</th>
<th>( \eta )</th>
<th>( z/Q^{2/5} )</th>
<th>( C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persistent flame</td>
<td>6.8 m(^{1/2})/s</td>
<td>1/2</td>
<td>&lt;0.08</td>
<td>0.9</td>
</tr>
<tr>
<td>Intermittent flame</td>
<td>1.9 m/kW(^{1/5})/s</td>
<td>0</td>
<td>0.08-0.2</td>
<td>0.9</td>
</tr>
<tr>
<td>Buoyant plume</td>
<td>1.1 m(^{1/3})/kW(^{1/3})/s</td>
<td>-1/3</td>
<td>&gt;0.2</td>
<td>0.9</td>
</tr>
</tbody>
</table>
2.4.3 Radiation

Heat transferred by electromagnetic waves is called radiation. From Planck’s law, energy emitted by a black body per unit area at a given temperature, $T$, within a narrow band of wavelengths, $\lambda$, is

$$E_{b,\lambda} = \frac{2\pi c^2 h \lambda^{-5}}{\exp(ch/\lambda kT) - 1}$$

(2.45)

where $c$ is the speed of light, $h$ the Planck’s constant and $k$ the Boltzman constant. Integrating eq. (2.45) over the whole spectrum of wavelengths gives

$$E_b = \int_0^\infty E_{b,\lambda} d\lambda = \frac{2 \pi^5 k^4 T^4}{15c^2 h^3}$$

(2.46)

The emissivity, $\epsilon$, is not unity for surfaces other than for black bodies and is defined as

$$\epsilon = \frac{E_\lambda}{E_{b,\lambda}}$$

(2.47)

Rewriting eq. (2.46) gives

$$E = \epsilon \sigma T^4$$

(2.48)

where $\sigma$ is the Stefan-Boltzman constant.

2.5 Eddy Dissipation Concept (EDC)


The Eddy Dissipation Concept (EDC) was proposed by Magnusen and Hjertager and has been developed for decades [1][2][3]. EDC model assumes that the chemical reaction in a turbulent flow takes place where reactants and hot products are molecularly mixed. As described in Section 2.1.3, the molecular mixing occurs where the time of diffusion over the diameter of an eddy is shorter than the time an eddy takes to rotate 1/2 revolution. These locations are referred as fine structures and are also the place where TKE is dissipated into heat.

2.5.1 Energy Cascade

The EDC model is developed for RANS, which treats average values and is taken into account in the energy cascade shown in Figure 2.8. Mechanical energy, $w'$, from the mean flow is transferred to the largest eddies with a characteristic length scale $L'$, velocity scale $u'$ and a frequency or strain rate $\omega' = u'/L'$. On the next level in the cascade, the characteristic frequency is $\omega'' = 2\omega'$ and length $L''$. Further, on the $n$-th level, the characteristic scales are $\omega_n = 2\omega_{n-1}$, $L_n$ and $\omega_n = u_n/L_n$. The energy transfer continues down to the smallest level, equals Kolmogorov scale $\omega^*$, $u^*$ and $L^*$. A total
eddy dissipation for the whole cascade equal the energy transferred from the largest eddies, $\epsilon = q' + w''$. Production of TKE equals the mechanical energy transferred from the main flow to the largest eddies and is the source term in eq. (2.13) \( w' = \bar{\rho} u_i' u_j' \frac{\partial \bar{u}_i}{\partial x_j} \). Dissipation of TKE on the first level in the cascade is proportional to the viscosity and the strain rate:

\[
w'' = \frac{3}{2} C_D_1 \omega' 2 u'^2
\]  

(2.49)

and

\[
q' = C_D_2 \nu \omega'^2
\]  

(2.50)

$C_{D1}$ and $C_{D2}$ are model constants. It is assumed that $w$ and $q$ are equal on all levels [1]. By inserting $\omega'' = 2 \omega'$ or $\omega_n = 2 \omega_{n-1}$, the received and dissipated energy on $n$-th level yields

\[
w_n = \frac{3}{2} C_D_1 \omega_n u_n^2
\]  

(2.51)

and

\[
q_n = C_D_2 \nu \omega_n^2
\]  

(2.52)
An energy balance gives $w_n = q_n + w_{n+1}$. In turbulent flows with high Reynolds number $q_n \ll w_n$ and $w_n \approx w_{n+1}$ for small steps, $n$, and $w'^2 = \frac{1}{2}u'^2$ yields

$$w'' = \frac{3}{2}C_{D1}\omega' u'^2 = C_{D1}\omega' k$$

(2.53)

and

$$q' = C_{D2}\nu\omega'^2$$

(2.54)

On the final step of the cascade, in the fine structure, $w^* = q^*$ and

$$w^* = \frac{3}{2}C_{D1}\omega^* u^* u'^2$$

(2.55)

and

$$q^* = C_{D2}\nu\omega^* u^* u'^2$$

(2.56)

At one level in the cascade, the dissipation rate is 1/4 of the dissipation rate on the level below [13]. Nearly no turbulence energy is dissipated into heat in the largest eddies and according to the model, 3/4 of the dissipation takes place in the fine structure [1]. 98% of the dissipated heat takes places in the last three steps of the cascade [13]. In flows with high Reynolds number $\epsilon \approx w''$ and $\epsilon = \frac{4}{3}q^*$, since $w'$ is far greater than $q'$:

$$\epsilon = w'' = \frac{3}{2}C_{D1}u'^3$$

(2.57)

$$\epsilon = \frac{4}{3}q^* = \frac{4}{3}C_{D2}\nu u'^2$$

(2.58)

A balance for last level, the fine structure is

$$\epsilon = \frac{4}{3}w^* = 2C_{D1}\frac{u^* u'^3}{L^*}$$

(2.59)

Now, the characteristic scales for the fine structures may be expressed as

$$L^* = \frac{2}{3} \left( \frac{3C_{D2}^3}{C_{D1}^2} \right)^{1/4} \left( \frac{\nu^3}{\epsilon} \right)^{1/4}$$

(2.60)

$$u^* = \left( \frac{C_{D2}}{3C_{D1}^2} \right)^{1/4} (\nu\epsilon)^{1/4}$$

(2.61)

### 2.5.2 Fine Structures and Mass Exchange

The modeled ratio of fine structure mass to the total mass is

$$\gamma^* = \left( \frac{u^*}{w} \right)^3 = \left( \frac{3C_{D2}}{4C_{D1}} \right)^{3/4} \left( \frac{\nu\epsilon}{k^2} \right)^{3/4} = 9.8 \left( \frac{\nu\epsilon}{k^2} \right)^{3/4}$$

(2.62)
CHAPTER 2. THEORY

The ratio can be seen as a thin layer of fine structure, \( L^* \), on a cylinder representing an eddy of diameter \( L' \), where the ratio is \( \gamma^* = \frac{L^*}{L'} \) [13]. In between large eddies, regions of fine structures appears. The ratio of this mass to the total mass is

\[
\gamma = \frac{u^*}{u'} = \left( \frac{3C_d D_2}{4C_d D_1} \right)^{1/4} \left( \frac{\nu \epsilon}{k^2} \right)^{1/4} = 2.1 \left( \frac{\nu \epsilon}{k^2} \right)^{1/4} \tag{2.63}
\]

Mass exchange between the fine structure to the surroundings, divided on the mass of the fine structure is expressed as

\[
\dot{m}^* = 2 \frac{u^*}{L^*} = \left( \frac{3}{C_d D_1} \right)^{1/2} \left( \frac{\epsilon}{\nu} \right)^{1/2} = 2.5 \left( \frac{\epsilon}{\nu} \right)^{1/2} \tag{2.64}
\]

Then, the mass exchange between the fine structure to the surroundings, divided on the total mass must be \( \dot{n} = \dot{m}^* \gamma^* \):

\[
\dot{n} = \frac{3}{4C_d D_1} \left( \frac{12C_d D_2}{C_d D_1} \right)^{1/4} \left( \frac{\nu \epsilon}{k^2} \right)^{1/4} \left( \frac{\epsilon}{k} \right)^{1/2} = 24 \left( \frac{\nu \epsilon}{k^2} \right)^{1/4} \frac{\epsilon}{k} \tag{2.65}
\]

The duration of mixing must be long enough that the hot products starts the reaction between fuel and oxygen. This time is given by

\[
\tau^* = \frac{1}{\dot{m}^*} \tag{2.66}
\]

2.5.3 Perfectly Stirred Reactor (PSR)

![Diagram of reactor model](adapted from Ertesvåg, 2000 [13]).

The fine structures, where the reaction occurs, are treated as perfectly stirred reactor (PSR) which means that the reactants are perfectly mixed and the mass is constant at each time step [5]. The combustion in the reactor can be modeled with fast chemistry, equilibrium or chemical kinetics. Mass balance in the reactor may be expressed as

\[
\dot{M}_{in} Y_k^0 - \dot{M}_{out} Y_k^* = -R_k^* \frac{M_{FS}}{\rho^*} \tag{2.67}
\]

\[
\dot{M}_F \rho_i \left( Y_i \right) = \frac{\dot{M}_F \rho_i Y_i}{\rho^0} \tag{2.68}
\]
where \( R^*_k \) is the reaction rate of the fine structure, the superscripts 0 and * represents values for surroundings and fine structure, respectively (Figure 2.9). At stationary conditions \( \dot{M}_{in} = \dot{M}_{out} = \dot{M} \). Inserting \( \dot{m}^* = \frac{\dot{M}}{M_{FS}} \) in eq. (2.67) yields

\[
-R^*_k = \rho^* \dot{m}^*(Y^*_k - Y^*_k) \tag{2.68}
\]

The average reaction rate is

\[
\bar{R}_k = \left( R^*_k \frac{M_{FS}}{\rho^*} \right) \left( \frac{M_{tot}}{\bar{\rho}} \right)^{-1} = \frac{\bar{\rho}}{\rho^*} \gamma^* R^*_k \tag{2.69}
\]

Not all of the fine structures are reacting, so a reaction fraction is given by a probability function \( \chi \):

\[
\bar{R}_k = \frac{\bar{\rho}}{\rho^*} \chi \gamma^* R^*_k \tag{2.70}
\]

Using \( \dot{m} = \gamma^* \dot{m}^* \) and inserting eq. (2.68) in eq. (2.70) yields

\[
-\bar{R}_k = \frac{\bar{\rho} \dot{m} \chi}{1 - \gamma^* \chi} (Y^*_k - Y^*_k) \tag{2.71}
\]

As mentioned previously, RANS is treating average values. Therefore must eq. (2.71) must be expressed by an average mass fraction:

\[
-\bar{R}_k = \frac{\bar{\rho} \dot{m} \chi}{1 - \gamma^* \chi} \left( \tilde{Y}_k - Y^*_k \right) \tag{2.72}
\]

A detailed description of averaging is found in Turbulent Flow and Combustion by Ertesvåg [13]. When assuming fast chemistry, the reaction rate is

\[
-\bar{R}_F = \frac{\bar{\rho} \dot{m} \chi}{1 - \gamma^* \chi} Y_{min} \tag{2.73}
\]

where

\[
Y_{min} = \min \left( Y_F, \frac{Y_{O_2}}{s} \right); s = \frac{W_f}{\nu_{O_2} W_{O_2}} \tag{2.74}
\]

The probability function consists three parameters

\[
\chi = \chi_1 \cdot \chi_2 \cdot \chi_3 \tag{2.75}
\]

where \( \chi_1 \) is the probability of coexistence of reactants

\[
\chi_1 = \frac{\left( \tilde{Y}_{min} + \tilde{Y}_P/(1 + s) \right)^2}{\left( \tilde{Y}_F + \tilde{Y}_P/(1 + s) \right) \left( \tilde{Y}_{O_2}/s + \tilde{Y}_P/(1 + s) \right)} \tag{2.76}
\]

\( \chi_2 \) the degree of heating

\[
\chi_2 = \min \left[ \frac{1}{\gamma \lambda} \cdot \frac{\tilde{Y}_P/(1 + s)}{\tilde{Y}_P/(1 + s) + \tilde{Y}_{min}}, 1 \right] \tag{2.77}
\]

and \( \chi_3 \) the limitation due to lack of reactants

\[
\chi_2 = \min \left[ \frac{\gamma \lambda \left( \tilde{Y}_P/(1 + s) + \tilde{Y}_{min} \right)}{\tilde{Y}_{min}}, 1 \right] \tag{2.78}
\]
Chapter 3

Fire Dynamics Simulator (FDS)

The Fire Dynamics Simulator (FDS) is an open source CFD code used world wide in fire engineering applications and science. For almost 25 years FDS has been developed, mainly by the National Institute of Standards and Technology (NIST) and is today accepted as industrial standard. However, the first public release was in February 2000 [4]. The main purposes of FDS are to study smoke spread, smoke venting and activation of detectors in natural building fires.

FDS calculation consists three parts: pre-processing, processing and post-processing. In the pre-processing part, a text file is written in a plain text editor where input values as boundary conditions and other initial conditions as temperature, pressure, HRR are specified. The processing part is the calculation itself, solved with FDS. In Smokeview, the post-processor, animations and images of output values can be analyzed. Output values may also be exported to Excel, MatLab or similar softwares. Third party softwares exist for pre-processing and/or post-processing. The most popular commercial software is PyroSim.

FDS solves numerically the Navier-Stokes equations with low Mach number approximation, $Ma < 0.3$, on a (uniform) cartesian grid. This limits simulations to thermally-driven flows with low pressure differences. The equations are solved explicitly in second order central difference schemes in space and time [25]. Reynolds stresses are closed either with Deardorff SGS Model (current default), Smagorinsky SGS model (constant or dynamic coefficient) or Vreman (in the unofficial version 6).

3.1 Local Heat Release Rate (HRR) and Reaction Rate

The reaction rate in a combustion process is assumed to be limited by either the chemical reaction time (finite-rate reaction) or by mixing time of fuel, oxygen and hot products (infinitely fast reaction). When using LES the grid resolution is too coarse to compute the reaction rate directly because the flame sheet is much thinner than a grid cell [25]. Le combustion is a subgrid phenomenon and need to be modeled. If the condition in a grid cell meets certain criteria and the reactants of the reaction are present, the local
HRR is modeled as
\[
\dot{Q}_c = \rho \min \left( Y_F, \frac{Y_{O_2}}{s}, \beta \frac{Y_P}{1+s} \right) \left( 1 - e^{-\delta t / \tau} \right) \Delta H_c \tag{3.1}
\]
if simple chemistry (one-step reaction) and eddy dissipation is specified in the input file, else HRR is
\[
\dot{Q}_c = \rho \min \left( Y_F, \frac{Y_{O_2}}{s}, \beta \frac{Y_P}{1+s} \right) \frac{1}{\tau} \Delta H_c \tag{3.2}
\]
where \(Y_F, Y_{O_2}\) and \(Y_P\) are mass fractions of fuel, oxygen and products, respectively. The empirical parameter \(\beta\) is equal 1, \(\delta t\) is the time step and \(s = \frac{W_f}{\nu_{O_2} W_{O_2}}\), \(\tau\) is a characteristic mixing time given by
\[
\tau = \max \left( \tau_{chem}, \min \left( \tau_d, \tau_u, \tau_g, \tau_{flame} \right) \right) \tag{3.3}
\]
where the diffusion time scale, turbulent time scale and acceleration time scale are
\[
\tau_d = \frac{Sc \rho \Delta^2}{\mu}; \tau_u = \frac{\Delta}{\sqrt{2k_{sgs}}}; \tau_g = \sqrt{\frac{2\Delta}{g}} \tag{3.4}
\]
\(Sc\) is the dimensionless Schmidt number, defined as the ratio of viscous diffusion rate and molecular diffusion rate (default is 0.5 [25]). The subgrid kinetic energy, \(k_{sgs}\), is calculated by integrating a model Kolmogorov spectrum [25]. \(\tau_g\) is acceleration time scale and is the time required to travel a distance \(\Delta\) under a constant acceleration, \(g = 9.81 \text{ m/s}^2\). The flame time scale (large), \(\tau_{flame}\), and the chemistry time scale (small), \(\tau_{chem}\) may be specified by the user. They are however, rarely the limiting mixing time. The simple mixing model used in FDS is grid depended and will overestimate the HRR when a too coarse grid resolution is applied. An upper bound on the total local heat release rate per unit volume (HRRPUV) is set to 2500 kW/m³.

![Figure 3.1](image)

Figure 3.1: A description of how the reaction rate is computed in FDS.
Figure 3.2: The threshold temperature for a given oxygen concentration [26].

When assuming finite rate reaction, the reaction rate is computed by Arrhenius rate, used for DNS calculations and post-flame measurements. A detailed description is found in FDS Technical Reference Guide [25].

For the two-step reaction, fuel and oxygen are reacting infinitely fast (eq. (3.1)) forming CO, soot and other products if the temperature for a given oxygen concentration fulfills Figure 3.2. Further, CO is reacting infinitely fast with oxygen if any oxygen is left in the grid cell after the first reaction. If no fuel is traced in the first reaction or the temperature is too low to support combustion, but CO is present, a finite rate assumed. A visual description is seen in Figure 3.1.

### 3.2 Mulit-Parameter Mixture Fraction

With a single parameter mixture fraction model information of the completeness of the reaction is not stored [26]. In FDS v4 a single parameter mixture fraction, \( Z \), given in eq. (3.5) was used. The scalar represents the original mass of fuel before the combustion.

\[
Z = Y_F + \frac{W_F}{xW_{CO_2}} Y_{CO_2} + \frac{W_F}{xW_{CO}} Y_{CO} + \frac{W_F}{xW_s} Y_s
\]  

(3.5)

In well-ventilated fires the single scalar approach is sufficient. In other cases as extinctions and under-ventilated fires resulting in incomplete reaction, a single parameter is not sufficient. In order to get information of CO and CO\(_2\) production, multi-parameter mixture fraction model is necessary.
FDS supports

• two parameters single-step reaction: $Z_0$, $Z_1$ and $Z_2$

• three parameters two-step reaction: $Z_0$, $Z_1$, $Z_2$ and $Z_3$

where $Z = \sum Z_\alpha$. For a single step reaction fuel is reacting with oxygen and do not allow post combustion of CO. A fixed amount of CO is formed. In contrast, a two-step approach allows post combustion of CO within a hot upper layer in under-ventilated fires, post flame combustion or CO production due to fire suppression [25][26]. Note that a yield of CO ($\nu_{CO}$) is still needed to be specified when using a two-step reaction. Also the yields of CO$_2$ and soot ($\nu_{CO_2}, \nu_s$) in eq. (3.25) and (3.6) must be specified by the user, for either a complete or incomplete reaction.

### 3.2.1 Two parameter single-step reaction

For a single-step approach, the reaction is

$$C_xH_yO_zN_a + \nu'_{O_2}O_2 \rightarrow (\nu_{CO_2}CO_2 + \nu_{H_2O}H_2O + \nu_{CO})CO + \nu_sS + \nu_{o_2}N_2$$ (3.6)

where $x$ is the number of carbon atoms, $y$ the number of hydrogen atoms, $z$ the number of oxygen atoms and $a$ the number of nitrogen atoms. A general form of the reaction is

$$Fuel + \nu_{air} \rightarrow Products$$ (3.7)

The species are $Z_0$ for air, $Z_1$ for fuel and $Z_2$ for products. $Z_1$ and $Z_2$ are tracked explicitly while $Z_0$ is tracked implicitly as the background species. The mass fractions are given as [25]:

**Z$_0$: Air**

$$Y_{N_2}(Z_0) = Y_{N_2}^\infty$$ (3.8)

$$Y_{O_2}(Z_0) = Y_{O_2}^\infty$$ (3.9)

$$Y_{CO_2}(Z_0) = Y_{CO_2}^\infty$$ (3.10)

$$Y_{H_2O}(Z_0) = Y_{H_2O}^\infty$$ (3.11)

**Z$_1$: Fuel**

$$Y_F(Z_1) = Y_F$$ (3.12)

**Z$_2$: Products**

$$Y_{N_2}(Z_2) = \frac{\nu_{air}W_{air}Y_{N_2}^\infty + \nu_{N_2}W_{N_2}}{W_F + \nu_{air}W_{air}}$$ (3.13)

$$Y_{CO_2}(Z_2) = \frac{\nu_{air}W_{air}Y_{CO_2}^\infty + \nu_{CO_2}W_{CO_2}}{W_F + \nu_{air}W_{air}}$$ (3.14)

$$Y_{H_2O}(Z_2) = \frac{\nu_{air}W_{air}Y_{H_2O}^\infty + \nu_{H_2O}W_{H_2O}}{W_F + \nu_{air}W_{air}}$$ (3.15)

$$Y_{CO}(Z_2) = \frac{\nu_{CO}W_{CO}}{W_F + \nu_{air}W_{air}}$$ (3.16)

$$Y_S(Z_2) = \frac{\nu_sW_S}{W_F + \nu_{air}W_{air}}$$ (3.17)
CHAPTER 3. FIRE DYNAMICS SIMULATOR (FDS)

The species yields given as:

\[ Y_{\alpha}(Z_0, Z_1, Z_2) = Y_{\alpha}(Z_0)(1 - Z_1 - Z_2) + Y_{\alpha}(Z_1)Z_1 + Y_{\alpha}(Z_2)Z_2 \]  

(3.18)

Stoichiometric coefficients in \( Z_2 \) are:

\[ \nu_{N_2} = \frac{a}{2} \]  

(3.19)

\[ \nu_{O_2} = \nu_{CO_2} + \frac{\nu_{CO} + \nu_{H_2O}}{2} - \frac{z}{2} \]  

(3.20)

\[ \nu_{CO_2} = x - \nu_{CO} - (1 - X_H)\nu_S \]  

(3.21)

\[ \nu_{H_2O} = \frac{y}{2} - X_H\nu_S \]  

(3.22)

\[ \nu_{CO} = \frac{W_F}{W_{CO}}\nu_{CO} \]  

(3.23)

\[ \nu_S = \frac{W_F}{W_S}\nu_S \]  

(3.24)

### 3.2.2 Three parameter two-step reaction

The reactions for a two-step reaction are

\[ C_xH_yO_zNa + \nu'_{O_2}O_2 \rightarrow \nu_{H_2O}H_2O + (\nu'_{CO} + \nu_{CO})CO + \nu_sS + \nu_{n_2}N_2 \]  

(3.25)

\[ \nu'_{CO} \left[ CO + \frac{1}{2}O_2 \rightarrow CO_2 \right] \]  

(3.26)

The species in the two-step approach are \( Z_0 \) for air, \( Z_1 \) for fuel, \( Z_2 \) for products in the incomplete reaction one and \( Z_2 \) for products in the complete reaction two. A simplified way of writing the two-step reaction is

\[ Fuel + \nu_{Air,1}Air \rightarrow Incomplete \, Products \]  

(3.27)

\[ Incomplete \, Products + \nu_{Air,2}Air \rightarrow Complete \, Products \]  

(3.28)

The species yields are:

**\( Z_0 \): Air**

\[ Y_{N_2}(Z_0) = Y_{N_2}^\infty \]  

(3.29)

\[ Y_{O_2}(Z_0) = Y_{O_2}^\infty \]  

(3.30)

\[ Y_{CO_2}(Z_0) = Y_{CO_2}^\infty \]  

(3.31)

\[ Y_{H_2O}(Z_0) = Y_{H_2O}^\infty \]  

(3.32)

**\( Z_1 \): Fuel**

\[ Y_F(Z_1) = Y_F \]  

(3.33)
Z₂: Products of Incomplete Reaction

\[
Y_{N_2}(Z_2) = \frac{\nu_{air,1} W_{air} Y_{N_2}^\infty + \nu_{N_2} W_{N_2}}{W_F + \nu_{air} W_{air}}
\]

(3.34)

\[
Y_{CO_2}(Z_2) = \frac{\nu_{air} W_{air} Y_{CO_2}^\infty}{W_F + \nu_{air} W_{air}}
\]

(3.35)

\[
Y_{H_2O}(Z_2) = \frac{\nu_{air} W_{air} Y_{H_2O}^\infty + \nu_{H_2O} W_{H_2O}}{W_F + \nu_{air} W_{air}}
\]

(3.36)

\[
Y_{CO}(Z_2) = \frac{\nu_{CO} W_{CO}}{W_F + \nu_{air} W_{air}}
\]

(3.37)

\[
Y_S(Z_2) = \frac{\nu_S W_S}{W_F + \nu_{air} W_{air}}
\]

(3.38)

Z₃: Products of Complete Reaction

\[
Y_{N_2}(Z_3) = \frac{\nu_{air,2} W_{air} Y_{N_2}^\infty + \nu_{N_2} W_{N_2}}{W_F + \nu_{air} W_{air}}
\]

(3.39)

\[
Y_{CO_2}(Z_3) = \frac{\nu_{air} W_{air} Y_{CO_2}^\infty}{W_F + \nu_{air} W_{air}}
\]

(3.40)

\[
Y_{H_2O}(Z_3) = \frac{\nu_{air} W_{air} Y_{H_2O}^\infty + \nu_{H_2O} W_{H_2O}}{W_F + \nu_{air} W_{air}}
\]

(3.41)

\[
Y_{CO}(Z_3) = \frac{\nu_{CO} W_{CO}}{W_F + \nu_{air} W_{air}}
\]

(3.42)

\[
Y_S(Z_3) = \frac{\nu_S W_S}{W_F + \nu_{air} W_{air}}
\]

(3.43)

The stoichiometric coefficients are:

\[
\nu_{N_2} = \frac{a}{2}
\]

(3.44)

\[
\nu'_O_2 = \frac{\nu'_O_2 + \nu_{H_2O} - z}{2}
\]

(3.45)

\[
\nu_{O_2} = \frac{\nu_{CO_2} + \nu_{CO} + \nu_{H_2O} - z}{2}
\]

(3.46)

\[
\nu_{CO} = x - (1 - X_H)c
\]

(3.47)

\[
\nu_{M} = b
\]

(3.48)

\[
\nu_{H_2O} = \frac{y}{2} - X_H\nu_{S}
\]

(3.49)

\[
\nu'_CO = \frac{W_{CO} y_{CO}}{W_F}
\]

(3.51)

\[
\nu_S = \frac{W_F}{W_S}\nu_{S}
\]

(3.52)
CHAPTER 3. FIRE DYNAMICS SIMULATOR (FDS)

3.3 Extinction Criteria

The model for local extinction in FDS consists two parts. First, the local temperature in a grid cell has to be above auto-ignition temperature to the fuel. Second, the local temperature rise must exceed the lower flame temperature, $T_{LFT}$, from the energy released when maximum limiting available amount of fuel or oxygen, in terms of fuel are consumed:

$$\Delta Z_{\text{air}} h_{\text{air}}(T) + \Delta Z_{F} (h_{F}(T) + \Delta H_{F}) > \Delta Z_{\text{air}} h_{\text{air}}(T_{LFT}) + \Delta Z_{F} h_{F}(T_{LFT})$$  (3.53)

3.4 Grid Resolution

To find the correct grid resolution is not straightforward. It depends on input values in the calculation and no accepted manual to this exist in the CFD community. Bjarne P. Husted et al. recommended that the grid cell size should be in such order that 90-99% of the turbulent energy is solved on the grid points [27]. In FDS User’s Manual a somewhat lower value on at least 80% is recommended. Generally, the correct grid cell size is found by simulating the scenario with relative coarse grid and then gradually refine it until only minor differences in the results are observed. This procedure is called a grid sensitivity analysis. For low Mach number LES approximation, $D^*/dx$ is a measure on how well the flow field is resolved. The ratio $D^*/dx$ may also be used to predict the correct grid cell size. The non-dimensional fire diameter is

$$D^* = \left( \frac{Q}{\rho_{\infty} c_{p} T_{\infty} \sqrt{g}} \right)^\frac{3}{4}$$  (3.54)

and $dx$ is the grid cell size. The ratio $D^*/dx$ should be between 4 and 16 [25][4]. $T_{\infty}$ is the ambient temperature and $\rho_{\infty}$ is the ambient density. Since CFD modeling is time consuming, it is desirable to not use a finer grid than necessary to obtain satisfactory results. It is worth noting that doubling the number of nodes in each direction, reduces the discretization error by a factor of 4. Furthermore, the computing time increases by a factor of 16 (a factor of 2 for the temporal and each spatial dimension) [28]. To enhance the calculation with respect to time, the domain can be divided in several grids and Multi Processor Interface (MPI) may be applied to calculate the grids in parallel.
Chapter 4

Implementation of LES-EDC in FDS

4.1 LES-EDC

Extension of EDC to LES is not a straightforward task and requires customizing. In LES, partial turbulence cascading or no cascading occurs. However, the fine structure regions are calculated based on full energy cascading in each grid cell in RANS-EDC. Panjwani et al. proposed two approaches for formulating the fine structure regions [5][6]. In contrast to RANS, the turbulent kinetic energy and dissipation are rarely computed explicitly (i.e. no conservation) in LES. Thus, the fine structure regions are formulated from the subgrid viscosity. The first approach is based on volume fraction of small structures, the second on inhomogenous distribution of the fine structure regions, both ending up with the expression

\[
\gamma_\lambda = C_{LES} \left( \frac{\nu}{\nu_{sgs}} \right)^{\frac{1}{4}}
\]

(4.1)

where the subgrid viscosity, \(\nu_{sgs}\), is tracked from the turbulence model. In case of larger fine structure velocity than subgrid velocity, \(\gamma_\lambda < 1\) is used [5]. The proposed EDC-LES model assumes that the fine structures are localized in nearly constant energy regions so that \(\gamma^* = \gamma_\lambda^2\). Furthermore, the time scale is reformulated [29]:

\[
\tau^* = \frac{1}{|S|}
\]

(4.2)

where \(S\) is the strain rate. The reaction rate is computed as

\[
\omega_k = \frac{\gamma_\lambda^2 \chi}{\tau^*} \left( \tilde{Y}_k^0 - \tilde{Y}_k^* \right)
\]

(4.3)

where \(Y_k^0\) is the surrounding mass fraction and \(Y_k^*\) the mass fraction of the fine structures, see Figure 2.9. In terms of limiting concentration of fuel or oxygen, the reaction rate is

\[
\omega_k = \frac{\chi}{\tau^* (1 - \chi \gamma_\lambda^2)} Y_{min}
\]

(4.4)
where

\[ Y_{\min} = \min \left( Y_F, \frac{Y_{O_2}}{s} \right); \quad s = \frac{W_f}{v_{O_2} W_{O_2}} \]  

(4.5)

The probability function, \( \chi \), is computed as for RANS-EDC given in eq. (2.76) - (2.78).

### 4.2 Numerical Procedure

FDS is a non-commercial code written in FORTRAN 90/95 and is available on the website [http://code.google.com/p/fds-smv/](http://code.google.com/p/fds-smv/). Minor releases or subversions of FDS are accessed through a Subversion Client (SVN), while major releases are published on NIST’s official website ([http://fire.nist.gov/fds/](http://fire.nist.gov/fds/)).

The FDS code is divided into modules and subroutines as CFD in general. In this thesis the subroutine “Eddy Dissipation” (infinitely fast reaction, eq. (3.1)) in module "Fire" (fire.f90 Appendix B.1) is substituted with the EDC combustion model modified for LES described in section 4.1. In addition, other subroutines were adapted to changes implemented in the combustion model. A schematic diagram of the subroutines in the module "Fire" is presented in Figure 4.1.

![Module Fire](image)

Figure 4.1: Numerical Procedure in FDS combustion model (fire.f90 in Appendix B.1).

- **Combustion:** First, the upper HRR in the grid cell is calculated, set to 2500 kW/m³. If reactants are present, an ordinary differential equation (ODE) solver is
chosen. The temperature is calculated by the energy released from the combustion, which is reported back from the ODE solver. Mixture fractions are also reported back from the ODE solver before the divergence term is updated.

- **ODE solver**: If simple chemistry (single-step reaction) and "Eddy Dissipation" reaction rate is specified in the input file, the "Single Exact" ODE solver is chosen. Otherwise, "Explicit Euler" is chosen. Use of other ODE solvers must be specified in the input file. The ODE solver reads the number of reaction steps before finding the limiting reactant in terms of fuel. The mixing time is reported from the reaction rate mode and HRR is calculated from eq. (3.1) in the "Single Exact" solver or eq. (3.2) for the other solvers. Then fuel consumption is calculated.

- **Hybrid Reaction Rate Mode**: The "Mixed" mode is following the description in Figure 3.1 and 3.2. For DNS calculations, finite rate reaction is assumed. "Eddy_Dissipation_Concept" is calculating both "Finite Rate" (RATE\_CONSTANT\_FR) and "Eddy Dissipation" (RATE\_CONSTANT\_ED). The reaction rate is then

\[
\frac{\text{RATE}\_\text{CONSTANT}\_\text{ED}\times\text{RATE}\_\text{CONSTANT}\_\text{FR}}{\text{RATE}\_\text{CONSTANT}\_\text{ED}+\text{RATE}\_\text{CONSTANT}\_\text{FR}}.
\]

"Eddy Dissipation" is supporting co-existing of fuel and oxygen in the same grid cell by a function called extinction, see section 3.3.

### 4.3 The Implemented Code

The major part of the changes are done in the module "fire.f90", under "Eddy Dissipation":

- First the kinematic viscosity is calculated, \( \nu = \mu / \rho \), because it is not a global variable in FDS.
- The fine structure is computed by eq. (4.1), an upper value value is set \( \gamma_\lambda < 1 \).
- Mass fractions for oxygen and fuel are tracked from the subroutine GET\_MASS\_FRACTION, and mass fraction for products is found from the relation \( Y_{\text{products}} = 1 - (Y_{O_2} + Y_{fuel}) \).
- Stoichiometric coefficient is computed.
- The probability function is computed by eq. (2.75), (2.76), (2.77) and (2.78).
- Rate constant is computed by eq. (4.3).
\!The Eddy Dissipation Concept (EDC) Combustion Model (by Hjertager and Magnussen) for LES proposed by Balram et al.

\[ \text{NU} = \frac{\text{MU}(I,J,K)}{\text{RHO}(I,J,K)} \]

\[ C_{\text{LES}} = 0.15 \text{EB} \]

\[ \text{GAMMA}_L = C_{\text{LES}} \times \left( \frac{\text{NU}}{\text{NU}_{\text{EDDY}}(I,J,K)} \right) \times 0.25 \text{EB} \]

\[ \text{IF} (\text{GAMMA}_L > 1.0) \text{THEN} \]

\[ \text{GAMMA}_L = 1.0 \text{EB} \]

\[ \text{ENDIF} \]

\[ \text{CALL GET\_MASS\_FRACTION(ZZ\_GET, FUEL\_INDEX, Y\_FUEL)} \]

\[ \text{CALL GET\_MASS\_FRACTION(ZZ\_GET, O2\_INDEX, Y\_O2)} \]

\[ Y_{\text{PRODUCT}} = 1.0 - (Y_{\text{FUEL}} + Y_{\text{O2}}) \]

\[ S = \frac{\text{SPECIES(FUEL\_INDEX)\%MW}}{(\text{RN\%NU\_O2}\times\text{SPECIES(O2\_INDEX)\%MW})} \]

\[ Y_{\text{O2}} = \frac{Y_{\text{O2}}}{S} \]

\[ Y_{\text{PRODUCT}} = \frac{Y_{\text{PRODUCT}}}{(1.0S + Y_{\text{PRODUCT}})} \]

\[ \text{CHI}_1 = \frac{(Y_{\text{FLIM}} + Y_{\text{PRODUCT}})^2}{(Y_{\text{FUEL}} + Y_{\text{PRODUCT}})} \]

\[ \text{CHI}_2 = \text{MIN}(Y_{\text{PRODUCT}}/(\text{GAMMA}_L(1.0 - Y_{\text{FUEL}}))) \]

\[ \text{CHI}_3 = \text{MIN}(\text{GAMMA}_L(1.0 - Y_{\text{FUEL}})) \]

\[ \text{CHI} = \text{CHI}_1 \times \text{CHI}_2 \times \text{CHI}_3 \]

\[ \text{RATE\_CONSTANT} = \frac{YY_{\text{FLIM}} \times \text{CHI}}{(\text{MIX\_TIME}(I,J,K) \times (1.0 - \text{CHI} \times \text{GAMMA}_L)**2)} \]

\[ !\text{RATE\_CONSTANT} = \frac{YY_{\text{FLIM}} \times \text{CHI} \times \text{GAMMA}_L**2}{(\text{MIX\_TIME}(I,J,K) \times (1.0 - \text{CHI} \times \text{GAMMA}_L)**2)} \]

The mixing time (eq. 4.2) is computed further up in the module. Other changes are done in modules "velo.f90", "mesh.f90" and "init.f90". Since the strain rate is only computed in the Smagorinsky turbulence model (eq. (2.18)), the subroutine COMPUTE\_STRAIN\_RATE is added in the subroutine COMPUTE\_VISCOSITY for the Deardorff (eq. (2.19)) and Vreman (eq. (2.20)) turbulence model in "velo.f90". The strain rate and the subgrid viscosity are made global variables by modifications in "mesh.f90" and "init.f90". Furthermore, parts of the code are commented out, so that they do not conflict with the implemented part. The ODE solve "Single Exact" is
adapted to eq. (3.1) only, and are for that reason substituted by the "Explicit Euler". Limiting mass fraction in terms of fuel, $Y_{\text{min}}$, is part of the original code. The module fire.f90 is found in its full length in Appendix B.1. Other modified subroutines are found in Appendix B whiles their full length is found at http://code.google.com/p/fds-smv/.
Chapter 5

Validation of FDS-EDC

An important aspect when implementing a new model in a CFD model is to validate this. Validation can be done by comparing simulations with experiments or other numerical tools that computes the exact physics instead of modeling it (typically DNS). In some cases CFD models are validated with flows which can be solved analytically, e.g. velocity profile in Couette flow.


- The process of determining the degree to which a calculation method is an accurate representation of the real world from the perspective of the intended uses of the calculation method.

and verification as

- The process of determining that the implementation of a calculation method accurately represents the developer’s conceptual description of the calculation method and the solution to the calculation method.

In FDS verification guide it is putted in a simple way and stated that verification is to check the math and validation to check the physics [31]. The choice of validation cases must be within the limits of CFD code and related to its purpose. In the matter of combustion model validation the quantities must be related to the local HRR. The energy released in the combustion increases the temperature, and through thermal expansion and buoyancy caused by density differences the flow is forced in an upward direction. The type of fire must equal those FDS is limited to and about the same size. There are several fire scenarios to consider and many ways to validate a CFD code. The selected scenarios in this thesis are identical to some of those in FDS validation guide [32], so that results in this thesis may in further work be compared with earlier and future versions of FDS. Based on the given arguments McCaffery centerline velocity and temperature profile [24], Heskestad flame height correlation [23] and velocity profile in
CHAPTER 5. VALIDATION OF FDS-EDC

Sandia plume experiments [33][34] are considered in this thesis. Temperature, velocity and flame height are important quantities for flames and are all influenced by local HRR which is calculated in the implemented code. Flame height validation is a way to check if the fuel consumed over the correct travelled distance while the center line temperature and velocity profile is a way to check the local fuel consumption is correct. Even though all the fuel is consumed after traveling the distance equal the flame height the local fuel consumption do not necessarily has to be correct. Correct modeling of the lower flame region is crucial to be able to model a flame correctly. The lower flame region, called the persistent flame, is where the flow is accelerating. Above, in the intermittent region, the flow is nearly constant. An over prediction of HRR in the lower region leads to an over prediction of the velocity, hence the height of the persistent region decreases. Furthermore, the temperature is over predicted in the lower region and the flame height under predicted. An under prediction of HRR in the persistent region leads to the opposite. By velocity profiles from Sandia experiments the persistent flame region may be studied in detail. In Section 6.1, information regarding the philosophy of arranging experiments for CFD validation is found.

In total, 720 simulations are presented in this thesis: 480 simulations for the Hestekstad correlation, 150 simulations for the McCaffery correlation and 90 simulations for the Sandia plume experiments. All simulations were run with FDS v.6, SVN revision number 10231 on six different computers:

- a) 2 x Intel(R) Xeon(R) CPU X5690 Hexa Core 3.47 GHz, (48 GB memory)
- b) 2 x Intel(R) Xeon(R) CPU X5690 Hexa Core 3.47 GHz, (48 GB memory)
- c) Intel(R) Xeon(R) CPU X5570 Quad Core 2.93 GHz, (12 GB memory)
- d) Intel(R) Xeon(TM) CPU Dual Core 3.0 GHz, (4 GB memory)
- e) 2 x Intel(R) Xeon(R) CPU E5540 Quad Core 2.53 GHz, (24 GB memory)
- f) 2 x Intel(R) Xeon(R) CPU E5630 Quad Core 2.53 GHz, (8 GB memory)

The Sandia plume experiments were simulated with 4 CPU cores with computer a) and b) to be able to compare CPU clock times\(^1\). Because of time constraint, results for the original version FDS v.6 (only referred as FDS6) were only compared with Deardorff turbulence model. Deardorff was chosen since that is the default turbulence model in the unofficial version 6 of FDS. Input for all simulations were limited to a two parameters mixture fraction with a single-step reactions for infinitely fast chemistry.

Example of input files are found in Appendix A. All input files in its full length are found at http://code.google.com/p/fds-smv/. Results are presented with three different $C_{LES}$ and three different grid resolutions.

\(^1\)Test 17 with Vreman turbulence model ($dx = 1.5$ cm) was simulated with 12 CPU cores and stopped at 15 seconds because of computer crash and time constraint in the final stages of the project.
5.1 McCaffery’s Plume Correlation, Velocity and Temperature Profiles

The McCaffery case is simulated with a 30 cm x 30 cm methane burner with HRR of 14 kW, 22 kW, 33 kW, 45 kW and 57 kW. Three different grids (30x30x100, 60x60x200 and 40x40x100) are applied with varying computational domain size such that \( D^*/dx = 5 \), \( D^*/dx = 10 \) and \( D^*/dx = 20 \). This corresponds a range of dimensionless heat release from \( Q^* = 0.005 \) to \( Q^* = 0.02 \). The centerline velocity and temperature profiles are average over the 30 seconds simulated. Ambient conditions are 20°C and atmospheric pressure. Results are presented for Deardorff, Vreman and (dynamic) Smagorinsky turbulence models with \( C_{LES} \) of 0.005, 0.01 and 0.015. An example of input file is found in Appendix A.5.
Figure 5.1: McCaffery correlation 14 kW, centerline velocity (left side) and temperature (right side) profiles with Deardorff turbulence model. Coarse grid at the top, medium grid in the middle and fine grid at the bottom.
Figure 5.2: McCaffery correlation 22 kW, centerline velocity (left side) and temperature (right side) profiles with Deardorff turbulence model. Coarse grid at the top, medium grid in the middle and fine grid at the bottom.
Figure 5.3: McCaffery correlation 33 kW, centerline velocity (left side) and temperature (right side) profiles with Deardorff turbulence model. Coarse grid at the top, medium grid in the middle and fine grid at the bottom.
Figure 5.4: McCaffery correlation 45 kW, centerline velocity (left side) and temperature (right side) profiles with Deardorff turbulence model. Coarse grid at the top, medium grid in the middle and fine grid at the bottom.
Figure 5.5: McCaffery correlation 57 kW, centerline velocity (left side) and temperature (right side) profiles with Deardorff turbulence model. Coarse grid at the top, medium grid in the middle and fine grid at the bottom.
Figure 5.6: McCaffery correlation 14 kW, centerline velocity (left side) and temperature (right side) profiles with Smagorinsky turbulence model. Coarse grid at the top, medium grid in the middle and fine grid at the bottom.
Figure 5.7: McCaffery correlation 22 kW, centerline velocity (left side) and temperature (right side) profiles with Smagorinsky turbulence model. Coarse grid at the top, medium grid in the middle and fine grid at the bottom.
CHAPTER 5. VALIDATION OF FDS-EDC

Figure 5.8: McCaffery correlation 33 kW, centerline velocity (left side) and temperature (right side) profiles with Smagorinsky turbulence model. Coarse grid at the top, medium grid in the middle and fine grid at the bottom.
Figure 5.9: McCaffery correlation 45 kW, centerline velocity (left side) and temperature (right side) profiles with Smagorinsky turbulence model. Coarse grid at the top, medium grid in the middle and fine grid at the bottom.
Figure 5.10: McCaffery correlation 57 kW, centerline velocity (left side) and temperature (right side) profiles with Smagorinsky turbulence model. Coarse grid at the top, medium grid in the middle and fine grid at the bottom.
Figure 5.11: McCaffery correlation 14 kW, centerline velocity (left side) and temperature (right side) profiles with Vreman turbulence model. Coarse grid at the top, medium grid in the middle and fine grid at the bottom.
Figure 5.12: McCaffery correlation 22 kW, centerline velocity (left side) and temperature (right side) profiles with Vreman turbulence model. Coarse grid at the top, medium grid in the middle and fine grid at the bottom.
Figure 5.13: McCaffery correlation 33 kW, centerline velocity (left side) and temperature (right side) profiles with Vreman turbulence model. Coarse grid at the top, medium grid in the middle and fine grid at the bottom.
Figure 5.14: McCaffery correlation 45 kW, centerline velocity (left side) and temperature (right side) profiles with Vreman turbulence model. Coarse grid at the top, medium grid in the middle and fine grid at the bottom.
Figure 5.15: McCaffery correlation 57 kW, centerline velocity (left side) and temperature (right side) profiles with Vreman turbulence model. Coarse grid at the top, medium grid in the middle and fine grid at the bottom.
Figure 5.16: McCaffery correlation, centerline velocity (left side) and temperature (right side) profiles with FDS6 Deardorff turbulence model.
Figure 5.17: McCaffery correlation, centerline velocity (left side) and temperature (right side) profiles with FDS6 Deardorff turbulence model.
CHAPTER 5. VALIDATION OF FDS-EDC

5.2 Heskestad Flame Height Correlation

The different fire cases for Heskestad validation is found in Table 5.1. Propane is used as fuel with $0.1 \leq Q^* \leq 10000$ at atmospheric pressure and $20 \, ^\circ C$. Three different grids; $17 \times 17 \times 40$, $33 \times 33 \times 80$ and $65 \times 65 \times 160$ and three different $C_{LES}$: 0.005, 0.01 and 0.015 are simulated for the turbulence models Deardorff, (dynamic) Smagorinsky and Vreman. $D^*$ and HRR is presented in Table 5.1.

Table 5.1: Heskestad Flame Height Simulations [32].

<table>
<thead>
<tr>
<th>$Q^*$</th>
<th>$Q$ [kW]</th>
<th>$D^*$ [-]</th>
<th>$dx_{10}$ [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>151</td>
<td>0.45</td>
<td>0.045</td>
</tr>
<tr>
<td>0.2</td>
<td>303</td>
<td>0.59</td>
<td>0.059</td>
</tr>
<tr>
<td>0.5</td>
<td>756</td>
<td>0.86</td>
<td>0.086</td>
</tr>
<tr>
<td>1</td>
<td>1513</td>
<td>1.13</td>
<td>0.113</td>
</tr>
<tr>
<td>2</td>
<td>3025</td>
<td>1.49</td>
<td>0.149</td>
</tr>
<tr>
<td>5</td>
<td>7564</td>
<td>2.15</td>
<td>0.215</td>
</tr>
<tr>
<td>10</td>
<td>15127</td>
<td>2.84</td>
<td>0.284</td>
</tr>
<tr>
<td>20</td>
<td>30255</td>
<td>3.75</td>
<td>0.375</td>
</tr>
<tr>
<td>50</td>
<td>75636</td>
<td>5.40</td>
<td>0.540</td>
</tr>
<tr>
<td>100</td>
<td>151273</td>
<td>7.13</td>
<td>0.713</td>
</tr>
<tr>
<td>200</td>
<td>302545</td>
<td>9.41</td>
<td>0.941</td>
</tr>
<tr>
<td>500</td>
<td>756363</td>
<td>13.6</td>
<td>1.36</td>
</tr>
<tr>
<td>1000</td>
<td>1512725</td>
<td>17.9</td>
<td>1.79</td>
</tr>
<tr>
<td>2000</td>
<td>3025450</td>
<td>23.6</td>
<td>2.36</td>
</tr>
<tr>
<td>5000</td>
<td>7563625</td>
<td>34.1</td>
<td>3.41</td>
</tr>
<tr>
<td>10000</td>
<td>15127250</td>
<td>45.0</td>
<td>4.50</td>
</tr>
</tbody>
</table>

In the validation of flame height by Heskestad correlation, outputs from the simulations must be post-processed to predict the flame height. A FORTRAN programme for this is available at http://code.google.com/p/fds-smv/. The output from the simulation is heat release per unit length ($HRR_{PUL} = \int \dot{q}'^\prime dx dy$). The flame height, $L_f$, is assumed to be where 99% of fuel is consumed on average. This is the same as the height where $\sum_{L_f} HRR_{PUL} = 0.99 \cdot \sum_{0}^\infty HRR_{PUL}$. In line 107 in Appendix C the flame height is found through interpolation.
Figure 5.18: Heskestad Flame Height Correlation Deardorff Turbulence Model.
Figure 5.19: Heskestad Flame Height Correlation Smagorinsky Turbulence Model.
Figure 5.20: Heskestad Flame Height Correlation, Vreman Turbulence Model.
Figure 5.21: Heskestad Flame Height Correlation FDS6 compared with FDS-EDC, Deardorff Turbulence Model.
5.3 Sandia Plume

Sandia Plume experiments are specifically designed for validating CFD models involving fire plumes. The Fire Laboratory for Accreditation of Models by Experimentation (FLAME) facility is located in New Mexico and is where the experiments are performed by Tieszen et al. [33]. The experiments are arranged with a 0.5 m steel plane surrounding the fire area of 1 m in diameter. Average velocities profiles are measured in heights of 0.3 m, 0.5 m and 0.9 m above the fire (see Figure 5.59) with Planar Laser Induced Fluorescence (PLIF).

Simulations is compared with the average velocity profiles between 10-20 seconds in the simulations. The computational domain size is 3 m x 3 m x 4 m meters divided in uniform rectangular grid cell of 1.5 cm, 3.0 cm and 6.0 cm. This corresponds to grid resolutions of 192 x 192 x 256, 96 x 96 x 128 and 48 x 48 x 64 respectively. The fuel releases were 0.04 kg/m²s (test 14), 0.053 kg/m²s (test 24) and 0.066 kg/m²s (test 17). Dimensionless HRR and fire diameter is given in Table 5.2. Input for all cases are specified after the actual test conditions. Example of inputs for the fires are seen in Appendix A.1-A.3 [33].

<table>
<thead>
<tr>
<th>Testnumber</th>
<th>Q</th>
<th>Q*</th>
<th>D*</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>1590</td>
<td>1.35</td>
<td>1.13</td>
</tr>
<tr>
<td>17</td>
<td>2610</td>
<td>2.22</td>
<td>1.38</td>
</tr>
<tr>
<td>24</td>
<td>2070</td>
<td>1.76</td>
<td>1.25</td>
</tr>
</tbody>
</table>
5.3.1 Methane Fire, Test 14

Figure 5.22: Sandia plume experiment test 14 with Deardorff turbulence model at $z = 0.3$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid dx = 6 cm, dx = 3 cm in the middle and dx = 1.5 cm at the bottom.
Figure 5.23: Sandia plume experiment test 14 with Deardorff turbulence model at z = 0.5 m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid dx = 6 cm, dx = 3 cm in the middle and dx = 1.5 cm at the bottom.
Figure 5.24: Sandia plume experiment test 14 with Deardorff turbulence model at z = 0.9 m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid dx = 6 cm, dx = 3 cm in the middle and dx = 1.5 cm at the bottom.
Figure 5.25: Sandia plume experiment test 14 with Smagorinsky turbulence model at $z = 0.3$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid $dx = 6$ cm, $dx = 3$ cm in the middle and $dx = 1.5$ cm at the bottom.
Figure 5.26: Sandia plume experiment test 14 with Smagorinsky turbulence model at z = 0.5 m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid dx = 6 cm, dx = 3 cm in the middle and dx = 1.5 cm at the bottom.
Figure 5.27: Sandia plume experiment test 14 with Smagorinsky turbulence model at $z = 0.9$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid $dx = 6$ cm, $dx = 3$ cm in the middle and $dx = 1.5$ cm at the bottom.
Figure 5.28: Sandia plume experiment test 14 with Vreman turbulence model at $z = 0.3$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid $dx = 6$ cm, $dx = 3$ cm in the middle and $dx = 1.5$ cm at the bottom.
Figure 5.29: Sandia plume experiment test 14 with Vreman turbulence model at z = 0.5 m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid dx = 6 cm, dx = 3 cm in the middle and dx = 1.5 cm at the bottom.
Figure 5.30: Sandia plume experiment test 14 with Vreman turbulence model at $z = 0.9$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid $dx = 6$ cm, $dx = 3$ cm in the middle and $dx = 1.5$ cm at the bottom.
Figure 5.31: Sandia plume experiment test 14 with FDS6. Vertical velocity to the left and radial velocity to the right. On the top at $z = 0.3$ m, $z = 0.5$ m in the middle and $z = 0.9$ m at the bottom.
5.3.2 Methane Fire, Test 17

Figure 5.32: Sandia plume experiment test 17 with Deardorff turbulence model at $z = 0.3$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid $dx = 6$ cm, $dx = 3$ cm in the middle and $dx = 1.5$ cm at the bottom.
Figure 5.33: Sandia plume experiment test 17 with Deardorff turbulence model at $z = 0.5$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid $dx = 6$ cm, $dx = 3$ cm in the middle and $dx = 1.5$ cm at the bottom.
Figure 5.34: Sandia plume experiment test 17 with Deardorff turbulence model at $z = 0.9$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid $dx = 6$ cm, $dx = 3$ cm in the middle and $dx = 1.5$ cm at the bottom.
Figure 5.35: Sandia plume experiment test 17 with Smagosinsky turbulence model at z = 0.3 m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid dx = 6 cm, dx = 3 cm in the middle and dx = 1.5 cm at the bottom.
CHAPTER 5. VALIDATION OF FDS-EDC

Figure 5.36: Sandia plume experiment test 17 with Smagorinsky turbulence model at $z = 0.5$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid $dx = 6$ cm, $dx = 3$ cm in the middle and $dx = 1.5$ cm at the bottom.
Figure 5.37: Sandia plume experiment test 17 with Smagorinsky turbulence model at \( z = 0.9 \) m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid \( dx = 6 \) cm, \( dx = 3 \) cm in the middle and \( dx = 1.5 \) cm at the bottom.
Figure 5.38: Sandia plume experiment test 17 with Vreman turbulence model at $z = 0.3$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid $dx = 6$ cm, $dx = 3$ cm in the middle and $dx = 1.5$ cm at the bottom.
Figure 5.39: Sandia plume experiment test 17 with Vreman turbulence model at $z = 0.5$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid $dx = 6$ cm, $dx = 3$ cm in the middle and $dx = 1.5$ cm at the bottom.
Figure 5.40: Sandia plume experiment test 17 with Vreman turbulence model at z = 0.9 m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid dx = 6 cm, dx = 3 cm in the middle and dx = 1.5 cm at the bottom.
Figure 5.41: Sandia plume experiment test 17 with FDS6. Vertical velocity to the left and radial velocity to the right. On the top at $z = 0.3$ m, $z = 0.5$ m in the middle and $z = 0.9$ m at the bottom.
5.3.3 Methane Fire, Test 24

Figure 5.42: Sandia plume experiment test 24 with Deardorff turbulence model at $z = 0.3$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid $dx = 6$ cm, $dx = 3$ cm in the middle and $dx = 1.5$ cm at the bottom.
Figure 5.43: Sandia plume experiment test 24 with Deardorff turbulence model at $z = 0.5$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid $dx = 6$ cm, $dx = 3$ cm in the middle and $dx = 1.5$ cm at the bottom.
Figure 5.44: Sandia plume experiment test 24 with Deardorff turbulence model at z = 0.9 m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid dx = 6 cm, dx = 3 cm in the middle and dx = 1.5 cm at the bottom.
Figure 5.45: Sandia plume experiment test 24 with Smagorinsky turbulence model at $z = 0.3$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid $dx = 6$ cm, $dx = 3$ cm in the middle and $dx = 1.5$ cm at the bottom.
Figure 5.46: Sandia plume experiment test 24 with Smagosinky turbulence model at z = 0.5 m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid dx = 6 cm, dx = 3 cm in the middle and dx = 1.5 cm at the bottom.
Figure 5.47: Sandia plume experiment test 24 with Smagorinsky turbulence model at $z = 0.9$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid $dx = 6$ cm, $dx = 3$ cm in the middle and $dx = 1.5$ cm at the bottom.
Figure 5.48: Sandia plume experiment test 24 with Vreman turbulence model at $z = 0.3$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid $dx = 6$ cm, $dx = 3$ cm in the middle and $dx = 1.5$ cm at the bottom.
Figure 5.49: Sandia plume experiment test 24 with Vreman turbulence model at $z = 0.5$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid $dx = 6$ cm, $dx = 3$ cm in the middle and $dx = 1.5$ cm at the bottom.
Figure 5.50: Sandia plume experiment test 24 with Vreman turbulence model at $z = 0.9$ m. Vertical velocity to the left and radial velocity to the right. On the top the coarse grid $dx = 6$ cm, $dx = 3$ cm in the middle and $dx = 1.5$ cm at the bottom.
Figure 5.51: Sandia plume experiment test 24 with FDS6. Vertical velocity to the left and radial velocity to the right. On the top at $z = 0.3$ m, $z = 0.5$ m in the middle and $z = 0.9$ m at the bottom.
5.3.4 CPU clock time

Figure 5.52: From top to bottom; CPU clock time for Sandia test 14, test 24 and test 17 with \( dx = 1.5 \) cm. Divided in groups of \( C_{LES} \).
Figure 5.53: From top to bottom; CPU clock time for Sandia test 14, test 24 and test 17 with $dx = 3$ cm. Divided in groups of $C_{LES}$. 
Figure 5.54: From top to bottom; CPU clock time for Sandia test 14, test 24 and test 17 with $dx = 6\text{ cm}$. Divided in groups of $C_{LES}$. 
Figure 5.55: Comparison of CPU clock time between FDS6 and FDS-EDC for Sandia simulations. At the top $dx = 1.5$ cm, in the middle $dx = 3$ cm and $dx = 6$ cm at the bottom. Divided in groups of test 14, test 24 and test 17.
CHAPTER 5. VALIDATION OF FDS-EDC

5.4 Discussion

5.4.1 McCaffery’s Plume Correlation, Velocity and Temperature Profiles

In the McCaffery simulation the connection between centerline temperature and velocity are investigated when \( C_{LES} \) and grid resolutions are varied. With the fine grid the velocity profile is acceptably predicted with all turbulence models for all \( C_{LES} \). The main difference is the peak value which is under estimated for \( C_{LES} = 0.005 \). However, the temperature is strongly under predicted just above the burner \((z < 1)\) for \( C_{LES} = 0.005 \). The lower region right above the burner, approximately below 1 m where the velocity and temperature peak is located, is the most challenging area to model correctly. Particularly the peak temperature and its location. It is meaningless to quantify the error for the the temperature profile because of the steep slope. The error is large in this area, but the curves have the same shape and are just displace. For all the turbulence models, the main difference in change of \( C_{LES} = 0.01 \) to \( C_{LES} = 0.015 \) is the maximum temperature.

Simulations with Deardorff turbulence model show that \( C_{LES} = 0.005 \) fits best for the velocity profile for 14 kW and 22 kW (5.1 and 5.2). As the HRR is increased \( C_{LES} = 0.01 \) and \( C_{LES} = 0.015 \) gives better results for the velocity even though the velocity peak is over estimated. With the finest grid, the temperature peaks are located too far away from the burner and becomes larger as the HRR is increased. This displacement is largest for \( C_{LES} = 0.005 \). For the other \( C_{LES} \) the error is about 50-70\%. But for \( C_{LES} = 0.015 \) the peak value are strongly over predicted with up to 600°C (57 kW in Figure 5.5). Because of the over prediction, \( C_{LES} = 0.01 \) is the most correct constant for the Deardorff model in total.

The Smagorinsky turbulence model predicts the temperature profiles in McCaffery simulations better than Deardorff and Vreman. The temperature profile is in excellent agreement for \( C_{LES} = 0.01 \) with the finest grid. The under predictions of the velocity profiles are larger with the Smagorinsky model, particularly for the two highest HRR of 45 kW and 57 kW. However, the error in maximum temperatures for both \( C_{LES} = 0.01 \) and \( C_{LES} = 0.015 \) are not as over predicted as for the other models. The results for velocity is almost identical for the coarse grid when \( C_{LES} \) is changed.

The errors in maximum temperature for the Vreman turbulence model with \( C_{LES} = 0.015 \) are the smallest. With \( C_{LES} = 0.01 \) the errors are about the same as for the other models.

In Figures 5.16 and 5.17 the displacement of temperature peak value and the location for FDS6 may be seen. The maximum temperatures are not over estimated as much as for FDS-EDC with \( C_{LES} = 0.015 \) (Deardorff) for the three scenarios with largest HRR. The under estimation of velocity profiles is slightly larger for FDS6 than for FDS-EDC in contrast to the peak value, which more correctly predicted.

In Figure 5.56, Temperature and velocity contours for McCaffery 57 kW with Deardorff turbulence model is presented. This figure can be seen in connection with Figure 5.58 where the contour plot of HRRPUV is seen. Even though the fire size not equal the principle is the same. For the lowest \( C_{LES} = 0.005 \) the HRRPUV is under estimated
and smeared out over a longer travelled distance than the flame height. This leads to an under estimation of velocity and temperature. For the flame height $C_{LES} = 0.015$ is in best agreement with Heskestad correlation while the temperature and velocity profiles with $C_{LES} = 0.01$ is in best agreement with McCaffery correlation. I.e the same value of the constant do not necessarily has to fit for flame height, velocity and temperature profile, even for the same fire size.

### 5.4.2 Heskestad Flame Height Correlation

Figures 5.18-5.21 show that FDS-EDC for all turbulence models are strongly grid dependent and are over predicting the flame height for coarse and medium grid resolution ($D^*/dx = 5$ and $D^*/dx = 10$). A constant can be established for all turbulence models to be in good agreement with the Heskestad correlation for large $Q^*$ with the fine grid ($D^*/dx = 20$). Deardorff and Vreman turbulence model do not capture the slope for low $Q^*$ as well as the Smagorinsky, especially not for the coarse grid. With the fine grid and
$C_{LES} = 0.015$, Vreman gives the worst results for $Q^* < 1$ with error ranging from -27% to 50% in contrast to Smagorinsky which captures the slope excellently. Except from the Smagorinsky model, the flame height is little influenced by the model constant for the coarse grid. However, the results are most sensitive to changes in $C_{LES}$ for low $Q^*$. 

![Figure 5.57: Comparison of percentages error in Heskestad flame height correlation with Smagorinsky turbulence model.](image)

$C_{LES} = 0.015$ gives results in best agreement with the Heskestad flame height correlation for the Deardorff turbulence model (in Figure 5.18). With $D^*/dx = 20$ the maximum error is 27% for $Q^* < 1$ (except when $Q^* = 0.5$) else it is 12%, and for $D^*/dx = 10$ the maximum error is 25% in the whole range of $Q^*$.

It is more difficult to establish a constant for the Smagorinsky turbulence model with the fine grid in Figure 5.19. The Smagorinsky model is the model that is most sensitive to changes in the model constant with $D^*/dx = 5$ and $D^*/dx = 10$. By plotting the percentage error as in Figure 5.57, it can be seen that both $C_{LES} = 0.015$ and $C_{LES} = 0.01$ gives excellent results for $D^*/dx = 20$, even for low $Q^*$. With $C_{LES} = 0.015$ the absolute error is less than 14.7%. For medium grid resolution $D^*/dx = 10$, $C_{LES} = 0.01$ predicts the overall flame height slightly better than $C_{LES} = 0.015$, particularly for $2 > Q^*$. Both constants under estimate the flame height for the lowest $Q^*$ with medium grid resolution.

With $C_{LES} = 0.015$ for the Vreman turbulence model (Figure 5.20), the flame height
Figure 5.58: Slice file for heat release per unit volume (HRRPUV) versus flame height averaged from 10-30 seconds for Smagorinsky turbulence model, $D^*/dx = 10$.

deviates by -27% to +50% in the range $0.1 \leq Q^* \leq 2$ and up to 10% for $2 < Q^*$ with $D^*/dx = 20$. For the coarse grid, the flame height is over predicted by about 40%.

In Figure 5.21 FDS6 and FDS-EDC are compared with Heskstad flame height correlation for the Deardorff turbulence model. Surprisingly, FDS6 has also difficulties to capture the slope correctly in the interval $0.1 \geq Q^* \geq 1$. FDS6 is over predicting the flame height as FDS-EDC in this interval. In contrast to FDS-EDC, FDS6 is not as grid depended and gives satisfactory results for all grid resolutions. The explanation to this is that the mixing times are functions of the grid cell size, seen in eq. (3.4). The LES-EDC model could simply be made grid independent by establishing a dynamic model constant, $C_{LES}$, as a function of the grid cell size. For FDS6 the absolute error is less than 13.7% with $D^*/dx = 20$ if $2 < Q^*$, else it is up to 46%.

Actually, the absolute error is less for the coarser grids for low $Q^*$. With $D^*/dx = 10$ the 10.3% and 12.9% with $D^*/dx = 5$ except from $Q^* = 0.1$ (63% error).

Figure 5.58 is an example on how $C_{LES}$ affects the flame height. Since the flame height is assumed to be where 99% of fuel is consumed on average, a slice of heat release per unit volume (HRRPUV) is a good way to illustrate how $C_{LES}$ affects the physics. When $C_{LES}$ is increased, the local HRR also increase (right side of Figure 5.58). In case of under prediction of HRRPUV (small $C_{LES}$) too little fuel is consumed in the lower
flame region and allows more fuel to raise due to buoyancy forces. As a result, the fuel moves a longer distance before it is consumed, hence flame length is over estimated (left side of Figure 5.58).

### 5.4.3 Sandia Plume

It is obvious from the results that a time average of 10 seconds is too short. Since the boundary conditions are symmetric it is reasonable to expect symmetric behavior around the centerline of the flame. The consequence of too short time average may be seen in test 14 with the Smagorinsky turbulence model for the coarse grid in the lowest positions $z = 0.3$ m and $z = 0.5$ m (Figures 5.25 and 5.26) where the vertical peak velocity is displaced away from the center in radial position. This misplacement is generally not observed in such extent for simulations with finer grid. Radial velocities are also influenced by the too short time average period because the velocities in few cases are zero along the centerline.

The results show that the trend is the same for all the turbulence models. Not surprisingly, the velocity profiles are more difficult to model correctly as the position increase away from the fuel source. This can especially be seen for the lowest HRR in test 14 in $z = 0.9$ m, Figures 5.24, 5.27 and 5.30. It is difficult to establish a single $C_{LES}$ that fits all the cases in all positions. The vertical velocity is overestimated in the edges of radial position in all heights even for the lowest $C_{LES}$ of 0.005. However, an error of the vertical velocities in the edges or the radial velocity (in high $z$ position) is not that crucial for the flame behavior. The reason is that in these places the velocities are low, and even though the percentage error is large the contribution to the over all upward mass rate in the flame is low. Actually, the velocity profiles should be considered in connection with densities to ensure correct upward mass transport. Experimental results could maybe reveal that the velocity is overestimated in place where the density is underestimated (i.e, that underestimation of density compensate for overestimation of velocity) so that the mass transport is correctly modeled.

Deviation of the vertical velocity between the $C_{LES}$ is largest in simulations with the coarse grid. The deviations are larger as the $z$-position increase. Velocities are over predicted with Smagorsinky turbulence model and FDS6 cases with the coarse grid which is in agreement with theory in Section 3.1 (over prediction of HRR if the grid is too coarse).

The radial velocity seems to be more difficult to model correctly than the vertical velocity. In Figures 5.25, 5.35 and 5.45 the radial velocity for the Smagorinsky turbulence model in $z = 0.3$ m appears almost to be random with the coarse grid. All over, Smagorinsky is the most grid dependent turbulence model. The error in the edges for the vertical velocities are slightly larger with Smagorsinsky compared with the other two turbulence models. In general, this error is less at the edges for all the models in $z = 0.5$ m and $z = 0.9$ m where the slope is steeper. In contrast to Vreman and Deardorff turbulence models, simulations with Smagorinsky and FDS6 are capturing the dip in the fuel rich area along the centerline of the flame in test 17. But for some reason the vertical velocity for all simulations, even for FDS6 (with the original combustion model),
Figure 5.59: An example of temperature contours for FDS6 test 14 with $dx = 1.5$ cm at 13 second. Be aware that the screenshot represents instantaneous values and is not representative for the whole simulation time. The green dots are measuring points in heights of 0.3 m, 0.5 m and 0.9 m.

are remarkably over predicted 0.5 m and 0.9 m above the burner in test 14.

If FDS6 is used as acceptance criteria, all turbulence models for FDS-EDC with $C_{LES} = 0.015$ predicts the vertical velocity as well as FDS6 or even better with the finest grid. However, FDS6 is not as grid dependent as FDS-EDC. Other $C_{LES}$ than 0.015 fits better in some cases with grids corresponds to $dx = 3$ cm and $dx = 6$ cm. But a constant should be establish for a fine grid ($dx = 1.5$ cm) and then in further work be modified to be grid independent. The deviation between $C_{LES} = 0.01$ and $C_{LES} = 0.015$ in the vertical velocity at $z = 0.3$ m is minimal in test 17 and 24 with all turbulence models. With $C_{LES} = 0.015$ the absolute maximum error of vertical velocities are approximately 15-20% while the radial velocity errors are some places up several hundred percent around the center.
Almost the same amount of code lines are added to the code as the amount of code lines which is commented out. Furthermore, one IF loop is commented out and one is added. If two similar cases is simulated with FDS6 and FDS-EDC contains the same amount of burning cells, then they should use the same CPU clock time in theory. Figure 5.55 confirms that this is almost correct. No systematic connection cannot be drawn whether CPU clock time for FDS6 simulations are larger than for FDS-EDC or the other way around. However, HRR release is related to the simulation time in most cases (see Table 5.2). Hence, test 17 takes longer time to simulate than test 24 and test 24 takes longer time to simulate than test 14. Figures 5.52 - 5.54 show that the Smagorinsky turbulence model is the most computational expensive. This is simply because the constant in the Smagorinsky model used in this thesis is dynamic and not static as for Deardorff and Vreman. The difference is much larger in percent if the grid is refined or the HRR increased. With the fine grid the difference is about 60%. Vreman and Deardorff turbulence models are roughly the same computational expensive.
Chapter 6

Flow Field Above Obstacle Inserted in Fire Plume

These experiments are inspired by experiments done by Lars Roar Skarsbøin a master’s thesis delivered June 2011 at the University of Bergen [35]. Some improvements are done to make it closer to how it is modeled. In these experiments, square pipes are used instead of circular. The first reason is to generate more turbulence. Secondly, because a rectangular grid is used in simulations. Since only the combustion model is evaluated in this thesis and not the evaporation/pyrolysis and thermal radiation, liquid is replaced with gas to control the HRR. The burner size is decreased to be able to increase the flow velocity and also the degree of turbulence at lower HRR with less fuel.

The motivation for these experiments is to study the flow field above turbulence generating obstacles inserted in a fire. By systematically increase the HRR and height of the obstacles, a simple correlation between a turbulence property (TKE, turbulence intensity, velocity fluctuations, strain rate, etc.) and a dynamic constant $C_{LES}$ may perhaps be established. It is also reasonable to believe that in such turbulent regime is where the largest difference between the sophisticated LES-EDC model and the already existing combustion model in FDS.

To finance the experiments, this study was linked up to the research program Prediction and validation of pool fire developed in enclosures by means of CFD models for risk assessment of nuclear power plants. During the planning, it turned out to be quite hard to find an institution that had the correct equipment and knowledge to operate Particle Image Velocimetry (PIV) to measure a velocity vector field. After some heavy delay, the experiments were performed at Lund University. Unfortunately, the outcome was no successful results. Hence, this study must be regarded as a pre-project for PIV measurements in the pool fire research programme and not a part of the validation of the combustion model.
6.1 Designing Experiments for CFD Validation

The philosophy of CFD model validation in matter of fire modeling may be separated in whether the fire is specified or predicted [32]. The choice is depended on the goal of validation. In this thesis the goal is to validate the combustion model, other models as evaporation, pyrolysis, soot, etc. is not of interest. Therefore, the fire should be specified rather than predicted. The road map for a CFD model in Figure 6.1 shows that the combustion is depended on other sub models in fire modeling. Sub models are only models consisting empirical correlations only valid in certain interval of physical values, and not necessarily physical formulas solving the actual physical problem. By specifying the fire the HRR is not influenced by quantities as radiation. Possible errors in sub models controlling the HRR are then eliminated. So in cases when single sub models are validated, it is preferable to involve a minimum of other sub models.

Figure 6.1: CFD road map; extension of CFD to fire modeling and overview of sub models.
In validations of CFD models, the experiments must be as close to the simulated scenario as possible. The HRR in the experiments must be controlled to be able to specify it in the input of simulations. A constant gas fire is therefore a good choice for validation of the combustion model. The HRR is then not influenced by the radiation which would be the case if a liquid fuel fire or a solid fuel fire is chosen. If a liquid fuel fire or a solid fuel fire is chosen, an error in the radiation modeling would influence the rate of pyrolysis or evaporation which further would lead to overestimation or underestimation of HRR.

Since uniform rectangular grid cells are applied in FDS, the geometrical shapes in the experimental setup should also be rectangular. So in these experiments, circular pipes used in Skarsbøs thesis were replaced with rectangular pipes. Additionally, geometrical dimensions smaller than the grid cell size should be avoided. However, most commercial CFD codes supports this by setting a porosity parameter in all faces of the grid cell.

Insertion of obstacles in flames leads to heat loss in the flame and affects the temperature, velocity, reaction rate, etc. Input in the simulations does not require inner temperature of the pipes, but it is an advantage to specify the temperature rather than predict it to eliminate a possible error in the heat transfer calculation. Therefore, 16 thermocouples were mounted in the pipes with improvised v-shaped wedges to find the steady state temperature. The placements of the thermocouples and numbering are seen in Figure 6.2.

### 6.2 Measurement Techniques

#### 6.2.1 Particle Image Velocimetry (PIV)

Particle Image Velocimetry (PIV) is a sophisticated measuring technique for measurements of instantaneous velocity vectors in cross-section of a fluid flow. Wind tunnel velocity experiments, experimental verification of CFD models, measurements in pipe flows, spray and combustion research are examples of application areas of PIV [37]. Normally, low-mass particles are seeded in the flow, which is assumed to move free with the local flow velocity [36]. Two short time laser pulses illuminate a plane in the flow. Light is scattered by the seeded particles and are recorded by a synchronized camera. Through the post-processing, the two images are subdivided in interrogation areas and
6.2.2 Thermocouples

A thermocouple is a device for temperature measurements. Thermocouples can operate in a broad range of temperature and are inexpensive, durable and easy to apply. Two different conductors are coupled to a voltage logger, see Figure 6.4. The voltage produced by the conductors are proportional to the temperature. Temperature at the cold junction must be known and preferably kept constant to calculate the temperature at the hot junction. The temperature difference is given by

\[ \Delta T = \sum_{n=0}^{N} a_n v^n \]  

(6.1)

where \( v \) is the output voltage and \( a_n \) the a coefficient depending upon the metal. In some cases the temperature is found from a database. Databases are usually implemented in modern logging softwares on computers.

In the experiments in this thesis, thermocouples type K was used. For type K metal 1 is chromel (90 % nickel and 10 %chromium) and metal 2 amuel (95 % nickel, 2 % manganese, 2% aluminum and 1 % silicon). Thermocouples type K measures in the range -200 °C to 1350 °C.
6.3 Experiments

Many possible scenarios were simulated before the experiments. The goal was to find an optimal set up with respect to fire size, HRR, cross-section pipe dimensions and the height of the pipes above the fire within limits of measurement range of equipments and other laboratory facilities. Another goal was to find in which combinations of the varied parameters that gave largest differences when comparing a fire plume with and without obstacles. Circular pipes, which were used in Skarsbøs experiments, were replaced with 60 mm x 60 mm (outer dimensions and thickness of 4 mm) square pipes. Unfortunately, only round-edged pipes were available in the area of Haugesund at moment the experiments were performed. However, a somewhat smaller dimension was preferable for an optimal turbulence intensity. But since a smaller dimension requires a finer resolution of the grid in simulations to capture the correct fluid flow between the pipes, 60 mm x 60 mm was an acceptable compromise.

Simulations showed that the pipes placed in the persistent flame region where the flow is accelerating (see Figure 2.7) gave highest rise in temperature and velocity. This is about equivalent to heights below 1/3 of the flame height. Furthermore, a distance of 60 mm between the pipes was used in the simulations. This distance was later changed to 40 mm after some test experiments revealed that the flow around and just above the obstacles got more stable.

In Table 6.1 the planned scenarios are given. The experiment set up is seen in Figure 6.5.
Table 6.1: Experimental scenarios.

<table>
<thead>
<tr>
<th>Number</th>
<th>Fuel</th>
<th>HRR</th>
<th>Heights of pipes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[-]</td>
<td>[kW/m²]</td>
<td>[m]</td>
</tr>
<tr>
<td>1.1</td>
<td>Propane</td>
<td>500</td>
<td>0.15</td>
</tr>
<tr>
<td>1.2</td>
<td>Propane</td>
<td>500</td>
<td>0.30</td>
</tr>
<tr>
<td>1.3</td>
<td>Propane</td>
<td>500</td>
<td>0.45</td>
</tr>
<tr>
<td>2.1</td>
<td>Propane</td>
<td>1000</td>
<td>0.15</td>
</tr>
<tr>
<td>2.2</td>
<td>Propane</td>
<td>1000</td>
<td>0.30</td>
</tr>
<tr>
<td>2.3</td>
<td>Propane</td>
<td>1000</td>
<td>0.45</td>
</tr>
<tr>
<td>3.1</td>
<td>Propane</td>
<td>1500</td>
<td>0.15</td>
</tr>
<tr>
<td>3.2</td>
<td>Propane</td>
<td>1500</td>
<td>0.30</td>
</tr>
<tr>
<td>3.3</td>
<td>Propane</td>
<td>1500</td>
<td>0.45</td>
</tr>
<tr>
<td>4.1</td>
<td>Heptane</td>
<td>-</td>
<td>0.15</td>
</tr>
<tr>
<td>4.2</td>
<td>Heptane</td>
<td>-</td>
<td>0.30</td>
</tr>
<tr>
<td>4.3</td>
<td>Heptane</td>
<td>-</td>
<td>0.45</td>
</tr>
</tbody>
</table>

Figure 6.5: Experimental setup: (1) laser, (2) lens, (3) laser beam absorber, (4) seeding box, (5) camera, (6) anti-reflection shield, (7) gas burner and (8) wind shield.
CHAPTER 6. FLOW FIELD ABOVE OBSTACLE INSERTED IN FIRE PLUME

6.3.1 Summary of the experiments

The laboratory that was booked for the experiments was not suited for fire experiments. It was used for enclosed bench scale flame experiments to study combustion processes in detail. Only four days were spent in the laboratory which turned out to be way too little time. Below, some limitations because of this and challenges during the experiments are described.

Scenarios

Initially, a series of 9 experiments was planned with a propane fire with HRR of 500 kW/m$^2$, 1000 kW/m$^2$ and 1500 kW/m$^2$, with pipes in heights of 0.15 m, 0.3 m and 0.45 m. Because the laboratory was loaded with a lot of expensive equipment and quite small relatively to the fire size, the plan was not achievable. Only experiments 500 kW/m$^2$ propane fire and 32 cm x 32 cm liquid methanol was performed.

Smoke hood and ventilation

The relative large fire size that was initially planned made it necessary to enlarged the smoke hood and set the ventilation on full power to prevent spread of smoke and hazardous seeding particles in the room. Another problem was then observed, the small fire got unstable because of the momentum from the ventilation and tilted in away from the laser. The problem was solved by placing a wind shield (number 8 in Figure 6.5) in front of the laser. However, this problem was not observed for larger fire (>500 kW/m$^2$).

Calibration of the Laser and Camera

Calibration of laser and camera is the most time consuming part of the experiments. First, the camera must be focused on the measurement area, as seen in Figure 6.6. In these calibration experiments, the measurement area was set to 4 cm x 4 cm between the two pipes on the left side. Large measurement areas requires a strong laser to penetrate flame, especially for sooty flames. One of the uncertainties before doing the experiments was whether the laser used could penetrate such a sooty flame produced by propane. That is why methanol was chosen in the calibration experiments.

The camera does not necessarily has to be placed perpendicular to the laser sheet, since the distance between the dots on the calibration plate is known (in Figure 6.6). The correction was automatically fixed by the logging software that was used.

Since the seeding particles are traced as they are illuminated by the laser pulses, disturbing incoming light on the camera lens must be minimized. For this reason a black anti-reflection shield (number (6) in Figure 6.5) was placed behind the measurement area to absorb light in all wave lengths. Another uncertainty that arose regarding the disturbing light was the use of propane as fuel. Propane flames are sootier than for example methanol and are also therefore also emitting more thermal radiation.

The laser used was only able to operate on 4 Hz. This is not sufficient to capture turbulent motions. So the plan about studying turbulent characteristics was discarded.
in the early stages. During the calibration, the shutter on the camera did only function sporadically. As a result, too much light was received on every second image making the tracing of particles impossible. From the experiments no conclusion could be draw whether a propane flame is too sooty to measure a velocity vector field with the equipment available at Lund University.

![Figure 6.6: Calibration of the camera.](image)

**Seeding**

Seeding of particles is the most challenging part of PIV measurements of flames. First of all, no universal method for seeding particles in flames exists. Secondly, the handling of seeding particles requires uttermost care due to hazards of inhaling it. Breathing masks and powerful ventilation are demanded. During these experiments, the flame was seeded in three different ways; by smoke sticks, smoke pellets and seeding of particles with pressurized air. All methods were some way unsuitable and the out coming results were of rather poor quality, primarily caused by the methods. Seeding with smoke was unsuitable because a stable concentration of smoke particles within measurable range was not achievable in the whole measurement area on a sufficient number of the images recorded. It was also attempted to seed the particles by pressurized in vertical direction, both upward and downward, and also horizontal direction faced in positive direction along the laser beam. When the particles were seeded upwards, a jet was clearly observed
in the results. It was obvious that the particles were strongly influenced by the release momentum and did not move free with the local flow velocity in the flame. In downward direction, seeding particles got stuck in burner. The measurement area was not provided with enough seeding when the seeding particles were supplied horizontally. Furthermore, the flame behavior was too much influenced when pressurized air method was applied for small fires.
Chapter 7

Conclusion

The PIV data from the experiments at Lund University were of rather poor quality and unsuitable for CFD validation. Too much light was received in the camera lens since the shutter on the camera only functioned sporadically on every second image. The seeding of particles turned out to be quite challenging and no successful method was found. Smoke pellets and smoke sticks did not provide enough seeding within measurable range in the desired area. The seeding particles did not move free with the local flow velocity when they were seeded by pressurized air, but were influenced by the release momentum. The flame behavior for small fires was also influenced by the pressurized air method.

The validation of FDS-EDC revealed that the model is quite grid dependent. Except from the grid dependency the implemented code is in satisfactory agreement with the validation cases, and as good as the already existing or even better. Both models are about the same computational expensive. However, a single model constant, $C_{LES}$, gives not the most correct result for all the cases. An overall recommended $C_{LES}$ of 0.015, as well as the already existing combustion model, over estimates the flame height for $Q^* \lesssim 1$. This over estimation of flame height is not observed with the Smagorinsky turbulence model. Furthermore, the maximum temperature is as much as 600°C with the Deardorf model for the McCaffery simulation of 57 kW. In addition, the temperature peak was located too far above the burner. The Smagorinsky model did not displace this peak in such extent as Deardorf and Vreman. For the McCaffery simulations, all turbulence models gave results in better agreement with $C_{LES} = 0.01$ than $C_{LES} = 0.015$. The Smagorinsky model, was the only turbulence model that captured the dip for the vertical velocity in the Sandia experiment test 17, but with the finest grid the CPU clock time was nearly 70% larger than the two other turbulence models.

Before LES-EDC can be applied in fire analysis the model must be modified to be grid independent. The model should also be be validated in a wider range for temperature and velocity profiles in practical fire scale to establish a more accurate constant. A dynamic constant is not necessary for buoyancy-driven fires in the context of fire engineering, but could a benefit for detailed investigation of flames.
7.1 Further Work

The author of this thesis and the participants of the experiments at Lund University discussed alternative ways to arrange experiments to achieve successful PIV measurements. Regarding the seeding, the most challenging part;

- the first step is to perform the experiments in a closed room where seeding particles are in no hazard to humans. Then it is possible to seed the flame with large amounts of particles.

- step two is to customize the burner and build in seeding point around the frame (maybe even in the middle of the burner too) and seed the particles with low-velocity controllable pressurized air.

The implemented code is validated against flame height in range $0.1 \leq Q^* \leq 10000$; radial and vertical (horizontal) velocity profiles in range $1.35 \leq Q^* \leq 2.2$; centerline velocity and temperature profiles in range $0.005 \leq Q^* \leq 0.02$. In further work, it could be interesting to investigate how the mentioned parameter interact with each other when $C_{LES}$ is varied. The code should also be validated for temperature and velocity in the range of $0.1 \lesssim Q^* \lesssim 1.5$ where the FDS-EDC did not fit well for the flame height (except from Smagorinsky turbulence model). The flame height is varying significantly because the unstable nature of fire [38]. This puff cycle could also be validated in further work [34]. At last the model has to be modified to be grid independent.
Bibliography


BIBLIOGRAPHY


[37] Particle image velocimetry (piv); www.dantecdynamics.com, March 2012.

Appendix A

FDS input files

A.1 Sandia test 14

&HEAD CHID='Sandia_CH4_1m_Test14_dx1p5cm', TITLE='Sandia 1m low flow rate methane pool fire (Test 14), 1.5 cm resolution '/

&MULT ID='mesh array', DX=1.5, DY=1.5, DZ=1.0, L_UPPER=1, L_UPPER =1, K_UPPER=3 '/
&MESH IJK=96,96,64, XB=-1.5,0.0,-1.5,0.0,-.0625,0.9375, MULT_ID = 'mesh array' '/

&TIME T_END=20. /
&MISC TMPA=12.
   P_INF=80600.
   TURBULENCE_MODEL='VREMAN' /

&REAC FUEL='METHANE'
   HEAT_OF_COMBUSTION=50611.
   CRITICAL_FLAME_TEMPERATURE=1207 /

&RADI RADIATIVE_FRACTION=0.2 /
&DUMP SIG FIGS=4, SIG FIGS EXP=2 /
&MATL ID='STEEL', CONDUCTIVITY=54., SPECIFIC_HEAT=0.465,
   DENSITY=7850., EMISSIVITY=0.9 /
&SURF ID='PLATE', COLOR='GRAY', DEFAULT=.TRUE., MATL_ID='STEEL'
   , THICKNESS=0.025 /
&SURF ID='POOL', MASS_FLUX (1)=0.04 , SPEC_ID (1)='METHANE',
   TMP_FRONT=11. /
APPENDIX A. FDS INPUT FILES

&VENT XB = -0.5, 0.5, -0.5, 0.5, -0.0625, -0.0625, COLOR = 'BLUE', SURF_ID = 'POOL'

&VENT XB = -1.5, -1.5, 1.5, 1.5, -0.625, -0.625, SURF_ID = 'OPEN'
&VENT XB = -1.5, 1.5, -1.5, -1.5, -0.625, -0.625, SURF_ID = 'OPEN'
&VENT XB = -1.5, 1.5, 1.5, -1.5, -0.625, -0.625, SURF_ID = 'OPEN'

&VENT MB = 'ZMAX', SURF_ID = 'OPEN'
&VENT MB = 'YMIN', SURF_ID = 'OPEN'
&VENT MB = 'XMIN', SURF_ID = 'OPEN'
&VENT MB = 'XMAX', SURF_ID = 'OPEN'

&SLCF PBY = -0.0625, QUANTITY = 'VELOCITY', VECTOR = .TRUE. /
&SLCF PBY = -0.0625, QUANTITY = 'DENSITY', CELL_CENTERED = .TRUE. /
&SLCF PBY = -0.0625, QUANTITY = 'TEMPERATURE', CELL_CENTERED = .TRUE. /
&SLCF PBY = -0.0625, QUANTITY = 'HRRPUV', CELL_CENTERED = .TRUE. /
&SLCF PBY = -0.0625, QUANTITY = 'HRRPUA', CELL_CENTERED = .TRUE. /
&SLCF PBY = -0.0625, QUANTITY = 'MASS FRACTION', SPEC_ID = 'METHANE', CELL_CENTERED = .TRUE. /
&SLCF PBY = -0.0625, QUANTITY = 'TURBULENCE RESOLUTION', CELL_CENTERED = .TRUE. /
&SLCF PBY = -0.0625, QUANTITY = 'WAVELET ERROR', QUANTITY2 = 'MASS FRACTION', SPEC_ID = 'METHANE', CELL_CENTERED = .TRUE. /

&OBST XB = -0.70711, 0.70711, -0.70711, 0.70711, -0.0625, 0 /
&OBST XB = -0.70711, 0.70711, -0.70711, 0.70711, -0.0625, 0 /
&OBST XB = -0.69211, 0.69211, -0.7218, 0.7218, -0.0625, 0 /
&OBST XB = -0.7218, 0.7218, -0.69211, 0.69211, -0.0625, 0 /
&OBST XB = -0.67711, 0.67711, -0.73588, 0.73588, -0.0625, 0 /
&OBST XB = -0.73588, 0.73588, -0.67711, 0.67711, -0.0625, 0 /
&OBST XB = -0.66211, 0.66211, -0.74941, 0.74941, -0.0625, 0 /
&OBST XB = -0.74941, 0.74941, -0.66211, 0.66211, -0.0625, 0 /
&OBST XB = -0.64711, 0.64711, -0.7624, 0.7624, -0.0625, 0 /
&OBST XB = -0.7624, 0.7624, -0.64711, 0.64711, -0.0625, 0 /
&OBST XB = -0.63211, 0.63211, -0.77488, 0.77488, -0.0625, 0 /
&OBST XB = -0.77488, 0.77488, -0.63211, 0.63211, -0.0625, 0 /
&OBST XB = -0.61711, 0.61711, -0.78688, 0.78688, -0.0625, 0 /
&OBST XB = -0.78688, 0.78688, -0.61711, 0.61711, -0.0625, 0 /
&OBST XB = -0.60211, 0.60211, -0.79842, 0.79842, -0.0625, 0 /
&OBST XB = -0.79842, 0.79842, -0.60211, 0.60211, -0.0625, 0 / &OBST XB = -0.58711, 0.58711, -0.80951, 0.80951, -0.0625, 0 / &OBST XB = -0.80951, 0.80951, -0.57211, 0.57211, -0.0625, 0 / &OBST XB = -0.82018, 0.82018, -0.57211, 0.57211, -0.0625, 0 / &OBST XB = -0.55711, 0.55711, -0.83044, 0.83044, -0.0625, 0 / &OBST XB = -0.83044, 0.83044, -0.55711, 0.55711, -0.0625, 0 / &OBST XB = -0.54211, 0.54211, -0.84031, 0.84031, -0.0625, 0 / &OBST XB = -0.84031, 0.84031, -0.54211, 0.54211, -0.0625, 0 / &OBST XB = -0.52711, 0.52711, -0.8498, 0.8498, -0.0625, 0 / &OBST XB = -0.8498, 0.8498, -0.52711, 0.52711, -0.0625, 0 / &OBST XB = -0.51211, 0.51211, -0.85892, 0.85892, -0.0625, 0 / &OBST XB = -0.85892, 0.85892, -0.51211, 0.51211, -0.0625, 0 / &OBST XB = -0.49711, 0.49711, -0.86769, 0.86769, -0.0625, 0 / &OBST XB = -0.86769, 0.86769, -0.49711, 0.49711, -0.0625, 0 / &OBST XB = -0.48211, 0.48211, -0.87611, 0.87611, -0.0625, 0 / &OBST XB = -0.87611, 0.87611, -0.48211, 0.48211, -0.0625, 0 / &OBST XB = -0.46711, 0.46711, -0.8842, 0.8842, -0.0625, 0 / &OBST XB = -0.8842, 0.8842, -0.46711, 0.46711, -0.0625, 0 / &OBST XB = -0.45211, 0.45211, -0.89196, 0.89196, -0.0625, 0 / &OBST XB = -0.89196, 0.89196, -0.45211, 0.45211, -0.0625, 0 / &OBST XB = -0.43711, 0.43711, -0.89941, 0.89941, -0.0625, 0 / &OBST XB = -0.89941, 0.89941, -0.43711, 0.43711, -0.0625, 0 / &OBST XB = -0.42211, 0.42211, -0.90655, 0.90655, -0.0625, 0 / &OBST XB = -0.90655, 0.90655, -0.42211, 0.42211, -0.0625, 0 / &OBST XB = -0.40711, 0.40711, -0.91338, 0.91338, -0.0625, 0 / &OBST XB = -0.91338, 0.91338, -0.40711, 0.40711, -0.0625, 0 / &OBST XB = -0.39211, 0.39211, -0.91992, 0.91992, -0.0625, 0 / &OBST XB = -0.91992, 0.91992, -0.39211, 0.39211, -0.0625, 0 / &OBST XB = -0.37711, 0.37711, -0.92617, 0.92617, -0.0625, 0 / &OBST XB = -0.92617, 0.92617, -0.37711, 0.37711, -0.0625, 0 / &OBST XB = -0.36211, 0.36211, -0.93214, 0.93214, -0.0625, 0 / &OBST XB = -0.93214, 0.93214, -0.36211, 0.36211, -0.0625, 0 / &OBST XB = -0.34711, 0.34711, -0.93783, 0.93783, -0.0625, 0 / &OBST XB = -0.93783, 0.93783, -0.34711, 0.34711, -0.0625, 0 / &OBST XB = -0.33211, 0.33211, -0.94324, 0.94324, -0.0625, 0 / &OBST XB = -0.94324, 0.94324, -0.33211, 0.33211, -0.0625, 0 / &OBST XB = -0.31711, 0.31711, -0.94839, 0.94839, -0.0625, 0 / &OBST XB = -0.94839, 0.94839, -0.31711, 0.31711, -0.0625, 0 / &OBST XB = -0.30211, 0.30211, -0.95327, 0.95327, -0.0625, 0 / &OBST XB = -0.95327, 0.95327, -0.30211, 0.30211, -0.0625, 0 / &OBST XB = -0.28711, 0.28711, -0.9579, 0.9579, -0.0625, 0 / &OBST XB = -0.9579, 0.9579, -0.28711, 0.28711, -0.0625, 0 /
&OBST XB = −0.27211, 0.27211, −0.96227, 0.96227, −0.0625, 0 /
&OBST XB = −0.96227, 0.96227, −0.27211, 0.27211, −0.0625, 0 /
&OBST XB = −0.25711, 0.25711, −0.96638, 0.96638, −0.0625, 0 /
&OBST XB = −0.96638, 0.96638, −0.25711, 0.25711, −0.0625, 0 /
&OBST XB = −0.24211, 0.24211, −0.97025, 0.97025, −0.0625, 0 /
&OBST XB = −0.97025, 0.97025, −0.24211, 0.24211, −0.0625, 0 /
&OBST XB = −0.22711, 0.22711, −0.97387, 0.97387, −0.0625, 0 /
&OBST XB = −0.97387, 0.97387, −0.22711, 0.22711, −0.0625, 0 /
&OBST XB = −0.21211, 0.21211, −0.97725, 0.97725, −0.0625, 0 /
&OBST XB = −0.97725, 0.97725, −0.21211, 0.21211, −0.0625, 0 /
&OBST XB = −0.19711, 0.19711, −0.98038, 0.98038, −0.0625, 0 /
&OBST XB = −0.98038, 0.98038, −0.19711, 0.19711, −0.0625, 0 /
&OBST XB = −0.18211, 0.18211, −0.98328, 0.98328, −0.0625, 0 /
&OBST XB = −0.98328, 0.98328, −0.18211, 0.18211, −0.0625, 0 /
&OBST XB = −0.16711, 0.16711, −0.98594, 0.98594, −0.0625, 0 /
&OBST XB = −0.98594, 0.98594, −0.16711, 0.16711, −0.0625, 0 /
&OBST XB = −0.15211, 0.15211, −0.98836, 0.98836, −0.0625, 0 /
&OBST XB = −0.98836, 0.98836, −0.15211, 0.15211, −0.0625, 0 /
&OBST XB = −0.13711, 0.13711, −0.99056, 0.99056, −0.0625, 0 /
&OBST XB = −0.99056, 0.99056, −0.13711, 0.13711, −0.0625, 0 /
&OBST XB = −0.12211, 0.12211, −0.99252, 0.99252, −0.0625, 0 /
&OBST XB = −0.99252, 0.99252, −0.12211, 0.12211, −0.0625, 0 /
&OBST XB = −0.10711, 0.10711, −0.99425, 0.99425, −0.0625, 0 /
&OBST XB = −0.99425, 0.99425, −0.10711, 0.10711, −0.0625, 0 /
&OBST XB = −0.092107, 0.092107, −0.99575, 0.99575, −0.0625, 0 /
&OBST XB = −0.99575, 0.99575, −0.092107, 0.092107, −0.0625, 0 /
&OBST XB = −0.077107, 0.077107, −0.99702, 0.99702, −0.0625, 0 /
&OBST XB = −0.99702, 0.99702, −0.077107, 0.077107, −0.0625, 0 /
&OBST XB = −0.062107, 0.062107, −0.99807, 0.99807, −0.0625, 0 /
&OBST XB = −0.99807, 0.99807, −0.062107, 0.062107, −0.0625, 0 /
&OBST XB = −0.047107, 0.047107, −0.99889, 0.99889, −0.0625, 0 /
&OBST XB = −0.99889, 0.99889, −0.047107, 0.047107, −0.0625, 0 /
&OBST XB = −0.032107, 0.032107, −0.99948, 0.99948, −0.0625, 0 /
&OBST XB = −0.99948, 0.99948, −0.032107, 0.032107, −0.0625, 0 /
&OBST XB = −0.017107, 0.017107, −0.99985, 0.99985, −0.0625, 0 /
&OBST XB = −0.99985, 0.99985, −0.017107, 0.017107, −0.0625, 0 /
&OBST XB = −0.0021068, 0.0021068, −1, 1, −0.0625, 0 /
&OBST XB = −1, 1, −0.0021068, 0.0021068, −0.0625, 0 /

&HOLE XB = −0.35355, 0.35355, −0.35355, 0.35355, −1.25,.1/
&HOLE XB = −0.35355, 0.35355, −0.35355, 0.35355, −1.25,.1/
&HOLE XB = −0.33755, 0.33755, −0.36886, 0.36886, −1.25,.1/
&HOLE XB = −0.36886, 0.36886, −0.33755, 0.33755, −1.25,.1/
APPENDIX A. FDS INPUT FILES

&HOLE XB = −0.32155, 0.32155, −0.38289, 0.38289, −0.125, 0.1
&HOLE XB = −0.38289, 0.32155, −0.30555, 0.30555, −0.125, 0.1
&HOLE XB = −0.39577, 0.39577, −0.29555, 0.29555, −0.125, 0.1
&HOLE XB = −0.30555, 0.30555, −0.32155, 0.32155, −0.125, 0.1
&HOLE XB = −0.28955, 0.28955, −0.39577, 0.39577, −0.125, 0.1
&HOLE XB = −0.29555, 0.29555, −0.32155, 0.32155, −0.125, 0.1
&HOLE XB = −0.27355, 0.27355, −0.40763, 0.40763, −0.125, 0.1
&HOLE XB = −0.28955, 0.28955, −0.39577, 0.39577, −0.125, 0.1
&HOLE XB = −0.27355, 0.27355, −0.41853, 0.41853, −0.125, 0.1
&HOLE XB = −0.25755, 0.25755, −0.42856, 0.42856, −0.125, 0.1
&HOLE XB = −0.24155, 0.24155, −0.43778, 0.43778, −0.125, 0.1
&HOLE XB = −0.22555, 0.22555, −0.44623, 0.44623, −0.125, 0.1
&HOLE XB = −0.19355, 0.19355, −0.46102, 0.46102, −0.125, 0.1
&HOLE XB = −0.16155, 0.16155, −0.47318, 0.47318, −0.125, 0.1
&HOLE XB = −0.13355, 0.13355, −0.48693, 0.48693, −0.125, 0.1
&HOLE XB = −0.10155, 0.10155, −0.49887, 0.49887, −0.125, 0.1
&HOLE XB = −0.069553, 0.069553, −0.5, 0.5, −0.125, 0.1
Radial profiles of velocity and mass fraction

\&DEVC XB = -0.50, 0.50, 0.0, 0.0, 0.3, 0.3, QUANTITY='W-VELOCITY', ID = 'Wp3', POINTS=21, X_ID='x' /
\&DEVC XB = -0.50, 0.50, 0.0, 0.0, 0.5, 0.5, QUANTITY='W-VELOCITY', ID = 'Wp5', POINTS=21, HIDE_COORDINATES=.TRUE. /
\&DEVC XB = -0.50, 0.50, 0.0, 0.0, 0.9, 0.9, QUANTITY='W-VELOCITY', ID = 'Wp9', POINTS=21, HIDE_COORDINATES=.TRUE. /
\&DEVC XB = -0.50, 0.50, 0.0, 0.0, 0.3, 0.3, QUANTITY='U-VELOCITY', ID = 'Up3', POINTS=21, HIDE_COORDINATES=.TRUE. /
\&DEVC XB = -0.50, 0.50, 0.0, 0.0, 0.5, 0.5, QUANTITY='U-VELOCITY', ID = 'Up5', POINTS=21, HIDE_COORDINATES=.TRUE. /
\&DEVC XB = -0.50, 0.50, 0.0, 0.0, 0.9, 0.9, QUANTITY='U-VELOCITY', ID = 'Up9', POINTS=21, HIDE_COORDINATES=.TRUE. /
\&DEVC XB = -0.50, 0.50, 0.0, 0.0, 0.3, 0.3, QUANTITY='W-VELOCITY', ID = 'Wp3_rms', STATISTICS='RMS', POINTS=21, HIDE_COORDINATES=.TRUE. /
\&DEVC XB = -0.50, 0.50, 0.0, 0.0, 0.5, 0.5, QUANTITY='W-VELOCITY', ID = 'Wp5_rms', STATISTICS='RMS', POINTS=21, HIDE_COORDINATES=.TRUE. /
\&DEVC XB = -0.50, 0.50, 0.0, 0.0, 0.9, 0.9, QUANTITY='W-VELOCITY', ID = 'Wp9_rms', STATISTICS='RMS', POINTS=21, HIDE_COORDINATES=.TRUE. /
\&DEVC XB = -0.50, 0.50, 0.0, 0.0, 0.3, 0.3, QUANTITY='U-VELOCITY', ID = 'Up3_rms', STATISTICS='RMS', POINTS=21, HIDE_COORDINATES=.TRUE. /
\&DEVC XB = -0.50, 0.50, 0.0, 0.0, 0.5, 0.5, QUANTITY='U-VELOCITY', ID = 'Up5_rms', STATISTICS='RMS', POINTS=21, HIDE_COORDINATES=.TRUE. /
\&DEVC XB = -0.50, 0.50, 0.0, 0.0, 0.9, 0.9, QUANTITY='U-VELOCITY', ID = 'Up9_rms', STATISTICS='RMS', POINTS=21, HIDE_COORDINATES=.TRUE. /

&TAIL /

A.2 Sandia test 17

\&HEAD CHID='Sandia_CH4_1m_Test17_dx1p5cm', TITLE='Sandia 1 m high flow rate methane pool fire (Test 17), 1.5 cm resolution' /
APPENDIX A. FDS INPUT FILES

&MULT ID='mesh array', DX=1.5, DY=1.5, DZ=1.0, I_UPPER=1, J_UPPER=3 /
&MESH IJK=96,96,64, XB=-1.5,0.0,-1.5,0.0,-.0625,.9375, MULT_ID = 'mesh array' /

&TIME T_END=20. /
&MISC TMPA=5.
  P_INF=81100.
  TURBULENCE_MODEL='VREMAN' /

&REAC FUEL='METHANE'
  HEAT_OF_COMBUSTION=50350.
  CRITICAL_FLAME_TEMPERATURE=1207 /

&RADI RADIATIVE_FRACTION=0.2 /
&DUMP SIG_FIGS=4, SIG_FIGS_EXP=2 /
&MATL ID='STEEL', CONDUCTIVITY=54., SPECIFIC_HEAT=0.465,
  DENSITY=7850., EMISSIVITY=0.9 /
&SURF ID='PLATE', COLOR='GRAY', DEFAULT=.TRUE., MATL_ID='STEEL',
  THICKNESS=0.025 /
&SURF ID='POOL', MASS_FLUX(1)=0.066, SPEC_ID(1)='METHANE',
  TMP_FRONT=1. /

&VENT XB=-0.5,0.5,-0.5,0.5,-.0625,-.0625, COLOR='BLUE', SURF_ID='POOL' /

&VENT XB=-1.5,-.5,-1.5,1.5,-.0625,-.0625, SURF_ID='OPEN' /
&VENT XB=.5,1.5,-1.5,1.5,-.0625,-.0625, SURF_ID='OPEN' /
&VENT XB=-1.5,1.5,-1.5,-.5,-.0625,-.0625, SURF_ID='OPEN' /
&VENT XB=-1.5,1.5,5.15,-.0625,-.0625, SURF_ID='OPEN' /
&VENT MB='ZMAX', SURF_ID='OPEN' /
&VENT MB='YMIN', SURF_ID='OPEN' /
&VENT MB='YMAX', SURF_ID='OPEN' /
&VENT MB='XMIN', SURF_ID='OPEN' /
&VENT MB='XMAX', SURF_ID='OPEN' /

&SLEC PBY=-.0625, QUANTITY='VELOCITY', VECTOR=.TRUE. /
&SLEC PBY=-.0625, QUANTITY='DENSITY', CELL_CENTERED=.TRUE. /
&SLEC PBY=-.0625, QUANTITY='TEMPERATURE', CELL_CENTERED=.TRUE. /
APPENDIX A. FDS INPUT FILES

&SLCF PBY=−.0625, QUANTITY='HRRPUV', CELL_CENTERED=.TRUE. / 
/&SLCF PBY=−.0625, QUANTITY='HRRPUA', CELL_CENTERED=.TRUE. / 
&SLCF PBY=−.0625, QUANTITY='MASS FRACTION', SPEC_ID='METHANE', 
CELL_CENTERED=.TRUE. / 
&SLCF PBY=−.0625, QUANTITY='CSMAG', CELL_CENTERED=.TRUE. / 
&SLCF PBY=−.0625, QUANTITY='TURBULENCE RESOLUTION', 
CELL_CENTERED=.TRUE. / 
&SLCF PBY=−.0625, QUANTITY='WAVELET ERROR', QUANTITY2='MASS 
FRACTION', SPEC_ID='METHANE', CELL_CENTERED=.TRUE. / 

&OBST XB = −0.70711,0.70711,0.70711,0.70711,−0.0625,0 / 
&OBST XB = −0.69211,0.69211,0.69211,0.69211,−0.0625,0 / 
&OBST XB = −0.69211,0.69211,0.69211,0.69211,−0.0625,0 / 
&OBST XB = −0.69211,0.69211,0.69211,0.69211,−0.0625,0 / 
&OBST XB = −0.69211,0.69211,0.69211,0.69211,−0.0625,0 / 
&OBST XB = −0.69211,0.69211,0.69211,0.69211,−0.0625,0 / 
&OBST XB = −0.69211,0.69211,0.69211,0.69211,−0.0625,0 / 
&OBST XB = −0.69211,0.69211,0.69211,0.69211,−0.0625,0 / 
&OBST XB = −0.69211,0.69211,0.69211,0.69211,−0.0625,0 / 
&OBST XB = −0.69211,0.69211,0.69211,0.69211,−0.0625,0 / 
&OBST XB = −0.69211,0.69211,0.69211,0.69211,−0.0625,0 / 
&OBST XB = −0.69211,0.69211,0.69211,0.69211,−0.0625,0 / 
&OBST XB = −0.69211,0.69211,0.69211,0.69211,−0.0625,0 / 
&OBST XB = −0.69211,0.69211,0.69211,0.69211,−0.0625,0 / 

\&OBST XB = -0.8842, 0.8842, -0.46711, 0.46711, -0.0625, 0 /
\&OBST XB = -0.45211, 0.45211, -0.89196, 0.89196, -0.0625, 0 /
\&OBST XB = -0.89196, 0.89196, -0.43711, 0.43711, -0.0625, 0 /
\&OBST XB = -0.42211, 0.42211, -0.90655, 0.90655, -0.0625, 0 /
\&OBST XB = -0.90655, 0.90655, -0.42211, 0.42211, -0.0625, 0 /
\&OBST XB = -0.40711, 0.40711, -0.91338, 0.91338, -0.0625, 0 /
\&OBST XB = -0.91338, 0.91338, -0.43711, 0.43711, -0.0625, 0 /
\&OBST XB = -0.39211, 0.39211, -0.91992, 0.91992, -0.0625, 0 /
\&OBST XB = -0.91992, 0.91992, -0.39211, 0.39211, -0.0625, 0 /
\&OBST XB = -0.37711, 0.37711, -0.92617, 0.92617, -0.0625, 0 /
\&OBST XB = -0.92617, 0.92617, -0.37711, 0.37711, -0.0625, 0 /
\&OBST XB = -0.36211, 0.36211, -0.93214, 0.93214, -0.0625, 0 /
\&OBST XB = -0.93214, 0.93214, -0.36211, 0.36211, -0.0625, 0 /
\&OBST XB = -0.34711, 0.34711, -0.93783, 0.93783, -0.0625, 0 /
\&OBST XB = -0.93783, 0.93783, -0.34711, 0.34711, -0.0625, 0 /
\&OBST XB = -0.33211, 0.33211, -0.94324, 0.94324, -0.0625, 0 /
\&OBST XB = -0.94324, 0.94324, -0.33211, 0.33211, -0.0625, 0 /
\&OBST XB = -0.31711, 0.31711, -0.94839, 0.94839, -0.0625, 0 /
\&OBST XB = -0.94839, 0.94839, -0.31711, 0.31711, -0.0625, 0 /
\&OBST XB = -0.30211, 0.30211, -0.95327, 0.95327, -0.0625, 0 /
\&OBST XB = -0.95327, 0.95327, -0.30211, 0.30211, -0.0625, 0 /
\&OBST XB = -0.28711, 0.28711, -0.9579, 0.9579, -0.0625, 0 /
\&OBST XB = -0.9579, 0.9579, -0.28711, 0.28711, -0.0625, 0 /
\&OBST XB = -0.27211, 0.27211, -0.96227, 0.96227, -0.0625, 0 /
\&OBST XB = -0.96227, 0.96227, -0.27211, 0.27211, -0.0625, 0 /
\&OBST XB = -0.25711, 0.25711, -0.96638, 0.96638, -0.0625, 0 /
\&OBST XB = -0.96638, 0.96638, -0.25711, 0.25711, -0.0625, 0 /
\&OBST XB = -0.24211, 0.24211, -0.97025, 0.97025, -0.0625, 0 /
\&OBST XB = -0.97025, 0.97025, -0.24211, 0.24211, -0.0625, 0 /
\&OBST XB = -0.22711, 0.22711, -0.97387, 0.97387, -0.0625, 0 /
\&OBST XB = -0.97387, 0.97387, -0.22711, 0.22711, -0.0625, 0 /
\&OBST XB = -0.21211, 0.21211, -0.97725, 0.97725, -0.0625, 0 /
\&OBST XB = -0.97725, 0.97725, -0.21211, 0.21211, -0.0625, 0 /
\&OBST XB = -0.19711, 0.19711, -0.98038, 0.98038, -0.0625, 0 /
\&OBST XB = -0.98038, 0.98038, -0.19711, 0.19711, -0.0625, 0 /
\&OBST XB = -0.18211, 0.18211, -0.98328, 0.98328, -0.0625, 0 /
\&OBST XB = -0.98328, 0.98328, -0.18211, 0.18211, -0.0625, 0 /
\&OBST XB = -0.16711, 0.16711, -0.98594, 0.98594, -0.0625, 0 /
\&OBST XB = -0.98594, 0.98594, -0.16711, 0.16711, -0.0625, 0 /
\&OBST XB = -0.15211, 0.15211, -0.98836, 0.98836, -0.0625, 0 /
\&OBST XB = -0.98836, 0.98836, -0.15211, 0.15211, -0.0625, 0 /
APPENDIX A. FDS INPUT FILES

&OBST XB = -0.13711,0.13711,-0.99056,0.99056,-0.0625,0 /
&OBST XB = -0.99056,0.99056,-0.13711,0.13711,-0.0625,0 /
&OBST XB = -0.12211,0.12211,-0.99252,0.99252,-0.0625,0 /
&OBST XB = -0.99252,0.99252,-0.12211,0.12211,-0.0625,0 /
&OBST XB = -0.10711,0.10711,-0.99425,0.99425,-0.0625,0 /
&OBST XB = -0.99425,0.99425,-0.10711,0.10711,-0.0625,0 /
&OBST XB = -0.092107,0.092107,-0.99575,0.99575,-0.0625,0 /
&OBST XB = -0.99575,0.99575,-0.092107,0.092107,-0.0625,0 /
&OBST XB = -0.077107,0.077107,-0.99702,0.99702,-0.0625,0 /
&OBST XB = -0.99702,0.99702,-0.077107,0.077107,-0.0625,0 /
&OBST XB = -0.062107,0.062107,-0.99807,0.99807,-0.0625,0 /
&OBST XB = -0.99807,0.99807,-0.062107,0.062107,-0.0625,0 /
&OBST XB = -0.047107,0.047107,-0.99889,0.99889,-0.0625,0 /
&OBST XB = -0.99889,0.99889,-0.047107,0.047107,-0.0625,0 /
&OBST XB = -0.032107,0.032107,-0.99948,0.99948,-0.0625,0 /
&OBST XB = -0.99948,0.99948,-0.032107,0.032107,-0.0625,0 /
&OBST XB = -0.017107,0.017107,-0.99985,0.99985,-0.0625,0 /
&OBST XB = -0.99985,0.99985,-0.017107,0.017107,-0.0625,0 /
&OBST XB = -0.0021068,0.0021068,-1,1,-0.0625,0 /
&OBST XB = -1,1,-0.0021068,0.0021068,-0.0625,0 /

&HOLE XB = -0.35355,0.35355,-0.35355,0.35355,-0.125,.1/
&HOLE XB = -0.35355,0.35355,-0.35355,0.35355,-0.125,.1/
&HOLE XB = -0.33755,0.33755,-0.36886,0.36886,-0.125,.1/
&HOLE XB = -0.36886,0.36886,-0.33755,0.33755,-0.125,.1/
&HOLE XB = -0.32155,0.32155,-0.38289,0.38289,-0.125,.1/
&HOLE XB = -0.38289,0.38289,-0.32155,0.32155,-0.125,.1/
&HOLE XB = -0.30555,0.30555,-0.39577,0.39577,-0.125,.1/
&HOLE XB = -0.39577,0.39577,-0.30555,0.30555,-0.125,.1/
&HOLE XB = -0.28955,0.28955,-0.40763,0.40763,-0.125,.1/
&HOLE XB = -0.40763,0.40763,-0.28955,0.28955,-0.125,.1/
&HOLE XB = -0.27355,0.27355,-0.41853,0.41853,-0.125,.1/
&HOLE XB = -0.41853,0.41853,-0.27355,0.27355,-0.125,.1/
&HOLE XB = -0.25755,0.25755,-0.42856,0.42856,-0.125,.1/
&HOLE XB = -0.42856,0.42856,-0.25755,0.25755,-0.125,.1/
&HOLE XB = -0.24155,0.24155,-0.43778,0.43778,-0.125,.1/
&HOLE XB = -0.43778,0.43778,-0.24155,0.24155,-0.125,.1/
&HOLE XB = -0.22555,0.22555,-0.44623,0.44623,-0.125,.1/
&HOLE XB = -0.44623,0.44623,-0.22555,0.22555,-0.125,.1/
&HOLE XB = -0.20955,0.20955,-0.45397,0.45397,-0.125,.1/
&HOLE XB = -0.45397,0.45397,-0.20955,0.20955,-0.125,.1/
&HOLE XB = -0.19355,0.19355,-0.46102,0.46102,-0.125,.1/
&HOLE XB = -0.46102,0.46102,-0.19355,0.19355,-0.125,.1/
APPENDIX A. FDS INPUT FILES

| &HOLE XB = \(-0.17755,0.17755,-0.46741,0.46741,-.125,.1/\) |
| &HOLE XB = \(-0.16155,0.16155,-0.47318,0.47318,-.125,.1/\) |
| &HOLE XB = \(-0.47318,0.47318,-0.16155,0.16155,-0.125,.1/\) |
| &HOLE XB = \(-0.14555,0.14555,-0.125,.1/\) |
| &HOLE XB = \(-0.12955,0.12955,-0.125,.1/\) |
| &HOLE XB = \(-0.48292,0.48292,-0.125,.1/\) |
| &HOLE XB = \(-0.11355,0.11355,-0.125,.1/\) |
| &HOLE XB = \(-0.14555,0.14555,-0.125,.1/\) |
| &HOLE XB = \(-0.12955,0.12955,-0.125,.1/\) |
| &HOLE XB = \(-0.097553,0.097553,-0.125,.1/\) |
| &HOLE XB = \(-0.49039,0.49039,-0.125,.1/\) |
| &HOLE XB = \(-0.081553,0.081553,-0.125,.1/\) |
| &HOLE XB = \(-0.4933,0.4933,-0.125,.1/\) |
| &HOLE XB = \(-0.065553,0.065553,-0.125,.1/\) |
| &HOLE XB = \(-0.49568,0.49568,-0.125,.1/\) |
| &HOLE XB = \(-0.49568,0.49568,-0.125,.1/\) |
| &HOLE XB = \(-0.49568,0.49568,-0.125,.1/\) |
| &HOLE XB = \(-0.49568,0.49568,-0.125,.1/\) |
| &HOLE XB = \(-0.49754,0.49754,-.125,.1/\) |
| &HOLE XB = \(-0.49754,0.49754,-.125,.1/\) |
| &HOLE XB = \(-0.033553,0.033553,-.125,.1/\) |
| &HOLE XB = \(-0.49887,0.49887,-.125,.1/\) |
| &HOLE XB = \(-0.033553,0.033553,-.125,.1/\) |
| &HOLE XB = \(-0.49887,0.49887,-.125,.1/\) |
| &HOLE XB = \(-0.017553,0.017553,-.125,.1/\) |
| &HOLE XB = \(-0.49969,0.49969,-.125,.1/\) |
| &HOLE XB = \(-0.0015534,0.0015534,-.125,.1/\) |
| &HOLE XB = \(-0.5,0.5,-.125,.1/\) |

Radial profiles of velocity and mass fraction

| &DEVC XB = \(-0.50,0.50,0.0,0.0,0.0,0.0,0.3,0.3, \text{QUANTITY}='W\text{-VELOCITY}', \text{ID} = 'Wp3', \text{POINTS}=21, \text{X_ID}='x' /\) |
| &DEVC XB = \(-0.50,0.50,0.0,0.0,0.0,0.5,0.5, \text{QUANTITY}='W\text{-VELOCITY}', \text{ID} = 'Wp5', \text{POINTS}=21, \text{HIDE\_COORDINATES=}=.TRUE. /\) |
| &DEVC XB = \(-0.50,0.50,0.0,0.0,0.9,0.9, \text{QUANTITY}='W\text{-VELOCITY}', \text{ID} = 'Wp9', \text{POINTS}=21, \text{HIDE\_COORDINATES=}=.TRUE. /\) |
| &DEVC XB = \(-0.50,0.50,0.0,0.0,0.3,0.3, \text{QUANTITY}='U\text{-VELOCITY}', \text{ID} = 'Up3', \text{POINTS}=21, \text{HIDE\_COORDINATES=}=.TRUE. /\) |
| &DEVC XB = \(-0.50,0.50,0.0,0.0,0.5,0.5, \text{QUANTITY}='U\text{-VELOCITY}', \text{ID} = 'Up5', \text{POINTS}=21, \text{HIDE\_COORDINATES=}=.TRUE. /\) |
| &DEVC XB = \(-0.50,0.50,0.0,0.0,0.9,0.9, \text{QUANTITY}='U\text{-VELOCITY}', \text{ID} = 'Up9', \text{POINTS}=21, \text{HIDE\_COORDINATES=}=.TRUE. /\) |
| &DEVC XB = \(-0.50,0.50,0.0,0.0,0.3,0.3, \text{QUANTITY}='W\text{-VELOCITY}', \text{ID} = 'Wp3\_rms', \text{STATISTICS}='\text{RMS}', \text{POINTS}=21, \text{HIDE\_COORDINATES=}=.TRUE. /\) |
APPENDIX A. FDS INPUT FILES

TRUE. / &DEVC XB = -0.50 , 0.50 , 0.0 , 0.0 , 0.5 , 0.5 , QUANTITY = 'W-VELOCITY' , ID = 'Wp5_rms' , STATISTICS = 'RMS' , POINTS = 21 , HIDE_COORDINATES =. TRUE. / &DEVC XB = -0.50 , 0.50 , 0.0 , 0.0 , 0.9 , 0.9 , QUANTITY = 'W-VELOCITY' , ID = 'Wp9_rms' , STATISTICS = 'RMS' , POINTS = 21 , HIDE_COORDINATES =. TRUE. / &DEVC XB = -0.50 , 0.50 , 0.0 , 0.0 , 0.3 , 0.3 , QUANTITY = 'U-VELOCITY' , ID = 'Up3_rms' , STATISTICS = 'RMS' , POINTS = 21 , HIDE_COORDINATES =. TRUE. / &DEVC XB = -0.50 , 0.50 , 0.0 , 0.0 , 0.5 , 0.5 , QUANTITY = 'U-VELOCITY' , ID = 'Up5_rms' , STATISTICS = 'RMS' , POINTS = 21 , HIDE_COORDINATES =. TRUE. / &DEVC XB = -0.50 , 0.50 , 0.0 , 0.0 , 0.9 , 0.9 , QUANTITY = 'U-VELOCITY' , ID = 'Up9_rms' , STATISTICS = 'RMS' , POINTS = 21 , HIDE_COORDINATES =. TRUE. / &TAIL /

A.3 Sandia test 24

&HEAD CHID = 'Sandia_CH4_1m_Test24_dx1p5cm' , TITLE = 'Sandia 1 m med flow rate methane pool fire (Test 24) , 1.5 cm resolution' / &MULT ID = 'mesh array' , DX = 1.5 , DY = 1.5 , DZ = 1.0 , I_UPPER = 1 , J_UPPER = 1 , K_UPPER = 3 /
&MESH IJK = 96,96,64 , XB = -1.5,0.0,-1.5,0.0,-.0625,0.9375 , MULT_ID = 'mesh array' /

&TIME T_END = 20. /
&MISC TMPA = 17.
  P_INF = 81300.
  TURBULENCE_MODEL = 'VREMAN' /

&REAC FUEL = 'METHANE'
  HEAT_OF_COMBUSTION = 49728.
  CRITICAL_FLAME_TEMPERATURE = 1207 /

&RADI RADIATIVE_FRACTION = 0.2 /
&DUMP SIG_FIGS = 4 , SIG_FIGS_EXP = 2 /
&MATL ID = 'STEEL' , CONDUCTIVITY = 54. , SPECIFIC_HEAT = 0.465 ,
  DENSITY = 7850. , EMISSIVITY = 0.9 /
APPENDIX A. FDS INPUT FILES

&SURF ID = 'PLATE', COLOR = 'GRAY', DEFAULT = .TRUE., MATL_ID = 'STEEL', THICKNESS = 0.025

&SURF ID = 'POOL', MASS_FLUX(1) = 0.053, SPEC_ID(1) = 'METHANE', TMP_FRONT = 15.

&VENT XB = -0.5,0.5,-0.5,0.5,-.0625,-.0625,COLOR = 'BLUE', SURF_ID = 'POOL'

&VENT XB = -1.5,-.5,-1.5,1.5,-.0625,-.0625,SURF_ID = 'OPEN'
&VENT XB = .5,1.5,-1.5,1.5,-.0625,-.0625,SURF_ID = 'OPEN'
&VENT XB = -1.5,1.5,-1.5,-.5,-.0625,-.0625,SURF_ID = 'OPEN'
&VENT XB = -1.5,1.5,.5,1.5,-.0625,-.0625,SURF_ID = 'OPEN'

&VENT MB = 'ZMAX', SURF_ID = 'OPEN'
&VENT MB = 'YMIN', SURF_ID = 'OPEN'
&VENT MB = 'YMAX', SURF_ID = 'OPEN'
&VENT MB = 'XMIN', SURF_ID = 'OPEN'
&VENT MB = 'XMAX', SURF_ID = 'OPEN'

&SLCF PBY = -.0625, QUANTITY = 'VELOCITY', VECTOR = .TRUE.
&SLCF PBY = -.0625, QUANTITY = 'DENSITY', CELL_CENTERED = .TRUE.
&SLCF PBY = -.0625, QUANTITY = 'TEMPERATURE', CELL_CENTERED = .TRUE.

&SLCF PBY = -.0625, QUANTITY = 'HRPPUV', CELL_CENTERED = .TRUE.
&SLCF PBY = -.0625, QUANTITY = 'HRPPUA', CELL_CENTERED = .TRUE.
&SLCF PBY = -.0625, QUANTITY = 'MASS FRACTION', SPEC_ID = 'METHANE', CELL_CENTERED = .TRUE.
&SLCF PBY = -.0625, QUANTITY = 'CSMAG', CELL_CENTERED = .TRUE.
&SLCF PBY = -.0625, QUANTITY = 'TURBULENCE RESOLUTION', CELL_CENTERED = .TRUE.
&SLCF PBY = -.0625, QUANTITY = 'WAVELET_ERROR', QUANTITY2 = 'MASS FRACTION', SPEC_ID = 'METHANE', CELL_CENTERED = .TRUE.

&OBST XB = -0.70711,.70711,-0.70711,.70711,-.0625,0
&OBST XB = -0.70711,.70711,-0.70711,.70711,-.0625,0
&OBST XB = -0.69211,.69211,-0.7218,.7218,-.0625,0
&OBST XB = -0.7218,.7218,-.69211,.69211,-.0625,0
&OBST XB = -0.67711,.67711,-0.73588,.73588,-.0625,0
&OBST XB = -0.73588,.73588,-0.67711,.67711,-.0625,0
&OBST XB = -0.66211,.66211,-0.74941,.74941,-.0625,0
&OBST XB = -0.74941,.74941,-.66211,.66211,-.0625,0
&OBST XB = -0.64711,.64711,-0.7624,.7624,-.0625,0
| &OBST XB | -0.7624, 0.7624, -0.6471, 0.6471, -0.0625, 0 |
| &OBST XB | -0.6321, 0.6321, -0.7748, 0.7748, -0.0625, 0 |
| &OBST XB | -0.7748, 0.7748, -0.6321, 0.6321, -0.0625, 0 |
| &OBST XB | -0.6171, 0.6171, -0.7868, 0.7868, -0.0625, 0 |
| &OBST XB | -0.7868, 0.7868, -0.6171, 0.6171, -0.0625, 0 |
| &OBST XB | -0.6021, 0.6021, -0.7984, 0.7984, -0.0625, 0 |
| &OBST XB | -0.7984, 0.7984, -0.6021, 0.6021, -0.0625, 0 |
| &OBST XB | -0.5871, 0.5871, -0.8095, 0.8095, -0.0625, 0 |
| &OBST XB | -0.8095, 0.8095, -0.5871, 0.5871, -0.0625, 0 |
| &OBST XB | -0.5721, 0.5721, -0.8201, 0.8201, -0.0625, 0 |
| &OBST XB | -0.8201, 0.8201, -0.5721, 0.5721, -0.0625, 0 |
| &OBST XB | -0.5571, 0.5571, -0.8304, 0.8304, -0.0625, 0 |
| &OBST XB | -0.8304, 0.8304, -0.5571, 0.5571, -0.0625, 0 |
| &OBST XB | -0.5421, 0.5421, -0.8403, 0.8403, -0.0625, 0 |
| &OBST XB | -0.8403, 0.8403, -0.5421, 0.5421, -0.0625, 0 |
| &OBST XB | -0.5271, 0.5271, -0.8498, 0.8498, -0.0625, 0 |
| &OBST XB | -0.8498, 0.8498, -0.5271, 0.5271, -0.0625, 0 |
| &OBST XB | -0.5121, 0.5121, -0.8589, 0.8589, -0.0625, 0 |
| &OBST XB | -0.8589, 0.8589, -0.5121, 0.5121, -0.0625, 0 |
| &OBST XB | -0.4971, 0.4971, -0.8676, 0.8676, -0.0625, 0 |
| &OBST XB | -0.8676, 0.8676, -0.4971, 0.4971, -0.0625, 0 |
| &OBST XB | -0.4821, 0.4821, -0.8761, 0.8761, -0.0625, 0 |
| &OBST XB | -0.8761, 0.8761, -0.4821, 0.4821, -0.0625, 0 |
| &OBST XB | -0.4671, 0.4671, -0.8842, 0.8842, -0.0625, 0 |
| &OBST XB | -0.8842, 0.8842, -0.4671, 0.4671, -0.0625, 0 |
| &OBST XB | -0.4521, 0.4521, -0.8919, 0.8919, -0.0625, 0 |
| &OBST XB | -0.8919, 0.8919, -0.4521, 0.4521, -0.0625, 0 |
| &OBST XB | -0.4371, 0.4371, -0.8994, 0.8994, -0.0625, 0 |
| &OBST XB | -0.8994, 0.8994, -0.4371, 0.4371, -0.0625, 0 |
| &OBST XB | -0.4221, 0.4221, -0.9065, 0.9065, -0.0625, 0 |
| &OBST XB | -0.9065, 0.9065, -0.4221, 0.4221, -0.0625, 0 |
| &OBST XB | -0.4071, 0.4071, -0.9133, 0.9133, -0.0625, 0 |
| &OBST XB | -0.9133, 0.9133, -0.4071, 0.4071, -0.0625, 0 |
| &OBST XB | -0.3921, 0.3921, -0.9199, 0.9199, -0.0625, 0 |
| &OBST XB | -0.9199, 0.9199, -0.3921, 0.3921, -0.0625, 0 |
| &OBST XB | -0.3771, 0.3771, -0.9261, 0.9261, -0.0625, 0 |
| &OBST XB | -0.9261, 0.9261, -0.3771, 0.3771, -0.0625, 0 |
| &OBST XB | -0.3621, 0.3621, -0.9321, 0.9321, -0.0625, 0 |
| &OBST XB | -0.9321, 0.9321, -0.3621, 0.3621, -0.0625, 0 |
| &OBST XB | -0.3471, 0.3471, -0.9378, 0.9378, -0.0625, 0 |
| &OBST XB | -0.9378, 0.9378, -0.3471, 0.3471, -0.0625, 0 |
| &OBST XB | -0.3321, 0.3321, -0.9432, 0.9432, -0.0625, 0 |
| &OBST XB | -0.9432, 0.9432, -0.3321, 0.3321, -0.0625, 0 |
APPENDIX A. FDS INPUT FILES

&OBST XB = -0.31711, 0.31711, -0.94839, 0.94839, -0.0625, 0 /
&OBST XB = -0.94839, 0.94839, -0.31711, 0.31711, -0.0625, 0 /
&OBST XB = -0.30211, 0.30211, -0.95327, 0.95327, -0.0625, 0 /
&OBST XB = -0.95327, 0.95327, -0.30211, 0.30211, -0.0625, 0 /
&OBST XB = -0.28711, 0.28711, -0.9579, 0.9579, -0.0625, 0 /
&OBST XB = -0.9579, 0.9579, -0.28711, 0.28711, -0.0625, 0 /
&OBST XB = -0.27211, 0.27211, -0.96227, 0.96227, -0.0625, 0 /
&OBST XB = -0.96227, 0.96227, -0.27211, 0.27211, -0.0625, 0 /
&OBST XB = -0.25711, 0.25711, -0.96638, 0.96638, -0.0625, 0 /
&OBST XB = -0.96638, 0.96638, -0.25711, 0.25711, -0.0625, 0 /
&OBST XB = -0.24211, 0.24211, -0.97025, 0.97025, -0.0625, 0 /
&OBST XB = -0.97025, 0.97025, -0.24211, 0.24211, -0.0625, 0 /
&OBST XB = -0.22711, 0.22711, -0.97387, 0.97387, -0.0625, 0 /
&OBST XB = -0.97387, 0.97387, -0.22711, 0.22711, -0.0625, 0 /
&OBST XB = -0.21211, 0.21211, -0.97725, 0.97725, -0.0625, 0 /
&OBST XB = -0.97725, 0.97725, -0.21211, 0.21211, -0.0625, 0 /
&OBST XB = -0.19711, 0.19711, -0.98038, 0.98038, -0.0625, 0 /
&OBST XB = -0.98038, 0.98038, -0.19711, 0.19711, -0.0625, 0 /
&OBST XB = -0.18211, 0.18211, -0.98328, 0.98328, -0.0625, 0 /
&OBST XB = -0.98328, 0.98328, -0.18211, 0.18211, -0.0625, 0 /
&OBST XB = -0.16711, 0.16711, -0.98594, 0.98594, -0.0625, 0 /
&OBST XB = -0.98594, 0.98594, -0.16711, 0.16711, -0.0625, 0 /
&OBST XB = -0.15211, 0.15211, -0.98836, 0.98836, -0.0625, 0 /
&OBST XB = -0.98836, 0.98836, -0.15211, 0.15211, -0.0625, 0 /
&OBST XB = -0.13711, 0.13711, -0.99056, 0.99056, -0.0625, 0 /
&OBST XB = -0.99056, 0.99056, -0.13711, 0.13711, -0.0625, 0 /
&OBST XB = -0.12211, 0.12211, -0.99252, 0.99252, -0.0625, 0 /
&OBST XB = -0.99252, 0.99252, -0.12211, 0.12211, -0.0625, 0 /
&OBST XB = -0.10711, 0.10711, -0.99425, 0.99425, -0.0625, 0 /
&OBST XB = -0.99425, 0.99425, -0.10711, 0.10711, -0.0625, 0 /
&OBST XB = -0.092107, 0.092107, -0.99575, 0.99575, -0.0625, 0 /
&OBST XB = -0.99575, 0.99575, -0.092107, 0.092107, -0.0625, 0 /
&OBST XB = -0.077107, 0.077107, -0.99702, 0.99702, -0.0625, 0 /
&OBST XB = -0.99702, 0.99702, -0.077107, 0.077107, -0.0625, 0 /
&OBST XB = -0.062107, 0.062107, -0.99807, 0.99807, -0.0625, 0 /
&OBST XB = -0.99807, 0.99807, -0.062107, 0.062107, -0.0625, 0 /
&OBST XB = -0.047107, 0.047107, -0.99889, 0.99889, -0.0625, 0 /
&OBST XB = -0.99889, 0.99889, -0.047107, 0.047107, -0.0625, 0 /
&OBST XB = -0.032107, 0.032107, -0.99948, 0.99948, -0.0625, 0 /
&OBST XB = -0.99948, 0.99948, -0.032107, 0.032107, -0.0625, 0 /
&OBST XB = -0.017107, 0.017107, -0.99985, 0.99985, -0.0625, 0 /
&OBST XB = -0.99985, 0.99985, -0.017107, 0.017107, -0.0625, 0 /
&OBST XB = -0.0021068, 0.0021068, -1, 1, -0.0625, 0 /
APPENDIX A. FDS INPUT FILES

&OBST XB = -1,1,-0.0021068,0.0021068,-0.0625,0 /

&HOLE XB = -0.35355,0.35355,-0.35355,0.35355,-0.125,1/
&HOLE XB = -0.33755,0.33755,-0.36886,0.36886,-0.125,1/
&HOLE XB = -0.36886,0.36886,-0.33755,0.33755,-0.125,1/
&HOLE XB = -0.32155,0.32155,-0.38289,0.38289,-0.125,1/
&HOLE XB = -0.38289,0.38289,-0.32155,0.32155,-0.125,1/
&HOLE XB = -0.30555,0.30555,-0.39577,0.39577,-0.125,1/
&HOLE XB = -0.39577,0.39577,-0.30555,0.30555,-0.125,1/
&HOLE XB = -0.28955,0.28955,-0.40763,0.40763,-0.125,1/
&HOLE XB = -0.40763,0.40763,-0.28955,0.28955,-0.125,1/
&HOLE XB = -0.27355,0.27355,-0.41853,0.41853,-0.125,1/
&HOLE XB = -0.41853,0.41853,-0.27355,0.27355,-0.125,1/
&HOLE XB = -0.25755,0.25755,-0.42856,0.42856,-0.125,1/
&HOLE XB = -0.42856,0.42856,-0.25755,0.25755,-0.125,1/
&HOLE XB = -0.24155,0.24155,-0.43778,0.43778,-0.125,1/
&HOLE XB = -0.43778,0.43778,-0.24155,0.24155,-0.125,1/
&HOLE XB = -0.22555,0.22555,-0.44623,0.44623,-0.125,1/
&HOLE XB = -0.44623,0.44623,-0.22555,0.22555,-0.125,1/
&HOLE XB = -0.20955,0.20955,-0.45397,0.45397,-0.125,1/
&HOLE XB = -0.45397,0.45397,-0.20955,0.20955,-0.125,1/
&HOLE XB = -0.19355,0.19355,-0.46102,0.46102,-0.125,1/
&HOLE XB = -0.46102,0.46102,-0.19355,0.19355,-0.125,1/
&HOLE XB = -0.17755,0.17755,-0.46741,0.46741,-0.125,1/
&HOLE XB = -0.46741,0.46741,-0.17755,0.17755,-0.125,1/
&HOLE XB = -0.16155,0.16155,-0.47318,0.47318,-0.125,1/
&HOLE XB = -0.47318,0.47318,-0.16155,0.16155,-0.125,1/
&HOLE XB = -0.14555,0.14555,-0.47835,0.47835,-0.125,1/
&HOLE XB = -0.47835,0.47835,-0.14555,0.14555,-0.125,1/
&HOLE XB = -0.12955,0.12955,-0.48292,0.48292,-0.125,1/
&HOLE XB = -0.48292,0.48292,-0.12955,0.12955,-0.125,1/
&HOLE XB = -0.11355,0.11355,-0.48693,0.48693,-0.125,1/
&HOLE XB = -0.48693,0.48693,-0.11355,0.11355,-0.125,1/
&HOLE XB = -0.097553,0.097553,-0.49039,0.49039,-0.125,1/
&HOLE XB = -0.49039,0.49039,-0.097553,0.097553,-0.125,1/
&HOLE XB = -0.081553,0.081553,-0.4933,0.4933,-0.125,1/
&HOLE XB = -0.4933,0.4933,-0.081553,0.081553,-0.125,1/
&HOLE XB = -0.065553,0.065553,-0.49568,0.49568,-0.125,1/
&HOLE XB = -0.49568,0.49568,-0.065553,0.065553,-0.125,1/
&HOLE XB = -0.049553,0.049553,-0.49754,0.49754,-0.125,1/
&HOLE XB = -0.49754,0.49754,-0.049553,0.049553,-0.125,1/
&HOLE XB = -0.033553,0.033553,-0.49887,0.49887,-0.125,1/
APPENDIX A. FDS INPUT FILES

&HOLE XB=−0.49887,0.49887,−0.03353,0.03353,−.125,1 /
&HOLE XB=−0.017553,0.017553,−0.49969,0.49969,−.125,1 /
&HOLE XB=−0.49969,0.49969,−0.017553,0.017553,−.125,1 /
&HOLE XB=−0.0015534,0.0015534,−.5,0.5,−.125,1 /
&HOLE XB=−.5,0.5,−0.0015534,0.0015534,−.125,1 /

Radial profiles of velocity and mass fraction

&DEVC XB=−0.5,0.5,0.0,0.0,0.3,0.3, QUANTITY=’W-VELOCITY’ , ID =’Wp3’ , POINTS=21, X_ID=’x’ /
&DEVC XB=−0.5,0.5,0.0,0.0,0.5,0.5, QUANTITY=’W-VELOCITY’ , ID =’Wp5’ , POINTS=21, HIDE_COORDINATES=.TRUE. /
&DEVC XB=−0.5,0.5,0.0,0.0,0.9,0.9, QUANTITY=’W-VELOCITY’ , ID =’Wp9’ , POINTS=21, HIDE_COORDINATES=.TRUE. /

&DEVC XB=−0.5,0.5,0.0,0.0,0.3,0.3, QUANTITY=’U-VELOCITY’ , ID =’Up3’ , POINTS=21, HIDE_COORDINATES=.TRUE. /
&DEVC XB=−0.5,0.5,0.0,0.0,0.5,0.5, QUANTITY=’U-VELOCITY’ , ID =’Up5’ , POINTS=21, HIDE_COORDINATES=.TRUE. /
&DEVC XB=−0.5,0.5,0.0,0.0,0.9,0.9, QUANTITY=’U-VELOCITY’ , ID =’Up9’ , POINTS=21, HIDE_COORDINATES=.TRUE. /

&DEVC XB=−0.5,0.5,0.0,0.0,0.3,0.3, QUANTITY=’W-VELOCITY’ , ID =’Wp3_rms’ , STATISTICS=’RMS’ , POINTS=21, HIDE_COORDINATES=. TRUE. /
&DEVC XB=−0.5,0.5,0.0,0.0,0.5,0.5, QUANTITY=’W-VELOCITY’ , ID =’Wp5_rms’ , STATISTICS=’RMS’ , POINTS=21, HIDE_COORDINATES=. TRUE. /
&DEVC XB=−0.5,0.5,0.0,0.0,0.9,0.9, QUANTITY=’W-VELOCITY’ , ID =’Wp9_rms’ , STATISTICS=’RMS’ , POINTS=21, HIDE_COORDINATES=. TRUE. /

&DEVC XB=−0.5,0.5,0.0,0.0,0.3,0.3, QUANTITY=’U-VELOCITY’ , ID =’Up3_rms’ , STATISTICS=’RMS’ , POINTS=21, HIDE_COORDINATES=. TRUE. /
&DEVC XB=−0.5,0.5,0.0,0.0,0.5,0.5, QUANTITY=’U-VELOCITY’ , ID =’Up5_rms’ , STATISTICS=’RMS’ , POINTS=21, HIDE_COORDINATES=. TRUE. /
&DEVC XB=−0.5,0.5,0.0,0.0,0.9,0.9, QUANTITY=’U-VELOCITY’ , ID =’Up9_rms’ , STATISTICS=’RMS’ , POINTS=21, HIDE_COORDINATES=. TRUE. /

&TAIL /
APPENDIX A. FDS INPUT FILES

A.4 Heskestad

&HEAD CHID='Qs=10000_RI=20', TITLE='Flame Height Test, Q *=10000' /
&MESH IJK=65.65.160, XB=-72.0,72.0,-72.0,72.0,-18.0,342. /
&MISC TURBULENCE_MODEL='VREMAN' /
&TIME T_END=200. /

&REAC FUEL='PROPANE', C=3., H=8., SOOT_YIELD=0.015 /
&SURF ID='burner', HRRPUA=15127250., COLOR='RED' /
&OBST XB=-0.5,0.5,-0.5,0.5,-5.0,0.0, SURF_IDS='burner','INERT ' , 'INERT', THICKEN=.TRUE. /

&VENT MB='XMIN', SURF_ID='OPEN' /
&VENT MB='XMAX', SURF_ID='OPEN' /
&VENT MB='YMIN', SURF_ID='OPEN' /
&VENT MB='YMAX', SURF_ID='OPEN' /
&VENT MB='ZMIN', SURF_ID='OPEN' /
&VENT MB='ZMAX', SURF_ID='OPEN' /

&SLECF PBY=0., QUANTITY='TEMPERATURE', VECTOR=.TRUE. /
&SLECF PBY=0., QUANTITY='HRRPUV' /
&SLECF PBY=0., QUANTITY='MIXING TIME' /
&DEV.readLine XB=0.0,0.0,0.0,0.0,1.12,340.77, QUANTITY='HRRPUL', POINTS=152, Z_ID='Height', ID='HRRPUL' /
&TAIL /

A.5 McCaffery

&HEAD CHID='McCaffrey_57_kW_fine', TITLE='McCaffrey, NBSIR 79–1910, 57 kW Natural Gas' /
&MISC TURBULENCE_MODEL='VREMAN' /
&MULT ID='mesh', DX=0.62, DY=0.62, DZ=1.55, I_UPPER=2, J_UPPER=2, K_UPPER=3 /
&MESH IJK=40,40,100, XB=-0.93,-0.31,-0.93,-0.31,-0.248,1.302, MULT_ID='mesh' /

&TIME T_END=30. /

&SURF ID='burner', HRRPUA=639., TEMP_FRONT=100., COLOR='RED' /
&OBST XB=-.15,.15,-.15,.15,-.10,0.00, SURF_IDS='burner','INERT ' , 'INERT' /
APPENDIX A. FDS INPUT FILES

&REAC FUEL = 'METHANE'
   C=1.
   H=4.
   CO_YIELD=0.0
   SOOT_YIELD=0.0 /

&RADI RADIATIVE_FRACTION=0.20 /

&VENT MB='XMIN', SURF_ID='OPEN' /
&VENT MB='XMAX', SURF_ID='OPEN' /
&VENT MB='YMIN', SURF_ID='OPEN' /
&VENT MB='YMAX', SURF_ID='OPEN' /
&VENT MB='ZMIN', SURF_ID='OPEN' /
&VENT MB='ZMAX', SURF_ID='OPEN' /

&DEVCE ID='temp20', XB=0.00,0.00,0.00,0.00,0.0155,5.9365, POINTS =192, QUANTITY='TEMPERATURE', Z_ID='Height' /
&DEVCE ID='velo20', XB=0.00,0.00,0.00,0.00,0.0155,5.9365, POINTS =192, QUANTITY='W-VELOCITY', HIDE_COORDINATES=.TRUE. /

&DEVCE XYZ=0,0,0,3, QUANTITY='W-VELOCITY' /

&SLCF PBY=0.0, QUANTITY='WAVELET ERROR', QUANTITY2='MASS FRACTION', SPEC_ID='METHANE' /
&SLCF PBY=0.0, QUANTITY='WAVELET ERROR', QUANTITY2='HRRPUV' /
&SLCF PBY=0.0, QUANTITY='WAVELET ERROR', QUANTITY2='TEMPERATURE' /
&SLCF PBY=0.0, QUANTITY='TURBULENCE RESOLUTION' /

&SLCF PBY=0.0, QUANTITY='TEMPERATURE', VECTOR=.TRUE. /
&SLCF PBY=0.0, QUANTITY='HRRPUV' /

&TAIL /
Appendix B

Implemented code

B.1 fire.f90

```fortran
MODULE FIRE
  ! Compute combustion
  USE PRECISION_PARAMETERS
  USE GLOBAL_CONSTANTS
  USE MESH_POINTERS
  USE COMP_FUNCTIONS, ONLY: SECOND

IMPLICIT NONE
PRIVATE
CHARACTER(255), PARAMETER :: fireid='$Id: fire.f90 10216 2012-03-08 16:22:22Z craigweinschenk$'
CHARACTER(255), PARAMETER :: firerev='$Revision: 10216$'
CHARACTER(255), PARAMETER :: firedate='$ Date: 2012-03-08 17:22:22+0100 (to , 08 mar 2012)$'

TYPE(REACTION_TYPE), POINTER :: RN=>NULL()
REAL(EB) :: Q_UPPER
PUBLIC COMBUSTION, GET_REV_fire
CONTAINS
SUBROUTINE COMBUSTION(NM)

INTEGER, INTENT(IN) :: NM
```
APPENDIX B. IMPLEMENTED CODE

27 REAL(EB) :: TNOW
28
29 IF (EVACUATION_ONLY(NM)) RETURN
30
31 TNOW=SECOND()
32
33 IF (INIT_HRRPUV) RETURN
34
35 CALL POINT_TO_MESH(NM)
36
37 ! CALL COMPUTE_STRAIN_RATE(NM)
38
39 ! Upper bounds on local HRR per unit volume
40
41 Q_UPPER = HRRPUA_SHEET/CELL_SIZE + HRRPUV_AVERAGE
42
43 ! Call combustion ODE solver
44 CALL COMBUSTION_GENERAL
45
46 TUSED(10 ,NM)=TUSED(10 ,NM)+SECOND()−TNOW
47
48 END SUBROUTINE COMBUSTION
49
50
51 SUBROUTINE COMBUSTION_GENERAL
52
53 ! Generic combustion routine for multi step reactions with
54 ! kinetics either mixing controlled, finite rate,
55 ! or a temperature threshold mixed approach
56
57 USE PHYSICAL_FUNCTIONS, ONLY: GET_SPECIFIC_GAS_CONSTANT,
GET_MASS_FRACTION_ALL, GET_SPECIFIC_HEAT, GET_MOLECULAR_WEIGHT
, &
58 GET_SENSIBLE_ENTHALPY_DIFF,
GET_MASS_FRACTION !ADDED
59
60 INTEGER :: I,J,K,NS,NR,II,JJ,KK,IIG,JJG,KKG,IW,N
61 REAL(EB) :: ZZ_GET(0:N_TRACKED_SPECIES),ZZ_MIN=1.E−10,EB,DZZ(0:
N_TRACKED_SPECIES),CP,HDIFF,Y_O2,Y_FUEL,Y_PRODUCT !ADDED
62 LOGICAL :: DO_REACTION,REACTANTS_PRESENT,Q_EXISTS
63 TYPE (REACTION_TYPE) POINTER :: RN
64 TYPE (SPECIES_MIXTURE_TYPE), POINTER :: SM,SM0
65
66 Q = 0.0EB
APPENDIX B. IMPLEMENTED CODE

142

DREACTION = 0. EB
Q_EXISTS = . FALSE.
SM0 \Rightarrow \text{SPECIES}_{\text{MIXTURE}}(0)

DO K=1,KBAR
  DO J=1,JBAR
    ILOOP: DO I=1,IBAR
      ! Check to see if a reaction is possible
      IF (SOLID(C\text{ELL\_INDEX}(I,J,K))) CYCLE ILOOP
      ZZ\_GET(1:N\_TRACKED\_SPECIES) = ZZ(I,J,K,1:
        N\_TRACKED\_SPECIES)
      ZZ\_GET(0) = 1. EB - MIN(1. EB, SUM(ZZ\_GET(1:
        N\_TRACKED\_SPECIES)))
      DO REACTION = . FALSE.
      DO NR=1,N\_REACTIONS
        RN = REACTION(NR)
        REACTANTS\_PRESENT = . TRUE.
        DO NS=0,N\_TRACKED\_SPECIES
          IF (RN \% NU(NS) < 0. EB . AND. ZZ\_GET(NS) < ZZ\_MIN)
            THEN
              REACTANTS\_PRESENT = . FALSE.
              EXIT
          ENDIF
        END DO
        IF (. NOT. DO\_REACTION) DO\_REACTION =
          REACTANTS\_PRESENT
      END DO
      IF (. NOT. DO\_REACTION) CYCLE ILOOP
      DZZ(1:N\_TRACKED\_SPECIES) = ZZ\_GET(1:N\_TRACKED\_SPECIES)
      ! Store old ZZ for divergence term
      ! Easily allow for user selected ODE solver
      SELECT CASE (COMBUSTION\_ODE)
        \text{CASE}(\text{SINGLE\_EXACT})
        CALL ODE\_EXACT(I,J,K,ZZ\_GET,Q(I,J,K))
        CALL ODE\_EXPLICIT\_EULER(I,J,K,ZZ\_GET,Q(I,J,K))
        \text{CASE}(\text{EXPLICIT\_EULER})
        CALL ODE\_EXPLICIT\_EULER(I,J,K,ZZ\_GET,Q(I,J,K))
        \text{CASE}(\text{RUNGE\_KUTTA}_2)
        CALL ODE\_RUNGE\_KUTTA_2(I,J,K,ZZ\_GET,Q(I,J,K))
        \text{CASE}(\text{RK2\_RICHARDSON})
        CALL ODE\_RK2\_RICHARDSON(I,J,K,ZZ\_GET,Q(I,J,K))
      END SELECT

101
APPENDIX B. IMPLEMENTED CODE

103 ! Update RSUM and ZZ
104 IF (ABS(Q(I,J,K)) > ZERO_P) THEN
105 Q_EXISTS = .TRUE.
106 CALL GET_SPECIFIC_GAS_CONSTANT(ZZ_GET,RSUM(I,J,K))
107 TMP(I,J,K) = PBAR(K,PRESSURE_ZONE(I,J,K))/(RSUM(I,J,K)*RHO(I,J,K))
108 ZZ(I,J,K,1:N_TRACKED_SPECIES) = ZZ_GET(1:N_TRACKED_SPECIES)
109 ! Divergence term
110 DZZ(1:N_TRACKED_SPECIES) = ZZ_GET(1:N_TRACKED_SPECIES) - DZZ(1:N_TRACKED_SPECIES)
111 CALL GET_SPECIFIC_HEAT(ZZ_GET, CP,TMP(I,J,K))
112 DO N=1,N_TRACKED_SPECIES
113 SM => SPECIES_MIXTURE(N)
114 CALL GET_SENSIBLE_ENTHALPY_DIFF(N,TMP(I,J,K),HDIFF)
115 D_REACTION(I,J,K) = D_REACTION(I,J,K) + ((SM%RCON-SM0%RCON)/RSUM(I,J,K)) * (HDIFF/(CP*TMP(I,J,K)) * DZZ(N)/DT)
116 ENDDO
117 ENDIF
118 ENDDO ILOOP
119 ENDDO
120 IF (.NOT. Q_EXISTS) RETURN
121 ENDIF
122 ENDDO
123
124 ! Set Q in the ghost cell, just for better visualization.
125 DO IW=1,N_EXTERNAL_WALL.Cells
126 IF (WALL(IW)%BOUNDARY_TYPE/=INTERPOLATED_BOUNDARY .AND. WALL(IW)%BOUNDARY_TYPE/=OPEN_BOUNDARY) CYCLE
127 II = WALL(IW)%ONE_D%II
128 ENDDO
JJP = WALL(IW)%ONE%JJ
KK = WALL(IW)%ONE%KK
IIG = WALL(IW)%ONE%IIG
JJG = WALL(IW)%ONE%JJG
KKG = WALL(IW)%ONE%KKG
Q(IIG, JJG, KKG) = Q(IIG, JJG, KKG)

END SUBROUTINE COMBUSTIONGENERAL

SUBROUTINE ODEEXACT(I, J, K, ZZGET, QNEW)
INTEGER, INTENT(IN) :: I, J, K
REAL(EB), INTENT(INOUT) :: ZZGET(0: NTRACKEDSPECIES)
REAL(EB) :: DZF, QBOUND_1, QBOUND_2, RATECONSTANT, ZLIMITER,
            REACTANTMIN, DT2
LOGICAL :: MIN FOUND
INTEGER :: NS
TYPE(REACTIONTYPE) :: RN = NULL(

QNEW = 0._EB
RN = REACTION(1)
CALL COMPUTE RATECONSTANT(1, RN%MODE, 1, 0._EB, RATECONSTANT,
                         ZZGET, I, J, K)

IF (RATECONSTANT < ZERO_P) RETURN

ZLIMITER = RATECONSTANT*MIX_TIME(I, J, K)

DZF = -1._EB
! Check for reactant (i.e. fuel or oxidizer) limited combustion
MIN FOUND = .FALSE.
REACTANTMIN = 1._EB
DO NS=0, NTRACKEDSPECIES
  IF (RN%NU(NS) < -ZERO_P) &
    REACTANTMIN = MIN(REACTANTMIN, -ZZGET(NS)*
                     SPECIESMIXTURE(RN%FUEL_SMIX_INDEX)%MW/
                       (SPECIESMIXTURE(NS)%MW*RN%NU(NS)))
  IF (ABS(ZLIMITER - REACTANTMIN) <= SPACING(ZLIMITER)) THEN
    MIN FOUND = .TRUE.
    DZF = REACTANTMIN*(1._EB-EXP(-DT/MIX TIME(I, J, K)))
  EXIT
END DO

APPENDIX B. IMPLEMENTED CODE
APPENDIX B. IMPLEMENTED CODE

ENDIF

ENDDO

! For product limited combustio n find time of switch from product
limited to reactant limited (if it occurs)
! and do two step exact solution

IF (.NOT. MIN_FOUND) THEN

DT2 = MIX_TIME(I,J,K) * LOG((Z_LIMITER+REACTANT_MIN) / (2.*EB*
Z_LIMITER))

IF (DT2 < DT) THEN

DZF = ZZ_GET(RN%FUEL_SMIX_INDEX) - Z_LIMITER*(EXP(DT2/
MIX_TIME(I,J,K)) - 1.*EB)

REACTANT_MIN = REACTANT_MIN - DZF

DZF = DZF + REACTANT_MIN* (1.*EB-EXP(-(DT-DT2)/MIX_TIME(I,
J,K)) )

ELSE

DZF = ZZ_GET(RN%FUEL_SMIX_INDEX) - Z_LIMITER*(EXP(DT/
MIX_TIME(I,J,K)) - 1.*EB)

ENDIF

ENDIF

DZF = MIN(DZF, ZZ_GET(RN%FUEL_SMIX_INDEX))

! ******** TEMP OVERRIDE TO ENSURE SAME RESULTS AS PREVIOUS

********

! DZF = Z_LIMITER* (1.*EB-EXP(-DT/MIX_TIME(I,J,K)) )

!

*******************************************************************

Q_BOUND_1 = DZF*RHO(I,J,K)*RN%HEAT_OF_COMBUSTION/DT

Q_BOUND_2 = Q_UPPER

Q_NEW = MIN(Q_BOUND_1, Q_BOUND_2)

DZF = Q_NEW*DT/(RHO(I,J,K)*RN%HEAT_OF_COMBUSTION)

ZZ_GET = ZZ_GET + DZF*RN%NU*SPECIES_MIXTURE%M/M*SPECIES_MIXTURE( RN%FUEL_SMIX_INDEX)%M

END SUBROUTINE ODE_EXACT

SUBROUTINE ODE_EXPLICIT_EULER(I,J,K,ZZ_GET, Q_OUT)

INTEGER,INTENT(IN) :: I,J,K
APPENDIX B. IMPLEMENTED CODE

202 REAL(EB) , INTENT(OUT) :: Q.OUT
203 REAL(EB) , INTENT(INOUT) :: ZZ.GET (0:N_TRACKED_SPECIES)
204 REAL(EB) :: ZZ.0 (0:N_TRACKED_SPECIES) , ZZ.I (0:N_TRACKED_SPECIES)
205 DT.ODE, DT.NEW, RATE_CONSTANT (1:N_REACTIONS) , Q.NR (1:N_REACTIONS) , Q.SUM, DT.SUM

206 INTEGER :: NR, I.TS , NS
207 INTEGER , PARAMETER :: NODETS = 20
208 TYPE(REACTION_TYPE) , POINTER :: RN => NULL()

210 Q.OUT = 0. _EB
211 ZZ.0 = MAX (0. _EB , ZZ.GET)
212 ZZ.I = ZZ.0
213 DT.ODE = DT/REAL(NODETS,EB)
214 DT.NEW = DT.ODE
215 DT.SUM = 0. _EB
216 I.TS = 1

ODE_LOOP: DO WHILE (DT.SUM < DT)

218 DZZDT = 0. _EB
219 RATE_CONSTANT = 0. _EB
220 Q.NR = 0. _EB
221 REACTION_LOOP: DO NR = 1, N_REACTIONS
222 RN => REACTION(NR)
223 CALL COMPUTE_RATE_CONSTANT(NR,RN%MODE, I.TS , Q.OUT, RATE_CONSTANT(NR) , ZZ.I , I , J, K)
224 IF (RATE_CONSTANT(NR) < ZERO_P) CYCLE REACTION_LOOP
225 Q.NR(NR) = RATE_CONSTANT(NR) * RN%HEAT_OF_COMBUSTION*RHO(I , J, K)
226 DZZDT = DZZDT + RN%NU * SPECIES_MIXTURE%MW/SPECIES_MIXTURE(RN%FUEL_SMIX_INDEX)%MW*RATE_CONSTANT(NR)
227 END DO REACTION_LOOP
228 IF (ALL(DZZDT < ZERO_P)) EXIT ODE_LOOP
229 ZZ.N = ZZ.I + DZZDT * DT.NEW
230
231 IF (ANY(ZZ.N < 0. _EB)) THEN
232 DO NS=0,N_TRACKED_SPECIES
233 IF (ZZ.N(NS) < 0. _EB .AND. ABS(DZZDT(NS))>ZERO_P)
234 DT.NEW = MIN(DT.NEW,−ZZ.I(NS)/DZZDT(NS))
235 ENDIF
236 ENDDO
237
238 Q.SUM = SUM(Q.NR)

211 ZZ.0 = MAX (0. _EB , ZZ.GET)
212 ZZ.I = ZZ.0
213 DT.ODE = DT/REAL(NODETS,EB)
214 DT.NEW = DT.ODE
215 DT.SUM = 0. _EB
216 I.TS = 1

ODE_LOOP: DO WHILE (DT.SUM < DT)

218 DZZDT = 0. _EB
219 RATE_CONSTANT = 0. _EB
220 Q.NR = 0. _EB
221 REACTION_LOOP: DO NR = 1, N_REACTIONS
222 RN => REACTION(NR)
223 CALL COMPUTE_RATE_CONSTANT(NR,RN%MODE, I.TS , Q.OUT, RATE_CONSTANT(NR) , ZZ.I , I , J, K)
224 IF (RATE_CONSTANT(NR) < ZERO_P) CYCLE REACTION_LOOP
225 Q.NR(NR) = RATE_CONSTANT(NR) * RN%HEAT_OF_COMBUSTION*RHO(I , J, K)
226 DZZDT = DZZDT + RN%NU * SPECIES_MIXTURE%MW/SPECIES_MIXTURE(RN%FUEL_SMIX_INDEX)%MW*RATE_CONSTANT(NR)
227 END DO REACTION_LOOP
228 IF (ALL(DZZDT < ZERO_P)) EXIT ODE_LOOP
229 ZZ.N = ZZ.I + DZZDT * DT.NEW
230
231 IF (ANY(ZZ.N < 0. _EB)) THEN
232 DO NS=0,N_TRACKED_SPECIES
233 IF (ZZ.N(NS) < 0. _EB .AND. ABS(DZZDT(NS))>ZERO_P)
234 DT.NEW = MIN(DT.NEW,−ZZ.I(NS)/DZZDT(NS))
235 ENDIF
236 ENDDO
237 Q.SUM = SUM(Q.NR)
APPENDIX B. IMPLEMENTED CODE

238 IF (Q\textsubscript{OUT} + Q\textsubscript{SUM} \cdot DT\textsubscript{NEW} > Q\textsubscript{UPPER} \cdot DT) THEN
239   DT\textsubscript{NEW} = MAX(0.\_EB, (Q\textsubscript{UPPER} \cdot DT - Q\textsubscript{OUT}))/Q\textsubscript{SUM}
240   Q\textsubscript{OUT} = Q\textsubscript{OUT} + Q\textsubscript{SUM} \cdot DT\textsubscript{NEW}
241   ZZ\_I = ZZ\_I + DZZDT \cdot DT\textsubscript{NEW}
242   EXIT ODE\_LOOP
243 ENDIF
244 Q\textsubscript{OUT} = Q\textsubscript{OUT} + Q\textsubscript{SUM} \cdot DT\textsubscript{NEW}
245 ZZ\_I = ZZ\_I + DZZDT \cdot DT\textsubscript{NEW}
246 DT\textsubscript{SUM} = DT\textsubscript{SUM} + DT\textsubscript{NEW}
247 IF (DT\textsubscript{NEW} < DT\textsubscript{ODE}) DT\textsubscript{NEW} = DT\textsubscript{ODE}
248 IF (DT\textsubscript{NEW} + DT\textsubscript{SUM} > DT) DT\textsubscript{NEW} = DT - DT\textsubscript{SUM}
249 L\_TS = L\_TS + 1
250 ENDDO ODE\_LOOP
251 ZZ\_GET = ZZ\_GET + ZZ\_I - ZZ\_0
252 Q\textsubscript{OUT} = Q\textsubscript{OUT} / DT
253 RETURN
254
255 END SUBROUTINE ODE\_EXPLICIT\_EULER
256
257 SUBROUTINE ODE\_RUNGE\_KUTTA\_2(I, J, K, ZZ\_GET, Q\textsubscript{OUT})
258 INTEGER, INTENT(IN) :: I, J, K
259 REAL(EB), INTENT(OUT) :: Q\textsubscript{OUT}
260 REAL(EB), INTENT(INOUT) :: ZZ\_GET (0:N\_TRACKED\_SPECIES)
261 REAL(EB) :: ZZ\_0 (0:N\_TRACKED\_SPECIES), ZZ\_I (0:N\_TRACKED\_SPECIES)
262 , ZZ\_N (0:N\_TRACKED\_SPECIES), &
263 DZZDT (0:N\_TRACKED\_SPECIES), DZZDT2 (0:
264 N\_TRACKED\_SPECIES), &
265 DT\_ODE, DT\_NEW, RATE\_CONSTANT (1:N\_REACTIONS), Q\_NR (1:
266 N\_REACTIONS), Q\_NR2 (1:N\_REACTIONS), Q\textsubscript{SUM}, DT\textsubscript{SUM}
267 INTEGER :: NR, L\_TS, NS
268 INTEGER, PARAMETER :: NODETS=20
269 TYPE(REACTION\_TYPE) , POINTER :: RN=>NULL()
270
271 Q\textsubscript{OUT} = 0.\_EB
272 ZZ\_0 = MAX(0.\_EB, ZZ\_GET)
273 ZZ\_I = ZZ\_0
274 DT\_ODE = DT/REAL(NODETS, EB)
275 DT\_NEW = DT\_ODE
276 DT\textsubscript{SUM} = 0.\_EB
APPENDIX B. IMPLEMENTED CODE

148

I_LS = 1

ODE_LOOP: DO WHILE (DT_SUM < DT)

DZZDT = 0. _EB

DZZDT2 = 0. _EB

Q_NR = 0. _EB

Q_NR2 = 0. _EB

RATE_CONSTANT = 0. _EB

REACTION_LOOP: DO NR = 1, N_REACTIONS

RN => REACTION(NR)

CALL COMPUTE_RATE_CONSTANT(NR, RN%MODE, I_LS , Q_OUT,
RATE_CONSTANT(NR) , ZZ_I , I, J, K)

IF (RATE_CONSTANT(NR) < ZERO_P) CYCLE REACTION_LOOP

Q_NR(NR) = RATE_CONSTANT(NR) * RN%HEAT_OF_COMBUSTION * RHO(I, J, K)

DZZDT = DZZDT + RN%NU * SPECIES_MIXTURE%M%MW / SPECIES_MIXTURE(RN%FUEL_SMIX_INDEX)%MW * RATE_CONSTANT(NR)

END DO REACTION_LOOP

IF (ALL(DZZDT < ZERO_P)) EXIT ODE_LOOP

ZZ_N = ZZ_I + DZZDT * DT_NEW

IF (ANY(ZZ_N < 0. _EB)) THEN

DO NS=0,N_TRACKED_SPECIES

IF (ZZ_N(NS) < 0. _EB AND ABS(DZZDT(NS))>ZERO_P)

DT_NEW = MIN(DT_NEW, -ZZ_I(NS)/DZZDT(NS))

ENDIF

ENDDO

ZZ_N = ZZ_I + DZZDT * DT_NEW

REACTION_LOOP2: DO NR = 1, N_REACTIONS

RN => REACTION(NR)

CALL COMPUTE_RATE_CONSTANT(NR, RN%MODE, I_LS , Q_OUT,
RATE_CONSTANT(NR) , ZZ_N , I , J_K)

IF (RATE_CONSTANT(NR) < ZERO_P) CYCLE REACTION_LOOP2

Q_NR2(NR) = RATE_CONSTANT(NR) * RN%HEAT_OF_COMBUSTION * RHO(I, J_K)

DZZDT2 = DZZDT2 + RN%NU * SPECIES_MIXTURE%M%MW / SPECIES_MIXTURE(RN%FUEL_SMIX_INDEX)%MW * RATE_CONSTANT(NR)

END DO REACTION_LOOP2

IF (ALL(DZZDT2 < ZERO_P)) EXIT ODELOOP

ZZ_N = ZZ_I + 0.5 _EB * (DZZDT + DZZDT2) * DT_NEW

EXIT
APPENDIX B. IMPLEMENTED CODE

149

312 IF (ANY(ZZ_N < 0. _EB )) THEN
313 DO NS=0,N_TRACKED_SPECIES
314 IF (ZZ_N(NS) < 0. _EB .AND. ABS(DZZDT(NS)+DZZDT2(NS)) >
315 ZERO_P) DT_NEW = MIN(DT_NEW, -2. _EB * ZZ_I(NS) / (DZZDT
316 (NS)+DZZDT2(NS) ) )
317 ENDDO
318 ENDIF
319 Q_SUM = SUM(0.5 _EB *(Q_NR+Q_NR2) )
320 IF (Q_OUT + Q_SUM*DT_NEW > Q_UPPER * DT) THEN
321 DT_NEW = MAX(0. _EB , (Q_UPPER * DT - Q_OUT) )/Q_SUM
322 Q_OUT = Q_OUT+Q_SUM*DT_NEW
323 ZZ_I = ZZ_I + 0.5 _EB *(DZZDT+DZZDT2) * DT_NEW
324 EXIT ODE_LOOP
325 ENDIF
326
327 ZZ_I = ZZ_I + 0.5 _EB *(DZZDT+DZZDT2) * DT_NEW
328 Q_OUT = Q_OUT+Q_SUM*DT_NEW
329 DT_SUM = DT_SUM + DT_NEW
330 IF (DT_NEW < DT_ODE) DT_NEW = DT_ODE
331 IF (DT_NEW + DT_SUM > DT) DT_NEW = DT - DT_SUM
332 ITS = ITS + 1
333 ENDDO ODE_LOOP
334 ZZ_GET = ZZ_GET + ZZ_I - ZZ_0
335 Q_OUT = Q_OUT / DT
336 RETURN
337 END SUBROUTINE ODE_RUNGE_KUTTA_2
338
339 SUBROUTINE ODE_RK2_RICHARDSON(I , J ,K ,ZZ_GET ,Q_OUT)
340 INTEGER,INTENT(IN) :: I , J , K
341 REAL,INTENT(INOUT) :: ZZ_GET(0:N_TRACKED_SPECIES)
342 REAL :: ZZ_0(0:N_TRACKED_SPECIES), DZZDT(0:N_TRACKED_SPECIES),
343 DZZDT2(0:N_TRACKED_SPECIES), RATE_CONSTANT(1:N_REACTIONS),
344 ERR_EST, TOL, INT_VECTOR(1:N_REACTIONS), ERR_TOL,
345 Q_NR_1(1:N_REACTIONS), Q_NR_2(1:N_REACTIONS),
APPENDIX B. IMPLEMENTED CODE

\begin{verbatim}
Q_NR2 (1:N_REACTIONS), Q_NR2 (1:N_REACTIONS),
Q_NR4 (1:N_REACTIONS), Q_SUM1 , Q_SUM2 , Q_SUM4 ,
A1 (0:N_TRACKED_SPECIES), A2 (0:N_TRACKED_SPECIES), A4
(0:N_TRACKED_SPECIES), DT_SUB, DT_SUB_NEW, DT_ITER
&
DT_A1, DT_A2, DT_A4, ZZ_STORE (0:N_TRACKED_SPECIES, 0:3)
, TV (0:2) , ZZ_DIFF (0:2) , Q1, Q2, Q4, Q_OUT2

INTEGER :: I_TS , NR, NS, NSS, ITER, TVI, RICH_ITER
INTEGER, PARAMETER :: NODETS = 20, SUB_DT1=1, SUB_DT2=2, SUB_DT4=4,
TV_ITER_MIN=5, Q_ITER_MAX=10, RICH_ITER_MAX=100

TYPE (REACTION_TYPE) , POINTER :: RN=>NULL()

Q_OUT = 0._EB
Q_OUT2 = 0._EB
RICH_ITER = 0
ITER = 0
I_TS = 1
DT_SUB = DT
DT_SUB_NEW = DT_SUB

! Setting up tolerance vector from inputs
DO NR = 1, N_REACTIONS
  RN => REACTION(NR)
  TOL_INT_VECTOR(NR) = RN%TOL_INT
ENDDO
ERR_TOL = MINVAL (ABS (TOL_INT_VECTOR))

INTEGRATION_LOOP : DO WHILE (DT_ITER < DT)
  ERR_EST = 10._EB*ERR_TOL
  RICH_EX_LOOP : DO WHILE (ERR_EST > ERR_TOL)
    DT_SUB = DT_SUB_NEW
    IF (DT_ITER + DT_SUB > DT) THEN
      DT_SUB = DT - DT_ITER
    ENDIF
    ZZ_0 = MAX (0._EB, ZZ_GET)
  Q1 = Q_OUT2
ENDIF

!-------------------------------
! Calculate A1 term
! Time step = DT_SUB
!-------------------------------

\end{verbatim}
APPENDIX B. IMPLEMENTED CODE

```fortran
ODD_LOOP1: DO NS = 1, SUB_DT1
  DZZDT = 0. _EB
  DZZDT2 = 0. _EB
  RATE_CONSTANT = 0. _EB

  REACTION_LOOP: DO NR = 1, N_REACTIONS
    RN => REACTION(NR)
    CALL COMPUTE_RATE_CONSTANT(NR, RN%MODE, I_TS, Q_OUT2, 
      RATE_CONSTANT(NR), ZZ_0, I, J, K)
    IF (RATE_CONSTANT(NR) <= 0.0 _EB) CYCLE
      REACTION_LOOP
    DZZDT = DZZDT + RN%NU*SPECIES_MIXTURE%M/W 
      SPECIES_MIXTURE(RN%FUEL_SMIX_INDEX)%M/W*
      RATE_CONSTANT(NR)
    Q_NR1(NR) = RATE_CONSTANT(NR)*RN% 
      HEAT_OF_COMBUSTION*RHO(I,J,K)
  END DO REACTION_LOOP

  IF (ALL(DZZDT < 0. _EB)) EXIT INTEGRATION_LOOP
  A1 = ZZ_0 + DZZDT*DT_SUB
  IF (ANY(A1 < 0. _EB)) THEN
    DO NSS=0,N_TRACKED_SPECIES
      IF (A1(NSS) < 0. _EB .AND. ABS(DZZDT(NSS))>ZERO_P ) THEN
        DT_SUB = MIN(DT_SUB,−ZZ_0(NSS)/DZZDT(NSS))
      END IF
    ENDDO
  END IF

  A1 = ZZ_0 + DZZDT*DT_SUB

REACTION_LOOP2: DO NR = 1, N_REACTIONS
  RN => REACTION(NR)
  CALL COMPUTE_RATE_CONSTANT(NR, RN%MODE, I_TS, Q_OUT2, 
    RATE_CONSTANT(NR), A1, I, J, K)
  IF (RATE_CONSTANT(NR) <= 0.0 _EB) CYCLE
    REACTION_LOOP2
  DZZDT2 = DZZDT2 + RN%NU*SPECIES_MIXTURE%M/W 
    SPECIES_MIXTURE(RN%FUEL_SMIX_INDEX)%M/W*
    RATE_CONSTANT(NR)
  Q_NR21(NR) = RATE_CONSTANT(NR)*RN% 
    HEAT_OF_COMBUSTION*RHO(I,J,K)
END DO REACTION_LOOP2

  IF (ALL(DZZDT2 < 0. _EB)) EXIT INTEGRATION_LOOP
  A1 = ZZ_0 + 0.5 _EB*(DZZDT+DZZDT2)*DT_SUB
```

---

388 ODD_LOOP1: DO NS = 1, SUB_DT1
389    DZZDT = 0. _EB
390    DZZDT2 = 0. _EB
391    RATE_CONSTANT = 0. _EB
392
393    REACTION_LOOP: DO NR = 1, N_REACTIONS
394     RN => REACTION(NR)
395     CALL COMPUTE_RATE_CONSTANT(NR, RN%MODE, I_TS, Q_OUT2,
396        RATE_CONSTANT(NR), ZZ_0, I, J, K)
397     IF (RATE_CONSTANT(NR) <= 0.0 _EB) CYCLE
398        REACTION_LOOP
399     DZZDT = DZZDT + RN%NU*SPECIES_MIXTURE%M/W 
400        SPECIES_MIXTURE(RN%FUEL_SMIX_INDEX)%M/W*
401        RATE_CONSTANT(NR)
402     Q_NR1(NR) = RATE_CONSTANT(NR)*RN% 
403        HEAT_OF_COMBUSTION*RHO(I,J,K)
404   END DO REACTION_LOOP
405
406   IF (ALL(DZZDT < 0. _EB)) EXIT INTEGRATION_LOOP
407   A1 = ZZ_0 + DZZDT*DT_SUB
408   IF (ANY(A1 < 0. _EB)) THEN
409     DO NSS=0,N_TRACKED_SPECIES
410       IF (A1(NSS) < 0. _EB .AND. ABS(DZZDT(NSS))>ZERO_P ) THEN
411         DT_SUB = MIN(DT_SUB,−ZZ_0(NSS)/DZZDT(NSS))
412       END IF
413     ENDDO
414   END IF
415   A1 = ZZ_0 + DZZDT*DT_SUB
416
417 REACTION_LOOP2: DO NR = 1, N_REACTIONS
418     RN => REACTION(NR)
419     CALL COMPUTE_RATE_CONSTANT(NR, RN%MODE, I_TS, Q_OUT2,
420        RATE_CONSTANT(NR), A1, I, J, K)
421     IF (RATE_CONSTANT(NR) <= 0.0 _EB) CYCLE
422        REACTION_LOOP2
423     DZZDT2 = DZZDT2 + RN%NU*SPECIES_MIXTURE%M/W 
424        SPECIES_MIXTURE(RN%FUEL_SMIX_INDEX)%M/W*
425        RATE_CONSTANT(NR)
426     Q_NR21(NR) = RATE_CONSTANT(NR)*RN% 
427        HEAT_OF_COMBUSTION*RHO(I,J,K)
428   END DO REACTION_LOOP2
429
430   IF (ALL(DZZDT2 < 0. _EB)) EXIT INTEGRATION_LOOP
431   A1 = ZZ_0 + 0.5 _EB*(DZZDT+DZZDT2)*DT_SUB
```
APPENDIX B. IMPLEMENTED CODE

152

IF (ANY(A1 < 0. _EB ) ) THEN
   DO NSS=0,N_TRACKED_SPECIES
      IF (A1(NSS) < 0. _EB .AND. ABS(DZZDT(NSS)+DZZDT2( NSS))>ZERO_P) THEN
         DT_SUB = MIN(DT_SUB,−2. _EB*ZZ_0(NSS)/(DZZDT( NSS)+DZZDT2(NSS)))
      ENDF
   ENDDO
   A1 = ZZ_0+0.5 _EB*(DZZDT+DZZDT2)*DT_SUB
ENDIF
147
148 Q_SUM1 = SUM(0.5 _EB*(Q_NR1+Q_NR21))
149 IF (Q1 + Q_SUM1*DT_SUB > Q_UPPER * DT) THEN
      DT_SUB_NEW = MAX(0.0 _EB,(Q_UPPER * DT−Q1)/Q_SUM1)
      Q1 = Q1+Q_SUM1*DT_SUB_NEW
      A1 = ZZ_0+0.5 _EB*(DZZDT+DZZDT2)*DT_SUB_NEW
      EXIT ODE_LOOP1
ENDIF
143
144 L_TS = L_TS + 1
ENDDO ODE_LOOP1
139
140 DT_A1 = DT_SUB
141
!--------------------------------------------------------
! Calculate A2 term
! Time step = DT_SUB/2
!--------------------------------------------------------
145 ZZ_0 = MAX(0. _EB ,ZZ_GET)
146 Q2 = Q_OUT2
ODE_LOOP2: DO NS = 1 , SUB_D2
148 DZZDT = 0. _EB
149 DZZDT2 = 0. _EB
150 RATE_CONSTANT = 0. _EB
REACTION_LOOP2: DO NR = 1 , N_REACTIONS
   RN => REACTION(NR)
   CALL COMPUTE_RATE_CONSTANT(NR,RN%MODE,L_TS,Q_OUT2,
      RATE_CONSTANT(NR) ,ZZ_0 ,I ,J ,K)
   IF (RATE_CONSTANT(NR) <= 0.0 _EB) CYCLE
   REACTION_LOOP2:
   DZZDT = DZZDT + RN%NU*SPECIES_MIXTURE%MW/
      SPECIES_MIXTURE(RN%FUEL_SMIX_INDEX)%MW*
      RATE_CONSTANT(NR)
APPENDIX B. IMPLEMENTED CODE

457  \[ Q_{NR,2}(NR) = \text{RATE\_CONSTANT}(NR) \times \text{RN\%} \times \text{HEAT\_OF\_COMBUSTION} \times \text{RHO}(I, J, K) \]

458  END DO REACTION\_LOOP2

459  A2 = \[ ZZ_0 + \text{DZZDT} \times (\text{DT\_SUB}/\text{REAL}(\text{SUB\_DT2}, \text{EB})) \]

460  IF (ANY(A2 < 0 \_EB)) THEN

461    DO NSS=0,N\_TRACKED\_SPECIES

462      IF (A2(NSS) < 0 \_EB AND ABS(DZZDT(NSS)) > ZERO_P)

463        DT\_SUB = MIN(DT\_SUB, -\[ ZZ_0(NSS)/DZZDT(NSS) \])

464      ENDIF

465  ENDDO

466  A2 = ZZ_0 + DZZDT \times (\text{DT\_SUB}/\text{REAL}(\text{SUB\_DT2}, \text{EB}))

467  ENDF

468  REACTION\_LOOP2: DO NR = 1, N\_REACTIONS

469   RN => REACTION(NR)

470   EXECUTE \text{COMPUTE\_RATE\_CONSTANT}(NR, RN\_MODE, I\_TS, Q\_OUT2, RATE\_CONSTANT(NR), A2, I, J, K)

471   IF (RATE\_CONSTANT(NR) <= 0.0 \_EB) CYCLE

472   \text{REACTION\_LOOP2}

473   DZZDT2 = DZZDT + RN\_NU \times \text{SPECIES\_MIXTURE}\_\text{MW} / \text{SPECIES\_MIXTURE}(RN\_\text{FUEL}, \text{SMIX_INDEX}) \_\text{MW} \times \text{RHO}(I, J, K)

474   \[ Q_{NR,2}(NR) = \text{RATE\_CONSTANT}(NR) \times \text{RN\%} \times \text{HEAT\_OF\_COMBUSTION} \times \text{RHO}(I, J, K) \]

475  END DO REACTION\_LOOP2

476  A2 = ZZ_0 + 0.5 \_EB \times (DZZDT + DZZDT2) \times (\text{DT\_SUB}/\text{REAL}(\text{SUB\_DT2}, \text{EB}))

477  IF (ANY(A2 < 0 \_EB)) THEN

478    DO NSS=0,N\_TRACKED\_SPECIES

479      IF (A2(NSS) < 0 \_EB AND ABS(DZZDT(NSS)) + DZZDT2(NSS) > ZERO_P)

480        DT\_SUB = MIN(DT\_SUB, -2 \_EB \times ZZ_0(NSS)/(DZZDT(NSS) + DZZDT2(NSS)))

481    ENDIF

482  ENDDO

483  A2 = ZZ_0 + 0.5 \_EB \times (DZZDT + DZZDT2) \times (\text{DT\_SUB}/\text{REAL}(\text{SUB\_DT2}, \text{EB}))

484  ENDF

485  Q\_SUM = SUM(0.5 \_EB \times (Q_{NR,2} + Q_{NR,2}^2))

486  IF (Q2 + Q\_SUM \times (\text{DT\_SUB}/\text{REAL}(\text{SUB\_DT2}, \text{EB})) > Q\_UPPER \times DT) THEN
APPENDIX B. IMPLEMENTED CODE

```plaintext
DT_SUB_NEW = MAX(0.0EB, (Q_UPPER * DT - Q2) / Q_SUM_2)
Q2 = Q2 + Q_SUM_2 * (DT_SUB_NEW)
A2 = ZZ_0 + 0.5*EB*(DZZDT + DZZDT2) * (DT_SUB_NEW)
EXIT ODE_LOOP2
ENDIF
Q2 = Q2 + Q_SUM_2 * (DT_SUB / REAL(SUB_DT2, EB))
I_TS = I_TS + 1
ZZ_0 = A2
ENDDO ODE_LOOP2
DT_A2 = DT_SUB
IF (DT_A2 < DT_A1) THEN
   DT_SUB_NEW = DT_A2
ENDIF
CYCLE RICH_EX_LOOP

!------------------------------------------------------------------------!
! Calculate A4 term
! Time step = DT_SUB/4
!------------------------------------------------------------------------!
ZZ_0 = MAX(0.0EB, ZZ_GET)
Q4 = Q_OUT2
ODE_LOOP4: DO NS = 1, SUB_DT4
   DZZDT = 0.0EB
   DZZDT2 = 0.0EB
   RATE_CONSTANT = 0.0EB
   REACTION_LOOP4: DO NR = 1, N_REACTIONS
      RN = REACTION(NR)
      CALL COMPUTE_RATE_CONSTANT(NR, RN%MODE, I_TS, Q_OUT2,
                                  RATE_CONSTANT(NR), ZZ_0, I, J, K)
      IF (RATE_CONSTANT(NR) <= 0.0EB) CYCLE
      REACTION_LOOP4
      DZZDT = DZZDT + RN%NU * SPECIES_MIXTURE%M / SPECIES_MIXTURE%(RN%FUEL_SMIX_INDEX)%MW
      RATE_CONSTANT(NR)
      Q_NR_i(NR) = RATE_CONSTANT(NR) * RN%HEAT_OF_COMBUSTION * RHO(I, J, K)
   END DO REACTION_LOOP4
   A4 = ZZ_0 + DZZDT * (DT_SUB / REAL(SUB_DT4, EB))
   IF (ANY(A4 < 0.0EB)) THEN
   DO NSS=0,N_TRACKED_SPECIES
      IF (A4(NSS) < 0.0EB .AND. ABS(DZZDT(NSS)) > ZERO_P ) THEN
         ! Do something
      ENDIF
   ENDDO
   EXIT ODE_LOOP4
```

DT_SUB = MIN(DT_SUB, -ZZ_0(NSS)/DZZDT(NSS))
ENDIF
ENDDO
A4 = ZZ_0 + DZZDT*(DT_SUB/REAL(SUB_DT4, EB))
ENDIF

REACTIONLOOP2: DO NR = 1, NREACTIONS
RN => REACTION(NR)
CALL COMPUTE_RATE_CONSTANT(NR, RN%MODE, I_T, Q_OUT2,
RATE_CONSTANT(NR), A4, I, J, K)
IF (RATE_CONSTANT(NR) <= 0.0EB) CYCLE
REACTION_LOOP2
DZZDT2 = DZZDT + RN%NU*SPECIES_MIXTURE%MW/
SPECIES_MIXTURE(RN%FUEL_SMIX_INDEX)%MW*
RATE_CONSTANT(NR)
Q_NR24(NR) = RATE_CONSTANT(NR)*RN%
HEAT_OF_COMBUSTION*RHO(I, J, K)
END DO REACTION_LOOP2
A4 = ZZ_0 + 0.5EB*(DZZDT+DZZDT2)*(DT_SUB/REAL(SUB_DT4, EB))
IF (ANY(A4 < 0.0EB)) THEN
DO NSS=0, NTRACKED_SPECIES
IF (A4(NSS) < 0.0EB AND ABS(DZZDT(NSS)+DZZDT2(NSS)) > ZERO_P) THEN
DT_SUB = MIN(DT_SUB, -2.0EB*ZZ_0(NSS)/(DZZDT(NSS)+DZZDT2(NSS)))
ENDIF
ENDDO
A4 = ZZ_0 + 0.5EB*(DZZDT+DZZDT2)*(DT_SUB/REAL(SUB_DT4, EB))
ENDIF

Q_SUM4 = SUM(0.5EB*(Q_NR4+Q_NR24))
IF (ABS(Q4+Q_SUM4*(DT_SUB/REAL(SUB_DT4, EB))) > Q_UPPER+DT) THEN
DT_SUB_NEW = MAX(0.0EB, (Q_UPPER+DT-Q4)/Q_SUM4)
Q4 = Q4+Q_SUM4*(DT_SUB_NEW)
A4 = ZZ_0 + 0.5EB*(DZZDT+DZZDT2)*(DT_SUB_NEW)
IF (ITER >= Q_ITER_MAX) THEN
Q_OUT = Q4/DT
ZZ_GET = A4
EXIT INTEGRATION_LOOP
ENDIF
APPENDIX B. IMPLEMENTED CODE

558     EXIT ODE_LOOP4
559     ENDIF
560     Q4 = Q4+Q\_SUM\_A\_4*(DT\_SUB/REAL\(\text{SUB}\_DT4,EB\))
561     I\_TS = I\_TS + 1
562     ZZ\_0 = A4
563     ENDDO ODE_LOOP4
564     DT\_A4 = DT\_SUB
565     IF \(\text{DT\_A4 < DT\_A2}\) THEN
566         DT\_SUB\_NEW = DT\_A4
567     CYCLE RICH\_EX\_LOOP
568     ENDIF
569
570     ! Species Error Analysis
571     ERR\_EST = MAXVAL(ABS\((4.\_EB*A4-A2) - (4.\_EB*A2-A1)))/45.\_EB ! Estimate Error
572     IF \(\text{ERR\_EST <= 0.0\_EB}\) THEN
573         DT\_SUB\_NEW = DT
574     ELSE
575         DT\_SUB\_NEW = DT\_SUB*(\text{ERR\_TOL}/(\text{ERR\_EST}))*^(0.25\_EB) ! Determine New Time Step
576     ENDIF
577
578     RICH\_ITER = RICH\_ITER+1
579     IF \(\text{RICH\_ITER >= RICH\_ITER\_MAX}\) EXIT RICH\_EX\_LOOP
580     ENDDO RICH\_EX\_LOOP
581
582     DT\_ITER = DT\_ITER + DT\_SUB
583     ITER = ITER + 1
584     MAX\_CHEM\_SUBIT = MAX(MAX\_CHEM\_SUBIT,ITER)
585     ZZ\_GET = (4.\_EB*A4-A2)/3.\_EB
586     Q\_OUT = (4.\_EB*Q4-A2)/3.\_EB/DT
587     Q\_OUT2 = (4.\_EB*Q4-A2)/3.\_EB
588
589     ! Total Variation Scheme
590     DO NS = 0,N\_TRACKED\_SPECIES
591         DO TVI = 0,2
592             ZZ\_STORE(NS,TVI)=ZZ\_STORE(NS,TVI+1)
593         ENDDO
594         ZZ\_STORE(NS,3) = (4.\_EB*A4(NS)-A2(NS))/3.\_EB
595     ENDDO
596
597     REACTION\_LOOP\_TV:  DO NR = 1, N\_REACTIONS
598         RN => REACTION(NR)
APPENDIX B. IMPLEMENTED CODE

IF (.NOT. RN%REVERSIBLE) CYCLE REACTION_LOOP_TV
DO TVI = 0,2
TV(TVI) = ABS(ZZ_STORE(RN%FUEL_SMIX_INDEX,TVI+1) -
ZZ_STORE(RN%FUEL_SMIX_INDEX,TVI))
ZZ_DIFF(TVI) = ZZ_STORE(RN%FUEL_SMIX_INDEX,TVI+1) -
ZZ_STORE(RN%FUEL_SMIX_INDEX,TVI)
ENDDO
IF (SUM(TV) > 0.0 EB .AND. SUM(TV) >= ABS(2.5 EB + SUM(
ZZ_DIFF)) .AND. ITER >= TV_ITER_MIN) EXIT
INTEGRATION_LOOP
ENDDO REACTION_LOOP_TV
ENDDO INTEGRATION_LOOP
RETURN
END SUBROUTINE ODE_RK2_RICHARDSON
RECURSIVE SUBROUTINE COMPUTE_RATE_CONSTANT(NR,MODE, I_TS ,Q_IN ,
RATE_CONSTANT, ZZ_GET , I ,J,K)
USE PHYSICAL_FUNCTIONS , ONLY : GET_MASS_FRACTION_ALL ,
GET_MASS_FRACTION, GET_VISCOSITY
REAL(EB) , INTENT(IN) :: ZZ_GET (0:N_TRACKED_SPECIES) , Q_IN
INTEGER, INTENT(IN) :: NR, L_TS,MODE, I ,J,K
REAL(EB) , INTENT(INOUT) :: RATE_CONSTANT
REAL(EB) :: YY_PRIMITIVE (1:N_SPECIES) , Y_F_MIN=1.E-15 , ZZ_MIN
=1.E-10 , YY_F_LIM , ZZ_REACTANT , ZZ_PRODUCT , &
TAU_D,TAU_U , DELTA , RATE_CONSTANT_ED ,
RATE_CONSTANT_FR , GAMMA , LAMBDA , NU , Y_FUEL , Y_O2 ,
Y_PRODUCT , S , &
CHI_1 , CHI_2 , CHI_3 , CHI , C_LES !ADDED
INTEGER :: NS
TYPE(REACTION_TYPE) , POINTER :: RN=>NULL(
RN => REACTION(NR)
SELECT CASE (MODE)
CASE(MIXED)
IF ( Q_IN > 0.0 EB .AND. RN%THRESHOLDES_TEMP >= TMP(I ,J,K) ) THEN
CALL COMPUTE_RATE_CONSTANT(NR, EDDY_DISSIPATION, I_TS ,
Q_IN , RATE_CONSTANT, ZZ_GET , I ,J,K)
ELSE
CALL COMPUTE_RATE_CONSTANT(NR, FINITE_RATE, I_TS, Q_IN, RATE_CONSTANT, ZZ_GET, I, J, K)
ENDIF
CASE(EDDY_DISSIPATION)
  IF_SUPPRESSION: IF (SUPPRESSION) THEN
    ! Evaluate empirical extinction criteria
    IF (I_TS==1) THEN
      IF (EXTINCTION(I, J, K, ZZ_GET)) THEN
        RATE_CONSTANT = 0. _EB
        RETURN
      ENDIF
      ! ELSE
      ! IF (RATE_CONSTANT <= ZERO_P) RETURN
      ENDF
    IF_SUPPRESSION
    FIXED_TIME: IF (FIXED_MIX_TIME>0. _EB) THEN
      MIX_TIME(I, J, K)=FIXED_MIX_TIME
    ELSE FIXED_TIME
      IF (TWO_D) THEN
        DELTA = MAX(DX(I), DZ(K))
      ELSE
        DELTA = MAX(DX(I), DY(J), DZ(K))
      ENDF
    !LES_IF: IF (LES) THEN
    !TAU_D = D_Z(MIN(4.999, NINT(TMP(I, J, K))), RN% FUEL_SMIX_INDEX)
    !TAU_D = DELTA**2/TAU_D ! diffusive time scale
    !IF (TURB_MODEL==DEARDORFF) THEN
    !TAU_U = 0.1 _EB*SC*RHO(I, J, K)*DELTAS**2/MU(I, J ; K) ! turbulent mixing time scale
    !ELSE
    !TAU_U = DELTA/SQRT(2. _EB*KSGS(I, J, K)+1.E-10 _EB) ! advective time scale
    !ENDIF
    !TAU_G = SQRT(2. _EB*DELTA/(GRAV+1.E-10 _EB)) ! acceleration time scale
APPENDIX B. IMPLEMENTED CODE

\[ \text{MIX\_TIME}(I, J, K) = \max(\text{TAU\_CHEM}, \min(\text{TAU\_D}, \text{TAU\_U}, \text{TAU\_G}, \text{TAU\_FLAME})) \] ! Eq. 7, McDermott, McGrattan, Floyd

! ELSE LES_IF

!TAU\_D = D\_Z(\min(4.999, \text{NINT}(%\text{TMP}(I, J, K))), \text{RN\% FUEL\_SMIX\_INDEX})

!TAU\_D = \text{DELTA}\_2^{*2}/\text{TAU\_D}

!MIX\_TIME(I, J, K) = \text{TAU\_D}

!ENDIF LES_IF

MIX\_TIME(I, J, K) = 1.\_EB/\text{ABS}(\text{STRAIN\_RATE}(I, J, K)) ! ADDED

ENDIF FIXED\_TIME

YY\_F\_LIM = 1.E15\_EB

IF (\text{N\_REACTIONS} > 1) THEN

DO NS=0,\text{N\_TRACKED\_SPECIES}

IF (\text{RN\%NU}(NS) < -ZERO\_P) THEN

IF (\text{ZZ\_GET}(NS) < \text{ZZ\_MIN}) THEN

\text{RATE\_CONSTANT} = 0.\_EB

RETURN

ENDIF

YY\_F\_LIM = \min(YY\_F\_LIM, &

\text{ZZ\_GET}(NS) * \text{SPECIES\_MIXTURE}(\text{RN\% FUEL\_SMIX\_INDEX})\%\text{MW}/(\text{ABS}(\text{RN \%NU}(NS)) * \text{SPECIES\_MIXTURE}(NS)\%\text{MW}))

ENDIF

ENDDO

ELSE

\text{ZZ\_REACTANT} = 0.\_EB

\text{ZZ\_PRODUCT} = 0.\_EB

DO NS=0,\text{N\_TRACKED\_SPECIES}

IF (\text{RN\%NU}(NS) < -ZERO\_P) THEN

IF (\text{ZZ\_GET}(NS) < \text{ZZ\_MIN}) THEN

\text{RATE\_CONSTANT} = 0.\_EB

RETURN

ENDIF

\text{ZZ\_REACTANT} = \text{ZZ\_REACTANT} - \text{RN\%NU}(NS) * \text{SPECIES\_MIXTURE}(NS)\%\text{MW}

YY\_F\_LIM = \min(YY\_F\_LIM, &
APPENDIX B. IMPLEMENTED CODE

704 \[ \text{ZZ}_\text{PRODUCT} = \text{ZZ}_\text{PRODUCT} + \text{ZZ}_\text{GET}(\text{NS}) \]

705 \[ \text{ELSEIF}(\text{RN}\%\text{NU}(\text{NS}) > \text{ZERO}_P) \] THEN

706 \[ \text{ZZ}_\text{PRODUCT} = \text{ZZ}_\text{PRODUCT} + \text{ZZ}_\text{GET}(\text{NS}) \]

707 ENDIF

708 ENDDO

709 \[ \text{ZZ}_\text{PRODUCT} = \text{BETA}_\text{EDC} \times \text{MAX}(\text{ZZ}_\text{PRODUCT} \times \text{SPECIES}_\text{MIXTURE}(\text{RN}\%\text{FUEL}_\text{SMIX_INDEX})/\text{MW}, \] \[ \text{ZZ}_\text{REACTANT}, \text{Y}_P, \text{MIN}_\text{EDC}) \]

710 \[ \text{YY}_F, \text{LIM} = \text{MIN}(\text{YY}_F, \text{LIM}, \text{ZZ}_\text{PRODUCT}) \]

711 ENDIF

712 \[ \text{YY}_F, \text{LIM} = \text{MAX}(\text{YY}_F, \text{LIM}, \text{Y}_F, \text{MIN}) \]

713 ! \text{RATE}_\text{CONSTANT} = \text{YY}_F, \text{LIM}/\text{MIX}_\text{TIME}(I, J, K)

714

715 ! The Eddy Dissipation Concept (EDC) Combustion Model (by Hjertager and Magnussen) for LES proposed by Balram et al.

716 \[ \text{NU} = \text{MU}(I, J, K)/\text{RHO}(I, J, K) \]

717 \[ \text{C}_\text{LES} = 0.10_\text{EB} \]

718 ! \text{IF}(\text{SELECT}_\text{TURB} == \text{DEARDORFF}) \] THEN

719 ! \[ \text{C}_\text{LES} = \]

720 ! \text{ELSEIF}(\text{SELECT}_\text{TURB} == \text{DYNSMAG}) \] THEN

721 ! \[ \text{C}_\text{LES} = \]

722 ! \text{ELSEIF}(\text{SELECT}_\text{TURB} == \text{VREMAN}) \] THEN

723 ! \[ \text{C}_\text{LES} = \]

724 ! \text{ELSE}

725 ! \[ \text{WRITE}(*,*) \] 'The chosen turbulence model is not 

726 ! supported by the combustion model'

727 ! \text{END}

728 \[ \text{GAMMA}_\text{LAMBDA} = \text{C}_\text{LES} \times (\text{NU}/\text{NU}_\text{EDDY}(I, J, K))^{0.25}_\text{EB} \]

729 ! \text{IF}(\text{GAMMA}_\text{LAMBDA} > 1.0_\text{EB}) \text{THEN}

730 \[ \text{GAMMA}_\text{LAMBDA} = 1.0_\text{EB} \]

731 \text{END IF}

732

733 \text{CALL GET_MASS_FRACTION(\text{ZZ}_\text{GET}, \text{FUEL_INDEX}, \text{Y}_\text{FUEL}) ! ADDED}

734 \text{CALL GET_MASS_FRACTION(\text{ZZ}_\text{GET}, \text{O2_INDEX}, \text{Y}_\text{O2}) !ADDED}

735 \[ \text{Y}_\text{PRODUCT} = 1.0_\text{EB} - (\text{Y}_\text{FUEL} + \text{Y}_\text{O2}) \]
\[ S = \text{SPECIES(FUEL\_INDEX)}\%\text{MW}/(\text{RN}\%\text{NU}_2\text{O}_2*\text{SPECIES(O2\_INDEX)}\%\text{MW}) \]

\[ Y_{O2} = Y_{O2}/S \]

\[ Y_{PRODUCT} = Y_{PRODUCT}/(1.\_EB + S) \]

\[ CHI_1 = ((Y_{F\_LIM} + Y_{PRODUCT})**2)/(Y_{FUEL} + Y_{PRODUCT})*(Y_{O2} + Y_{PRODUCT}) \]

\[ CHI_2 = \text{MIN}(Y_{PRODUCT}/(\text{GAMMA}\_\text{LAMBD}\_2 \text{**2}), 1.\_EB) \]

\[ CHI_3 = \text{MIN}((\text{GAMMA}\_\text{LAMBD}2 \text{**2})/(YY_{F\_LIM}), 1.\_EB) \]

\[ CHI = CHI_1*CHI_2*CHI_3 \]

\[ \text{RATE\_CONSTANT} = YY_{F\_LIM}*CHI/(\text{MIX\_TIME}(I, J, K)*(1.\_EB - CHI*GAMMA\_LAMBD_2**2)) \]

\[ \text{RATE\_CONSTANT} = YY_{F\_LIM}*GAMMA\_LAMBD_2**2/((\text{MIX\_TIME}(I, J, K)*(1.\_EB - CHI*GAMMA\_LAMBD_2**2)) \]

\[ \text{CASE(FINITE\_RATE)} \]

\[ \text{RATE\_CONSTANT} = 0.\_EB \]

\[ \text{CALL GET\_MASS\_FRACTION\_ALL(ZZ\_GET, YY\_PRIMITIVE)} \]

\[ \text{RATE\_CONSTANT} = \text{RN}\%\text{A}\_\text{RHO}(I, J, K) * \text{RN}\%\text{RHO}\_\text{EXPONENT}*\text{EXP}(-\text{RN}\%\text{E}/(\text{R0}\_\text{TMP}(I, J, K)))*\text{TMP}(I, J, K)*\text{RN}\%\text{N\_T} \]

\[ \text{IF (ALL(RN}\%\text{N}\_S<\text{−998.}_\text{EB})} \] \THEN

\[ \text{DO NS=0, }N\_\text{TRACKED\_SPECIES} \]

\[ \text{IF (RN}\%\text{N}\_S(NS)<0.\_EB .AND. ZZ\_GET(NS)<ZZ\_MIN) } \THEN

\[ \text{RATE\_CONSTANT} = 0.\_EB \]

\[ \text{RETURN} \]

\[ \text{ENDIF} \]

\[ \text{ENDDO} \]

\[ \text{ELSE} \]

\[ \text{DO NS=1, }N\_\text{SPECIES} \]

\[ \text{IF (ABS(RN}\%\text{N}\_S(NS)) \text{<= ZERO\_P }) \text{ CYCLE} \]

\[ \text{IF (RN}\%\text{N}\_S(NS) \text{>= −998.}_\text{EB}) } \THEN

\[ \text{IF (YY\_PRIMITIVE(NS)<ZZ\_MIN) } \THEN

\[ \text{RATE\_CONSTANT} = 0.\_EB \]

\[ \text{ELSE} \]

\[ \text{RATE\_CONSTANT} = YY\_PRIMITIVE(NS)**(RN}\%\text{N}\_S(NS) \]

\[ \text{ENDIF} \]

\[ \text{ENDIF} \]

\[ \text{ENDDO} \]

\[ \text{ENDIF} \]
APPENDIX B. IMPLEMENTED CODE

772 CASE(EDDY DISSIPATION_CONCEPT)
773 CALL COMPUTE_RATE_CONSTANT(NR, EDDY DISSIPATION, I_TS, Q_IN,
774       RATE_CONSTANT, ZZ_GET, I, J, K)
775 RATE_CONSTANT_ED=RATE_CONSTANT
776 CALL COMPUTE_RATE_CONSTANT(NR, FINITE_RATE, I_TS, Q_IN,
777       RATE_CONSTANT, ZZ_GET, I, J, K)
778 RATE_CONSTANT_FR=RATE_CONSTANT
779 IF (ABS(RATE_CONSTANT_ED) < ZERO_P .AND. ABS(RATE_CONSTANT_FR) < ZERO_P) THEN
780 ELSE
781 RATE_CONSTANT = (RATE_CONSTANT_ED+RATE_CONSTANT_FR) /
782       (RATE_CONSTANT_ED+RATE_CONSTANT_FR)
783 ENDIF
784 RETURN
785 END SELECT
786 CONTAINS
787
788 LOGICAL FUNCTION EXTINCTION(I, J, K, ZZ_IN)
789 !This routine determines if local extinction occurs for a
790 !mixing controlled reaction.
791 !This is determined as follows:
792 !1) Determine how much fuel can burn (DZ_FUEL) by finding the
793    !limiting reactant and expressing it in terms of fuel mass
794 !2) Remove that amount of fuel form the local mixture,  
795    !everything else is "air"
796 !3) Search to find the minimum reactant other than fuel.
797 !4) Using the reaction stoichiometry, determine how much "air"  
798    !(DZ_AIR) is needed to burn the fuel.
799 !5) Check to see if the heat released from burning DZ_FUEL can  
800    !raise the current temperature of DZ_FUEL and DZ_AIR
801 ! above the critical flame temp.
802 USE PHYSICAL_FUNCTIONS, ONLY: GET_AVERAGE_SPECIFIC_HEAT
803 REAL(EB), INTENT(IN) :: ZZ_IN (0:N_TRACKED_SPECIES)
804 REAL(EB) :: DZ_AIR, DZ_FUEL, CPBAR_F_0, CPBAR_F_N, CPBAR_G_0, 
805       CPBAR_G_N, ZZ_GET (0:N_TRACKED_SPECIES)
806 INTEGER, INTENT(IN) :: I, J, K
APPENDIX B. IMPLEMENTED CODE

```
INTEGER :: NS

EXTINCTION = .FALSE.
IF (TMP(I,J,K) < RN%AUTO IGNITION_TEMPERATURE) THEN
  EXTINCTION = .TRUE.
ELSE
  DZFUEL = 1.0EB
  DZAIR = 0.0EB
  ! Search reactants to find limiting reactant and express it
  ! as fuel mass. This is the amount of fuel
  DO NS = 0,N,TRACKED_SPECIES
    IF (RN%NU(NS)<-ZERO_P) &
      DZFUEL = MIN(DZFUEL,-ZZ_IN(NS)*SPECIES_MIXTURE(RN%
        FUEL_SMIX_INDEX)%MW/(RN%NU(NS)*SPECIES_MIXTURE(NS)%
        MW))
  ENDDO
  ZZ_GET = 0.0EB
  ZZ_GET(RN%FUEL_SMIX_INDEX) = 1.0EB
  CALL GET_AVERAGE_SPECIFIC_HEAT(ZZ_GET, CPBAR_F_0,TMP(I,J,K))
  CALL GET_AVERAGE_SPECIFIC_HEAT(ZZ_GET, CPBAR_F_N,RN%
    CRIT_FLAME_TMP)
  ZZ_GET = ZZ_GET/SUM(ZZ_GET)
  ZZ_GET(RN%FUEL_SMIX_INDEX) = ZZ_GET(RN%FUEL_SMIX_INDEX) -
    DZFUEL
  ZZ_GET = ZZ_GET/SUM(ZZ_GET)
  ! Get the specific heat for the "air"
  CALL GET_AVERAGE_SPECIFIC_HEAT(ZZ_GET, CPBAR_G_0,TMP(I,J,K))
  CALL GET_AVERAGE_SPECIFIC_HEAT(ZZ_GET, CPBAR_G_N,RN%
    CRIT_FLAME_TMP)
  ! Loop over non-fuel reactants and find the minimum.
  ! Determine how much "air" is needed to provide the limiting
  ! reactant
  DO NS = 0,N,TRACKED_SPECIES
    IF (RN%NU(NS)<-ZERO_P .AND. NS/=RN%FUEL_SMIX_INDEX) &
      DZAIR = MAX(DZAIR,-DZFUEL*RN%NU(NS)*
        SPECIES_MIXTURE(NS)%MW/SPECIES_MIXTURE(RN%
        FUEL_SMIX_INDEX)%MW/ZZ_GET(NS))
```
ENDDO

! See if enough energy is released to raise the fuel and
required "air" temperatures above the critical flame temp

IF ((DZ_FUEL*CPBAR_F_0 + DZ_AIR*CPBAR_G_0)*TMP(I,J,K) +
    DZ_FUEL*RN%HEAT_OF_COMBUSTION < &
    (DZ_FUEL*CPBAR_F_N + DZ_AIR*CPBAR_G_N)*RN%
    CRIT_FLAME_TMP) EXTINCTION = .TRUE.

ENDIF

END FUNCTION EXTINCTION

REAL(EB) FUNCTION KSGS(I,J,K)
INTEGER, INTENT(IN) :: I,J,K
REAL(EB) :: EPSK

! ke dissipation rate, assumes production=dissipation
EPSK = MU(I,J,K)*STRAIN_RATE(I,J,K)**2/RHO(I,J,K)

KSGS = 2.25*EPSK*(DELTA/P1)**TWTH ! estimate of subgrid ke, from Kolmogorov spectrum

END FUNCTION KSGS

END SUBROUTINE COMPUTE_RATE_CONSTANT

SUBROUTINE GET_REV_fire(MODULE_REV,MODULE_DATE)
INTEGER, INTENT(INOUT) :: MODULE_REV
CHARACTER(255), INTENT(INOUT) :: MODULE_DATE

WRITE(MODULE_DATE, '(A)') firerev(INDEX(firerev,':') + 2:LEN_TRIM(firerev) - 2)
READ (MODULE_DATE, '(I5)') MODULE_REV
WRITE(MODULE_DATE, '(A)') firedate

END SUBROUTINE GET_REV_fire

END MODULE FIRE
APPENDIX B. IMPLEMENTED CODE

B.2 velo.f90

Modified subroutine in velo.f90:

```fortran
SUBROUTINE COMPUTE_VISCOITY(NM)

USE PHYSICAL_FUNCTIONS, ONLY: GET_VISCOSITY
USE TURBULENCE, ONLY: VARDEN_DYNSMAG, TEST_FILTER, EX2G3D

INTEGER, INTENT(IN) :: NM
REAL(EB) :: ZZ, GET(0:N_TRACKED_SPECIES), DELTA, NU_G, GRAD_RHO(3), U2, V2, W2, AA, AIJ(3,3), BB, BIJ(3,3)

INTEGER :: I, J, K, IIG, JJG, KKG, IJ, JJ, KK, IW, TURB_MODEL_TMP, IOR
REAL(EB), POINTER, DIMENSION(:,:,:,:) :: RHOP => NULL(), UP => NULL(), VP => NULL(), WP => NULL(), UHAT => NULL(), VHAT => NULL(), WHAT => NULL(), UU => NULL(), VV => NULL(), WW => NULL(), ZZP => NULL()

TYPE(WALL_TYPE), POINTER :: WC => NULL()

CALL POINT_TO_MESH(NM)

IF (PREDICTOR) THEN
    RHOP => RHOS
    UU => US
    VV => VS
    WW => WS
    IF (N_TRACKED_SPECIES > 0) ZZP => ZZ
ELSE
    RHOP => RHOS
    UU => US
    VV => VS
    WW => WS
    IF (N_TRACKED_SPECIES > 0 .AND. NOT_EVACUATIONONLY(NM))
        ZZP => ZZS
ENDIF

! Compute viscosity for DNS using primitive species/mixture fraction
```

APPENDIX B. IMPLEMENTED CODE

!$OMP PARALLEL DEFAULT(NONE) &
!$OMP SHARED(N_TRACKED_SPECIES, EVACUATION_ONLY, KBAR, JBAR, IBAR, SOLID, CELL_INDEX, ZZP, MU, TMP, &
!$OMP LES, NM, CS_MAGORINSKY, TWO_D, DX, DY, DZ, RDX, RDY, RDZ, UU, VV, WW, RHOP, CSD2, &
!$OMP NEXTERNAL_WALL CELLS, NINTERNAL_WALL CELLS, KRES, &
!$OMP IBP1, JBP1, KBP1, TURB_MODEL_TMP, TURB_MODEL, PREDICTOR, STRAIN_RATE, UP, VP, WP, WORK1, WORK2, WORK3, WC, WALL, U_GHOST, V_GHOST, &
!$OMP W_GHOST, UP_HAT, VP_HAT, WP_HAT, WORK4, WORK5, WORK6, DELTA, KSGS, NU_EDDY, C_DEARDORFF, DUDX, DVDY, DWDZ, DUDY, DUDZ, DVDX, DWDY, &
!$OMP Dwdx, Dwdy, II, JJ, KK, A_IJ, AA, B_IJ, BB, C_VREMAN, GRAV_VISC, GRAD_RHO, NU_G, C_G, GVEC, IOR, MU_DNS) &
!$OMP PRIVATE(ZZ_GET)

IF (N_TRACKED_SPECIES>0 .AND. EVACUATION_ONLY(NM)) ZZ_GET (1:N_TRACKED_SPECIES) = 0. _EB

!$OMP DO COLLAPSE(3) SCHEDULE(STATIC) &
!$OMP PRIVATE(K, J, I)

DO K=1,KBAR
  DO J=1,JBAR
    IF (SOLID(CELL_INDEX(I,J,K))) CYCLE
    IF (N_TRACKED_SPECIES>0 .AND. .NOT. EVACUATION_ONLY(NM))
      ZZ_GET (1:N_TRACKED_SPECIES) = ZZP(I, J, K, 1:N_TRACKED_SPECIES)
    CALL GET_VISCOSITY(ZZ_GET, MU(I, J, K), TMP(I, J, K))
  ENDDO
ENDDO
!$OMP END DO

TURB_MODEL_TMP = TURB_MODEL
IF (EVACUATION_ONLY(NM)) TURB_MODEL_TMP = CONSMAG

SELECT_TURB: SELECT CASE (TURB_MODEL_TMP)

CASE (CONSMAG, DYN_SMAG) SELECT_TURB ! Smagorinsky (1963) eddy viscosity
APPENDIX B. IMPLEMENTED CODE

CALL COMPUTE_STRAIN_RATE(NM)

IF (PREDICTOR .AND. TURB_MODEL_TMP==DYNSMAG) CALL
  YARDEN_DYNSMAG(NM) ! dynamic procedure, Moin et al. (1991)

!$OMP DO COLLAPSE(3) SCHEDULE(STATIC) &
!$OMP PRIVATE(K, J, I)
DO K=1,KBAR
  DO J=1,JBAR
    DO I=1,IBAR
      IF (SOLID(CELL_INDEX(I,J,K))) CYCLE
      MU(I,J,K) = MU(I,J,K) + RHOP(I,J,K) * CSD2(I,J,K) *
        STRAIN_RATE(I,J,K)
      !NU_EDDY = MU(I,J,K)/RHO(I,J,K)
      NU_EDDY(I,J,K) = MU(I,J,K)/RHO(I,J,K)
    ENDDO
  ENDDO
ENDDO

CASE (DEARDORFF) SELECT_TURB ! Deardorff (1980) eddy viscosity model (current default)

! Velocities relative to the p-cell center

!! CALL COMPUTE_STRAIN_RATE(NM) ! til forbrenningsmodell

UP => WORK1
VP => WORK2
WP => WORK3
UP=0._EB
VP=0._EB
WP=0._EB

DO K=1,KBAR
  DO J=1,JBAR
    DO I=1,IBAR
      UP(I,J,K) = 0.5_EEB*(UU(I,J,K) + UU(I-1,J,K))
      VP(I,J,K) = 0.5_EEB*(VV(I,J,K) + VV(I,J-1,K))
      WP(I,J,K) = 0.5_EEB*(WW(I,J,K) + WW(I,J,K-1))
    ENDDO
  ENDDO
ENDDO
ENDDO

DO IW=1,N_EXTERNAL_WALL CELLS
    WC => WALL(IW)
    CALL EX2G3D(UP, -1.E10, 1.E10)
    CALL EX2G3D(VP, -1.E10, 1.E10)
    CALL EX2G3D(WP, -1.E10, 1.E10)

    DO IW=1,N_EXTERNAL_WALL CELLS
        WC => WALL(IW)
        IF (WC%BOUNDARY_TYPE/=INTERPOLATED_BOUNDARY) CYCLE
        JJ = WC%JJ
        KK = WC%KK
        UP(I,J,K) = U_GHOST(IW)
        VP(I,J,K) = V_GHOST(IW)
        WP(I,J,K) = W_GHOST(IW)
    ENDDO

UP_HAT => WORK4
VP_HAT => WORK5
WP_HAT => WORK6
UP_HAT = 0.EB
VP_HAT = 0.EB
WP_HAT = 0.EB

CALL TEST_FILTER(UP_HAT, UP, -1.E10, 1.E10)
CALL TEST_FILTER(VP_HAT, VP, -1.E10, 1.E10)
CALL TEST_FILTER(WP_HAT, WP, -1.E10, 1.E10)

DO K=1,KBAR
    DO J=1,JBAR
        DO I=1,IBAR
            IF (SOLID(CELL_INDEX(I,J,K))) CYCLE
            IF (TWO_D) THEN
                DELTA = MAX(DX(I),DZ(K))
            ELSE
                DELTA = MAX(DX(I),DY(J),DZ(K))
            ENDIF
            KSGS = 0.5*EB*( (UP(I,J,K)-UP_HAT(I,J,K))**2 + (VP(I,J,K)-VP_HAT(I,J,K))**2 + (WP(I,J,K)-WP_HAT(I,J,K))**2 )
APPENDIX B. IMPLEMENTED CODE

\[
WP\_HAT(I,J,K) \ast \ast 2
\]

!KSGS(I,J,K) = 0.5_EB*( (UP(I,J,K)-UP\_HAT(I,J,K)) \ast \ast 2 + (VP(I,J,K)-VP\_HAT(I,J,K)) \ast \ast 2 + (WP(I,J,K)-WP\_HAT(I,J,K)) \ast \ast 2)

!NU\_EDDY(I,J,K) = C\_DEARDORFF*DELTA*SQRT(KSGS)

NU\_EDDY(I,J,K) = C\_DEARDORFF*DELTA*SQRT(KSGS)

MU(I,J,K) = MU(I,J,K) + RHOP(I,J,K)*NU\_EDDY(I,J,K)

ENDDO

CASE (VREMAN) SELECT_TURB! Vreman (2004) eddy viscosity model (experimental)


DO K=1,KBAR
  DO J=1,JBAR
    IF (SOLID(CELL\_INDEX(I,J,K))) CYCLE
    DUDX = RDX(I)*(UU(I,J,K)-UU(I-1,J,K))
    DVDY = RDY(J)*(VV(I,J,K)-VV(I,J-1,K))
    DWDZ = RDZ(K)*(WW(I,J,K)-WW(I,J,K-1))
    DUDY = 0.25_EB*RDY(J)*(UU(I,J+1,K)-UU(I,J-1,K)+UU(I-1,J+1,K)-UU(I-1,J-1,K))
    DUDZ = 0.25_EB*RDZ(K)*(UU(I,J,K+1)-UU(I,J,K-1)+UU(I-1,J,K+1)-UU(I-1,J,K-1))
    DVDX = 0.25_EB*RDZ(K)*(VV(I+1,J,K)-VV(I-1,J,K)+VV(I+1,J-1,K)-VV(I-1,J-1,K))
    DVDZ = 0.25_EB*RDZ(K)*(VV(I,J,K+1)-VV(I,J,K-1)+VV(I,J-1,K+1)-VV(I,J-1,K-1))
    DWDX = 0.25_EB*RDZ(K)*(WW(I+1,J,K)-WW(I-1,J,K)+WW(I+1,J-1,K)-WW(I-1,J-1,K))
    DWDY = 0.25_EB*RDY(J)*(WW(I,J+1,K)-WW(I,J-1,K)+WW(I,J+1,K-1)-WW(I,J-1,K-1))

! Vreman, Eq. (6)
A_\_IJ (1,1)=DUDX; A_\_IJ (2,1)=DUDY; A_\_IJ (3,1)=DWDX
A$_{IJ}(1,2)$=DVDX; A$_{IJ}(2,2)$=DVDY; A$_{IJ}(3,2)$=DVDZ
A$_{IJ}(1,3)$=DVDX; A$_{IJ}(2,3)$=DVDY; A$_{IJ}(3,3)$=DVDZ

AA=0._EB
DO JJ=1,3
    AA = AA + A$_{IJ}(II, JJ)$*A$_{IJ}(II, JJ)$
ENDDO

DO II=1,3
    AA = AA + A$_{IJ}(II, JJ)$*A$_{IJ}(II, JJ)$
ENDDO

! Vreman, Eq. (7)
B$_{IJ}(1,1)$=(DX(1)*A$_{IJ}(1,1)$)**2 + (DY(J)*A$_{IJ}(2,1)$)**2 + (DZ(K)*A$_{IJ}(3,1)$)**2
B$_{IJ}(2,2)$=(DX(1)*A$_{IJ}(1,2)$)**2 + (DY(J)*A$_{IJ}(2,2)$)**2 + (DZ(K)*A$_{IJ}(3,2)$)**2
B$_{IJ}(3,3)$=(DX(1)*A$_{IJ}(1,3)$)**2 + (DY(J)*A$_{IJ}(2,3)$)**2 + (DZ(K)*A$_{IJ}(3,3)$)**2

B$_{IJ}(1,2)$=DX(1)**2*A$_{IJ}(1,1)$*A$_{IJ}(1,2)$ + DY(J)
   **2*A$_{IJ}(2,1)$*A$_{IJ}(2,2)$ + DZ(K)**2*A$_{IJ}(3,1)$*A$_{IJ}(3,2)$
B$_{IJ}(1,3)$=DX(1)**2*A$_{IJ}(1,1)$*A$_{IJ}(1,3)$ + DY(J)
   **2*A$_{IJ}(2,1)$*A$_{IJ}(2,3)$ + DZ(K)**2*A$_{IJ}(3,1)$*A$_{IJ}(3,3)$
B$_{IJ}(2,3)$=DX(1)**2*A$_{IJ}(1,2)$*A$_{IJ}(1,3)$ + DY(J)
   **2*A$_{IJ}(2,2)$*A$_{IJ}(2,3)$ + DZ(K)**2*A$_{IJ}(3,2)$*A$_{IJ}(3,3)$

BB = B$_{IJ}(1,1)$*B$_{IJ}(2,2)$ - B$_{IJ}(1,2)$**2 &
     + B$_{IJ}(1,1)$*B$_{IJ}(3,3)$ - B$_{IJ}(1,3)$**2 &
     + B$_{IJ}(2,2)$*B$_{IJ}(3,3)$ - B$_{IJ}(2,3)$**2  !
Vreman, Eq. (8)

IF (ABS(AA)>ZERO_P) THEN
    NU_EDDY(I, J, K) = C_VREMAN*SQRT(BB/AA)  !
Vreman, Eq. (5)
ELSE
    NU_EDDY(I, J, K)=0._EB
ENDIF

MU(I, J, K) = MU(I, J, K) + RHOP(I, J, K)*NU_EDDY(I, J, K)
APPENDIX B. IMPLEMENTED CODE

171

END SELECT SELECT_TURB

! Add viscosity for stably stratified flows (experimental)

GRAVITY_IF: IF (LES AND GRAV, VISC) THEN

DO K=1,KBAR
  DO J=1,JBAR
    DO I=1,IBAR
      IF (SOLID(CELL_INDEX(I, J, K))) CYCLE
      IF (TWO_D) THEN
        DELTA = MAX(DX(I), DZ(K))
      ELSE
        DELTA = MAX(DX(I), DY(J), DZ(K))
      ENDIF
      GRAD_RHO(1) = 0.5 * EB * RDX(I) * (RHOP(I+1, J, K) - RHOP(I-1, J, K))
      GRAD_RHO(2) = 0.5 * EB * RDY(J) * (RHOP(I, J+1, K) - RHOP(I, J-1, K))
      GRAD_RHO(3) = 0.5 * EB * RDZ(K) * (RHOP(I, J, K+1) - RHOP(I, J, K-1))
      NU_G = C_G * DELTA**2 * SQRT(MAX(ZERO_P, DOT_PRODUCT(GRAD_RHO, GVEC))/RHOP(I, J, K))
      MU(I, J, K) = MAX(MU(I, J, K), RHOP(I, J, K) * NU_G)
    ENDDO
  ENDDO
 ENDDO
ENDIF GRAVITY_IF

! Compute resolved kinetic energy per unit mass

!$OMP DO COLLAPSE(3) SCHEDULE(STATIC) PRIVATE(K, I, U2, V2, W2)

DO K=1,KBAR
  DO J=1,JBAR
    DO I=1,IBAR


APPENDIX B. IMPLEMENTED CODE

U2 = 0.25*EB*(UU(I-1,J,K)+UU(I,J,K))**2
V2 = 0.25*EB*(VV(I,J-1,K)+VV(I,J,K))**2
W2 = 0.25*EB*(WW(I,J,K-1)+WW(I,J,K))**2
KRES(I,J,K) = 0.5*EB*(U2+V2+W2)

ENDDO
ENDDO
ENDDO
!$OMP END DO NOWAIT

! Mirror viscosity into solids and exterior boundary cells

!$OMP DO SCHEDULE(STATIC) &
!$OMP PRIVATE(IW, II, JJ, KK, IIG, JJG, KKG)

WALL_LOOP: DO IW=1,NEXTERNAL_WALL CELLS+NINTERIAL_WALL CELLS
W => WALL(IW)
IF (WBOUNDARY_TYPE= NULL BOUNDARY) CYCLE WALL LOOP
II = W%II
JJ = W%JJ
KK = W%KK
IOR = W%IOR
IIG = W%IIG
JJG = W%JJG
KKG = W%KKG

SELECT CASE(WBOUNDARY_TYPE)
CASE(SOLID BOUNDARY)
IF (LES) THEN
IF (NTRACKED_SPECIES>0 .AND. .NOT.EVACUATION_ONLY(NM)) &
ZZ_GET(1:NTRACKED_SPECIES) = ZZP(IIG,JJG,KKG,1:NTRACKED_SPECIES)
CALL GET VISCOSITY(ZZ_GET,MU_DNS,TMP(IIG,JJG,KKG))
SELECT CASE(IOR)
CASE(1): MU(IIG,JJG,KKG) = MAX(MU_DNS,ONTH*MU(IIG+1,JJG,KKG))
CASE(-1): MU(IIG,JJG,KKG) = MAX(MU_DNS,ONTH*MU(IIG-1,JJG,KKG))
CASE(2): MU(IIG,JJG,KKG) = MAX(MU_DNS,ONTH*MU(IIG,JJG+1,KKG))
CASE(-2): MU(IIG,JJG,KKG) = MAX(MU_DNS,ONTH*MU(IIG,JJG-1,KKG))
CASE(3): MU(IIG,JJG,KKG) = MAX(MU_DNS,ONTH*MU(IIG,JJG,KKG+1))
CASE (-3) ; \(\text{MU}(\text{IIG}, \text{JJG}, \text{KKG}) = \max(\text{MU}_{\text{DNS}}, \text{ONTH} \times \text{MU}(\text{IIG}, \text{JJG}, \text{KKG} - 1))\)

END SELECT

ENDIF

IF (\(\text{SOLID}((\text{CELL\_INDEX}(\text{II}, \text{JJ}, \text{KK})))\)) \(\text{MU}(\text{II}, \text{JJ}, \text{KK}) = \text{MU}(\text{IIG}, \text{JJG}, \text{KKG})\)

CASE (OPEN\_BOUNDARY, MIRROR\_BOUNDARY)

\(\text{MU}(\text{II}, \text{JJ}, \text{KK}) = \text{MU}(\text{IIG}, \text{JJG}, \text{KKG})\)

\(\text{KRES}(\text{II}, \text{JJ}, \text{KK}) = \text{KRES}(\text{IIG}, \text{JJG}, \text{KKG})\)

END SELECT

ENDDO WALL LOOP

!$OMP END DO

!$OMP WORKSHARE

\(\text{MU}(0, 0: \text{JBP1}, 0) = \text{MU}(1, 0: \text{JBP1}, 1)\)

\(\text{MU}(\text{IBP1}, 0: \text{JBP1}, 0) = \text{MU}(\text{IBAR}, 0: \text{JBP1}, 1)\)

\(\text{MU}(\text{IBP1}, 0: \text{JBP1}, \text{KBP1}) = \text{MU}(\text{IBAR}, 0: \text{JBP1}, \text{KBAR})\)

\(\text{MU}(0: \text{IBP1}, 0, 0) = \text{MU}(0: \text{IBP1}, 1, 1)\)

\(\text{MU}(0: \text{IBP1}, \text{JBP1}, 0) = \text{MU}(0: \text{IBP1}, \text{JBAR}, 1)\)

\(\text{MU}(0: \text{IBP1}, \text{JBP1}, \text{KBP1}) = \text{MU}(0: \text{IBP1}, \text{JBAR}, \text{KBAR})\)

\(\text{MU}(0, \text{IBP1}, 0: \text{KBP1}) = \text{MU}(0: \text{IBP1}, 1, \text{KBAR})\)

\(\text{MU}(0, 0: \text{KBP1}) = \text{MU}(1, 0: \text{KBP1})\)

\(\text{MU}(\text{IBP1}, 0, 0: \text{KBP1}) = \text{MU}(\text{IBAR}, 1, 0: \text{KBP1})\)

\(\text{MU}(\text{IBP1}, \text{JBP1}, 0: \text{KBP1}) = \text{MU}(\text{IBAR}, \text{JBAR}, 0: \text{KBP1})\)

\(\text{MU}(0, \text{JBP1}, 0: \text{KBP1}) = \text{MU}(1, \text{JBAR}, 0: \text{KBP1})\)

!$OMP END WORKSHARE

!$OMP END PARALLEL

END SUBROUTINE COMPUTE VISCOSITY
APPENDIX B. IMPLEMENTED CODE

B.3 mesh.f90

Modified subroutine in mesh.f90:

```fortran
MODULE MESH_VARIABLES

! Data structure for mesh-dependent variables

USE PRECISION_PARAMETERS
USE TYPES
IMPLICIT NONE

CHARACTER(255), PARAMETER :: meshid='$Id: mesh.f90 · 10087 · 2012-02-15 21:06:17Z · randy.mcdermott$'
CHARACTER(255), PARAMETER :: meshrev='$Revision: 10087 $'
CHARACTER(255), PARAMETER :: meshdate='$Date: 2012-02-15 22:06:17 +0100 (on, 15 feb 2012) $'

TYPE MESH_TYPE
  REAL(EB), POINTER, DIMENSION(:, :, :) :: &
    U, V, W, US, VS, WS, DDDT, D, DS, H, HS, KRES, FVX, FVY, FVZ, RHO, RHOS, &
    MU, TMP, Q, FRHO, KAPPA, QR, QRW, U11, RSUM, DLAGRANGIAN, DREACTION, &
    CSD2, MIX_TIME, STRAIN_RATE, KFST4, RHO_HS_OVER_PBAR, D_RHSOP_DT, D_RHSOP_DT_S, NU_EDDY !ADDED
  REAL(EB), POINTER, DIMENSION(:, :, :, :) :: ZZ, ZZS, DEL_RHO_D_DEL_Z
  REAL(EB), POINTER, DIMENSION(:, :, :) :: AVG_DROP_DEN, AVG_DROP_TMP, AVG_DROP_RAD, AVG_DROP_AREA
  REAL(EB), POINTER, DIMENSION(:, :) :: AVG_DROP_DEN_ALL
  REAL(EB), POINTER, DIMENSION(:, :) :: UVW_GHOST
  ! ADDED
  REAL(EB) :: POIS_PTBP, POIS_ERR
  REAL(EB), POINTER, DIMENSION(:, :) :: SAVE1, SAVE2, WORK
  REAL(EB), POINTER, DIMENSION(:, :) :: PRHS
  REAL(EB), POINTER, DIMENSION(:, :) :: BXS, BXS, BYF, BYF, BZS, BZF, BXST, BXFT, BYST, BYFT, BZST, BZFT
  INTEGER :: LSAVE, LWORK, LBC, MBC, NBC, ITRN, JTRN, KTRN, IPS
  REAL(EB), POINTER, DIMENSION(:, :) :: P_0, RHO_0, TMP_0, D_PBAR_DT, D_PBAR_S_DT, U_LEAK, U_DUCT
  REAL(EB), POINTER, DIMENSION(:, :) :: PBAR, PBAR_S, R_PBAR
```

APPENDIX B. IMPLEMENTED CODE

32 INTEGER, POINTER, DIMENSION(::,:), :: PRESSURE_ZONE
33 INTEGER, POINTER, DIMENSION(::) :: PRESSURE_BC_INDEX
34 REAL(EB), POINTER, DIMENSION(::,:,:,:) :: WORK1, WORK2, WORK3, WORK4, WORK5, WORK6, WORK7, WORK8
35 
36 REAL(EB), POINTER, DIMENSION(::,:,:,:) :: TURB_WORK1, TURB_WORK2, TURB_WORK3, TURB_WORK4
37 REAL(EB), POINTER, DIMENSION(::,:,:,:) :: TURB_WORK5, TURB_WORK6, TURB_WORK7, TURB_WORK8
38 REAL(EB), POINTER, DIMENSION(::,:) :: TURB_WORK9, TURB_WORK10
39 REAL(EB), POINTER, DIMENSION(::) :: TURB_WORK11, TURB_WORK12
40 
41 REAL(EB), POINTER, DIMENSION(::,:) :: IBM_SAVE1, IBM_SAVE2, IBM_SAVE3, IBM_SAVE4, IBM_SAVE5, IBM_SAVE6
42 INTEGER, POINTER, DIMENSION(::,:) :: U_MASK, V_MASK, W_MASK, P_MASK
43 
44 REAL(EB), POINTER, DIMENSION(::) :: WALL_WORK1, WALL_WORK2
45 REAL(FB), POINTER, DIMENSION(::,:,:) :: QQ
46 REAL(FB), POINTER, DIMENSION(::,:) :: PP, PPN
47 INTEGER, POINTER, DIMENSION(::,:) :: IBK
48 INTEGER, POINTER, DIMENSION(::,:) :: IBLK
49 
50 REAL(EB) :: DT, DT_PREV, DT_NEXT, DT_INIT
51 REAL(EB) :: CFL, DIVMX, DIVMN, VN, RESMAX, PART_CFL
52 INTEGER :: ICFL, JCFL, KCFL, IMX, JMX, KMX, IMN, JMN, KMN, I_VN, J_VN, K_VN, IRM, JRM, KRM
53 
54 INTEGER :: N_EDGES
55 INTEGER, POINTER, DIMENSION(::,:) :: IJKE, EDGE_INDEX
56 REAL(EB), POINTER, DIMENSION(::) :: TAU_E, OME_E
57 INTEGER, POINTER, DIMENSION(::,:) :: EDGE_TYPE
58 
59 INTEGER :: MESH_LEVEL
60 INTEGER :: IBAR, JBAR, KBAR, IBM1, JBM1, KBM1, IBP1, JBP1, KBP1
61 INTEGER, POINTER, DIMENSION(::) :: RGB
62 REAL(EB) :: DXI, ETA, DZETA, RDXI, RDETA, RDZETA, &
63 DXMIN, DXMAX, DYMIN, DYMAX, DZMIN, DZMAX, &
64 XS, XF, YS, YF, ZS, ZF, RDXINT, RDYINT, RDZINT, CELL_SIZE
65 REAL(EB), POINTER, DIMENSION(::) :: R, RC, X, Y, Z, XC, YC, ZC, HC, HY, &
66 HZ, &
67 DX, RDX, DXN, RDXN, DY, RDY, DYN, RDYN, DZ, RDZ, DZN, RDZN, &
APPENDIX B. IMPLEMENTED CODE

67 CELLSI, CELLSJ, CELLSK, RRN
68 REAL(FB), POINTER, DIMENSION(:) :: XPLT, YPLT, ZPLT
69  INTEGER :: N_OBST
70  TYPE(OBSTRACTION_TYPE), POINTER, DIMENSION(:) :: OBSTRUCTION
71  INTEGER :: N_VENT
72  TYPE(VENTS_TYPE), POINTER, DIMENSION(:) :: VENTS
73  INTEGER, POINTER, DIMENSION(:,:), :: CELL_INDEX
74  INTEGER, POINTER, DIMENSION(:,), :: L_CELL, J_CELL, K_CELL,
75  OBST_INDEX, C
76  INTEGER, POINTER, DIMENSION(:), :: WALL_INDEX
77  LOGICAL, POINTER, DIMENSION(:) :: SOLID, EXTERIOR
78  INTEGER :: N_INTERNAL_WALL_CELLS, N_EXTERNAL_WALL_CELLS,
79  N_VIRTUAL_WALL_CELLS, N_GHOST_WALL_CELLS, &
80  CELL_COUNT, WALL_COUNTER
81  REAL(EB) :: BC_CLOCK
82  INTEGER, POINTER, DIMENSION(:,,:) ::
83  EDGE_INTERPOLATION_FACTOR, AWM_AEROSOL
84  REAL(EB), POINTER, DIMENSION(:,,:) ::
85  D_DUWT, &
86  D_CORR, DS_CORR, UVW_SAVE, U_GHOST, V_GHOST, W_GHOST
87  TYPE(WALL_TYPE), POINTER, DIMENSION(:) :: WALL
88  TYPE(OMESH_TYPE), POINTER, DIMENSION(:) :: OMESH
89  TYPE(LAGRANGIAN_PARTICLE_TYPE), POINTER, DIMENSION(:) ::
90  LAGRANGIAN_PARTICLE
91  INTEGER :: NLP, NLPDIM
92  TYPE(HUMAN_TYPE), POINTER, DIMENSION(:) :: HUMAN
93  INTEGER :: N_HUMANS, N_HUMANS_DIM
94  TYPE(HUMAN_GRID_TYPE), POINTER, DIMENSION(:, :) :: HUMAN_GRID
95  INTEGER :: N_SLCF
96  TYPE(SLICE_TYPE), POINTER, DIMENSION(:) :: SLICE
97  INTEGER, POINTER, DIMENSION(:, :) :: INC
98  INTEGER :: NPATCH
99  REAL(EB), POINTER, DIMENSION(:, :, :) :: UUID
100 INTEGER :: RADCALL_COUNTER, ANGLE_INC_COUNTER
101 INTEGER, POINTER, DIMENSION(:, :, :) :: INTERPOLATED_MESH
APPENDIX B. IMPLEMENTED CODE

106 CHARACTER(80), POINTER, DIMENSION(:) :: STRING
107 INTEGER :: N_STRINGS, N_STRINGS_MAX
108
109 !rm ->
110 ! REAL(EB), POINTER, DIMENSION(:,:,:) :: DMPVDT_{FM,VEG}
111 INTEGER, POINTER, DIMENSION(:,:) :: K_AGL_SLICE
112 REAL(EB), POINTER, DIMENSION(,:) :: LS_Z_TERRAIN, VEG_DRAG
113 INTEGER :: N_TERRAIN_SLCF
114 REAL(EB) :: VEG_CLOCK_BC !surf veg
115 !rm <-
116
117 END TYPE MESH_TYPE
118
119 TYPE (MESH_TYPE) , SAVE, DIMENSION(:) , ALLOCATABLE, TARGET :: MESHES
120
121 END MODULE MESH_VARIABLES
122
123 MODULE MESH_POINTERS
124
125 USE PRECISION_PARAMETERS
126 USE MESH_VARIABLES
127 IMPLICIT NONE
128
129 REAL(EB), POINTER, DIMENSION(:,:,:) :: &
130 U, V, W, US, VS, WS, DDDT, D, DS, H, HS, KRES, FVX, FVY, FVZ, RHO, RHOS, &
131 MU, TMP, Q, FRHO, KAPPA, QR, QR_W, UII, RSUM, DLAGRANGIAN, &
132 DREACTION, &
133 CSD2, MTR, MRS, WEM, MIX_TIME, STRAIN_RATE, KFST4, &
134 RHO H S OVER PBAR, D RHSOP DT, D RHSOP DT S, NU EDDY &ADDED
135 REAL(EB), POINTER, DIMENSION(:,:,:) :: ZZ, ZZS, DELRHO_D DEL_Z
136 REAL(EB), POINTER, DIMENSION(:,:,:) :: AVG_DROP_DEN, &
137 AVG DROP TMP, AVG DROP_RAD, AVG DROP AREA
138 REAL(EB), POINTER, DIMENSION(,:) :: AVG DROP_DEN_ALL
139 REAL(EB), POINTER, DIMENSION(,:) :: UVW_GHOST
140 REAL(EB), POINTER :: POIS_PTB, POIS_ERR
141 REAL(EB), POINTER, DIMENSION(,:) :: PRHS
142 REAL(EB), POINTER, DIMENSION(,:) :: BXS, BXF, BYS, BYF, BZS, BZF, &
143 BXST, BXF_T, BYST, BYFT, BZT, BZF_T
APPENDIX B. IMPLEMENTED CODE

142 INTEGER, POINTER :: LS AVE, IWORK, LBC, MBC, NBC, ITRN, JTRN, KTRN, IPS
143 REAL(EB), POINTER, DIMENSION(:) :: P_0, RHO_0, TMP_0, D_PBAR_DT,
   D_PBAR_S, DT, U_LEAK, U_UDCT
144 REAL(EB), POINTER, DIMENSION(:, :) :: PBAR, PBAR_S, R_PBAR
145 INTEGER, POINTER, DIMENSION(:, :, :) :: PRESSURE_ZONE
146 INTEGER, POINTER, DIMENSION(:, :, :) :: PRESSURE_BC_INDEX
147 REAL(EB), POINTER, DIMENSION(:, :, :) :: WORK1, WORK2, WORK3, WORK4,
   WORK5, WORK6, WORK7, WORK8
148 REAL(EB), POINTER, DIMENSION(:, :, :) :: TURB_WORK1, TURB_WORK2,
   TURB_WORK3, TURB_WORK4
149 REAL(EB), POINTER, DIMENSION(:, :, :) :: TURB_WORK5, TURB_WORK6,
   TURB_WORK7, TURB_WORK8
150 REAL(EB), POINTER, DIMENSION(:, :, :) :: TURB_WORK9, TURB_WORK10
151 REAL(EB), POINTER, DIMENSION(:, :) :: U_Mask, V_Mask, W_Mask,
   P_Mask
152 REAL(EB), POINTER, DIMENSION(:, :, :) :: WALL_WORK1, WALL_WORK2
153 REAL(FB), POINTER, DIMENSION(:, :, :) :: QQ
154 INTEGER, POINTER, DIMENSION(:, :, :) :: IBK
155 INTEGER, POINTER, DIMENSION(:, :, :) :: IBLK
156 REAL(EB), POINTER, DIMENSION(:, :) :: DT, DT_PREV, DT_NEXT, DT_INIT
157 REAL(EB), POINTER, DIMENSION(:, :, :) :: CFL, DIVM_X, DIVM_N, RESMAX, PART_CFL
158 INTEGER, POINTER, DIMENSION(:, :) :: ICFL, JCFL, KCFL, IMX, JMX, KMX, IMN, JMN, KMN, I_VN,
   J_VN, K_VN, IRM, JRM, KRM
159 INTEGER, POINTER, DIMENSION(:, :) :: N_EDGES
160 INTEGER, POINTER, DIMENSION(:, :) :: IJKE, EDGE_INDEX
161 INTEGER, POINTER, DIMENSION(:, :) :: TAU_E, OME_E
162 INTEGER, POINTER, DIMENSION(:, :) :: EDGE_TYPE
163 INTEGER, POINTER, DIMENSION(:, :) :: MESH_LEVEL
164 INTEGER, POINTER, DIMENSION(:, :) :: IBAR, JBAR, KBAR, IBM1, JBM1, KBM1, IBP1, JBP1,
   KBP1
165 INTEGER, POINTER, DIMENSION(:, :) :: RGB
166 INTEGER, POINTER, DIMENSION(:, :) :: DXI, DEFA, DZETA, RDXI, RDETA, RZETA, &
   DXMIN, DXMAX, DYMIN, DYMIX, DZMIN, DZMAX, &
   XS, XF, YS, YF, ZS, ZF, RDXINT, RDXINT, RDZINT, CELL_SIZE
appEnB. Implemented Code

176 REAL(EB) , POINTER, DIMENSION(:) :: R, RC, X, Y, Z, XC, YC, ZC, HC, HY, HZ

177 &

178 & DX, RDX, DXN, RDXN, DY, RDY, DYN, RDN, DZ, RDZ, DZN, RDZN, &

179 CELLSI, CELLSJ, CELSK, RRN

179 REAL(FB) , POINTER, DIMENSION(:) :: XPLT, YPLT, ZPLT

180 INTEGER, POINTER :: N_OBST

181 TYPE(OBSTURATION,TYPE) , POINTER, DIMENSION(:) :: OBSTRUCTION

182 INTEGER, POINTER :: N_VENT

183 TYPE(VENTS,TYPE) , POINTER, DIMENSION(:) :: VENTS

184 INTEGER, POINTER, DIMENSION(:, :, :) :: CELL_INDEX

185 INTEGER, POINTER, DIMENSION(:, :) :: l_CELL, j_CELL, k_CELL, 

186 OBST_INDEX_C

186 INTEGER, POINTER, DIMENSION(:, :) :: WALL_INDEX

187 LOGICAL, POINTER, DIMENSION(:) :: SOLID, EXTERIOR

188 INTEGER, POINTER :: N_INTERNAL_WALL_CEILS, N_EXTERNAL_WALL_CEILS 

188 , N_VIRTUAL_WALL_CEILS, N_GHOST_WALL_CEILS, &

189 CELL_COUNT, WALL_COUNTER

190 REAL(EB) . POINTER :: BC_CLOCK

191 REAL(EB) , POINTER, DIMENSION(:, :) :: EDGE_INTERPOLATION_FACTOR, 

192 AWI_AEROSOL

193 REAL(EB) , POINTER, DIMENSION(:, :) :: DUWDT, &

194 D_CORR, DS_CORR, UVW_SAVE, U_GHOST, V_GHOST, W_GHOST

194 TYPE(WALL,TYPE) , POINTER, DIMENSION(:) :: WALL

195 TYPE(OMESH,TYPE) , POINTER, DIMENSION(:) :: OMESH

196 TYPE(LAGRANGIAN_PARTICLE,TYPE) , POINTER, DIMENSION(:) :: 

197 LAGRANGIAN_PARTICLE

198 INTEGER, POINTER :: NLP, NLPDIM

199 TYPE(HUMAN,TYPE) , POINTER, DIMENSION(:) :: HUMAN

200 INTEGER, POINTER :: N_HUMANS, N_HUMANS_DIM

200 TYPE(HUMAN,GRID,TYPE) , POINTER, DIMENSION(:, :) :: HUMAN_GRID

201 INTEGER, POINTER :: N_SLCF

202 TYPE(SLICE,TYPE) , POINTER, DIMENSION(:, :) :: SLICE

203 INTEGER, POINTER, DIMENSION(:, :) :: INC

204 INTEGER, POINTER :: NPATCH

205 REAL(EB) , POINTER, DIMENSION(:, :, :) :: UID

206 INTEGER, POINTER :: RAD_CALL_COUNTER, ANGLE_INC_COUNTER

207 INTEGER, POINTER, DIMENSION(:, :, :) :: INTERPOLATED_MESH

208 CHARACTER(80) , POINTER, DIMENSION(:, :) :: STRING

209 INTEGER, POINTER :: N_STRING, N_STRING_MAX

210 !rm ->

211 !REAL(EB) , POINTER, DIMENSION(:, :, :) :: DMPVDT_FM, VEG

212 INTEGER, POINTER, DIMENSION(:, :) :: K_AGL_SLICE

213 REAL(EB) , POINTER, DIMENSION(:, :) :: LS_Z_TERRAIN, VEG_DRAG
APPENDIX B. IMPLEMENTED CODE

214 INTEGER, POINTER :: N_TERRAIN_SLCF
215 REAL(EB), POINTER :: VEG_CLOCK_BC ! surf veg
216 ! rm <-
217
218 CONTAINS
APPENDIX B. IMPLEMENTED CODE

B.4 init.f90

Modified subroutine in init.f90:

```fortran
SUBROUTINE INITIALIZE_MESH_VARIABLES(NM)

USE PHYSICAL_FUNCTIONS, ONLY: GET_VISCOSITY, GET_SPECIFIC_GAS_CONSTANT, GET_SPECIFIC_HEAT
USE GEOMETRY_FUNCTIONS, ONLY: ASSIGN_PRESSURE_ZONE
USE RADCONS, ONLY: UIDIM
USE CONTROL_VARIABLES
INTEGER :: N, I, J, K, II, JJ, KK, IPTS, JPTS, KPTS, N_EDGES_DIM,
N_TOTAL_WALL_CELLS, IW, IWE, IWG, IC, SURF_INDEX, IOR, IOPZ, &
IERR, IB, JB, KB, IPZ
INTEGER, INTENT(IN) :: NM
REAL(EB) :: MU, N, ZZ, GET(0:N_TRACKED_SPECIES), VC, RTRM, CP, CS,
DELTA
INTEGER, POINTER :: IBP1, JBP1, KBP1, IBAR, JBAR, KBAR, N_EDGES
REAL(EB), POINTER :: XS, XF, YS, YF, ZS, ZF
TYPE(INITIALIZATION_TYPE), POINTER :: IN
TYPE(PZONE_TYPE), POINTER :: PZ
TYPE(DEVICE_TYPE), POINTER :: DV
TYPE(VENTS_TYPE), POINTER :: VT

IERR = 0
M => MESHES(NM)
IBP1 => M(IBP1)
JBP1 => M(JBP1)
KBP1 => M(KBP1)
IBAR => M(IBAR)
JBAR => M(JBAR)
KBAR => M(KBAR)
N_EDGES => M(N_EDGES)
XS => M(XS)
YS => M(YS)
ZS => M(ZS)
XF => M(XF)
YF => M(YF)
ZF => M(ZF)
ALLOCATE(M%RHO(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
CALL ChkMemErr('INIT','RHO',IZERO)
ALLOCATE(M%RHOS(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
CALL ChkMemErr('INIT','RHOS',IZERO)
M%RHOS = RHOA
```
APPENDIX B. IMPLEMENTED CODE

38  ALLOCATE(M%TMP(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
39  CALL ChkMemErr('INIT','TMP',IZERO)
40  ALLOCATE(M%FRHO(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
41  CALL ChkMemErr('INIT','FRHO',IZERO)
42  ALLOCATE(M%U(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
43  CALL ChkMemErr('INIT','U',IZERO)
44  ALLOCATE(M%V(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
45  CALL ChkMemErr('INIT','V',IZERO)
46  ALLOCATE(M%W(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
47  CALL ChkMemErr('INIT','W',IZERO)
48  ALLOCATE(M%US(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
49  CALL ChkMemErr('INIT','US',IZERO)
50  ALLOCATE(M%VS(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
51  CALL ChkMemErr('INIT','VS',IZERO)
52  ALLOCATE(M%WS(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
53  CALL ChkMemErr('INIT','WS',IZERO)
54  ALLOCATE(M%FVX(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
55  CALL ChkMemErr('INIT','FVX',IZERO)
56  ALLOCATE(M%FVY(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
57  CALL ChkMemErr('INIT','FVY',IZERO)
58  ALLOCATE(M%FVZ(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
59  CALL ChkMemErr('INIT','FVZ',IZERO)
60  ALLOCATE(M%H(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
61  CALL ChkMemErr('INIT','H',IZERO)
62  ALLOCATE(M%HS(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
63  CALL ChkMemErr('INIT','HS',IZERO)
64  ALLOCATE(M%KRES(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
65  CALL ChkMemErr('INIT','KRES',IZERO)
66  ALLOCATE(M%DDDT(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
67  CALL ChkMemErr('INIT','DDDT',IZERO)
68  ALLOCATE(M%D(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
69  CALL ChkMemErr('INIT','D',IZERO)
70  ALLOCATE(M%D(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
71  CALL ChkMemErr('INIT','D',IZERO)
72  ALLOCATE(M%M(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
73  CALL ChkMemErr('INIT','M',IZERO)
74  ALLOCATE(M%M(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
75  CALL ChkMemErr('INIT','M',IZERO)
76  ALLOCATE(M%M(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
77  CALL ChkMemErr('INIT','M',IZERO)
78  ALLOCATE(M%M(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
79  CALL ChkMemErr('INIT','M',IZERO)
80
APPENDIX B. IMPLEMENTED CODE

IF (.NOT. EVACUATION_ONLY(NM)) THEN
    ALLOCATE(M%Q(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
    CALL ChkMemErr('INIT','Q',IZERO)
ENDIF

ALLOCATE(M%MIX_TIME(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
CALL ChkMemErr('INIT','MIX_TIME',IZERO)
M%MIX_TIME=M%DT

ALLOCATE(M%MIX_TIME(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
CALL ChkMemErr('INIT','MIX_TIME',IZERO)
M%MIX_TIME=M%DT

ALLOCATE(M%PBAR(0:KBP1,0:N_ZONE),STAT=IZERO)
CALL ChkMemErr('INIT','PBAR',IZERO)
ALLOCATE(M%PBAR_S(0:KBP1,0:N_ZONE),STAT=IZERO)
CALL ChkMemErr('INIT','PBAR_S',IZERO)
ALLOCATE(M%R_PBAR(0:KBP1,0:N_ZONE),STAT=IZERO)
CALL ChkMemErr('INIT','R_PBAR',IZERO)
ALLOCATE(M%D_PBAR_DT(0:N_ZONE),STAT=IZERO)
CALL ChkMemErr('INIT','D_PBAR_DT',IZERO)
ALLOCATE(M%D_PBAR_S_DT(0:N_ZONE),STAT=IZERO)
CALL ChkMemErr('INIT','D_PBAR_S_DT',IZERO)
ALLOCATE(M%P_0(0:KBP1),STAT=IZERO)
CALL ChkMemErr('INIT','P_0',IZERO)
ALLOCATE(M%TMP_0(0:KBP1),STAT=IZERO)
CALL ChkMemErr('INIT','TMP_0',IZERO)
ALLOCATE(M%RHO_0(0:KBP1),STAT=IZERO)
CALL ChkMemErr('INIT','RHO_0',IZERO)

ALLOCATE(M%U_LEAK(0:N_ZONE),STAT=IZERO)
CALL ChkMemErr('INIT','U_LEAK',IZERO)
M%U_LEAK = 0._EB

ALLOCATE(M%U_LEAK(0:N_ZONE),STAT=IZERO)
CALL ChkMemErr('INIT','U_LEAK',IZERO)
M%U_LEAK = 0._EB

ALLOCATE(M%Z_Z(0:IBP1,0:JBP1,0:KBP1,N_TRACKED_SPECIES),STAT=IZERO)
CALL ChkMemErr('INIT','Z_Z',IZERO)
M%Z_Z = 0._EB
APPENDIX B. IMPLEMENTED CODE

121  
122  ALLOCATE(M\%ZZS (0:IBP1,0:JBP1,0:KBP1,N\_TRACKED\_SPECIES),STAT=IZERO)
123  CALL ChkMemErr ('INIT', 'ZZS',IZERO)
124  M\%ZZS = 0 ._EB
125  ALLOCATE(M\%DEL\_RHO\_D\_DEL\_Z(0:IBP1,0:JBP1,0:KBP1,
126      N\_TRACKED\_SPECIES),STAT=IZERO)
127  CALL ChkMemErr ('INIT', 'DEL\_RHO\_D\_DEL\_Z',IZERO)
128  M\%DEL\_RHO\_D\_DEL\_Z = 0 ._EB
129  ENDIF
130
131  ALLOCATE(M\%RSUM(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
132  CALL ChkMemErr ('INIT', 'RSUM',IZERO)
133  M\%RSUM = RSUM0
134
135  ! Allocate reaction divergence
136  IF (N\_REACTIONS > 0) THEN
137  ALLOCATE(M\%D\_REACTION(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
138  CALL ChkMemErr ('INIT', 'D\_REACTION',IZERO)
139  M\%D\_REACTION = 0 ._EB
140  ENDIF
141
142  ! Enthalpy arrays (experimental)
143  IF (ENTHALPY\_TRANSPORT) THEN
144  ALLOCATE(M\%RHO\_H\_S\_OVER\_PBAR (0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
145  CALL ChkMemErr ('INIT', 'RHO\_H\_S\_OVER\_PBAR',IZERO)
146  M\%RHO\_H\_S\_OVER\_PBAR = 0 ._EB ! initialized in DENSITY
147  ALLOCATE(M\%D\_RHSOP\_DT(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
148  CALL ChkMemErr ('INIT', 'D\_RHSOP\_DT',IZERO)
149  M\%D\_RHSOP\_DT = 0 ._EB
150  ALLOCATE(M\%D\_RHSOP\_DT\_S(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
151  CALL ChkMemErr ('INIT', 'D\_RHSOP\_DT\_S',IZERO)
152  M\%D\_RHSOP\_DT\_S = 0 ._EB
153  ENDIF
154
155  ! Allocate water mass arrays if sprinklers are present
156  IF (PARTICLE\_FILE) PARTICLE\_TAG = NM
157
158  IF (N\_LAGRANGIAN\_CLASSES > 0 .AND. .NOT. EVACUATION\_ONLY(NM)) THEN
APPENDIX B. IMPLEMENTED CODE

185

ALLOCATE(M%AVG_DROP_DEN_ALL(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
CALL ChkMemErr('INIT','AVG_DROP_DEN_ALL',IZERO)
M%AVG_DROP_DEN_ALL=0._EB
ENDIF

IF (N_LP_ARRAY_INDICES>0 .AND. .NOT.EVACUATION_ONLY(NM)) THEN
ALLOCATE(M%QR_W(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
CALL ChkMemErr('INIT','QR_W',IZERO)
M%QR_W = 0._EB
ALLOCATE(M%AVG_DROP_DEN(0:IBP1,0:JBP1,0:KBP1,
N_LP_ARRAY_INDICES),STAT=IZERO)
CALL ChkMemErr('INIT','AVG_DROP_DEN',IZERO)
M%AVG_DROP_DEN=0._EB
ALLOCATE(M%AVG_DROP_AREA(0:IBP1,0:JBP1,0:KBP1,
N_LP_ARRAY_INDICES),STAT=IZERO)
CALL ChkMemErr('INIT','AVG_DROP_AREA',IZERO)
M%AVG_DROP_AREA=0._EB
ALLOCATE(M%AVG_DROP_TMP(0:IBP1,0:JBP1,0:KBP1,
N_LP_ARRAY_INDICES),STAT=IZERO)
CALL ChkMemErr('INIT','AVG_DROP_TMP',IZERO)
M%AVG_DROP_TMP=TMP
ALLOCATE(M%AVG_DROP_RAD(0:IBP1,0:JBP1,0:KBP1,
N_LP_ARRAY_INDICES),STAT=IZERO)
CALL ChkMemErr('INIT','AVG_DROP_RAD',IZERO)
M%AVG_DROP_RAD=0._EB
ALLOCATE(M%D_LAGRANGIAN(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
CALL ChkMemErr('INIT','D_LAGRANGIAN',IZERO)
M%D_LAGRANGIAN = 0._EB
ENDIF

! If radiation absorption desired allocate arrays

IF (.NOT.EVACUATION_ONLY(NM)) THEN
ALLOCATE(M%QR(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
CALL ChkMemErr('INIT','QR',IZERO)
M%QR = 0._EB
ALLOCATE(M%KAPPA(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
CALL ChkMemErr('INIT','KAPPA',IZERO)
M%KAPPA = KAPPA0
ALLOCATE(M%UII(0:IBP1,0:JBP1,0:KBP1),STAT=IZERO)
CALL ChkMemErr('INIT','UII',IZERO)
M%UII = 0._EB
APPENDIX B. IMPLEMENTED CODE

186

198 ALLOCATE(M%Kfst4(0 : IBP1, 0 : JBP1, 0 : KBP1), STAT=IZERO)
199 CALL ChkMemErr('INIT', 'KFST4', IZERO)
200 M%Kfst4 = 0, _EB
201 ELSE
202 ALLOCATE(M%QR(0 : IBP1, 0 : JBP1, 0 : KBP1), STAT=IZERO)
203 CALL ChkMemErr('INIT', 'QR', IZERO)
204 M%QR = 0, _EB
205 ENDIF
206
207 ! Work arrays
208
209 ALLOCATE(M%work1(0 : IBP1, 0 : JBP1, 0 : KBP1), STAT=IZERO)
210 CALL ChkMemErr('INIT', 'WORK1', IZERO)
211 ALLOCATE(M%work2(0 : IBP1, 0 : JBP1, 0 : KBP1), STAT=IZERO)
212 CALL ChkMemErr('INIT', 'WORK2', IZERO)
213 ALLOCATE(M%work3(0 : IBP1, 0 : JBP1, 0 : KBP1), STAT=IZERO)
214 CALL ChkMemErr('INIT', 'WORK3', IZERO)
215 ALLOCATE(M%work4(0 : IBP1, 0 : JBP1, 0 : KBP1), STAT=IZERO)
216 CALL ChkMemErr('INIT', 'WORK4', IZERO)
217 ALLOCATE(M%work5(0 : IBP1, 0 : JBP1, 0 : KBP1), STAT=IZERO)
218 CALL ChkMemErr('INIT', 'WORK5', IZERO)
219 ALLOCATE(M%work6(0 : IBP1, 0 : JBP1, 0 : KBP1), STAT=IZERO)
220 CALL ChkMemErr('INIT', 'WORK6', IZERO)
221 ALLOCATE(M%work7(0 : IBP1, 0 : JBP1, 0 : KBP1), STAT=IZERO)
222 CALL ChkMemErr('INIT', 'WORK7', IZERO)
223 ALLOCATE(M%work8(0 : IBP1, 0 : JBP1, 0 : KBP1), STAT=IZERO)
224 CALL ChkMemErr('INIT', 'WORK8', IZERO)
225
226 IF (IMMERSED_BOUNDARY_METHOD==2) THEN
227 ALLOCATE(M%IBM_save1(0 : IBP1, 0 : JBP1, 0 : KBP1), STAT=IZERO)
228 CALL ChkMemErr('INIT', 'IBM_SAVE1', IZERO)
229 ALLOCATE(M%IBM_save2(0 : IBP1, 0 : JBP1, 0 : KBP1), STAT=IZERO)
230 CALL ChkMemErr('INIT', 'IBM_SAVE2', IZERO)
231 ALLOCATE(M%IBM_save3(0 : IBP1, 0 : JBP1, 0 : KBP1), STAT=IZERO)
232 CALL ChkMemErr('INIT', 'IBM_SAVE3', IZERO)
233 ALLOCATE(M%IBM_save4(0 : IBP1, 0 : JBP1, 0 : KBP1), STAT=IZERO)
234 CALL ChkMemErr('INIT', 'IBM_SAVE4', IZERO)
235 ALLOCATE(M%IBM_save5(0 : IBP1, 0 : JBP1, 0 : KBP1), STAT=IZERO)
236 CALL ChkMemErr('INIT', 'IBM_SAVE5', IZERO)
237 ALLOCATE(M%IBM_save6(0 : IBP1, 0 : JBP1, 0 : KBP1), STAT=IZERO)
238 CALL ChkMemErr('INIT', 'IBM_SAVE6', IZERO)
239 ENDIF
240
APPENDIX B. IMPLEMENTED CODE

! Boundary file patch counter
ALLOCATE(M%INC(-3:3,0:M%N,OBST),STAT=IZERO)
CALL ChkMemErr(’INIT’,’INC’,IZERO)

! Initialize background pressure, temperature and density
M%DPBAR_DT = 0._EB
M%DPBAR_SDT = 0._EB

IF (STRATIFICATION .AND. .NOT.EVACUATIONONLY(NM)) THEN
   DO K=0,M%KBP1
      M%TMP_0(K) = TMPA + LAPSE_RATE*(M%ZC(K)-GROUNDLLEVEL)
      IF (ABS(LAPSE_RATE)>ZERO.P) THEN
         M%P_0(K) = P_INF*(M%TMP_0(K)/M%TMP_0(0))**(GVEC(3)/RSUM0/LAPSE_RATE)
      ELSE
         M%P_0(K) = P_INF*EXP(GVEC(3)*(M%ZC(K)-GROUNDLLEVEL)/(RSUM0*TMPA))
      ENDIF
   ENDDO
ELSE
   M%TMP_0(:) = TMPA
   M%P_0(:) = P_INF
ENDIF

DO K=0,M%KBP1
   M%PBAR(K,:) = M%P_0(K)
   M%PBAR_S(K,:) = M%P_0(K)
   M%RHO_0(K) = M%P_0(K)/(M%TMP_0(K)*RSUM0)
ENDDO

! Initialize various time step variables
M%DT_PREV = M%DT
M%DT_NEXT = M%DT
M%DT_INIT = M%DT

! Initialize major arrays
DO K=0,M%KBP1
   M%RHO(:,:,K) = M%RHO_0(K)
   M%TMP(:,:,K) = M%TMP_0(K)
ENDDO
APPENDIX B. IMPLEMENTED CODE

282 IF (.NOT.EVACUATION ONLY(NM)) M\%FRHO = 0.\_EB
283 M\%U = U0
284 M\%V = V0
285 M\%W = W0
286 M\%US = U0
287 M\%VS = V0
288 M\%WS = W0
289 M\%FVX = 0.\_EB
290 M\%FVY = 0.\_EB
291 M\%FVZ = 0.\_EB
292 M\%H = H0
293 M\%HS = H0
294 M\%KRES = 0.\_EB
295 M\%DDDT = 0.\_EB
296 M\%D = 0.\_EB
297 M\%DS = 0.\_EB
298 IF (.NOT.EVACUATION ONLY(NM)) THEN
299 M\%Q = 0.\_EB
300 ENDIF
301 IF (EVACUATION ONLY(NM)) THEN
302 M\%U = 0.\_EB
303 M\%V = 0.\_EB
304 M\%W = 0.\_EB
305 M\%US = 0.\_EB
306 M\%VS = 0.\_EB
307 M\%WS = 0.\_EB
308 M\%H = 0.\_EB
309 M\%HS = 0.\_EB
310 ENDIF
311 IF (N\_TRACKED\_SPECIES > 0 .AND. .NOT.EVACUATION ONLY(NM)) THEN
312 M\%DEL\_RHO\_D\_DEL\_Z = 0.\_EB
313!
314 Viscosity
315
316 IF (N\_TRACKED\_SPECIES>0) ZZ\_GET (1:N\_TRACKED\_SPECIES) =
317 SPECIES\_MIXTURE (1:N\_TRACKED\_SPECIES)%ZZ0
318 CALL GET\_VISCOITY (ZZ\_GET, MU_N, TMPA)
319 M\%MU = MU_N
320 CS = C\_SMAGORINSKY
321 IF (EVACUATION ONLY(NM)) CS=0.9\_EB
322 DO K=0,KBP1
APPENDIX B. IMPLEMENTED CODE

189

\begin{verbatim}
323  DO J=0,JB1
324      DO I=0,IB1
325       IF (TWO(J)) THEN
326          DELTA = MAX(M%DX(I),M%DZ(K))
327       ELSE
328          DELTA = MAX(M%DX(I),M%DY(J),M%DZ(K))
329       ENDIF
330       M%CSD2(I,J,K) = (CS*DELTA)**2
331    ENDDO
332  ENDDO
333 ENDDO
334
335 ! Initialize mass fraction arrays
336
337 IF (N_TRACKED_SPECIES > 0 .AND. .NOT.EVACUATION_ONLY(NM)) THEN
338  DO N=1,N_TRACKED_SPECIES
339      M%ZZ(:, :, :, N) = SPECIES_MIXTURE(N)%ZZ0
340      M%ZZS(:, :, :, N) = SPECIES_MIXTURE(N)%ZZ0
341  ENDDO
342 ENDIF
343
344 ! Initialize pressure ZONES
345
346 ALLOCATE(M%PRESSURE_ZONE(0:IBP1,0:JB1,0:KB1),STAT=IZERO)
347 CALL ChkMemErr( 'INIT', 'PRESSURE_ZONE', IZERO)
348 M%PRESSURE_ZONE = 0
349 ZONE_LOOP: DO N=1,N_ZONE
350       IF (EVACUATIONONLY(NM)) CYCLE ZONE_LOOP
351       PZ => P_ZONE(N)
352       DO K=0,KB1
353         DO J=0,JB1
354           DO I=0,IB1
355             IF (M%PRESSURE_ZONE(I,J,K)==N) CYCLE
356             IF (M%XC(I) > PZ%X1 .AND. M%XC(I) < PZ%X2 .AND. &
357                M%YC(J) > PZ%Y1 .AND. M%YC(J) < PZ%Y2 .AND. &
358                M%ZC(K) > PZ%Z1 .AND. M%ZC(K) < PZ%Z2) THEN
359                M%PRESSURE_ZONE(I,J,K) = N
360             DO IOPZ=0,N_ZONE
361               IF (PZ%LEAK_AREA(IOPZ) > 0 .EB)
362                  ACTUAL_LEAK_AREA(N,IOPZ) = PZ%LEAK_AREA(IOPZ)
363             ENDIF
364       ENDIF
365     ENDDO
366   ENDDO
367 ENDDO
\end{verbatim}
IF (.NOT.M%SOLID(M%CELL_INDEX(I,J,K))) CALL ASSIGN_PRESSURE_ZONE(NM,M%XC(I),M%YC(J),M%ZC(K),N)

ENDDO

ENDIF

ENDDO

ENDDO

ENDDO ZONE_LOOP

! Over-ride default ambient conditions with user-prescribed INITIALizations

DO N=1,N_INIT
   IF (EVACUATION_ONLY(NM)) CYCLE
   IN => INITIALIZATION(N)
   DO K=0,KBP1
   DO J=0,JBP1
   DO I=0,IBP1
      IF (M%XC(I) > IN%X1 .AND. M%XC(I) < IN%X2 .AND. & M%YC(J) > IN%Y1 .AND. M%YC(J) < IN%Y2 .AND. & M%ZC(K) > IN%Z1 .AND. M%ZC(K) < IN%Z2) THEN
         M%TMP(I,J,K) = IN%TEMPERATURE
         M%RHO(I,J,K) = IN%DENSITY
         IF (N_TRACKED_SPECIES > 0) M%ZZ(I,J,K,1:N_TRACKED_SPECIES) = IN%MASS_FRACTION(1:N_TRACKED_SPECIES)
         IF (IN%ADJUST_DENSITY) M%RHO(I,J,K) = M%RHO(I,J,K) * M%P_0(K) / P_INF
         IF (IN%ADJUST_TEMPERATURE) M%TMP(I,J,K) = M%TMP(I,J,K) * M%P_0(K) / P_INF
         M%Q(I,J,K) = IN%HRRPUV
      ENDIF
   ENDDO
   ENDDO
   ENDDO
ENDDO

! Compute molecular weight term RSUM=R0*SUM( Y_i/M_i )

IF (N_TRACKED_SPECIES > 0 .AND. .NOT.EVACUATION_ONLY(NM)) THEN
DO K=1,KBAR
  DO J=1,JBAR
    DO I=1,IBAR
      ZZ_GET(1:N_TRACKED_SPECIES) = M%ZZ(I,J,K,1:
      N_TRACKED_SPECIES)
      CALL GET_SPECIFIC_GAS_CONSTANT(ZZ_GET,M%RSUM(I,J,K)
      )
    ENDDO
  ENDDO
ENDDO
ENDIF

! Allocate and Initialize Mesh-Dependent Radiation Arrays
M%QR = 0._EB
IF (.NOT.EVACUATION_ONLY(NM)) THEN
  M%KAPPA = KAPPA0
  M%UII = 4._EB*SIGMA*TMPA4
ENDIF
M%ANGLE_INC_COUNTER = 0
M%RAD_CALL_COUNTER = 0
IF (RADIATION .AND. .NOT.EVACUATION_ONLY(NM)) THEN
  ALLOCATE(M/UIID(0:M%IBP1,0:M%JBP1,0:M%KBP1,1:UIIDIM),STAT=
  IZERO)
  CALL ChkMemErr('INIT','UIID',IZERO)
  M/UIID = 0.
ENDIF

! General work arrays
M%WORK1 = 0._EB
M%WORK2 = 0._EB
M%WORK3 = 0._EB
M%WORK4 = 0._EB
M%WORK5 = 0._EB
M%WORK6 = 0._EB
IF (.NOT.EVACUATION_ONLY(NM)) M%WORK7 = 0._EB

! Immersed Boundary Method
IF (IMMERSED_BOUNDARY_METHOD==2) THEN
  M%IBM_SAVE1 = 0._EB
  M%IBM_SAVE2 = 0._EB
ENDIF
APPENDIX B. IMPLEMENTED CODE

192

M%IBM_SAVE3 = 0._EB
M%IBM_SAVE4 = 0._EB
M%IBM_SAVE5 = 0._EB
M%IBM_SAVE6 = 0._EB

ENDIF

IF (IMMERSED_BOUNDARY_METHOD>=0) THEN
  ALLOCATE(M%U_MASK(0:M%IBP1,0:M%JBP1,0:M%KB1),STAT=IZERO)
  CALL ChkMemErr ( 'INIT.ibm', 'U_MASK', IZERO)
  ALLOCATE(M%V_MASK(0:M%IBP1,0:M%JBP1,0:M%KB1),STAT=IZERO)
  CALL ChkMemErr ( 'INIT.ibm', 'V_MASK', IZERO)
  ALLOCATE(M%W_MASK(0:M%IBP1,0:M%JBP1,0:M%KB1),STAT=IZERO)
  CALL ChkMemErr ( 'INIT.ibm', 'W_MASK', IZERO)
  ALLOCATE(M%P_MASK(0:M%IBP1,0:M%JBP1,0:M%KB1),STAT=IZERO)
  CALL ChkMemErr ( 'INIT.ibm', 'P_MASK', IZERO)
  M%U_MASK=1
  M%V_MASK=1
  M%W_MASK=1
  M%P_MASK=1

ENDIF

! Determine the total number of wall cells to allocate

M%N_INTERNAL_WALL CELLS = 0

OBST_LOOP1: DO N=1,M%N_OBST
  OB=OBSTRUCSTION(N)
  IF (OB%CONSUMABLE .AND. .NOT.EVACUATION_ONLY(NM) ) THEN
    IB = OB%I2-OB%I1
    JB = OB%J2-OB%J1
    KB = OB%K2-OB%K1
    M%N_INTERNAL_WALL CELLS = M%N_INTERNAL_WALL CELLS + 2*(
      MAX(1,IB)*JB*KB + MAX(1,JB)*IB*KB + MAX(1,KB)*IB*JB)
  ELSE
    DO K=OB%K1+1,OB%K2
      DO J=OB%J1+1,OB%J2
        IC = M%CELL_INDEX(OB%I1,OB%J1,OB%K1)
        IF (.NOT.M%SOLID(IC) .OR.M%OBSTRUCSTION(M%OBST_INDEX,C(1))%REMOVABLE) M%
          N_INTERNAL_WALL CELLS = M%N_INTERNAL_WALL CELLS +1
        IC = M%CELL_INDEX(OB%I2+1,OB%J2,OB%K2)
        ENDIF
      END DO J
    END DO K
  ENDIF

END
APPENDIX B. IMPLEMENTED CODE

476 IF (.NOT.M%SOLID(IC).OR.M%OBSTRUCTION(M%OBST_INDEX_C(IC))%REMOVABLE) M%
N INTERNAL WALL CELLS = M%N INTERNAL WALL CELLS +1

477 ENDDO
478 ENDDO
479 DO K=OB%K1+1,OB%K2
480 DO I=OB%I1+1,OB%I2
481 IC = M%CELL_INDEX(I,OB%J1,K)
482 IF (.NOT.M%SOLID(IC).OR.M%OBSTRUCTION(M%OBST_INDEX_C(IC))%REMOVABLE) M%
N INTERNAL WALL CELLS = M%N INTERNAL WALL CELLS +1
483 IC = M%CELL_INDEX(I,OB%J2+1,K)
484 IF (.NOT.M%SOLID(IC).OR.M%OBSTRUCTION(M%OBST_INDEX_C(IC))%REMOVABLE) M%
N INTERNAL WALL CELLS = M%N INTERNAL WALL CELLS +1
485 ENDDO
486 ENDDO
487 DO J=OB%J1+1,OB%J2
488 DO I=OB%I1+1,OB%I2
489 IC = M%CELL_INDEX(I,J,OB%K1)
490 IF (.NOT.M%SOLID(IC).OR.M%OBSTRUCTION(M%OBST_INDEX_C(IC))%REMOVABLE) M%
N INTERNAL WALL CELLS = M%N INTERNAL WALL CELLS +1
491 IC = M%CELL_INDEX(I,J,OB%K2+1)
492 IF (.NOT.M%SOLID(IC).OR.M%OBSTRUCTION(M%OBST_INDEX_C(IC))%REMOVABLE) M%
N INTERNAL WALL CELLS = M%N INTERNAL WALL CELLS +1
493 ENDDO
494 ENDDO
495 ENDDO OBST LOOP 1
496 ! Add wall cells for VIRTUAL devices
497 500 M%N_VIRTUAL_WALL CELLS = 0
498 501 DO N=1,N_DEV
499 502 DV => DEVICE(N)
APPENDIX B. IMPLEMENTED CODE

504 IF (DV%MESH/=NM) CYCLE
505 IF (EVACUATION_ONLY(NM)) CYCLE
506 IF (DV%QUANTITY/=’CABLE_TEMPERATURE’) CYCLE
507 M%N_VIRTUAL_WALL CELLS = M%N_VIRTUAL_WALL CELLS + 1
508 ENDO
509
510 ! Compute the number of ghost wall cells (external wall cells outside the computational domain)
511 M%N_GHOST_WALL CELLS = 8*(IBP1+JBP1+KBP1)
512
513 ! Allocate arrays indexed by wall cells (IW). Note the order of the cells in the overall array.
514 N_TOTAL_WALL CELLS = M%N_EXTERNAL_WALL CELLS + M%
515 N_INTERNAL_WALL CELLS + M%N_VIRTUAL_WALL CELLS + M%
516 N_GHOST_WALL CELLS
517
518 ALLOCATE(M%WALL(0:N_TOTAL_WALL CELLS) ,STAT=IZERO)
519 CALL ChkMemErr (’INIT’, ’WALL’, IZERO)
520 M%WALL%RHO_F = RHOA
521 M%WALL%ONE_D%EMISSIVITY = 1. _EB
522 M%WALL%U_TAU = 0. _EB
523
524 NOT_EVAC_IF_1: IF (.NOT.EVACUATION_ONLY(NM)) THEN
525
526 M%WALL%TMP_F = TMPA
527 M%WALL%TMP_B = TMPA
528 DO IW=1,N TOTAL_WALL CELLS
529 ALLOCATE(M%WALL(IW)%ZZ_F (N_TRACKED SPECIES) ,STAT=IZERO)
530 CALL ChkMemErr (’INIT’, ’WALL(IW)%ZZ_F’, IZERO)
531 M%WALL(IW)%ZZ_F (1:N TRACKED SPECIES) = SPECIES_MIXTURE (1:
532 N TRACKED_SPECIES)%ZZ0
533 ENDO
534
535 M%WALL%ONE_D%QRADIN = SIGMA*TMA4
536 M%WALL%ONE_D%QRADOUT = SIGMA*TMA4
537 M%WALL%ONE_D%QCONF = 0. _EB
538 M%WALL%ONE_D%HEAT_TRANS_COEF = 0. _EB
539
540 ALLOCATE(M%D_CORR(M%N_EXTERNAL_WALL CELLS) ,STAT=IZERO)
541 CALL ChkMemErr (’INIT’, ’D_CORR’, IZERO)
542 ALLOCATE(M%DS_CORR(M%N_EXTERNAL_WALL CELLS) ,STAT=IZERO)
543 CALL ChkMemErr (’INIT’, ’DS_CORR’, IZERO)
APPENDIX B. IMPLEMENTED CODE

542 parameter(M%D_CORR = 0., EB)
543 parameter(M%DS_CORR = 0., EB)
544 ALLOCATE(M%UVW_SAVE(M%N_EXTERNAL_WALL CELLS), STAT=IZERO)
545 CALL ChkMemErr('INIT', 'UVW_SAVE', IZERO)
546 M%UVW_SAVE = 0., EB
547
548 ALLOCATE(M%U_GHOST(M%N_EXTERNAL_WALL CELLS), STAT=IZERO)
549 CALL ChkMemErr('INIT', 'U_GHOST', IZERO)
550 ALLOCATE(M%V_GHOST(M%N_EXTERNAL_WALL CELLS), STAT=IZERO)
551 CALL ChkMemErr('INIT', 'V_GHOST', IZERO)
552 ALLOCATE(M%W_GHOST(M%N_EXTERNAL_WALL CELLS), STAT=IZERO)
553 CALL ChkMemErr('INIT', 'W_GHOST', IZERO)
554 M%U_GHOST = 0., EB
555 M%V_GHOST = 0., EB
556 M%W_GHOST = 0., EB
557 ENDIF NOT_EVAC_IF_1
558
559 ! Allocate arrays for turbulent inflow boundary conditions (experimental)
560
561 VENT_LOOP: DO N=1,M%N_VENT
562 VT => M%VENTS(N)
563 EDDY_IF: IF (VT%N_EDDY > 0) THEN
564   SELECT CASE(ABS(VT%IOR))
565   CASE(1)
566     ALLOCATE(VT%U_EDDY(VT%J1+1:VT%J2, VT%K1+1:VT%K2), STAT=IZERO)
567     CALL ChkMemErr('READ_VENT', 'U_EDDY', IZERO)
568     ALLOCATE(VT%V_EDDY(VT%J1+1:VT%J2, VT%K1+1:VT%K2), STAT=IZERO)
569     CALL ChkMemErr('READ_VENT', 'V_EDDY', IZERO)
570     ALLOCATE(VT%W_EDDY(VT%J1+1:VT%J2, VT%K1+1:VT%K2), STAT=IZERO)
571     CALL ChkMemErr('READ_VENT', 'W_EDDY', IZERO)
572   CASE(2)
573     ALLOCATE(VT%U_EDDY(VT%I1+1:VT%I2, VT%K1+1:VT%K2), STAT=IZERO)
574     CALL ChkMemErr('READ_VENT', 'U_EDDY', IZERO)
575     ALLOCATE(VT%V_EDDY(VT%I1+1:VT%I2, VT%K1+1:VT%K2), STAT=IZERO)
576     CALL ChkMemErr('READ_VENT', 'V_EDDY', IZERO)
APPENDIX B. IMPLEMENTED CODE

578  ALLOCATE (VT%W_EDDY (VT%I1 + 1:VT%I2 , VT%K1 + 1:VT%K2) , STAT=IZERO)
579  CALL ChkMemErr ( 'READVENT' , 'W_EDDY' , IZERO)
580  CASE (3)
581  ALLOCATE (VT%U_EDDY (VT%I1 + 1:VT%I2 , VT%J1 + 1:VT%J2) , STAT=IZERO)
582  CALL ChkMemErr ( 'READVENT' , 'U_EDDY' , IZERO)
583  ALLOCATE (VT%V_EDDY (VT%I1 + 1:VT%I2 , VT%J1 + 1:VT%J2) , STAT=IZERO)
584  CALL ChkMemErr ( 'READVENT' , 'V_EDDY' , IZERO)
585  ALLOCATE (VT%W_EDDY (VT%I1 + 1:VT%I2 , VT%J1 + 1:VT%J2) , STAT=IZERO)
586  CALL ChkMemErr ( 'READVENT' , 'W_EDDY' , IZERO)
587  END SELECT
588  ALLOCATE (VT%X_EDDY (VT%N_EDDY) , STAT=IZERO)
589  CALL ChkMemErr ( 'READVENT' , 'X_EDDY' , IZERO)
590  ALLOCATE (VT%Y_EDDY (VT%N_EDDY) , STAT=IZERO)
591  CALL ChkMemErr ( 'READVENT' , 'Y_EDDY' , IZERO)
592  ALLOCATE (VT%Z_EDDY (VT%N_EDDY) , STAT=IZERO)
593  CALL ChkMemErr ( 'READVENT' , 'Z_EDDY' , IZERO)
594  ALLOCATE (VT%CU_EDDY (VT%N_EDDY) , STAT=IZERO)
595  CALL ChkMemErr ( 'READVENT' , 'CU_EDDY' , IZERO)
596  ALLOCATE (VT%CV_EDDY (VT%N_EDDY) , STAT=IZERO)
597  CALL ChkMemErr ( 'READVENT' , 'CV_EDDY' , IZERO)
598  ALLOCATE (VT%CW_EDDY (VT%N_EDDY) , STAT=IZERO)
599  CALL ChkMemErr ( 'READVENT' , 'CW_EDDY' , IZERO)
600  VT%U_EDDY = 0._EB
601  VT%V_EDDY = 0._EB
602  VT%W_EDDY = 0._EB
603  VT%X_EDDY = 0._EB
604  VT%Y_EDDY = 0._EB
605  VT%Z_EDDY = 0._EB
606  VT%CU_EDDY = 0._EB
607  VT%CV_EDDY = 0._EB
608  VT%CW_EDDY = 0._EB
609 ENDIF EDDY_IF
610 ENDDO VENT_LOOP
611
612 M%WALL%IW = T_BEGIN
613
614 NOT_EVAC_IF_2: IF (.NOT.EVACUATION_ONLY(NM)) THEN
615  M%WALL%IW = 0._EB
616  M%WALL%KW = 0._EB
APPENDIX B. IMPLEMENTED CODE

617 DO IW=1,N_TOT_WALL CELLS
618   ALLOCATE(M%WALL(IW)%RHODW(N_TOTAL_SPECIES),STAT=IZERO)
619   CALL ChkMemErr(‘INIT’,’WALL(IW)%RHODW’,IZERO)
620   M%WALL(IW)%RHODW = 0.1_EB ! Do not initialize to zero to avoid divide by zero in the first time step
621 ENDDO
622 M%WALL%AREA_ADJUST = 1._EB
623 DO IW=1,N_TOT_WALL CELLS
624   ALLOCATE(M%WALL(IW)%ONE_D%MASSFLUX(0:N_TOTAL_SPECIES),STAT=IZERO)
625   CALL ChkMemErr(‘INIT’,’WALL(IW)%ONE_D%MASSFLUX’,IZERO)
626   M%WALL(IW)%ONE_D%MASSFLUX = 0._EB
627   ALLOCATE(M%WALL(IW)%ONE_D%MASSFLUX_ACTUAL(0:N_TOTAL_SPECIES),STAT=IZERO)
628   CALL ChkMemErr(‘INIT’,’WALL(IW)%ONE_D%MASSFLUX_ACTUAL’,IZERO)
629   M%WALL(IW)%ONE_D%MASSFLUX_ACTUAL = 0._EB
630 ENDDO
631 M%WALL%NPFW = 1
632 M%WALL%BACK_INDEX = 0
633 M%WALL%AW = 0._EB
634 M%WALL%RAW = 0._EB
635 ENDF IF NOT_EVAC_IF_2
636
637 M%WALL%RDN = 1._EB
638 M%WALL%UW0 = 0._EB
639 M%WALL%UW = 0._EB
640 M%WALL%UWS = 0._EB
641 M%WALL%OBST_INDEX = 0
642 M%WALL%VENT_INDEX = 0
643 ALLOCATE(M%DUWT(N_TOT_WALL CELLS),STAT=IZERO)
644 CALL ChkMemErr(‘INIT’,’DUWT’,IZERO)
645 M%DUWT = 0._EB
646 ALLOCATE(M%PRESSURE_BC_INDEX(0:M%N_EXTERNAL_WALL CELLS),STAT=IZERO)
647 CALL ChkMemErr(‘INIT’,’PRESSURE_BC_INDEX’,IZERO)
648 M%PRESSURE_BC_INDEX = NEUMANN
649 M%WALL%PRESSURE_BC_INDEX = NEUMANN
650 M%WALL%SURF_INDEX_ORIG = 0
651 M%WALL%BOUNDARY_TYPE = NULL
652 ALLOCATE(M%WALL_INDEX(0:M%CELL_COUNT,-3:3),STAT=IZERO)
653 CALL ChkMemErr(‘INIT’,’WALL_INDEX’,IZERO)
654 M%WALL_INDEX = 0
APPENDIX B. IMPLEMENTED CODE

ALLOCATE(M$EDGE_INDEX(0:M$CELL_COUNT,1:12),STAT=IZERO)
CALL ChkMemErr('INIT','EDGE_INDEX',IZERO)
M$EDGE_INDEX = 0
ALLOCATE(M$UVW_GHOST(0:M$CELL_COUNT,3),STAT=IZERO)
CALL ChkMemErr('INIT','UVW_GHOST',IZERO)
M$UVW_GHOST = 0

! Surface soot array

IF (N$SURFACE_DENSITY_SPECIES > 0 .AND. .NOT.EVACUATION_ONLY(NM)) THEN
ALLOCATE(M$AWM_AEROSOL(N_TOTAL_WALL CELLS,
   N$SURFACE_DENSITY_SPECIES),STAT=IZERO)
CALL ChkMemErr('INIT','AWM_AEROSOL',IZERO)
M$AWM_AEROSOL = 0._EB
DO IW=1,N_TOTAL_WALL CELLS
   ALLOCATE(M$WALL(IW)%AWM_AEROSOL(N$SURFACE_DENSITY_SPECIES),STAT=IZERO)
   CALL ChkMemErr('INIT','WALL(IW)%AWM_AEROSOL',IZERO)
   M$WALL(IW)%AWM_AEROSOL = 0._EB
ENDDO
ENDIF

! Surface water arrays

IF (ACCUMULATE_WATER .AND. .NOT.EVACUATION_ONLY(NM)) THEN
   DO IW = 1,N_TOTAL_WALL CELLS
      ALLOCATE(M$WALL(IW)%L_P_MPUA(N_L_P_ARRAY_INDICES),STAT=IZERO)
      CALL ChkMemErr('INIT','WALL(IW)%L_P_MPUA',IZERO)
      M$WALL(IW)%L_P_MPUA = 0._EB
   ENDDO
ENDIF

IF (.NOT.EVACUATION_ONLY(NM)) THEN
   DO IW = 1,N_TOTAL_WALL CELLS
      ALLOCATE(M$WALL(IW)%L_P_CPUA(N_L_P_ARRAY_INDICES),STAT=IZERO)
      CALL ChkMemErr('INIT','WALL(IW)%L_P_CPUA',IZERO)
      M$WALL(IW)%L_P_CPUA = 0._EB
   ENDDO
CALL ChkMemErr ( 'INIT', 'WALL(IW)%LP_CPUA' , IZERO )
M_WALL(IW)%LP_CPUA = 0 ._EB
ENDO
ENDIF

! Surface work arrays
ALLOCATE(M_WALL_WORK1(N_TOTAL_WALL.Cells) ,STAT=IZERO)
CALL ChkMemErr ( 'INIT', 'WALL_WORK1' , IZERO )
ALLOCATE(M_WALL_WORK2(N_TOTAL_WALL.Cells) ,STAT=IZERO)
CALL ChkMemErr ( 'INIT', 'WALL_WORK2' , IZERO )

! Vegetation surface drag
ALLOCATE(M_VEGRAG(0 : IBP1 , 0 : JBP1) ,STAT=IZERO)
CALL ChkMemErr ( 'INIT', 'VEG_DRAG' , IZERO )
M_VEGRAG = 0 ._EB

! Set up boundary arrays for external boundaries of the current
mesh
IWE = 0
IWG = M_N_EXTERNAL_WALL.Cells + M_N_INTERNAL_WALL.Cells + M_%
N_VIRTUAL_WALL.Cells

DO K=0,KBP1
  DO J=0,JBP1
    I = 0
    SURF_INDEX = DEFAULT_SURF_INDEX
    IOR = 1
    IF (J==0 .OR. J==JBP1 .OR. K==0 .OR. K==KBP1) THEN
      IWG = IWG + 1
      IW = IWG
    ELSE
      IWE = IWE + 1
      IW = IWE
    ENDIF
  CALL INIT_WALL_CELL(NM, I, J, K, 0 ,IW, IOR, SURF_INDEX, IERR)
  IF (IERR>0) RETURN
ENDO
DO K=0,KBP1
  DO J=0,JBP1
    I = IBP1
APPENDIX B. IMPLEMENTED CODE

733  SURF_INDEX = DEFAULT_SURF_INDEX
734  IOR = −1
735  IF (J==0 . OR. J==JBP1 . OR. K==0 . OR. K==KBP1) THEN
736      IWG = IWG + 1
737      IW = IWG
738  ELSE
739      IWE = IWE + 1
740      IW = IWE
741  ENDIF
742  CALL INIT_WALL_CELL(NM, I, J, K, 0, IW, IOR, SURF_INDEX, IERR)
743  IF (IERR>0) RETURN
744  ENDDO
745  ENDDO
746
747  DO K=0,KBP1
748      DO I=0,IBP1
749          J = 0
750          SURF_INDEX = DEFAULT_SURF_INDEX
751          IOR = 2
752          IF (I==0 . OR. I==IBP1 . OR. K==0 . OR. K==KBP1) THEN
753              IWG = IWG + 1
754              IW = IWG
755          ELSE
756              IWE = IWE + 1
757              IW = IWE
758          ENDIF
759          CALL INIT_WALL_CELL(NM, I, J, K, 0, IW, IOR, SURF_INDEX, IERR)
760      IF (IERR>0) RETURN
761  ENDDO
762  ENDDO
763  DO K=0,KBP1
764      DO I=0,IBP1
765          J = JBP1
766          SURF_INDEX = DEFAULT_SURF_INDEX
767          IOR = −2
768          IF (I==0 . OR. I==IBP1 . OR. K==0 . OR. K==KBP1) THEN
769              IWG = IWG + 1
770              IW = IWG
771          ELSE
772              IWE = IWE + 1
773              IW = IWE
774          ENDIF
775      CALL INIT_WALL_CELL(NM, I, J, K, 0, IW, IOR, SURF_INDEX, IERR)
APPENDIX B. IMPLEMENTED CODE

776 IF (IERR>0) RETURN
777 ENDDO
778 ENDDO
779
780 IF (.NOT.EVACUATION_ONLY(NM)) THEN
781 DO J=0,JBP1
782 DO I=0,IBP1
783 K = 0
784 SURF_INDEX = DEFAULT_SURF_INDEX
785 IOR = 3
786 IF (I==0 .OR. I==IBP1 .OR. J==0 .OR. J==JBP1) THEN
787 IWG = IWG + 1
788 IW = IWG
789 ELSE
790 IWE = IWE + 1
791 IW = IWE
792 ENDIF
793 CALL INIT_WALL_CELL(NM, I, K, 0, IW, IOR, SURF_INDEX, IERR)
794 IF (IERR>0) RETURN
795 ENDDO
796 ENDDO
797 DO J=0,JBP1
798 DO I=0,IBP1
799 K = KBP1
800 SURF_INDEX = DEFAULT_SURF_INDEX
801 IOR = -3
802 IF (I==0 .OR. I==IBP1 .OR. J==0 .OR. J==JBP1) THEN
803 IWG = IWG + 1
804 IW = IWG
805 ELSE
806 IWE = IWE + 1
807 IW = IWE
808 ENDIF
809 CALL INIT_WALL_CELL(NM, I, K, 0, IW, IOR, SURF_INDEX, IERR)
810 IF (IERR>0) RETURN
811 ENDDO
812 ENDDO
813 ENDIF
814
815 ! Go through all obstructions and decide which cell faces ought to be given a wall cell index and initialized

816 M%N_INTERNAL_WALL.Cells = 0
APPENDIX B. IMPLEMENTED CODE

818
819 OBST_LOOP_2: DO \( N = 1 \), \( M \)\%OBST
820 OB\%=M\%OBSTRUCTION(\( N \))
821
822 DO \( K = \text{OB}\%K1 + 1, \text{OB}\%K2 \)
823 \( \text{DO} \ J = \text{OB}\%J1 + 1, \text{OB}\%J2 \)
824 \( I = \text{OB}\%I1 + 1 \)
825 IF (\( I == 1 \)) CYCLE ! Don’t assign wall cell index to obstruction face pointing out of the computational domain
826 IC = \( M\%\text{CELL_INDEX}(I-1,J,K) \)
827 IF (\( M\%\text{SOLID}(IC) \).AND. .NOT.\( M\%\text{OBSTRUCTION}(M\%\text{OBST_INDEX,C(IC)}\)%REMOVABLE) CYCLE ! Permanently covered face
828 IOR = \(-1 \)
829 SURF_INDEX = \( \text{OB}\%\text{SURF_INDEX}(\text{IOR}) \)
830 IW = \( M\%\text{WALL_INDEX}(IC,-\text{IOR}) \)
831 IF (\( IW == 0 \)) THEN
832 \( M\%\text{INTERNAL_WALL_CELLS} = M\%\text{INTERNAL_WALL_CELLS} + 1 \)
833 IW = \( M\%\text{EXTERNAL_WALL_CELLS} + M\%\text{INTERNAL_WALL_CELLS} \)
834 ENDIF
835 CALL INIT_WALL_CELL\((\text{NM, I, J, K, N, IW, IOR, SURF_INDEX, IERR})\)
836 IF (\( \text{IERR} > 0 \)) RETURN
837 ENDDO
838 ENDDO
839
840 DO \( K = \text{OB}\%K1 + 1, \text{OB}\%K2 \)
841 \( \text{DO} \ J = \text{OB}\%J1 + 1, \text{OB}\%J2 \)
842 \( I = \text{OB}\%I2 \)
843 IF (\( I == \text{M}\%\text{IBAR} \)) CYCLE ! Don’t assign wall cell index to obstruction face pointing out of the computational domain
844 IC = \( M\%\text{CELL_INDEX}(I+1,J,K) \)
845 IF (\( M\%\text{SOLID}(IC) \).AND. .NOT.\( M\%\text{OBSTRUCTION}(M\%\text{OBST_INDEX,C(IC)}\)%REMOVABLE) CYCLE ! Permanently covered face
846 IOR = \( 1 \)
847 SURF_INDEX = \( \text{OB}\%\text{SURF_INDEX}(\text{IOR}) \)
848 IW = \( M\%\text{WALL_INDEX}(IC,-\text{IOR}) \)
849 IF (\( IW == 0 \)) THEN
APPENDIX B. IMPLEMENTED CODE

\begin{verbatim}
M\%N\_INTERNAL\_WALL\_CELLS = M\%N\_INTERNAL\_WALL\_CELLS + 1
IW = M\%N\_EXTERNAL\_WALL\_CELLS + M\%
N\_INTERNAL\_WALL\_CELLS
ENDIF
CALL INIT\_WALL\_CELL(NM, I, J, K, N, IW, IOR, SURF\_INDEX, IERR)
IF (IERR>0) RETURN
ENDO

DO K=OB%K1+1,OB%K2
   DO I=OB%I1+1,OB%I2
      J = OB%J1+1
      IF (J==1) CYCLE ! Don’t assign wall cell index to obstruction face pointing out of the computational domain
      IC = M\%CELL\_INDEX(I, J-1,K)
      IF (M\%SOLID(IC) .AND. .NOT.M\%OBSTRUCTION(M\%OBST\_INDEX\_C(IC))%REMOVABLE) CYCLE ! Permanently covered face
      IOR = -2
      SURF\_INDEX = OB\%SURF\_INDEX(IOR)
      IW = M\%WALL\_INDEX(IC,-IOR)
      IF (IW==0)
         M\%N\_INTERNAL\_WALL\_CELLS = M\%N\_INTERNAL\_WALL\_CELLS + 1
         IW = M\%N\_EXTERNAL\_WALL\_CELLS + M\%
         N\_INTERNAL\_WALL\_CELLS
      ENDIF
      CALL INIT\_WALL\_CELL(NM, I, J, K, N, IW, IOR, SURF\_INDEX, IERR)
      IF (IERR>0) RETURN
   ENDDO
   DO K=OB%K1+1,OB%K2
      DO I=OB%I1+1,OB%I2
         J = OB%J1+2
         IF (J==M\%JBAR) CYCLE ! Don’t assign wall cell index to obstruction face pointing out of the computational domain
         IC = M\%CELL\_INDEX(I, J+1,K)
         IF (M\%SOLID(IC) .AND. .NOT.M\%OBSTRUCTION(M\%OBST\_INDEX\_C(IC))%REMOVABLE) CYCLE ! Permanently
\end{verbatim}
covered face

\[ \text{IOR} = 2 \]

\[ \text{SURF}_\text{INDEX} = \text{OB}_\text{SURF}_\text{INDEX}(\text{IOR}) \]

\[ \text{IW} = \text{M}_\text{WALL}_\text{INDEX}(\text{IC}, -\text{IOR}) \]

\[ \text{IF} \ (\text{IW}==0) \ \text{THEN} \]

\[ \text{M}_\text{N}_\text{INTERNAL}_\text{WALL}_\text{CELLS} = \text{M}_\text{N}_\text{INTERNAL}_\text{WALL}_\text{CELLS} + 1 \]

\[ \text{IW} = \text{M}_\text{N}_\text{EXTERNAL}_\text{WALL}_\text{CELLS} + \text{M}_\text{N}_\text{INTERNAL}_\text{WALL}_\text{CELLS} \]

\[ \text{ENDIF} \]

\[ \text{CALL INIT}_\text{WALL}_\text{CELL(}\text{NM}, \text{I} , \text{J} , \text{K} , \text{N} , \text{IW}, \text{IOR}, \text{SURF}_\text{INDEX}, \text{IERR}) \]

\[ \text{IF} \ (\text{IERR}>0) \ \text{RETURN} \]

\[ \text{ENDDO} \]

\[ \text{ENDDO} \]

\[ \text{DO} \ J=\text{OB}_\text{J1} + 1, \text{OB}_\text{J2} \]

\[ \text{DO} \ I=\text{OB}_\text{I1} + 1, \text{OB}_\text{I2} \]

\[ \text{K} = \text{OB}_\text{K1} + 1 \]

\[ \text{IF} \ (\text{K}==1) \ \text{CYCLE} \quad \text{! Don’t assign wall cell index to obstruction face pointing out of the computational domain} \]

\[ \text{IC} = \text{M}_\text{N}_\text{CELL}_\text{INDEX}(\text{I} , \text{J} , \text{K}-1) \]

\[ \text{IF} \ (\text{M}_\text{SOLID}(\text{IC}) \ \text{AND} \ .\text{NOT.} \text{M}_\text{OBSTRUCTION(M}_\text{N}_\text{OBST}_\text{INDEX.C}(\text{IC}))%\text{REMOVABLE}) \ \text{CYCLE} \quad \text{! Permanently covered face} \]

\[ \text{IOR} = -3 \]

\[ \text{SURF}_\text{INDEX} = \text{OB}_\text{SURF}_\text{INDEX}(\text{IOR}) \]

\[ \text{IW} = \text{M}_\text{WALL}_\text{INDEX}(\text{IC}, -\text{IOR}) \]

\[ \text{IF} \ (\text{IW}==0) \ \text{THEN} \]

\[ \text{M}_\text{N}_\text{INTERNAL}_\text{WALL}_\text{CELLS} = \text{M}_\text{N}_\text{INTERNAL}_\text{WALL}_\text{CELLS} + 1 \]

\[ \text{IW} = \text{M}_\text{N}_\text{EXTERNAL}_\text{WALL}_\text{CELLS} + \text{M}_\text{N}_\text{INTERNAL}_\text{WALL}_\text{CELLS} \]

\[ \text{ENDIF} \]

\[ \text{CALL INIT}_\text{WALL}_\text{CELL(}\text{NM}, \text{I} , \text{J} , \text{K} , \text{N} , \text{IW}, \text{IOR}, \text{SURF}_\text{INDEX}, \text{IERR}) \]

\[ \text{IF} \ (\text{IERR}>0) \ \text{RETURN} \]

\[ \text{ENDDO} \]

\[ \text{ENDDO} \]

\[ \text{DO} \ J=\text{OB}_\text{J1} + 1, \text{OB}_\text{J2} \]

\[ \text{DO} \ I=\text{OB}_\text{I1} + 1, \text{OB}_\text{I2} \]

\[ \text{K} = \text{OB}_\text{K2} \]
APPENDIX B. IMPLEMENTED CODE

915 IF (K=M%KBAR) CYCLE ! Don't assign wall cell index to obstruction face pointing out of the computational domain

916 IC = M%CELL_INDEX(1, J, K+1)

917 IF (M%SOLID(IC) .AND. .NOT.M%OBSTRUCTION(M%OBST_INDEX,C(1IC))%REMOVABLE) CYCLE ! Permanently covered face

918 IOR = 3

919 SURF_INDEX = OB%SURF_INDEX(IOR)

920 IW = M%WALL_INDEX(1C, -IOR)

921 IF (IW==0) THEN

922 M%N_INTERNAL_WALL.Cells = M%N_INTERNAL_WALL.Cells + 1

923 IW = M%N_EXTERNAL_WALL.Cells + M%N_INTERNAL_WALL.Cells

924 ENDF

925 CALL INIT_WALL_CELL(NM, I, J, K, N, IW, IOR, SURF_INDEX, IERR)

926 IF (IERR>0) RETURN

927 ENDDO

928 ENDDO

929 ENDDO OBST_LOOP

930 ! Initialize PSUM for zone cases

931 IF (N_ZONE > 0) THEN

932 N_ZONE_LOOP: DO IPZ = 1, N_ZONE

933 PSUM(IPZ, NM) = 0. * EB

934 IF (EVACUATIONONLY(NM)) EXIT N_ZONE_LOOP

935 DO K=1, M%KBAR

936 DO J=1, M%JBAR

937 DO I=1, M%IBAR

938 IF (M%PRESSURE_ZONE(I, J, K) /= IPZ) CYCLE

939 IF (M%SOLID(M%CELL_INDEX(I, J, K))) CYCLE

940 VC = M%DX(I) * M%RC(I) + M%DY(J) * M%DZ(K)

941 IF (N_TRACKED_SPECIES>0) ZZ_GET(1: N_TRACKED_SPECIES) = M%ZZ(I, J, K, 1: N_TRACKED_SPECIES)

942 CALL GET_SPECIFIC_HEAT(ZZ_GET, CP, M%TMP(I, J, K))

943 RTRM = M%RSUM(I, J, K)/(CP*M%PBAR(K, IPZ))

944 PSUM(IPZ, NM) = PSUM(IPZ, NM) + VC(1. * EB/M%PBAR(K, IPZ) - RTRM)

945 ENDDO

946 ENDDO
enddo
enddo
enddo n_zone_loop
endif
!
Set up wall cell arrays for virtual boundaries
iw = m%external_wall_cells + m%internal_wall_cells
device_loop: do n=1,n_dev
   if (evacuation_only(nm)) cycle device_loop
   dv => device(n)
   if (dv%mesh/=nm) cycle device_loop
   if (dv%quantity/=’cable_temperature’) cycle device_loop
   iw = iw + 1
   dv%virtual_wall_index = iw
   i = dv%i
   j = dv%j
   k = dv%k
   surf_index = dv%surf_index
   call init_wall_cell(nm, i, j, k, 0, iw, 0, surf_index, ierr)
endo device_loop
!
Determine back wall index for exposed surfaces
!
do iw=m%external_wall_cells+1,m%external_wall_cells+m%
   n_internal_wall_cells
   if (evacuation_only(nm)) cycle
   ! only assign back_index to wall cells that are not attached
to the exterior boundary of the computational domain
   sf => surface(m%wall(iw)%surf_index)
   if (sf%backing==exposed) then
      ii = m%wall(iw)%ii
      jj = m%wall(iw)%jj
      kk = m%wall(iw)%kk
      ic = m%cell_index(ii, jj, kk)
      ior = m%wall(iw)%ior
      if (.not.m%solid(ic)) m%wall(iw)%back_index = m%
         wall_index(ic, ior)
      if (m%solid(ic)) then
         select case(ior)
            case(-1)
            ii=ii+1
APPENDIX B. IMPLEMENTED CODE

CASE (1)
    II = II - 1
CASE (2)
    JJ = JJ + 1
CASE (3)
    KK = KK - 1
CASE (4)
    KK = KK + 1
END SELECT

IC = M\%CELL\_INDEX (II, JJ, KK)
M\%WALL\_INDEX = M\%WALL\_INDEX (IC, IOR)
ENDIF
ENDIF
ENDDO

! Set clocks and counters related to frequency of solid phase conduction updates

M\%BC\_CLOCK = T\_BEGIN
M\%WALL\_COUNTER = 0

! Set clock for boundary fuel vegetation model

M\%VEG\_CLOCK\_BC = T\_BEGIN

! Allocate arrays for storing velocity boundary condition info

N\_EDGES\_DIM = 4\*(IBP1*JBP1+IBP1*KBP1+JBP1*KBP1)

IF (EVACUATION\_ONLY(NM)) N\_EDGES\_DIM = 4\*(IBP1*KBP1+JBP1*KBP1)
DO N=1,M\%OBST
    OB = M\%OBSTRUCTION(N)
    IPTS = OB\%I2-OB\%I1
    JPTS = OB\%J2-OB\%J1
    KPTS = OB\%K2-OB\%K1
    IF (EVACUATION\_ONLY(NM)) THEN
        N\_EDGES\_DIM = N\_EDGES\_DIM + 4\*(IPTS*KPTS+JPTS*KPTS)
    ELSE
        N\_EDGES\_DIM = N\_EDGES\_DIM + 4\*(IPTS*JPTS+IPTS*KPTS+JPTS*KPTS)
    ENDIF
ENDDO
APPENDIX B. IMPLEMENTED CODE

1030
1031 ALLOCATE(M%IJKE(16,N_EDGES_DIM),STAT=IZERO)
1032 CALL ChkMemErr(‘INIT’,’IJKE’,IZERO)
1033 M%IJKE = 0
1034 ALLOCATE(M%OME_E(0:N_EDGES_DIM,-2:2),STAT=IZERO)
1035 CALL ChkMemErr(‘INIT’,’OME_E’,IZERO)
1036 M%OME_E = 0._EB
1037 ALLOCATE(M%TAU_E(0:N_EDGES_DIM,-2:2),STAT=IZERO)
1038 CALL ChkMemErr(‘INIT’,’TAU_E’,IZERO)
1039 M%TAU_E = 0._EB
1040 ALLOCATE(M%EDGE_TYPE(N_EDGES_DIM,2),STAT=IZERO)
1041 CALL ChkMemErr(‘INIT’,’EDGE_TYPE’,IZERO)
1042 M%EDGE_TYPE = SOLID_EDGE
1043 ALLOCATE(M%EDGE_INTERPOLATION_FACTOR(N_EDGES_DIM,2),STAT=IZERO)
1044 CALL ChkMemErr(‘INIT’,’EDGE_INTERPOLATION_FACTOR’,IZERO)
1045 M%EDGE_INTERPOLATION_FACTOR = 1._EB
1046 ! Initialize and allocate lagrangian particle/PARTICLE arrays
1047 1048 M%NLP = 0
1049 M%NLPDIM = 1000
1050 IF (PARTICLE_FILE .AND. .NOT.EVACUATION_ONLY(NM)) THEN
1051 ALLOCATE(M%LAGRANGIAN_PARTICLE(M%NLPDIM),STAT=IZERO)
1052 ENDIF
1053 CALL ChkMemErr(‘INIT’,’PARTICLE’,IZERO)
1054 1055 ! Allocate array to hold character strings for Smokeview file
1056 1057 M%N_STRINGS = 0
1058 M%N_STRINGS_MAX = 100
1059 ALLOCATE(M%STRING(M%N_STRINGS_MAX),STAT=IZERO)
1060 CALL ChkMemErr(‘INIT’,’STRING’,IZERO)
1061 ! Set up arrays to hold velocity boundary condition info
1062 1063 CALL INITIALIZE_EDGES
1064 1065 ! Initialize Pressure solver
1066 1067 CALL INITIALIZE_POISSON_SOLVER
1068 IF (IERR/=0) RETURN
1069
! Determine which wall cells to assign for solid phase
thermocouples and profiles

CALL INITIALIZE_DEV
IF (IERR/=0) RETURN

CALL INITIALIZE_PROF
IF (IERR/=0) RETURN

! Initialize Mesh Exchange
CALL INITIALIZE_INTERPOLATION

CONTAINS
Appendix C

Flame Height Calculation, post-prosessing

```fortran
program compute_flame_height

character(30) :: infile (16,3)
real :: z(0:200), hrrpul(200), height (16,3), qstar(16), diameter, sumold
integer :: i, n, npts

diameter = 1.13 ! Equivalent diameter of 1 m² square

qstar(1) = 0.1
qstar(2) = 0.2
qstar(3) = 0.5
qstar(4) = 1.0
qstar(5) = 2.0
qstar(6) = 5.0
qstar(7) = 10.
qstar(8) = 20.
qstar(9) = 50.
qstar(10) = 100.
qstar(11) = 200.
qstar(12) = 500.
qstar(13) = 1000.
qstar(14) = 2000.
qstar(15) = 5000.
qstar(16) = 10000.

infile (1,1) = 'Qs=p1_RI=05_fds2ascii.csv'
```

210
APPENDIX C. FLAME HEIGHT CALCULATION, POST-PROSESSING

27 infile (2,1) = 'Qs=p2_RI=05_fds2ascii.csv'
28 infile (3,1) = 'Qs=p5_RI=05_fds2ascii.csv'
29 infile (4,1) = 'Qs=1_RI=05_fds2ascii.csv'
30 infile (5,1) = 'Qs=2_RI=05_fds2ascii.csv'
31 infile (6,1) = 'Qs=5_RI=05_fds2ascii.csv'
32 infile (7,1) = 'Qs=10_RI=05_fds2ascii.csv'
33 infile (8,1) = 'Qs=20_RI=05_fds2ascii.csv'
34 infile (9,1) = 'Qs=50_RI=05_fds2ascii.csv'
35 infile (10,1) = 'Qs=100_RI=05_fds2ascii.csv'
36 infile (11,1) = 'Qs=200_RI=05_fds2ascii.csv'
37 infile (12,1) = 'Qs=500_RI=05_fds2ascii.csv'
38 infile (13,1) = 'Qs=1000_RI=05_fds2ascii.csv'
39 infile (14,1) = 'Qs=2000_RI=05_fds2ascii.csv'
40 infile (15,1) = 'Qs=5000_RI=05_fds2ascii.csv'
41 infile (16,1) = 'Qs=10000_RI=05_fds2ascii.csv'
42
43 infile (1,2) = 'Qs=p1_fds2ascii.csv'
44 infile (2,2) = 'Qs=p2_fds2ascii.csv'
45 infile (3,2) = 'Qs=p5_fds2ascii.csv'
46 infile (4,2) = 'Qs=1_fds2ascii.csv'
47 infile (5,2) = 'Qs=2_fds2ascii.csv'
48 infile (6,2) = 'Qs=5_fds2ascii.csv'
49 infile (7,2) = 'Qs=10_fds2ascii.csv'
50 infile (8,2) = 'Qs=20_fds2ascii.csv'
51 infile (9,2) = 'Qs=50_fds2ascii.csv'
52 infile (10,2) = 'Qs=100_fds2ascii.csv'
53 infile (11,2) = 'Qs=200_fds2ascii.csv'
54 infile (12,2) = 'Qs=500_fds2ascii.csv'
55 infile (13,2) = 'Qs=1000_fds2ascii.csv'
56 infile (14,2) = 'Qs=2000_fds2ascii.csv'
57 infile (15,2) = 'Qs=5000_fds2ascii.csv'
58 infile (16,2) = 'Qs=10000_fds2ascii.csv'
59
60 infile (1,3) = 'Qs=p1_RI=20_fds2ascii.csv'
61 infile (2,3) = 'Qs=p2_RI=20_fds2ascii.csv'
62 infile (3,3) = 'Qs=p5_RI=20_fds2ascii.csv'
63 infile (4,3) = 'Qs=1_RI=20_fds2ascii.csv'
64 infile (5,3) = 'Qs=2_RI=20_fds2ascii.csv'
65 infile (6,3) = 'Qs=5_RI=20_fds2ascii.csv'
66 infile (7,3) = 'Qs=10_RI=20_fds2ascii.csv'
67 infile (8,3) = 'Qs=20_RI=20_fds2ascii.csv'
68 infile (9,3) = 'Qs=50_RI=20_fds2ascii.csv'
69 infile (10,3) = 'Qs=100_RI=20_fds2ascii.csv'
APPENDIX C. FLAME HEIGHT CALCULATION, POST-PROSEESSING

70 `in_file(11,3) = 'Qs=200,RI=20,fds2ascii.csv'`
71 `in_file(12,3) = 'Qs=500,RI=20,fds2ascii.csv'`
72 `in_file(13,3) = 'Qs=1000,RI=20,fds2ascii.csv'`
73 `in_file(14,3) = 'Qs=2000,RI=20,fds2ascii.csv'`
74 `in_file(15,3) = 'Qs=5000,RI=20,fds2ascii.csv'`
75 `in_file(16,3) = 'Qs=10000,RI=20,fds2ascii.csv'`
76
77 write(6,"(a)") "Q*,L/D_{\omega}(RI=5),L/D_{\omega}(RI=10),L/D_{\omega}(RI=20)"
78
79 file_loop: do n=1,16
80    resolution_loop: do i=1,3
81        if (i==1) npts=39
82        if (i==2) npts=76
83        if (i==3) npts=151
84
85 open(10, file=in_file(n,i), form='formatted', status='old')
86 read(10,*)
87 read(10,*)
88 z = 0.
89 z(0) = 0.
90 do k=1,npts
91    read(10,*,end=20) z(k), hrrpul(k)
92 enddo
93 20 continue
94 do k=1,npts
95    sum = 0.
96    do k=1,npts
97      sum = sum + hrrpul(k)*(z(k)-z(k-1))
98    enddo
99 sum1 = 0.
100 do k=1,npts
101    sum1 = sum1 + hrrpul(k)
102    sumold = sum1
103    sum1 = sum1 + hrrpul(k)
104    if (sum1/sum>0.99) then
105      height(n,i) = z(k-1) + (z(k)-z(k-1))*(0.99*sum-
106        sumold)/(sum1-sumold)
107    exit
108 endif
109 enddo
APPENDIX C. FLAME HEIGHT CALCULATION, POST-PROCESSING

112     enddo resolution_loop
113
114     write(6,“(f8.1,’,’f8.2,’’,f8.2,’’,f8.2)”) qstar(n),height
115         (n,1)/diameter, height(n,2)/diameter, height(n,3)/diameter
116     close(10)
117
118     enddo file_loop
119
120 end program