Modeling Non-premixed Turbulent Combustion in Industrial Rotary Kiln

Application in OpenFOAM

by

Dimas Ramadhan Abdillah Fikri

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on March 29, 2017.

Student number:	4501403	
Project duration:	September 1, 2016 – March 29, 2017	
Thesis committee:	Prof. dr. Ir. C. Kleijn,	CE TU Delft
	Dr. D. J. P. Lahaye,	EEMCS TU Delft, supervisor
	Dr. L. Portela,	CE TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft Delft University of Technology

ALMATIS cfMesh lpdp Lembaga Pengelola Dana Pendidikan

Preface

It is an honor for me to be able to gain knowledge and experience in one of the most prestigious university in the world. Being involved in this master thesis project is a precious opportunity to learn and deal with the application of various lectures given by the university in the previous quarters. This project involves ups and downs of tackling choking point in the setup of computational simulation. It is impossible for me to finish this project without kind support from people around me.

Therefore, I would like to thank Dr. Domenico Lahaye for the opportunity, support, and openness. I had no idea that a small conversation in a Christmas dinner can lead to a year long project and firm bond of partnership. Thank you for always be there when I face even a very small problem in the project. Also, thank you for the support with your wide range of connection that made the project is valuable by validation with published data.

I also want to appreciate the Assessment Committee, Prof. Dr. Ir. Chris Kleijn, for your time, support and advice in this project; Dr. Luis Portela, for the discussion during the class, constructive chats, and promise for the future projects. The project will never meet its end without your involvement.

The project will never be available if Almatis B.V. never thought a need of computational fluid dynamics for their rotary kiln case. Thank you, Mr. Rudi Sadi, Stefan Kuiper, and Almatis B.V. staffs for the opportunity that allows me to investigate your rotary kiln.

This project will only become a brief summary of work by a student without any valuable information if there is no support from various third-parties involved in the project. I would like to appreciate the help from Dr. Marco Talice in the buoyant flow case validation; dr. Franjo Juretic, CEO of cfMesh, for the provided mesh and advice on case configuration; Mohamed El Abassi, our group's Ph.D. candidate, for the advise, mesh, and the escape rope from the 3D mesh geometrical error; also Ali Kadar and Michelle Pisaroni for the suggestions and reference of the previous project.

I will never come to TU Delft if not by the fund support from Indonesian Endowment Fund for Education, Lembaga Pengelola Dana Pendidikan (LPDP). Your dream is high, to see the bright future of Indonesia from its experienced and well-educated youth. I will fulfill my promise, go back to Indonesia, and contribute my knowledge for the sake of Indonesia.

Lastly, I want to thank my mental support during the project. My family, Dad, and sister, for their smile that strengthens me, to keep my chin up during the simulation's dead-end; my future wife, Hajar Prastyani Hapsari, who keeps reminding me to take care of my health, eat properly, and for the late night calls from 16000 km away; friends from Indonesian Students Association (PPI) Delft, that always make me remember that I am not fighting alone; and friends from Chemical Engineering Department, we will finish this!

There are a lot of room for improvement from this project. Many unfinished problems which still need to be resolved in future projects. However, I hope this thesis project can contribute to better understanding of the application of OpenFOAM in industrial cases, especially in the turbulent combustion in a rotary kiln.

Wherever you are, always give your best for your people, nation, country, and the world.

Dimas Ramadhan Abdillah Fikri
Delft, March 2017

Contents

1 Introduction

1.1. Rotary Kiln

Rotary kilns are industrial device used in drying, combustion, and metallurgical processing [21]. This device features long horizontal cylinder shell with slight inclination towards its solid material outlet and can rotate in its axis. Solid material enters from one end of the kiln and processed product leave at the other end. Most common rotary kilns have counter-current configuration which allows combustion gas enters the kiln through burner from the opposite end of solid material inlet. Inside the kiln, solid material is gradually moving due to the kiln inclination angle and rotation movement. During the process, the solid material is calcinated in high temperature provided by heat released from combustion process. This creates complex chemistry occurs at the solid material region, the combustion gas region, and the interface between both region. A schematic diagram of counter-current rotary kiln is shown in Figure 1.1 [6].

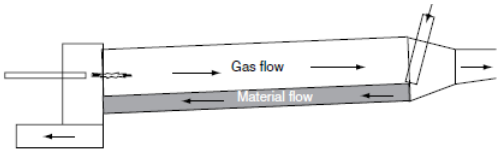


Figure 1.1: Schematic diagram of countercurrent rotary kiln.

Rotary kiln can handle different kinds of feedstock, from slumped and granular solids to wet slurry. Furthermore, it can also process material into desired grain size or particulate form and able to maintain different redox condition in the solid material and combustion gas regions. This allows the rotary kiln to have wide application in industry. Rotary kiln can commonly be found in pigment, lime or alumina hydrates calcination and production of cement clinker. Rotary kiln can also handle the calcination of bulkier material such as petroleum coke. In the metallurgy application, the rotary kiln is used in roasting or reduction of ores, recycling of zinc and lead oxides, etc. The most recent application of rotary kiln is incineration of waste material into non-toxic solids or recycling their contents as reusable products [21].

Two essential parameters of rotary kiln operation are operating temperature and residence time of the solid material. These two parameters determine the resulting end product of the process. To accommodate suitable operating condition of specific rotary kiln application, adaptation in kiln geometry and addition of structural support may be needed. As an example, installation of a refractory wall inside the steel cylinder shell is usually found in a rotary kiln which operates at more than 2000 K. Another example is an addition of dam to increase the solid material residence time and installation of lifter or tumblers to improve axial movement and mixing.

The present work considers a direct fired rotary kiln used in cement production by Almatis B.V., a company with expertise in alumina processing based in Rotterdam. This kiln also features slightly inclined horizontal steel cylinder with counter-current configuration. In this rotary kiln, heat is provided by combustion of hydrocarbon gas, creating a flame which has direct contact with solid materials. The combustion occurs as a

result of the reaction between hydrocarbon mixture as fuel with oxygen supplied by atmospheric air. The fuel and air enters the kiln from holes in the cylindrical burner and mixed within the kiln to create combustion reaction. Due to very high temperature, a secondary air inlet is provided as cooler and a refractory lining constructed from bricks is installed to protect steel cylinder. Even though the kiln operation involves solid materials, this work only considers the gas combustion to create a temperature profile inside the rotary kiln.

1.2. Computational Fluid Dynamics Modeling of Rotary Kilns

Recently, Computational Fluid Dynamic (CFD) approach for troubleshooting industrial process begins to be a mainstream approach. This exercise relies only on digital data rather than complex setup and consumable materials needed by direct laboratory experiment. A variation on process parameters is done by changing a digital text compared to reconstruct new mechanical parts to accommodate the change. Even a miss-conception only resulting in an error in the calculation rather than a disastrous accident that may affect personnel. This more cost-efficient and relatively safer procedure compared to direct laboratory experiment aspect drives the preference towards CFD utilization for troubleshooting or optimizing existing industrial processes. Moreover, development of faster and more reliable computers provide faster and finer computational simulation. However, CFD simulation highly depends on the details of simulation setup. Nonphysical simulation result can be obtained from the false application of the computational model, hence a validation of CFD simulation by direct laboratory experiment is often still needed. Therefore, an extra cautious consideration needs to be done when constructing a CFD model to produce a reliable result. Moreover, some physical phenomena can only be described by complex mathematical equations which need enormous computational time to calculate. To overcome this, several simplified approximations are often needed in CFD simulation which sometimes limits the achievement of a hyper-realistic model.

Application of CFD simulation for rotary kiln cases had been implemented in several publications. However, the modeling of complex physical phenomena involving chemical reactions, heat and mass transfer in a heterogeneous system has yet remained as an incompletely solved problem [21]. The common practice of the modeling is creating mathematical analysis on the heat and mass balance in the solid material and combustion gas domains while chemical and phase change is approximated by the reaction kinetics. Some of the works of CFD application in rotary kiln cases with this approach are [1], [5], [8], [20], [22], and [26]. Another approach to the simulation is by using predictive control model as seen in [29], where rough compartment segmentation of rotary kiln is further applied by sacrificing the detailed chemistry.

Based on [22], two common approaches is used in rotary kiln modeling: 1-dimensional and 'quasi 3-dimensional'. The first approach solves material and energy balances for both solid material and combustion gas in a 1-dimensional domain. The later approach solves in a 2-dimensional presentation of transverse surface of the solid material and 1-dimensional model for combustion gas. Both approaches calculate the chemical reaction as selected stoichiometric reactions, either equilibrium or mechanistic reaction rate [21].

To create a CFD simulation, several points are needed to be considered. First, all physical phenomena related to the process shall be known and expressed as a mathematical equation. This allows the model to resemble actual physical phenomena by a set of numbers. Then, a geometrical model of process system is needed and converted to meshes. This geometrical mesh represents the calculation domain of the simulation. Creating a reliable mesh is an essential step in CFD simulation since it will affect the overall convergence and result. The bad mesh can result in false calculation or even worse, non-converged simulation. Lastly, the equations are then calculated in the mesh domain by a computer software to achieve simulation result.

In the present work, physical phenomena which occur inside rotary kiln are considered. These include the turbulent flow of combustion gas, the reaction between fuel and oxygen to create combustion, and the heat transfer within the kiln which contributed by convective heat transfer of gas, conductive heat transfer at the brick lining, and radiative heat transfer. A simplified geometry of the kiln is constructed for the present work. This geometry includes the general feature of the kiln including the kiln cylinder, the burner, and secondary air inlet. A trade-off between finer and more reliable mesh with faster computational time is considered in the present work, allowing only scaled cases to be simulated in the present work. The simulation is run in an open-source CFD software named OpenFOAM (Open Field Operation And Manipulation) on a Linux-based machine provided by Delft Institute of Applied Mathematics (DIAM) of Delft Technology University.

The project of rotary kiln CFD simulation in DIAM had been done for several generations, including by M. Pisaroni and A. Kadar [14]. M. Pisaroni focused on creating a simulation of real rotary kiln geometry in commercial software StarCCM while A. Kadar initiated the usage of open source software OpenFOAM in arbitrary kiln geometry. The present work will be focused on combining both stepping stones: creating a simulation of real rotary kiln geometry in open source software OpenFOAM.

1.3. Objectives

The present work is conducted as an approach towards complete simulation for industrial rotary kiln using OpenFOAM. An application of solver developed by A. Kadar [14] to a more comprehensive geometry of rotary kiln is the focus of the present work. The challenge of this work comes from the high computational cost of the simulation in full three-dimensional kiln domain. Developed geometry model by M. Pisaroni [26] consists of about two million of cells which contributes to an enormous computational time. This issue hinders swift access to the resulting data in order to properly analyze and revise simulation setup to obtain a better result. In that basis, simplified models are developed in order to obtain a quick glance of the behavior of the simulation system, allowing faster troubleshooting of the simulation.

The simplified geometry derived from full three-dimensional kiln domain covers an axial slice of the geometry, constructing two-dimensional domain with major features such as the dimension of the kiln, location of the secondary air inlet, and the shape of the burner is incorporated in the simplified domain. This simplified geometry is used as calculation domain to analyze changing behavior of the system with respect to the change in input variables. This allows swift verification of the solver to underlying physics which is valuable information before moving towards more complex domain. The simplified two-dimensional geometry is also being used to profile the performance of the solver to identify the highest contributor to the computational cost. This identification can be a stepping stone for increasing the performance of the solver in the future project.

Another simplified geometry is developed based on a paper by Elattar and Specht [9], which has similar features to the intended full three-dimensional kiln geometry. The geometry has a high degree of symmetry, allowing partial domain simulation to reduce the computational cost. Available data from the paper also allows verification of the developed solver with commercial software used by the author. The verification will increase the confidence level of the solver.

In short, the objectives of the present works are:

- To identify the highest contribution of computational cost in order to improve the computational performance of the developed solver in OpenFOAM
- To verify the behavior of the system with respect to change in input variables
- To validate the result of the simulation in OpenFOAM with result in commercial software on published data

1.4. Structure of This Report

The report will be initiated by introduction of the simulation model in Chapter 2. The underlying physics are elaborated as governing equations and several models to approximate complex equations are introduced. Chapter 3 will explain the approach of the numerical model, exclusively in the software OpenFOAM. Several terminologies and general setup of the software are described as a basis for later explanation. The first study of this work is presented in Chapter 4, which identify the contribution of the code execution to total computational cost and the attempts to improve the computational performance. Chapter 5 will elaborate the behavior of the two-dimensional system towards changing input variables. Chapter 6 consists of reconstruction of published work using commercial software in OpenFOAM. Finally, a conclusion of the overall project is presented in Chapter 7.

2

Model of Combustion in Rotary Kiln

2.1. Introduction

Understanding the underlying physics in the studied phenomena is a highly influential step of creating a computational model. However, a complete elaboration of multi-physics in a given phenomena may require high complexity in the developed mathematical model. A trade-off between reliability and complexity of the model shall be cautiously considered to allows an acceptable degree of accuracy to the real physical phenomena in a reasonable computational cost. To achieve that, the general feature of the physical phenomena shall be captured in a precise mathematical equation and several simplification approach is developed to accommodate complex terms in the equation while keeping consistency with actual physics.

One of the major physical phenomena which occur in combustion in the rotary kiln is turbulent combustion. This physical phenomenon features two important aspects: turbulence and reaction. The turbulence is related to the characteristic of the flow and expressed as one of the term in momentum balance equation. The reaction occurs as the result of contacting different chemical species at a certain energy and rise a change in enthalpy. These two aspects are related as turbulence creates eddies which can lead to a vortex. These eddies prompt higher mixing of the chemical species within the flow. Since chemical reaction occurs due to contact of different chemical species, higher mixing induced higher reaction and hence higher change in enthalpy. The change of enthalpy affects the buoyancy in the flow and hence, driving the eddies to create more turbulence. In the model, these two aspects are very related in two-ways coupling between flow and chemistry. In short, the combustion phenomena in rotary kiln incorporates four main balance equations: mass, momentum, energy, and chemical species. These four equations encompass the basic fundamental of physics where both physical entities: matter, represents by mass and chemical species, and energy represents by momentum and energy itself, are balanced within the system. The balance accounts convection, diffusion, generation, and consumption of the two physical entities. The mass balance equation, or known as continuity equation, is often coupled with the momentum equation as two basic equations which describe the characteristic of the flow in a system. The energy equation in the present work is represented as enthalpy to allows relation with the chemical reaction which is governed by the fourth equation.

In its own, turbulence is probably the most significant unsolved problem in classical physics. The mechanism of turbulence generation has not been fully understood. Although numerous approaches of modeling turbulence are available at present days, most of the models introduce highly complex and coupled mathematical expressions which unlikely solvable analytically and require high cost of numerical computation, which is known as Direct Numerical Simulation (DNS) method. In respect to this, several approaches are developed to minimize the computational cost while maintaining the important aspect of the physical phenomena. Reynolds Averaged Navier-Stokes (RANS) and Large Eddy Simulation (LES) are two common methods of modeling turbulence generation and dissipation. RANS, as the name suggests, take the average feature of the turbulent flow by introducing the statistical approach of a Favre-averaged value of physical terms in the equation, including the density, viscosity, and velocity. From this basis, several models are derived to quantify the generation and dissipation of the turbulence, for example the common k-ϵ model, k-ω model, Spalart-Allmaras model, etc. Each model were developed from various specific cases, thus the application of the

model to given cases is shall be considered carefully. The LES model focuses more to the large eddies in the turbulence. This model lies between RANS, which only takes the statistical average of the eddies in the flow, and DNS, which takes all quantifiable eddies. This allows more detailed simulation than RANS in considerably cheaper computational than DNS. Graphical comparison between RANS, LES, and DNS is illustrated in Figure 2.1. More detail on RANS model application in present work is elaborated in the following subsection.

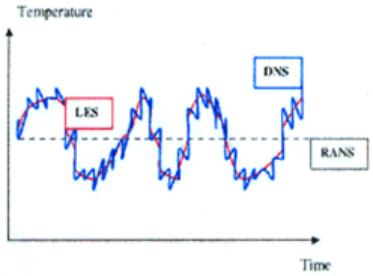


Figure 2.1: Graphical illustration of a comparison between RANS, LES, and DNS. [15]

As stated before, the chemical reaction occurs due to contact between several chemical species with sufficient energy. Those two are the key factors for chemical reaction, if one fails to be present, no chemical reaction will occur. In this respect, the turbulent mixing of chemical species plays a big role on the occurrence of the chemical reaction, since it provides the contact between chemical species. The combustion reaction occurs in a rotary kiln is considerably fast, faster than the turbulent mixing. Hence, it can be assumed that once the fuel and oxygen are mixed, the reaction occurs instantaneously. Therefore, the occurrence of combustion reaction in the rotary kiln is determined more by the mixing of the chemical species rather than the actual reaction itself. This phenomenon is taken into consideration when creating a model for the combustion reaction. Several models are developed based on this behavior such as eddy dissipation model, eddy-break-up model, and infinitely fast chemistry model, which takes more weight on turbulent mixing time scale than the chemical reaction time scale in the determination of chemical reaction rate. The detailed explanation regarding this model is also elaborated in the following subsection.

Lastly, rotary kiln operates at considerably high temperature, around 2000 K. At this high temperature, radiative heat transfer becomes significant compared to two other heat transfer phenomena, namely convection and conduction. Neglecting radiation term at this temperature, as commonly done in simulation at a lower temperature, will overestimate the resulting temperature field. However, calculation of radiative heat transfer often contributes to the high computational cost, since it needs to consider the geometrical configuration of cells in the calculation domain. Again, a trade-off between the physical accuracy and computational cost shall be carefully considered here. The elaboration of radiation model is presented in the following subsection.

2.2. Governing Equations

This section contains elaboration of governing equations mentioned in the introduction. Explanation of mathematical equation and contribution of each term is described.

2.2.1. Continuity

Continuity equation comes from the very basic physical law: conservation of mass, where it is stated that mass can not be generated nor destroyed. In the finite volume basis, in a control volume, the confined space where the system is considered, the mass that stays in the considered space shall be equal to the difference between incoming and outgoing mass flux.

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho u_j\right) = 0 \tag{2.1}$$

where ρ represent the mass per volume, t is time, u_j is the velocity of mass at arbitrary axis j, and x_j indicate the length of the control volume on arbitrary axis j. The first term of the equation (2.1) represents

the time rate of change while the second term shows the net flux of the mass into or from the control volume. The velocity term present in the equation comes from the characteristic of the flow which is calculated by momentum balance elaborated in the next subsection. This equation must hold for each control volume, which in the present work considered as a cell within calculation domain. Since the velocity term is also being calculated in the momentum balance equation, the continuity equation is often coupled with the momentum balance to iteratively calculate the velocity field which holds both equations.

In several simulation cases done in the present work, an assumption of an incompressible fluid is taken. This assumption treats the density of fluid as a constant, and hence, the first term of accumulation over time in (2.1) can be omitted, and density term can be excluded from the differential form.

$$\rho \frac{\partial u_j}{\partial x_j} = 0 \tag{2.2}$$

$$\frac{\partial u_j}{\partial x_j} = 0 \tag{2.3}$$

2.2.2. Momentum Balance (Navier-Stokes)

Momentum is the product of the mass and velocity. This quantity indicates how the object, which in the rotary kiln case is the mixture of fuel and oxidizer, flows within the system. The momentum balance equates the accumulation over time of momentum within control volume with the net fluxes and forces applied to the system [30]. The force can act as two types:

- surface forces
  - pressure force
  - viscous force
  - gravity force
- body forces
  - centrifugal force
  - Corolis force
  - electromagnetic force

However, to simplify the model, it is assumed that only pressure and viscous stresses, is accounted for momentum balance equation.

$$\underbrace{\frac{\partial}{\partial t}\left(\rho u_i\right)}_{\text{transient}} + \underbrace{\frac{\partial}{\partial x_j}\left(\rho u_i u_j\right)}_{\text{flux}} = \underbrace{S_{Mi}}_{\text{source}} \tag{2.4}$$

where ρu_i is the momentum at arbitrary axis i and S_{Mi} represent the forces applied to the system. Similar to continuity, the first term and second term on the left-hand side of equation (2.4) represent accumulation over time and flux respectively.

The source term in the equation comes from the derivative of stress tensor $\frac{\partial}{\partial x_j}\sigma_{ij}$, which as stated before, the present work only accounts for pressure and viscous stresses. Therefore, the forces terms in equation (2.4) can be elaborated as:

$$\sigma_{ij} = -p\delta_{ij} + \tau_{ij} \tag{2.5}$$

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{x_k}\delta_{ij}\right) \tag{2.6}$$

$$S_{Mi} = \frac{\partial}{\partial x_j}\sigma_{ij} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{x_k}\delta_{ij}\right)\right] \tag{2.7}$$

which completes the momentum balance as:

$$\underbrace{\frac{\partial}{\partial t}\left(\rho u_i\right)}_{\text{transient}} + \underbrace{\frac{\partial}{\partial x_j}\left(\rho u_i u_j\right)}_{\text{flux}} = \underbrace{-\frac{\partial p}{\partial x_i}}_{\text{pressure}} + \underbrace{\frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{x_k}\delta_{ij}\right)\right]}_{\text{viscous stress}} \tag{2.8}$$

where μ denotes dynamic viscosity of the fluid. This balance equation is known as Navier-Stokes equation.

For incompressible cases, the density term can be excluded from differential and kinematic viscosity (ν) is used instead of dynamic viscosity.

$$\frac{\partial u_i}{\partial t} + u_j\frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{x_k}\delta_{ij}\right)\right] \tag{2.9}$$

In a turbulent flow, a modification is needed for the Navier-Stokes equation. Turbulence can be seen as generation and dissipation of *eddies*, "lumps of fluid with concentrated swirling motion, vorticity blobs" (Kenejres), within the flow. These eddies are the result of continuous stretching of vortex filaments, which is related to viscous stress. The eddies create disturbance in the velocity field, resulting in a complex calculation of velocity gradient in viscous stress term. The modification of Navier-Stokes equation to take the behavior of eddies into account is elaborated in subsection 2.3.1.

2.2.3. Chemical Species Balance

During the combustion process, the chemical species are transported, generated, or consumed within the system. The transport of the chemical species is done by both convection, which is affected by the flow; and diffusion, which depends on the chemical interaction between the species. Generation or consumption of chemical species occurs due to the reaction, which as stated before, depends on the contact between the chemical species and the energy.

Given there are m chemical species $S_1, S_2, \ldots, S_m$ present in the mixture with their mass fraction are represented as $Y_1, Y_2, \ldots Y_m$, the balance equation for chemical species S_s is given as:

$$\underbrace{\frac{\partial}{\partial t}\left(\rho Y_s\right)}_{\text{transient}} + \underbrace{\frac{\partial}{\partial x_j}\left(\rho u_j Y_s\right)}_{\text{convection}} = \underbrace{\frac{\partial}{\partial x_j}\left(\rho J_s\right)}_{\text{diffusion}} + \underbrace{\dot{\omega}_s}_{\text{source}} \quad s = 1,2,\ldots,m \tag{2.10}$$

where J_s stands for the diffusive flux of chemical species and $\dot{\omega}_s$ represent the chemical reaction source term, which can be either generation or consumption of the species. By using Fick's law for diffusive flux, which assumes that it is proportional to the gradient of the mass of the species in arbitrary x_j direction, and *pseudobinary* approximation for diffusion constant D, the equation (2.10) is elaborated to:

$$\underbrace{\frac{\partial}{\partial t}\left(\rho Y_s\right)}_{\text{transient}} + \underbrace{\frac{\partial}{\partial x_j}\left(\rho u_j Y_s\right)}_{\text{convection}} = \underbrace{\frac{\partial}{\partial x_j}\left[\rho D_s\frac{\partial Y_s}{x_j}\right]}_{\text{diffusion}} + \underbrace{\dot{\omega}_s}_{\text{source}} \quad s = 1,2,\ldots,m \tag{2.11}$$

The chemical reaction source term is calculated by using the reaction rate law which indicates the mass change of a species over time. This rate law is commonly expressed as Arrhenius rate law which relates reaction rate is proportional to the concentration of involved species with a power of specific order. An empirical rate constant specific to the reaction is required to complete the proportionality. However, the discussion in the beginning of this chapter, at section 2.1, mentions that the chemical reaction is also highly affected by turbulent mixing. Often, in turbulent reaction, the mixing time is much slower than reaction time, causing the reaction rate is limited by the mixing phenomena. This behavior drives the development of other chemical source models which take turbulent mixing as dominating factor of reaction rates such as Eddy Dissipation Concept (EDC) and Eddy-Break-up Model (EBM). The elaboration of chemical source term model is presented in subsection 2.3.2.

2.2.4. Energy Balance

Conservation of energy can be expressed in several forms of the scalar transport equation. The most common forms are internal energy (e) and enthalpy (h). Internal energy is a quantity of energy calculated in the base of constant volume, while enthalpy is based on constant pressure. In this respect, internal energy is commonly used to quantify energy within liquid as a system of liquid with constant volume is commonly found in nature. In contrary, quantification of energy in the gas is often done in a constant pressure environment, allowing enthalpy becomes proper quantity for a gaseous system. Given that combustion phenomena in rotary kiln occur in a gaseous system, enthalpy balance is chosen as the form of energy balance used in the present work.

Enthalpy of a mixture of m chemical species $S_1, S_2, \ldots, S_m$ is expressed as mass-weighted sum of specific enthalpy h_s of species s

$$h = \sum_s Y_s h_s \quad h_s = h_s^o + \int_{T^o}^{T} Cp_s(T)\,\mathrm{d}T \tag{2.12}$$

where enthalpy with superscript h_s^o indicates enthalpy of formation of species s at s reference temperature ($T^o = 298.15K$) and pressure ($p^o = 1atm$) properties while Cp_s is the specific thermal capacity of the species at constant pressure and given temperature T. These two thermodynamic properties for various species can be obtained from standard libraries namely `JANAF` and `CHEMKIN` for enthalpy and thermal capacity respectively.

Similar to chemical species balance which features scalar transport equation, energy balance also consists of accumulation over time, convection, diffusion, and source term. Assuming q_h as the diffusive flux of enthalpy, the equation reads:

$$\underbrace{\frac{\partial}{\partial t}(\rho h)}_{\text{transient}} + \underbrace{\frac{\partial}{\partial x_j}(\rho u_j h)}_{\text{convection}} = \underbrace{-\frac{\partial}{\partial x_j} q_h}_{\text{diffusion}} + \underbrace{\dot{\omega}_h}_{\text{source}} \tag{2.13}$$

where $\dot{\omega}$ being source term for enthalpy due to various phenomena.

The diffusion of enthalpy occurs as result of conduction, interdiffusion, and Dufour effect [27]. The conduction is enthalpy transfer due to temperature gradient as stated by Fourier's Law. The interdiffusion effect is enthalpy transfer caused by species diffusion, while Dufour effect occurs due to the gradient of species concentration, which in most cases can be neglected.

$$q_h = q_{cond} + q_{interdiffusion} \tag{2.14}$$

$$q_{cond} = \alpha \frac{\partial h}{\partial x_j} \tag{2.15}$$

$$q_{interdiffusion} = \mu \left(\frac{1}{Sc_s} - \frac{1}{Pr_h} \right) \sum_s h_s \frac{\partial Y_s}{\partial x_j} \tag{2.16}$$

where α is thermal diffusivity of the mixture, defined as a ratio between thermal conductivity (k_c) and thermal capacity of the mixture (Cp) times density of the mixture;

$$\alpha = \frac{k_c}{\rho Cp} \tag{2.17}$$

Sc_s is Schmidt number, the ratio between viscous diffusivity of the flow and diffusivity of species s;

$$Sc_s = \frac{\nu}{D_s} = \frac{\mu}{\rho D_s} \tag{2.18}$$

and Pr_h is Prandtl number for enthalpy, the ratio between viscous diffusivity of the flow and thermal diffusivity of the mixture.

$$Pr_h = \frac{\nu}{\alpha} = \frac{\mu Cp}{k_c} \tag{2.19}$$

Common assumption taken for combustion simulation is that the mixture is homogeneous as such the diffusivity of species is taken to be equal and constant, $D_s = D$ and the Lewis number Le_s of species, the ratio between Schmidt and Prandtl number, is unity. In this assumption, the interdiffusion term is canceled out, allowing simple diffusion only due to enthalpy gradient.

The source term of enthalpy can be contributed by viscous heating ($\nabla \cdot \tau \vec{u}$), body forces such as pressure convection ($\vec{u} \cdot p$), transient pressure $\left(\frac{\partial p}{\partial t}\right)$, radiation $\dot{\omega}_{rad}$, and chemical reaction $\dot{\omega}_{chem}$. The first two sources: viscous heating and body forces, are only significant in high velocity and hence negligible for the combustion process in small Mach number as simulated in the present work. The pressure transient term can also be omitted in open flame case, where the pressure is approximately constant and equal to static pressure [14]. The radiation source term and chemical reaction source term is explained separately in subsection 2.3.3 and 2.3.2 respectively.

The complete energy balance equation is then:

$$\underbrace{\frac{\partial}{\partial t}(\rho h)}_{\text{transient}} + \underbrace{\frac{\partial}{\partial x_j}(\rho u_j h)}_{\text{convection}} = \underbrace{-\frac{\partial}{\partial x_j}\left[\frac{\mu}{Pr_h}\frac{\partial h}{\partial x_j}\right]}_{\text{diffusion}} + \underbrace{\dot{\omega}_{rad} + \dot{\omega}_{chem}}_{\text{source}} \tag{2.20}$$

2.3. Physical Models

In the previous section, four governing equations representing various physical phenomena is explained. However, additional unknown terms will appear when modeling turbulence, as will be explained in this section. Therefore models are needed to close the balance equation. This section elaborates physical models used in the simulation, namely Reynolds Averaged Navier-Stokes (RANS) with $k-\epsilon$ and Spalart-Allmaras models for the approximation of turbulence; infinitely fast chemistry and diffusion model derived from Eddy Dissipation Concept (EDC) for chemical source term; and P1 radiation model for radiative source term.

2.3.1. Reynolds Averaged Navier-Stokes (RANS) for Turbulent Stress

As mentioned earlier, turbulence is a disturbance in form of fluctuation in fluid flow due to generation and dissipation of eddies. The eddies can have a length as large as the flow length scale, or very small called *Kolmogorov* scale. In this respect, the disturbance in fluid flow also varies in a large spectrum. It is possible to resolve all the disturbance caused by eddies in turbulence, but the cost of computational time can be enormous. Therefore, several approaches are taken to minimize the effort of computation for turbulence but still takes general features into account. One method of the approximation is done by a statistical approach of fluctuation in the flow field, by taking the average to resolve the largest feature of turbulent phenomena. Given any fluctuating property of the flow $f(x,t)$, for example, fluctuation of velocity $u(x,t)$, the instantaneous fluctuation $f'(x,t)$ is given by:

$$f'(x,t) = f(x,t) - \overline{f}(x) \tag{2.21}$$

where $\overline{f}(x)$ is the time-average value of property $f(x,t)$ defined as:

$$\overline{f}(x) = \lim_{T\to\infty} \frac{1}{T}\int_0^T f(x,t)\mathrm{d}t \tag{2.22}$$

For compressible case, further treatment by weighing the fluctuation with time-averaged density is made. This allows even more simplification to the balance equation. The density weighted mean $\tilde{f}(x)$ is defined as:

$$\tilde{f}(x) = \frac{\overline{\rho f}(x)}{\overline{\rho}(x)} \tag{2.23}$$

this allows calculation of fluctuation from density-weighted mean $f''(x,t)$ as:

$$f''(x,t) = f(x,t) - \tilde{f}(x) \tag{2.24}$$

Using the time-averaged and density weighted mean approximation, the continuity equation is modified into:

$$\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial}{\partial x_i}\left(\overline{\rho}\tilde{u}_i\right) = 0 \tag{2.25}$$

Applying density weighted mean and subtracting the fluctuation on momentum balance equation resulting the Reynolds Averaged Navier-Stokes:

$$\underbrace{\frac{\partial}{\partial t}\left(\overline{\rho}\tilde{u}_i\right)}_{\text{transient}} + \underbrace{\frac{\partial}{\partial x_j}\left(\overline{\rho}\tilde{u}_i\tilde{u}_j\right)}_{\text{flux}} = \underbrace{-\frac{\partial \overline{p}}{\partial x_i}}_{\text{pressure}} + \underbrace{\frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right)\right]}_{\text{viscous stress}} \underbrace{-\frac{\partial}{\partial x_j}\left(\overline{\rho}\widetilde{u_i'' u_j''}\right)}_{\text{Reynolds stress}} \tag{2.26}$$

The Reynolds stress term emerges as an unknown variable in the momentum balance equation. This term shall be closed by introducing a model to approximate the instantaneous value. As proposed by Boussinesq in 1877, Reynolds stress is proportional to mean rate of deformation [30]. The proportionality is closed by a constant called *turbulent viscosity* (μ_t) which construct a similar equation to laminar viscous stress.

$$-\overline{\rho}\widetilde{u_i'' u_j''} = \mu_t\left[\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right] - \frac{2}{3}\overline{\rho}\tilde{k}\delta_{ij} \tag{2.27}$$

where k is turbulent kinetic energy per unit mass, defined as half of trace of density weighted velocity fluctuation:

$$k = \frac{1}{2}\left(u_j''\right)^2 \quad \text{or} \quad \tilde{k} = \frac{1}{2}\widetilde{\left(u_j''\right)^2} \tag{2.28}$$

By using the approximation in (2.27), the RANS equation (2.26) can be expanded to:

$$\underbrace{\frac{\partial}{\partial t}\left(\overline{\rho}\tilde{u}_i\right)}_{\text{transient}} + \underbrace{\frac{\partial}{\partial x_j}\left(\overline{\rho}\tilde{u}_i\tilde{u}_j\right)}_{\text{flux}} = \underbrace{-\frac{\partial}{\partial x_i}\left[\overline{p} + \frac{2}{3}\overline{\rho}\tilde{k}\right] + \frac{\partial}{\partial x_j}\left[\mu_{eff}\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right)\right]}_{\text{pressure, viscous, and Reynolds stress}} \tag{2.29}$$

where μ_{eff} is effective dynamic viscosity defined as:

$$\mu_{eff} = \mu + \mu_t \tag{2.30}$$

The approximation made the unknown for the equation narrowed down to one constant of turbulent viscosity μ_t and one scalar of turbulent kinetic energy k. Three models are being used in the present work to approximate the unknowns, namely standard $k-\epsilon$ model, realizable $k-\epsilon$ model, and Spalart-Allmaras model.

Standard $k-\epsilon$ Model

The model developed by Launder and Spalding in 1974 [30] is the most common model used in a simulation of turbulent flow. The model introduces two scalar transport equations to characterize the turbulent: turbulent kinetic energy (k) and turbulent energy dissipation rate (ϵ). The model is widely used since the usage of two equations allow calculation of small changes in the turbulent flow. The model calculates turbulent viscosity as a constant:

$$\mu_t = \rho C_\mu \frac{k^2}{\epsilon} = C_\mu \frac{\tilde{k}^2}{\tilde{\epsilon}} \tag{2.31}$$

where C_μ is model constant and turbulent energy dissipation rate ϵ is defined as

$$\epsilon = \frac{\mu}{\rho}\widetilde{\left(\frac{\partial u_j''}{\partial x_i}\right)\left(\frac{\partial u_i''}{\partial x_j}\right)} \quad \text{or} \quad \tilde{\epsilon} = \frac{\mu}{\overline{\rho}}\widetilde{\left(\frac{\partial u_j''}{\partial x_i}\right)\left(\frac{\partial u_i''}{\partial x_j}\right)} \tag{2.32}$$

The transport equation for k is:

$$\underbrace{\frac{\partial}{\partial t}\left(\overline{\rho}\tilde{k}\right)}_{\text{transient}} + \underbrace{\frac{\partial}{\partial x_j}\left(\overline{\rho}\tilde{u}_j\tilde{k}\right)}_{\text{convection}} = \underbrace{\frac{\partial}{\partial x_j}\left[\mu_{eff}^k \frac{\partial}{\partial x_j}\tilde{k}\right]}_{diffusion} + \underbrace{P_k}_{\text{production}} - \underbrace{\overline{\rho}\tilde{\epsilon}}_{destruction} \tag{2.33}$$

where μ^k_{eff} is effective viscosity for turbulent kinetic energy:

$$\mu^k_{eff} = \mu + \frac{\mu_t}{\sigma_k} \tag{2.34}$$

and σ_k is Prandtl number for turbulent kinetic energy. The production of turbulent kinetic energy P_k is defined as:

$$P_k = -\overline{\rho u_i'' u_j''} \frac{\partial \tilde{u}_i}{\partial x_j} = \mu_t \frac{\partial \tilde{u}_i}{\partial x_j} \left[\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} \right] \tag{2.35}$$

Substituting equation (2.35) to (2.33):

$$\underbrace{\frac{\partial}{\partial t}\left(\overline{\rho}\tilde{k}\right)}_{\text{transient}} + \underbrace{\frac{\partial}{\partial x_j}\left(\overline{\rho}\tilde{u}_j\tilde{k}\right)}_{\text{convection}} = \underbrace{\frac{\partial}{\partial x_j}\left[\mu^k_{eff}\frac{\partial}{\partial x_j}\tilde{k}\right]}_{diffusion} + \underbrace{\mu_t \frac{\partial \tilde{u}_i}{\partial x_j}\left[\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right]}_{\text{production}} - \underbrace{\overline{\rho}\tilde{\epsilon}}_{destruction} \tag{2.36}$$

Similar to transport equation for k, the transport equation for turbulent energy dissipation rate ϵ consists of accumulation over time, convection, diffusion, production and destruction.

$$\underbrace{\frac{\partial}{\partial t}\left(\overline{\rho}\tilde{\epsilon}\right)}_{\text{transient}} + \underbrace{\frac{\partial}{\partial x_j}\left(\overline{\rho}\tilde{u}_j\tilde{\epsilon}\right)}_{\text{convection}} = \underbrace{\frac{\partial}{\partial x_j}\left[\mu^\epsilon_{eff}\frac{\partial}{\partial x_j}\tilde{\epsilon}\right]}_{diffusion} + \underbrace{\frac{\tilde{\epsilon}}{\tilde{k}}\left(C_{\epsilon_1}P_k - C_{\epsilon_2}\overline{\rho}\tilde{\epsilon}\right)}_{\text{production and destruction}} \tag{2.37}$$

where C_{ϵ_1} and C_{ϵ_2} are model constant and similar to μ^k_{eff}, μ^ϵ_{eff} is effective viscosity for turbulent energy dissipation rate

$$\mu^\epsilon_{eff} = \mu + \frac{\mu_t}{\sigma_\epsilon} \tag{2.38}$$

where σ_ϵ is Prandtl number for turbulent energy dissipation rate. Substituting equation (2.35) to (2.37) makes:

$$\underbrace{\frac{\partial}{\partial t}\left(\overline{\rho}\tilde{\epsilon}\right)}_{\text{transient}} + \underbrace{\frac{\partial}{\partial x_j}\left(\overline{\rho}\tilde{u}_j\tilde{\epsilon}\right)}_{\text{convection}} = \underbrace{\frac{\partial}{\partial x_j}\left[\mu^\epsilon_{eff}\frac{\partial}{\partial x_j}\tilde{\epsilon}\right]}_{diffusion} + \underbrace{C_{\epsilon_1}\frac{\tilde{\epsilon}}{\tilde{k}}\mu_t\frac{\partial \tilde{u}_i}{\partial x_j}\left[\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right]}_{\text{production}} - \underbrace{C_{\epsilon_2}\overline{\rho}\frac{\tilde{\epsilon}^2}{\tilde{k}}}_{\text{destruction}} \tag{2.39}$$

By defining two transport equations above, the Reynolds stress term is closed. The remaining task is prescribe the model constants. The common values taken for the constants are:

$$\sigma_k = 1, \quad \sigma_\epsilon = 1.3, \quad C_\mu = 0.09, \quad C_{\epsilon_1} = 1.44, \quad C_{\epsilon_2} = 1.92$$

These values are used in the present work simulation.

Realizable $k-\epsilon$ Model

This model is a refinement of standard $k-\epsilon$ mentioned in subsubsection 2.3.1 model by giving mathematical constraint to assure the calculated Reynolds stress is physically consistent. The benefit of using realizable over the standard model is it predicts more accurately the spreading rate both planar and round jets. It is also better to model rotating flow, flow with a strong pressure gradient, separation, and recirculation. [13] The model uses a similar method with standard $k-\epsilon$ by introducing two scalar transport equation. The differences are:

1. Instead of a given constant value for C_μ is assumed, the Realizable $k-\epsilon$ model calculate it as a variable.
2. A slight modification of turbulent energy dissipation rate ϵ is made by derivation from the exact equation for the mean-square vorticity fluctuation [28] instead of from the production of turbulent kinetic energy k.

The calculation of C_μ is:

$$C_\mu = \frac{1}{A_0 + A_s\frac{\tilde{k}U^*}{\tilde{\epsilon}}} \tag{2.40}$$

where

$$U^* \equiv \sqrt{S_{ij}S_{ij} + \tilde{\Omega}_{ij}\tilde{\Omega}_{ij}}$$
$$S_{ij} \equiv \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$$
$$\tilde{\Omega}_{ij} = \Omega_{ij} - 2\epsilon_{ijk}\omega_k$$
$$\Omega_{ij} = \overline{\Omega_{ij}} - \epsilon_{ijk}\omega_k$$

and $\overline{\Omega_{ij}}$ is the mean rate-of-rotation tensor viewed in a rotating reference frame with the angular velocity ω_k. The model constants are given by:

$$A_0 = 4.04, \quad A_s = \sqrt{6}\cos\phi$$
$$\phi = \frac{1}{3}\cos^{-}1\left(\sqrt{6}W\right), \quad W = \frac{S_{ij}S_{jk}S_{ki}}{\tilde{S}^3}, \quad \tilde{S} = \sqrt{S_{ij}S_{ij}}$$

The modification to the transport equation for ϵ is:

$$\underbrace{\frac{\partial}{\partial t}\left(\overline{\rho}\tilde{\epsilon}\right)}_{\text{transient}} + \underbrace{\frac{\partial}{\partial x_j}\left(\overline{\rho}\tilde{u}_j\tilde{\epsilon}\right)}_{\text{convection}} = \underbrace{\frac{\partial}{\partial x_j}\left[\mu^{\epsilon}_{eff}\frac{\partial}{\partial x_j}\tilde{\epsilon}\right]}_{diffusion} + \underbrace{\overline{\rho}C_1 S\epsilon}_{\text{production}} \underbrace{-\overline{\rho}C_2\frac{\tilde{\epsilon}^2}{\tilde{k}+\sqrt{\nu\tilde{\epsilon}}}}_{\text{destruction}} \tag{2.41}$$

and the constant are:

$$C_1 = \max\left[0.43, \frac{\eta}{\eta+5}\right], \quad \eta = S\frac{\tilde{k}}{\tilde{\epsilon}}, \quad S = \sqrt{2S_{ij}S_{ij}}$$
$$C_2 = 1.9$$

This modification allows calculation of production directly from the ϵ not from the k, and the destruction term will never have singularity due to small value of k which can occurs in standard $k-\epsilon$ model.

Spalart-Allmaras Model

Unlike the two $k-\epsilon$ models mentioned before, this model only uses one scalar transport equation to predict the turbulent viscosity which is called kinematic eddy viscosity $\tilde{\nu}$ or known as *nuTilda* or *Spalart-Allmaras variable*. Other than that, the model involves specification of length scale by the algebraic formula [30]. This model is developed by Spalart and Allmaras in 1992. The usage of this model is commonly found in aerospace and turbomachinery application which involve low Reynolds number flow. Due to this fact, the usage of this model is limited. The combustion model presented in the present work do not use this model as it is poorly validated for industrial flow practice. However, a case mentioned in chapter 5 utilize this model as it only resolves non-isothermal flow without combustion. This model assumes that the turbulent kinetic energy in Boussinesq approximation mentioned in equation (2.27) can be neglected, and the only variable which needs to be resolved is the turbulent viscosity μ_t which is proportional to nuTilda:

$$\mu_t = \rho\tilde{\nu}f_{v1} \tag{2.42}$$

where

$$f_{v1} = \frac{\chi^3}{\chi^3 + C^3_{v1}}$$
$$\chi \equiv \frac{\tilde{\nu}}{\nu}$$

and C_{v1} is a model constant.

The scalar transport equation of $\tilde{\nu}$ is given as:

$$\underbrace{\frac{\partial}{\partial t}\left(\rho\tilde{\nu}\right)}_{\text{transient}}+\underbrace{\frac{\partial}{\partial x_j}\left(\rho\tilde{u}_j\tilde{\nu}\right)}_{\text{convection}}=\underbrace{\frac{1}{\sigma_{\tilde{\nu}}}\left[\frac{\partial}{\partial x_j}\left\{\left(\mu+\rho\tilde{\nu}\right)\frac{\partial\tilde{\nu}}{\partial x_j}\right\}+C_{b2}\rho\left(\frac{\partial\tilde{\nu}}{\partial x_j}\right)^2\right]}_{diffusion}+\underbrace{P_\nu}_{\text{production}}-\underbrace{Y_\nu}_{destruction} \tag{2.43}$$

where $\sigma_{\tilde{\nu}}$ and C_{b2} are model constants.

The production term is defined as:

$$P_\nu = C_{b1}\rho\tilde{S}\tilde{\nu} \tag{2.44}$$

where

$$\tilde{S}\equiv\Omega+\frac{\tilde{\nu}}{\kappa^2 d^2}f_{\nu 2}$$
$$f_{\nu 2}=1-\frac{\chi}{1+\chi f_{\nu 1}}$$

C_{b1} and κ are model constants, d is the distance from the wall and Ω is the magnitude of vorticity defined as:

$$\Omega\equiv\sqrt{2W_{ij}W_{ij}}$$
$$W_{ij}=\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}-\frac{\partial u_j}{\partial x_i}\right)$$

The destruction term for Spalart-Allmaras variable is elaborated as:

$$Y_\nu = C_{w1}\rho f_w\left(\frac{\tilde{\nu}}{d}\right)^2 \tag{2.45}$$

where

$$f_w = g\left[\frac{1+C_{w3}^6}{g^6+C_{w3}^6}\right]^{1/6}$$
$$g = r+C_{w2}\left(r^6-r\right)$$
$$r\equiv\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}$$

and C_{w1}, C_{w2}, C_{w3} being model constants.

The complete scalar transport equation for $\tilde{\nu}$ is:

$$\underbrace{\frac{\partial}{\partial t}\left(\rho\tilde{\nu}\right)}_{\text{transient}}+\underbrace{\frac{\partial}{\partial x_j}\left(\rho\tilde{u}_j\tilde{\nu}\right)}_{\text{convection}}=\underbrace{\frac{1}{\sigma_{\tilde{\nu}}}\left[\frac{\partial}{\partial x_j}\left\{\left(\mu+\rho\tilde{\nu}\right)\frac{\partial\tilde{\nu}}{\partial x_j}\right\}+C_{b2}\rho\left(\frac{\partial\tilde{\nu}}{\partial x_j}\right)^2\right]}_{diffusion}+\underbrace{C_{b1}\rho\tilde{S}\tilde{\nu}}_{\text{production}}-\underbrace{C_{w1}\rho f_w\left(\frac{\tilde{\nu}}{d}\right)^2}_{destruction} \tag{2.46}$$

and the model constants which is used in the present work are:

$$C_{b1}=0.1355,\quad C_{b2}=0.622,\qquad \sigma_{\tilde{\nu}}=\frac{2}{3},\quad C_{v1}=7.1$$
$$C_{w1}=\frac{C_{b1}}{\kappa}+\frac{1+C_{b2}}{\sigma_{\tilde{\nu}}},\quad C_{w2}=0.3,\qquad C_{w3}=2.0,\quad \kappa=0.4187$$

2.3.2. Model for Chemical Source Term

The chemical reaction is one of the hearts of combustion phenomena. This event does not only describes generation and consumption of chemical species $\dot{\omega}_s$ but also being a source term of enthalpy $\dot{\omega}_{chem}$ as enthalpy of reaction $\Delta H_r(T)$ in energy balance equation (2.20). The combustion phenomena can be described as a reaction between fuel and oxidizer. In the present work, the main fuel is methane (CH_4) and the oxidizer is oxygen (O_2). This combustion reaction ultimately releases carbon dioxide (CO_2) and water (H_2O).

$$CH_{4(g)} + 2O_{2(g)} \longrightarrow CO_{2(g)} + 2H_2O_{(g)}$$

Chemical source term is defined as the rate of generation or consumption of chemical species s at an instant:

$$\dot{\omega}_s \equiv \frac{\mathrm{d}}{\mathrm{d}t}\left(\overline{\rho}\tilde{Y}_s\right) \tag{2.47}$$

where $\overline{\rho}\tilde{Y}_s$ is the mass of species s per volume. The source term of enthalpy is defined as multiplication of chemical source term to the enthalpy of that reaction $\Delta H_r(T)$ in kJ/kg.

$$\dot{\omega}_{chem} = \dot{\omega}_s \Delta H_r(T) \tag{2.48}$$

Several models are developed to determine the value of the chemical source term, from the basic Arrhenius model which only account the chemical reaction itself to Eddy-Break-up Model which consider the turbulent mixing as the dominant factor of the chemical reaction.

Arrhenius Model

The most common basic expression for reaction rate is Arrhenius type reaction kinetic law. This model only considers the chemical reaction that occurs between several species at given temperature and energy while assuming the species are mixed well. This law expresses a single-step reaction rate as the concentration of involved species to the power of empirical order multiplied by a reaction rate constant k.

$$r = k\prod^{m} C_s^{\upsilon} \tag{2.49}$$

The reaction mechanism is essential in the determination of Arrhenius reaction rate since reaction rate constant and species reaction order is determined for a specific single-reaction mechanism. A chemical reaction such as combustion can be seen as single-step global reaction mechanism or several steps detailed reaction mechanism. The previously mentioned reaction is the example of global reaction mechanism which simplifies real mechanism of involved chemical reaction. This simplification often overestimates the rate of the reaction which leads to discrepancy to the real physical phenomena. A detailed reaction mechanism such as GRI-3.0 for methane combustion and several other models are developed to achieve more realistic value in combustion simulation. However, the usage of such detailed mechanism leads to high computational cost. In the present work, a two-step reaction mechanism of methane combustion is considered [4]. This mechanism involves an irreversible step of carbon monoxide (CO) from incomplete combustion of methane and a reversible step of carbon monoxide oxidation to carbon dioxide. The sum of both steps results in the global reaction mechanism presented before.

$$CH_4 + 1.5O_2 \longrightarrow CO + 2H_2O$$

$$CO + 0.5O_2 \rightleftharpoons CO_2$$

The reaction rate progress as the difference between forward and backward reaction [4].

$$r_j = k_{f_j}\prod_{s=1}^{m} C_s^{\upsilon'_{sj}} - k_{b_j}\prod_{s=1}^{m} C_s^{\upsilon''_{sj}}, \quad s = 1,2,\ldots m \tag{2.50}$$

where index j represent the reaction step, k_f and k_b are reaction rate constant for forward and backward reaction respectively, index s indicate the species, C_s is concentration of species s in mole per volume, υ'_{sj} and υ''_{sj} is reaction order of species s in step j in forward and backward reaction respectively which for detailed mechanism is equal to the reaction coefficient, while r_j is the reaction rate progress of step j. The reaction rate constant is given by Arrhenius equation:

$$k_j = A_j T^{\beta_j} \exp\left(-\frac{E_j}{RT}\right) \tag{2.51}$$

where A_j is empirical Arrhenius constant specific to reaction j, T is temperature at which reaction occurs, β_j is temperature exponent, $-E_j$ is activation energy of reaction j and R is ideal gas constant.

For the two-steps mechanism, the reaction rate for step 1 is given by:

$$r_1 = k_{f_1} C_{CH_4}^1 C_{O_2}^{1.5} \tag{2.52}$$

and step 2 is:

$$r_2 = k_{f_2} C_{CO}^1 C_{O_2}^{0.5} - k_{b_2} C_{CO_2} \tag{2.53}$$

The molar based reaction rate for given specie s, $\dot{q}_s$ is given as:

$$\dot{q}_s \equiv \frac{\mathrm{d}C_s}{\mathrm{d}t} = \sum_j \upsilon_{sj} r_j \tag{2.54}$$

where υ_{sj} is reaction coefficient of species s in reaction j which has a positive value when the species s is the product of the reaction, meaning it is located on the right hand side of reaction equation, and a negative value if the species s is the reactant of the reaction.

Taken species carbon monoxide (CO) as an example, this species is treated as the product in the first step and reactant in the second step with a coefficient of 1 for both steps. Hence, the change of concentration of species CO is given by:

$$\dot{q}_{CO} = \frac{\mathrm{d}C_{CO}}{\mathrm{d}t} = 1 \cdot r_1 - 1 \cdot r_2 \tag{2.55}$$

$$= k_{f_1} C_{CH_4}^1 C_{O_2}^{1.5} - \left[k_{f_2} C_{CO}^1 C_{O_2}^{0.5} - k_{b_2} C_{CO_2}\right] \tag{2.56}$$

In the present work, the reaction constants which are used in the simulation are given in Table 2.1.

step-1		
A_1	2×10^{15}	
$\upsilon'_{CH_4\,1}$	0.9	
$\upsilon'_{O_2\,1}$	1.1	
E_1	35000	cal/mole
step-2		
A_2	2×10^{9}	
$\upsilon'_{CO\,2}$	1	
$\upsilon'_{O_2\,2}$	0.5	
$\upsilon''_{CO_2\,2}$	1	
E_2	12000	cal/mole

Table 2.1: Reaction Constants

The relation between molar based reaction rate $\dot{q}_s$ to the source term of species s in chemical species balance equation (2.11) is given by:

$$\dot{\omega}_s = Mr_s \dot{q}_s \tag{2.57}$$

where Mr_s is molecular weight of species s in $g/mole$.

Arrhenius model is simple and highly related to the actual reaction. The time scale in equation (2.47) is defined as the chemical reaction time scale which only depends on the reaction rate constant k. Ability to

define detailed chemistry is also the advantage of using this model. The model is suited well for stagnant or perfectly-mixed laminar flow simulation. However, turbulent flow as in rotary kiln case in the present work accounts for non-perfectly mixed flow which affects the reaction as the contact between species depends on the mixing. Furthermore, the computation of Arrhenius model is considerably expensive. Therefore, another reaction source model which accounts more on mixing time scale than the chemical reaction time scale and computationally cheaper is then developed.

Eddy Break-Up Model

The Eddy break-up model developed by Spalding 1971 treats chemical reaction as an equilibrium reaction which is limited by the turbulent mixing. This model assumes that the chemical reaction τ_{chem} occurs much faster, hence smaller, than the residence time τ_{res}, the average time for chemical species to be inside a given control volume before transported away through convection or diffusion. A dimensionless number which compares these two time scales is the Damkohler number:

$$Da = \frac{\text{reaction rate}}{\text{mass transport rate}} = \frac{\tau_{res}}{\tau_{chem}} \tag{2.58}$$

Eddy-break-up model is a good approximation when the Da is high (≥ 10). At this condition, the chemistry is assumed to always sufficient time to consume fresh mixture and create an equilibrium [24]. In this respect, computation of Arrhenius model which is considerably expensive can be prevented and hence eddy-break-up model can give faster computation result.

The eddy-break-up model can be explained by considering the global mechanism of stoichiometric methane combustion:

$$\underset{\text{fuel:Fu}}{CH_4} + \underset{\text{oxidizer:Ox}}{2O_2} \rightleftharpoons \underset{\text{product:Pr}}{CO_2 + 2H_2O}$$

which in mass based can be simplified to:

$$1\,\text{Fu} + s\,\text{Ox} \rightleftharpoons (1+s)\,\text{Pr}$$

with s being the stoichiometric mass ratio of oxidizer to fuel.

The rate of consumption of fuel is then given as a function of local flow properties:

$$\tilde{\dot{\omega}}_{Fu} = -C_R \frac{\overline{\rho}}{\tau_{mix}} \widetilde{Y_{Fu}} \tag{2.59}$$

where C_R is model constant and τ_{mix} is the reaction rate limited by turbulent mixing time given as:

$$\tau_{mix} = \frac{\tilde{k}}{\tilde{\epsilon}} \tag{2.60}$$

Similarly, the rate of consumption of oxidizer and generation of product are given as:

$$\tilde{\dot{\omega}}_{Ox} = -C_R \frac{\overline{\rho}}{\tau_{mix}} \frac{\widetilde{Y_{Ox}}}{s} \tag{2.61}$$

$$\tilde{\dot{\omega}}_{Pr} = -C'_R \frac{\overline{\rho}}{\tau_{mix}} \frac{\widetilde{Y_{Pr}}}{1+s} \tag{2.62}$$

where the commonly used model constants are:

$$C_R = 1.0, \quad C'_R = 0.5$$

Eddy-break-up model resolve one transport equation for the mass fraction of fuel $\widetilde{Y_{Fu}}$ with the actual reaction rate is the minimum of three rates above:

$$\tilde{\dot{\omega}}_{chem} = \tilde{\dot{\omega}}_{Fu} = -\frac{\overline{\rho}}{\tau_{mix}} \min\left[C_R \widetilde{Y_{Fu}}, C_R \frac{\widetilde{Y_{Ox}}}{s}, C'_R \frac{\overline{\rho}}{\tau_{mix}} \frac{\widetilde{Y_{Pr}}}{1+s} \right] \tag{2.63}$$

In addition, the Arrhenius model reaction rate can also be considered as ignition limit which can contribute to the reaction rate. The Arrhenius reaction rate for fuel is given as:

$$\widetilde{\omega}_{Fu,Arr} = -A\overline{\rho}^{a}\widetilde{Y_{Fu}}^{b}\widetilde{Y_{Ox}}^{c}\exp\left(-E/RT\right) \tag{2.64}$$

where A is Arrhenius constant, a, b, and c are model constants, E is activation energy, R is ideal gas constant, and T is the temperature of the reaction. The equation (2.63) becomes:

$$\widetilde{\omega}_{Fu} = -\min\left[\frac{\overline{\rho}}{\tau_{mix}}C_R\widetilde{Y_{Fu}}, \frac{\overline{\rho}}{\tau_{mix}}C_R\frac{\widetilde{Y_{Ox}}}{s}, \frac{\overline{\rho}}{\tau_{mix}}C_R'\frac{\overline{\rho}}{\tau_{mix}}\frac{\widetilde{Y_{Pr}}}{1+s}, -\widetilde{\omega}_{Fu,Arr}\right] \tag{2.65}$$

The usage of eddy-break-up model is more suitable for the present work case since it is not only lower in computational cost but also accounts mixing time scale more than the actual chemical reaction rate. The eddy-break-up model is currently unavailable as a library in OpenFOAM-v1606+ software. The application of eddy-break-up model in OpenFOAM is approximated with two other models which are derived from eddy break-up model: infinitely fast chemistry and diffusion multi-component. The concept of two models is similar to eddy-break-up which considers the reaction rate as the minimum of fuel/oxidizer reaction rate.

Infinitely Fast Chemistry

The underlying assumption for this model is also chemical equilibrium is reached faster than the mixing phenomena. Instead of using mixing time scale τ_{mix} in eddy-break-up model, this model directly using computational time step Δt determined from Courant number at given simulation step.

$$\widetilde{\omega}_{Fu} = -C\frac{\overline{\rho}}{\Delta t}\min\left[\widetilde{Y_{Fu}}, \frac{\widetilde{Y_{Ox}}}{s}\right] \tag{2.66}$$

with C being model constant taken as

$$C = 5.0$$

As EBU, the application of infinitely fast chemistry model in OpenFOAM also benefits from no necessity of computing detailed Arrhenius law. However, although computational time step Δt is affected by turbulence, this model still lack direct contribution from turbulent properties k and ϵ. Albeit, the utilization of infinitely fast chemistry model is the closest OpenFOAM library to eddy-break-up model and even it is not recommended, this model can also being used for laminar flow as no turbulent properties are needed in the model. The serious drawback of this model, however, is this model can only account for single global mechanism reaction step. For simulating detailed reaction mechanism, another model based on diffusion with the similar assumption is used.

Diffusion Multi-Component

This model also assumes that mixed is burnt, that the turbulent mixing time is the bottleneck of reaction rather than reaction rate itself. However, unlike the two previous EBU and infinitely fast chemistry which calculate directly with the time scale, this model relies on the diffusion of the chemical species. The model is capable of handling detailed reaction mechanism. It assumes that the fuel and oxidizer for each reaction are single streams. The calculation of reaction rate is based on the magnitude of the product of fuel and oxidizer turbulent fluxes [16]. The reaction rate for this model is given as:

$$\widetilde{\omega}_{chem} = C_i D_{eff} P_{i,Gauss}\left|\frac{\partial\widetilde{Y_{Fu}}}{\partial x_j}\right|\left|\frac{\partial\widetilde{Y_{Ox}}}{\partial x_j}\right| \tag{2.67}$$

where D_{eff} is effective diffusivity, $P_{i,Gauss}$ is normalized Gaussian distribution centered on the stoichiometric value, and C_i is model coefficient:

$$C_i = C\left(1+\left(\frac{\widetilde{Y_{O_2}}}{oxRes}\right)^2\right)$$

where C is model constant with the default value of 1 and $oxRes$ is remaining oxidizer in each reaction which for the present work is taken as:

$$oxRes_1 = 0.015 \quad \text{for the first reaction step}$$
$$oxRes_2 = 0.005 \quad \text{for the second reaction step}$$

By using this model for the chemical source term, the calculation of turbulent combustion is done by weighing more to turbulent mixing time while still compensate the detailed reaction.

2.3.3. P1 model for Radiative Heat Transfer

All matter with the temperature greater than absolute zero emits thermal radiation which proportional to the fourth power of its absolute temperature as given by Stefan-Boltzmann law [14]. Although the increasing rate of this radiative heat transfer is high, considering the factor of fourth power of its absolute temperature, the Stefan-Boltzmann constant which accompanies it is considerably small, resulting in low contribution of radiative heat transfer on common industrial cases with operating condition lower than 1000 K and hence, often neglected. However, the case of rotary kiln operates at 1500-3000 K which makes the radiative heat transfer has comparable contribution with convective and conductive heat transfer.

Radiation can be seen as an emission of broadband electromagnetic wave distribution from radiation sources with the maximum content of energy is carried by specific wavelength determined by the source temperature. The transfer of radiation is then expressed as the rate of change of radiation beam spectral intensity traveling in the medium and propagating along a certain direction. This travel and propagation constitute absorption, emission, and scattering expressed as:

$$\frac{\mathrm{d}I_\lambda(\mathbf{r},\mathbf{s})}{\mathrm{d}\mathbf{s}} = \underbrace{-\kappa_\lambda I_\lambda(\mathbf{r},\mathbf{s})}_{\text{absorption}} + \underbrace{\kappa_\lambda I_{b\lambda}(\mathbf{r})}_{\text{emission}} \underbrace{-\sigma_{s\lambda} I_\lambda(\mathbf{r},\mathbf{s}) + \frac{\sigma_{s\lambda}}{4\pi}\int_{4\pi} I_\lambda(\mathbf{r},\mathbf{s}^*)\Omega(\mathbf{s}^*,\mathbf{s})\mathrm{d}\Omega^*}_{\text{scattering}} \tag{2.68}$$

this equation is known as Radiative Transfer Equation (RTE) where I_λ is spectral radiation intensity at point $\mathbf{r}$ along direction $\mathbf{s}$, κ and $\sigma_{s\lambda}$ are absorption and scattering coefficients of the medium respectively, $\Omega(\mathbf{s}^*,\mathbf{s})$ is scattering phase function, and subscript b indicates black body while subscript λ indicates wavelength.

The spectral radiation intensity I_λ is the extent of energy which radiates at specified wavelength λ measured as:

$$I_\lambda = \frac{\partial I_{e,\Omega}}{\partial \lambda} \tag{2.69}$$

where λ is the wavelength of interest and $I_{e,\Omega}$ is

$$I_{e,\Omega} = \frac{\partial \Phi_e}{\partial \Omega} \tag{2.70}$$

where Φ_e is radiant flux, energy per unit area, emitted, reflected, transmitted, or received; and Ω is solid angle into which the light is emitted.

For a black body, the radiation intensity at temperature T is following Planck equation:

$$I_{b\lambda}(T) = \frac{2hc^2}{\lambda^5}\frac{1}{e^{\frac{hc}{\lambda k_B T}} - 1} \tag{2.71}$$

where k_B is Boltzmann constant, h is Planck constant, and c is the speed of light in the medium. This implies shorter wavelength will have more intensity than a longer one, and the intensity is increasing with the temperature. The total amount of radiation from all wavelength is then given by:

$$I_b = \int_0^\infty I_{b\lambda}\mathrm{d}\lambda = \frac{\sigma}{\pi}T^4 \tag{2.72}$$

where $\sigma = 5.67 \times 10^{-8}\left[W/m^2K^4\right]$ is the Stefan-Boltzmann constant.

The radiative source term in energy balance equation (2.20) is the sum of the extent of gained and lost heat due to absorption and emission. The scattering effect is not accounted since it only redistributes radiative energy among the directions, not change the extent of local energy. It is given as the gradient of radiative flux (**q**):

$$\dot{\omega}_{rad} = -\nabla \cdot \mathbf{q} = \underbrace{\kappa G}_{\text{heat gain}} \underbrace{-4\kappa\sigma T^4}_{\text{heat loss}} \tag{2.73}$$

where radiative flux **q** is

$$\mathbf{q} = \int_0^\infty \int_{4\pi} I_\lambda \mathbf{s} \mathrm{d}\Omega \tag{2.74}$$

and total radiation intensity G is

$$G = \int_0^\infty \int_{4\pi} I_\lambda \mathrm{d}\Omega \tag{2.75}$$

This term of total radiation intensity G is not readily available for the computation. Approximation models to calculate this property are developed such as finite volume discrete ordinates method (fvDOM) and P1. Due to its simplicity, low computational cost with reasonable accuracy, the P1 model is used in the present work. The P1 approximation for radiative heat flux is:

$$\mathbf{q} = -\frac{1}{3\kappa_\lambda} \nabla G \tag{2.76}$$

$$\text{hence,} \quad \nabla \cdot \left(\frac{1}{3\kappa_\lambda} \nabla G \right) = \kappa G - 4\kappa\sigma T^4 \tag{2.77}$$

with the flux at the wall can be given as Marshak boundary condition:

$$\mathbf{q_w} = -\frac{1}{3\kappa_\lambda} \mathbf{n} \cdot \nabla G = -\frac{\epsilon_w}{2\,(2-\epsilon_w)} \left(4\sigma T_w^4 - G_w \right) \tag{2.78}$$

where T_w and G_w are calculated temperature and total radiation intensity at the wall and ϵ_w is wall emissivity taken as 1 for open boundaries which resemble a black body, including inlet and outlet of the rotary kiln and 0 for symmetry planes and perfectly reflecting boundaries.

The accuracy of P1 model is high if the radiation intensity in the system is mostly isotropic, for example in optically thick media, which is actually not true for rotary kiln combustion case [14]. The combustion gas in the rotary kiln is considered as optically thin media which makes P1 approximation is less accurate than more comprehensive model such as DOM. DOM method solves a finite number of discrete solid angle associated to vector $\overrightarrow{s}$ for RTE equation. Due to its high computational cost and adaptation of method from published data used as a comparison for present work, the P1 model is preferable.

2.4. Thermophysical Models

The present work involves various chemical species in its model. The behavior of chemical species and their interaction among themselves shall be accounted for changing physical behavior of the flow, compared to common simplified model for atmospheric air. The bulk properties of the flow such as density and dynamic viscosity which are used in the model shall be calculated in each computational cell domains to give the physical result from the simulation. Hence, various thermophysical models built-in to OpenFOAM software are being used in the simulation. The models are explained in the following subsections.

2.4.1. Thermophysical and Mixture Model: psiReactionThermo and Reacting Mixture

The thermophysical model for present work is taken as fixed composition model based on compressibility, $\psi = (RT)^{-1}$. The mixture contains various chemical species which are able to interact and react. Each species has its own library of a set of thermophysical properties such as molecular weight, `JANAF` constants, Sutherland coefficients, etc.

2.4.2. Equation of State: Perfect Gas Model

The flow in the rotary kiln consists of fuel as methane; air, which is oxygen and nitrogen; and combustion products, namely carbon dioxide, carbon monoxide, and water. These involved molecules are simple diatomic or triatomic molecules, which in the normal operating condition of rotary kiln behaves similarly to the ideal gasses. Therefore, the density of gas flow is assumed to follow simple perfect gas, where:

$$\rho = \frac{1}{RT} p \tag{2.79}$$

where ρ is the gas density, R is ideal gas constant, T and p are local temperature and pressure respectively.

2.4.3. Thermodynamics: `JANAF` Thermo Model

The thermal capacity at constant pressure Cp which is used in energy balance equation (2.20) is a function of temperature. The present work assumes the change of thermal capacity due to temperature as `JANAF` equation:

$$Cp = R\left((((a_4 T + a_3)\, T + a_2)\, T + a_1)\, T + a_0\right) \tag{2.80}$$

where constants $a_n,\ n = 0, 1, \ldots 4$ are taken from `JANAF` library which valid for temperature T between lower and upper limits T_l and T_h, and two sets of constants for temperature above or lower than reference temperature T_c where $T_l < T_c < T_h$.

2.4.4. Transport: Sutherland and Model

The transport model concerning dynamic viscosity μ is taken as Sutherland model, while thermal conductivity k_c is calculated using Eucken approximation. The Sutherland model for dynamic viscosity is:

$$\mu = \frac{A_s \sqrt{T}}{1 + T_s / T} \tag{2.81}$$

where A_s and T_s are Sutherland coefficient and temperature respectively. The thermal conductivity is then calculated as:

$$k_c = \frac{\left(Cp + \frac{5}{4} R\right) \mu}{Mr} \tag{2.82}$$

where Cp is thermal capacity calculated from `JANAF` and Mr is molecular weight of the mixture.

3

Numerical Model with OpenFOAM

3.1. Introduction

This chapter explains about the implementation of finite volume method to calculate governing equations given in chapter 2 using software OpenFOAM. The beginning of the chapter contains elaboration of discretization of scalar transport equation such as energy, chemical species, and turbulent scalars balance equation; and vector transport equation such as momentum balance. The last part of the chapter explains how the OpenFOAM software implements the discretized equation to its solver and various numerical method available to compute the equations.

3.2. Finite Volume Discretization Method

The basic of computational fluid dynamics (CFD) is instead of calculating the governing equations as a continuous domain with continuous calculus operator such as integral and differential, it discretize the system domain into numerous discrete sub-domains and calculate the governing equation within the sub-domains simultaneously and iteratively until a convergence reached. The key of CFD is then how to discrete the system domain into the sub-domains which computers able to calculate. Several methods of discretization for CFD are available such as Finite Discrete Method (FDM), Finite Element Method (FEM), and Finite Volume Method (FVM). The first method, FDM, is the oldest method where the equation is discretized using Taylor expansion to approximate the differential term with a difference in the variable, e.g. Δx, and hence it is considered as *discretization of functional operators*. Unlike FDM, FEM and FVM discretize the space of calculation domain instead, known as *discretization of functional space*. The widely used method for CFD is by far FVM due to its natural behavior of preserving the conservative properties of discretized numerical domain [12].

The idea of FVM is constructing a set of small sub-domains, called cells, which are treated as control volumes (CVs) and every governing equations are valid for each CVs. This set is often called grid or mesh and treated as a set of matrices on the computer. CV can have an arbitrary shape, but the most common shape is hexahedral, a cube with six faces as shown in Figure 3.1, which constructs a structured mesh. Due to its simplicity and the limitation of OpenFOAM, which has a high numerical error in handling unstructured mesh, all discretization done in the present work create structured meshes. Each CV in the structured mesh has values for computed fields, such as pressure, velocity, temperature, etc. at its center P, which communicates with the values on its six neighboring cells, N, E, W, S, H, L, through the faces as shown in Figure 3.2.
The integral form of transport equations within each CV are considered. The integration covers the entire volume of CV and for transient equations, the entire time domain. For arbitrary scalar or vector variable ϕ, the general form of the transport equation is given by:

$$\underbrace{\frac{\partial}{\partial t}\left(\rho\phi\right)}_{\text{transient}} + \underbrace{\nabla\cdot\left(\rho\vec{u}\,\phi\right)}_{\text{convection}} = \underbrace{\nabla\cdot\left(\Gamma\nabla\phi\right)}_{\text{diffusion}} + \underbrace{S_\phi}_{\text{source}} \tag{3.1}$$

where in the integral form, reads:

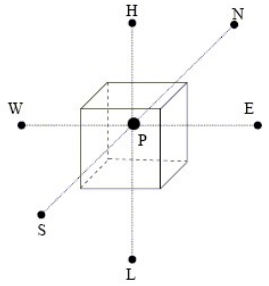


Figure 3.1: Hexahedral Control Volume [25]

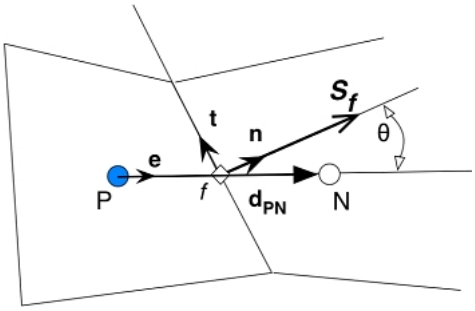


Figure 3.2: Arbitrary Face [23]

$$\int_t^{t+\delta t}\left[\frac{\partial}{\partial t}\int_\Omega \rho\phi \mathrm{d}V + \int_\Omega \nabla\cdot\left(\rho\vec{u}\phi\right)\mathrm{d}V - \int_\Omega \nabla\cdot\left(\Gamma\nabla\phi\right)\mathrm{d}V\right]\mathrm{d}t = \int_t^{t+\delta t}\int_\Omega S_\phi(\phi)\mathrm{d}V\mathrm{d}t \tag{3.2}$$

where Ω is the volume of CV, $\vec{u}$ is the flow velocity, and Γ is diffusion coefficient. The discretization of this integral form is done by changing continuous integral with a sum over control volumes. The discretization of each term is detailed in the following subsections.

3.2.1. Diffusion Term Discretization

The volume integral of diffusion term can be transformed to surface integral by Gaussian divergence theorem [7] which states:

$$\int_\Omega \nabla\cdot\vec{\Phi}\mathrm{d}V = \int_{\partial\Omega}\vec{n}\cdot\vec{\Phi}\mathrm{d}S = \int_{\partial\Omega}\vec{\Phi}\cdot\vec{S} \tag{3.3}$$

where $\vec{\Phi}$ is an arbitrary flux, including diffusive flux, $\partial\Omega = \vec{S}$ is the surface area at given normal vector $\vec{n}$. The discretized form of this integral form is:

$$\int_{\partial\Omega}\vec{\Phi}\mathrm{d}\vec{S} = \sum_{faces}\vec{\Phi}_f\mathrm{d}\vec{S}_f \tag{3.4}$$

where subscript f indicates the faces of CVs. Hence, the discretized diffusion term is given by:

$$\int_\Omega \nabla\cdot\left(\Gamma\nabla\phi\right)\mathrm{d}V = \int_{\partial\Omega}\left(\Gamma\nabla\phi\right)\mathrm{d}\vec{S} = \sum_{faces}\Gamma_f\nabla\phi_f\cdot\vec{S}_f \tag{3.5}$$

and diffusive flux at face f, D_f is given by:

$$D_f = \Gamma_f\nabla\phi_f\cdot\vec{S}_f = \Gamma_f\left[\left(\phi_N-\phi_P\right)\left|\frac{\vec{S}_f}{\overrightarrow{d_{PN}}}\right|\right] \tag{3.6}$$

where Γ_f is face average diffusivity, ϕ_P and ϕ_N denote central value, the value at the cell center point, of variable ϕ at local and neighboring cells respectively, while d is the distance between those two cell center points.

3.2.2. Convection Term Discretization

Similar to diffusion term in subsection 3.2.1, using Gaussian divergence theorem, the discretized convection term is given as:

$$\int_{\Omega} \nabla \cdot \left(\rho \vec{u} \phi\right) \mathrm{d}V = \int_{\partial\Omega} \left(\rho \vec{u} \phi\right) \cdot \mathrm{d}\vec{S} = \sum_{faces} \phi_f \left(\rho \vec{u}\right)_f \cdot \vec{S}_f = \sum_{faces} \phi_f \vec{F}_f \tag{3.7}$$

where $F_f = \left(\rho \vec{u}\right)_f \cdot \vec{S}_f$ is convective flux at face f. Since convective flux always contains vector of velocity $\vec{u}$ which has magnitude and direction, a numerical error can arise by interpolating convective flux at face simply by averaging the value from neighboring central values or often called *Central Difference Scheme*. Therefore, other differencing schemes for convective flux are considered. One example of commonly used difference scheme is the *Upwind Difference Scheme*, where the upwind central value is projected onto cell faces, depending on the flow direction [12]. The formulation is:

$$F_f \phi_f = \phi_P \max\left(F_f, 0\right) - \phi_N \max\left(-F_f, 0\right) \tag{3.8}$$

where operation $\max(A,B)$ will take the maximum value of either A or B. Other differencing schemes such as *Total Variation Diminishing (TVD)* [12] or *Normalized Variable Diagram (NVD)* are also available in OpenFOAM. These difference schemes are used throughout the simulation within the present work for convective term appears in the governing equations.

3.2.3. Source Term Discretization

The source term in transport equation can be non-linear to the variable ϕ such as in the chemical species balance equation (2.47). Hence, a linearization shall be applied to the source term before applying volume integral.

$$S_\phi(\phi) = S_c + S_p \phi \tag{3.9}$$

$$\int_{\Omega} S_\phi(\phi) \mathrm{d}V = S_c V_P + S_p \phi_P V_P \tag{3.10}$$

where V_P denotes the volume of CV.

3.2.4. Time Discretization

Transient simulation, where the transport equation contains time differentiation term, needs to resolve the equation from time to time. By substituting all discretized terms into the general transport equation (3.2) and rearrange to make the accumulation over time as the left-hand side, the equation becomes:

$$\int_t^{t+\delta t} \left(\frac{\partial \rho \phi}{\partial t}\right)_P V_P \mathrm{d}t = \int_t^{t+\delta t} \left[- \sum_{faces} \phi_f \vec{F}_f + \sum_{faces} \Gamma_f \nabla \phi_f \cdot \vec{S}_f + S_c V_P + S_p \phi_P V_P \right] \mathrm{d}t \tag{3.11}$$

The time discretization is then read:

$$\begin{aligned} \frac{\rho_P^1 \phi_P^1 - \rho_P^0 \phi_P^0}{\Delta t} = & c \left[- \sum_{faces} \phi_f \vec{F}_f + \sum_{faces} \Gamma_f \nabla \phi_f \cdot \vec{S}_f + S_c V_P + S_p \phi_P V_P \right]^1 + \\ & (1-c) \left[- \sum_{faces} \phi_f \vec{F}_f + \sum_{faces} \Gamma_f \nabla \phi_f \cdot \vec{S}_f + S_c V_P + S_p \phi_P V_P \right]^0 \end{aligned} \tag{3.12}$$

where superscript 0 and 1 denote the current and the next time steps and c is method constant where

$$c = \begin{cases} 0 & : \text{Euler explicit} \\ 0.5 & : \text{Crank-Nicholson} \\ 1 & : \text{Euler implicit} \end{cases} \tag{3.13}$$

Several cases within the present work diminish the time discretization completely to achieve steady-state modeling. However, the combustion case needs to be resolved in transient condition and Euler implicit method is used throughout the present work.

3.2.5. Pressure and Velocity Coupling

Unlike the other scalar transport equations, the momentum balance or Navier-Stokes equation (2.8) contains two fields which need to be resolved simultaneously: pressure p and velocity u_j which has a highly non-linear relationship. The fact that velocity shall also satisfy continuity equation (2.1) while pressure does not appear in that equation, arise necessity to treat pressure-velocity coupling carefully to minimize numerical error since they will correct each other. Some algorithms are developed to tackle this issue, including two methods which are used in the present work: Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) and Pressure Implicit with Split Operator (PISO). Both methods are based on de-coupling the pressure and velocity fields, allowing to calculate them separately, then use one field to correct another field iteratively until convergence achieved.

Semi-Implicit Method for Pressure-Linked Equations (SIMPLE)

The method is developed by Patankar and used for calculating steady-state problem iteratively. This method considers that fully resolving pressure-velocity coupling is not necessary for the steady-state problem since the changes between consecutive solutions are no longer small [30]. The method calculates the discretized momentum and pressure correct equation implicitly, while the velocity correction is calculated explicitly. The features of the SIMPLE algorithm are:

- The velocity field is approximated by solving momentum equation. Pressure distribution from previous iteration or initial guess is used to calculate the pressure gradient term.
- The new pressure distribution is calculated using pressure equation.
- Correction for velocity field is applied, new set of conservative fluxes is calculated.

The SIMPLE algorithm is done through steps shown in Figure 3.3 [30] [12].

Pressure Implicit with Split Operator (PISO)

This algorithm is developed by Issa in 1986 and extension to the SIMPLE algorithm [30]. This algorithm was developed originally to solve non-iterative unsteady compressible flow but had been adapted to accommodate steady state problem. This method applies no iteration, large time steps, and lesser computing effort compared to SIMPLE. This algorithm introduces one prediction step and two correction steps to satisfy mass conservation. The steps are shown in Figure 3.4 [30]
The predictor step and correction steps are as follows:

- Predictor step
  1. Guess the pressure p^* and velocity u_j^* using discretized momentum equation
  2. The guess can be either correct or not
- Corrector step 1
  1. Velocity component from predictor step may be not correct, correction factor p' and u_j' is defined
  2. Solve momentum equation by using correct pressure p^{**} to get corresponding correct velocity u_j^{**}

$$p' = p^{**} - p^*$$
$$u_j' = u_j^{**} - u_j^*$$

- Corrector step 2
  1. Define second correction factor p'' and u_j''

$$p'' = p^* + p'$$
$$u_j'' = u_j^* + u_j'$$

  2. Calculate second corrected field p^{***} and u_j^{***}

$$p^{***} = p^{**} + p''$$
$$u_j^{***} = u_j^{**} + u_j''$$

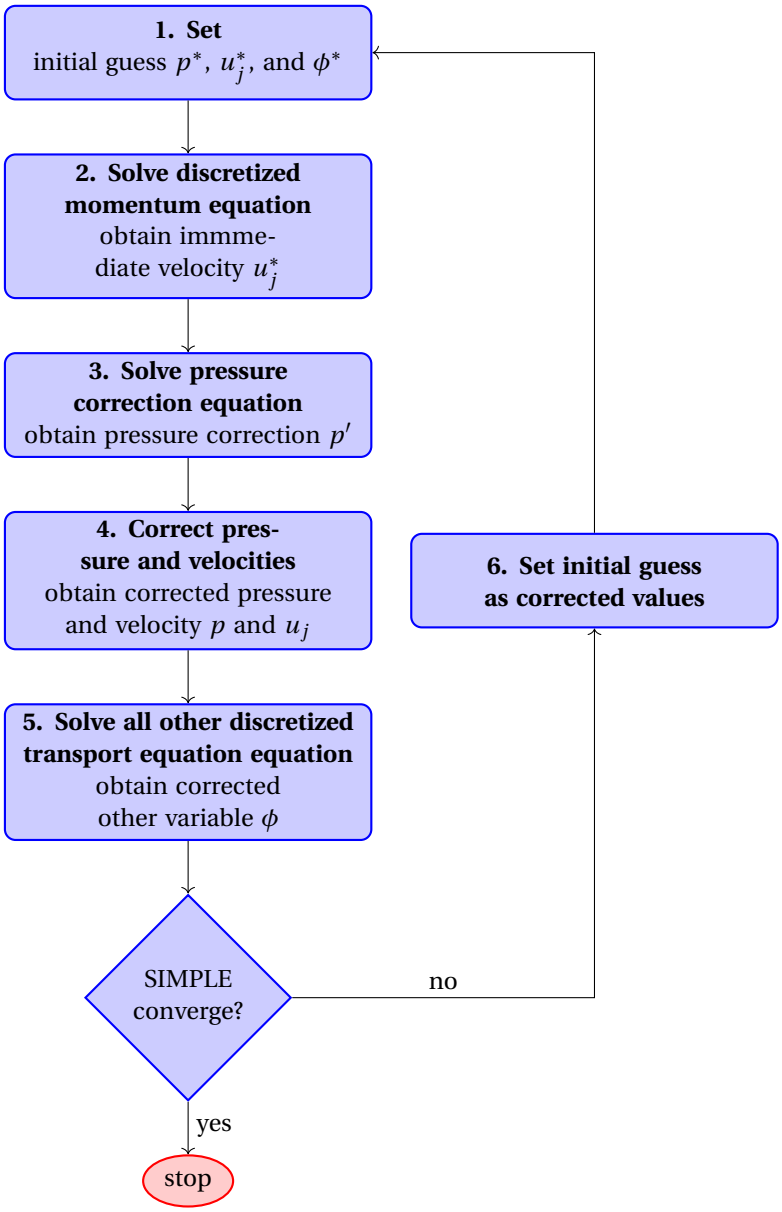


Figure 3.3: SIMPLE Algorithm

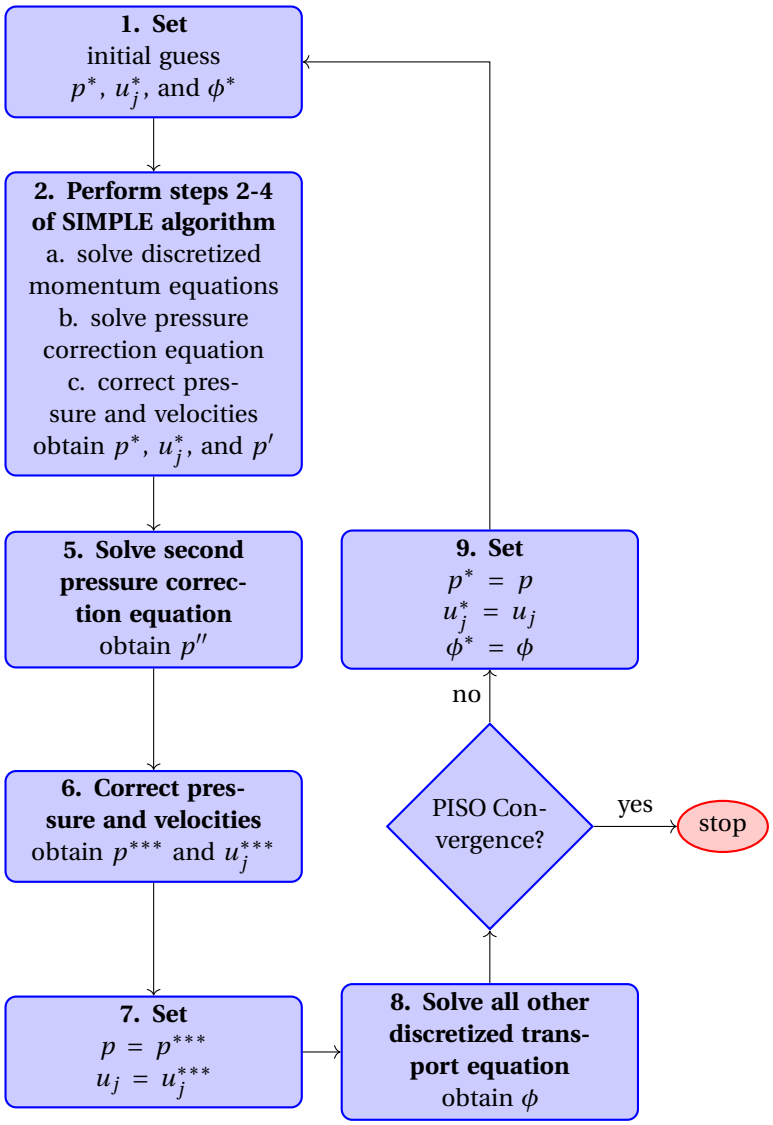


Figure 3.4: PISO Algorithm

3. Set the final correct pressure and velocity component field p and u_j

$$p = p^{***}$$
$$u_j = u_j^{***}$$

This PISO algorithm is generally more stable and takes less CPU time than SIMPLE especially for cases which have weak or no coupling between momentum and scalar equation. However, this method is not suitable for all cases. PISO algorithm is used in the present work in transient turbulent combustion cases.

3.3. Implementation in OpenFOAM

The present work is using CFD software called Open Field Operation and Manipulation (OpenFOAM) which written in C++ and run on a Linux platform. This open source software offers two types of executable files or known as an application: *solvers* and *utilities* [18]. The solvers are a comprehensive routine which designed to solve a wide variety of problem in continuum mechanics such as simple to complex fluid flows involving reaction or heat transfer, to solid dynamics and even electromagnetism. The utilities are additional toolbox which performs manipulation of data such as creating meshed geometry, post-process calculation, and data visualization. Almost all application in OpenFOAM, which the overall structures is given in Figure 3.5 [18], are capable of being executed in parallel processors, allowing optimization of available computing resources.

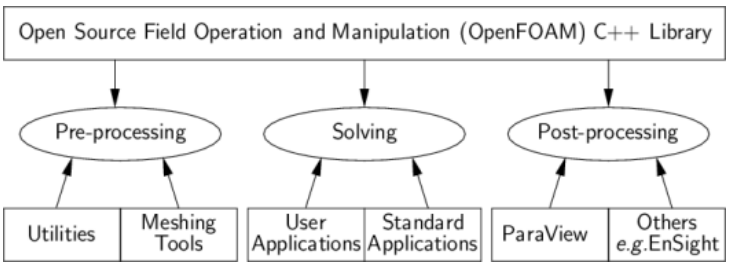


Figure 3.5: Overall Structure of OpenFOAM

Although numerous other CFD software, either commercial such as STAR CCM+, ANSYS Fluent, and COMSOL; or open source as SU2 are available, OpenFOAM has its own advantages which drive its usage in the present work. Being an open source software means not only the expensive license is unnecessary, but also ability to dive into the source code provides freedom for the users to apply their own solver or method. Although this freedom can divert to invalid codes or solvers, the large and highly active user community from various academic institutions and industry ensures bug or invalid method is reported and revised. However, the license-free and its flexibility do not come without cost. Although many efforts had been done through present years, complete and fully structured documentation of OpenFOAM is still not yet available. The lack of fully functional graphical user interface (GUI) - there is some development of GUI for OpenFOAM, but not all features are available - also force the user to set up their OpenFOAM cases by writing text-only C++ codes. Albeit, considering its flexibility and considerably robust toolbox, the present work relies on built-in solvers provided by OpenFOAM 1606+ from www.openfoam.com.

3.3.1. Structure of OpenFOAM Case

A case in OpenFOAM is set up by placing required files into three main folders:

- Time folder or Zero (0) folder, where the initial and boundary conditions of fields are defined.
- Constant folder, where constant values and libraries, such as thermophysical or physical model constant explained in section 2.4 and 2.3 respectively, is stored as a reference. The mesh information is also stored in this folder as the polyMesh file.
- System folder, which defines how the case will be run: steady-state or transient, how many iteration or time step, which numerical solver is used, etc.

Other libraries can also be part of an OpenFOAM case folder, for example, chemical species libraries of `CHEMKIN` which contains information about thermophysical properties and chemical reaction constant for

each chemical species. All the mesh used in the present work, except mesh for the buoyant flow validation case from cfMesh, are built using OpenFOAM utility blockMesh. This utility creates structured mesh from given points and arches defined in a blockMeshDict file. Another mesh which was planned to be used in the present work is the full 3D domain of rotary kiln provided from previous work of Pisaroni, using his own software to create the mesh from given 3D geometry file. The set up of each case in the present work is detailed in their respective case chapter.

3.3.2. Numerical Schemes and Algorithm

In resolving a discretized equation in OpenFOAM, the numerical schemes and algorithm for each term are need to be determined. The schemes are determined in the *fvSchemes* file under *system* folder in an OpenFOAM case [19]. The prescribed schemes dictate which method is used by OpenFOAM when resolving the discretized terms:

- Interpolation: to calculate interpolated values, for example from cell centers to face centers
- Surface Normal Gradient: to calculate gradient, normal to a face, of values at the center of 2 cells connected by that face
- Gradient: to calculate discretized gradient term ∇, such as ∇p in momentum equation
- Laplacian: to calculate discretized laplacian term ∇^2, for example, $\nabla^2 \nu u_j$ in incompressible momentum equation
- Divergence: to calculate discretized divergence term $\nabla\cdot$, such as convective term $\nabla\cdot\left(\rho u_i \phi\right)$ which needs consideration of differencing scheme as explained in section 3.2.2
- Time: to calculate discretized time derivative term $\frac{\partial}{\partial t}$ as explained in section 3.2.4 or using steady-state which makes this term zero.

The numerical algorithm is defined in *fvSolution* file under *system* folder and contains information about linear solver used to compute variable fields. This files also determine the relaxation factors, tolerance control, and pressure-velocity coupling algorithm explained in section 3.2.5. The numerical algorithms which are used in the present works are:

- Preconditioned (bi-)conjugate gradient (*PCG*/*PBiCG*), *PCG* for symmetric matrices such as pressure, *PBiCG* for asymmetric matrices such as velocity. The available preconditioners are:
  - Diagonal incomplete-Cholesky (*DIC*) for symmetric matrices
  - Diagonal incomplete-LU (*DILU*) for asymmetric matrices
- Solver using smoother (*smoothSolver*) such as Diagonal incomplete-Cholesky (*DIC*) or Gauss-Seidel algorithm
- Generalized geometric-algebraic multi-grid (*GAMG*)

The usage of the linear solver for each case is chosen from the characteristic of the solved equation and detailed in their respective case chapter.

3.3.3. OpenFOAM Solvers

The *solver* class application contains a comprehensive algorithm to solve a specific physical problem. Commonly, solvers are started by calling all necessary libraries such as model type and constants, creating required variable fields, and solve governing equations to get the final solution of the fields. Numerous built-in solvers are available in OpenFOAM out of the box, including solvers for turbulent flow and turbulent combustion. Users can also modify the behavior of the solvers or adapt the governing equation to accommodate their developed models. The governing equations in OpenFOAM solvers are written by determining several classes of matrices which are based on the discretized equation. The commonly used classes are [17]

- fvm: Finite Volume Matrix, calculate an implicit derivative, returning matrix
- fvc: Finite Volume Calculus, calculate an explicit derivative

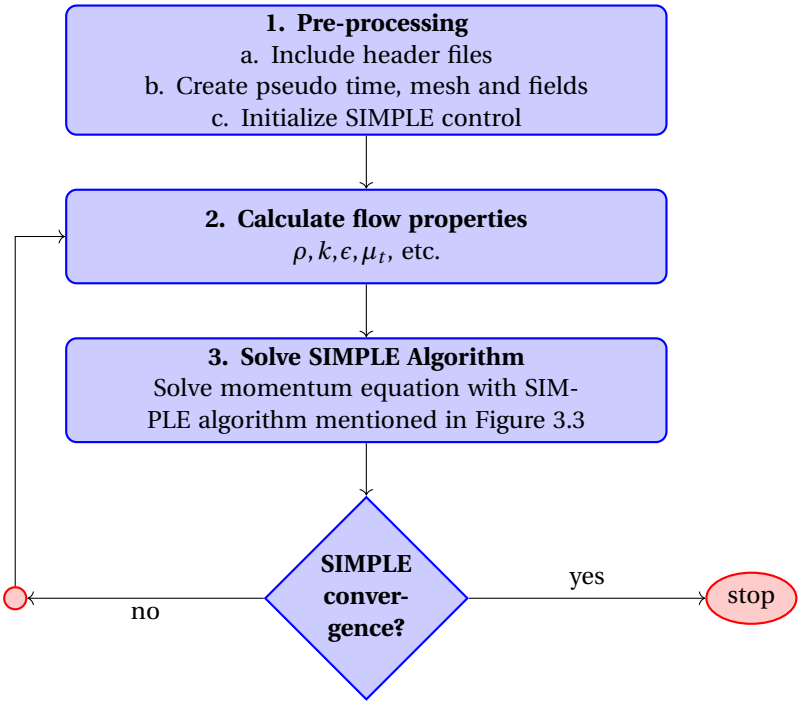


Figure 3.6: simpleFoam Algorithm

An example of discretized equation in OpenFOAM solver laplacianFoam which solves simple laplacian type temperature scalar equation:

$$\frac{\partial}{\partial t} T - D_T \nabla^2 T \tag{3.14}$$

which is translated to:

```
fvm::ddt(T) - fvm::laplacian(DT, T)
```

where the first part of the command returns a matrix of implicit time derivation of field T and second part returns a matrix of implicit laplacian of field T multiplied with diffusion constant D_T. The solvers used in the present work are:

- simpleFoam: solve incompressible Navier-Stokes equation to obtain flow profile as pressure and velocity, commonly used to generate steady-state field as initial condition for another solver
- buoyantSimpleFoam: solve compressible convective heat transfer of a flow, used in comparative study with published work in chapter 4
- reactingFoam: solve compressibility-based reacting flow, main solver that is used to simulate turbulent combustion
- furnaceFoam: user defined solver, extension of reactingFoam which includes the radiation source term

simpleFoam
The solver simpleFoam resolve incompressible continuity equation (2.2) and momentum balance equation (2.9). The algorithm of simpleFoam is given in Figure 3.6 [14]

buoyantSimpleFoam
Solver buoyantSimpleFoam is an extension of simpleFoam with an addition of energy balance (2.20) calculation. The solver takes buoyancy effect to the flow due to the difference in density caused by the temperature gradient. The algorithm of buoyantSimpleFoam is shown in Figure 3.7

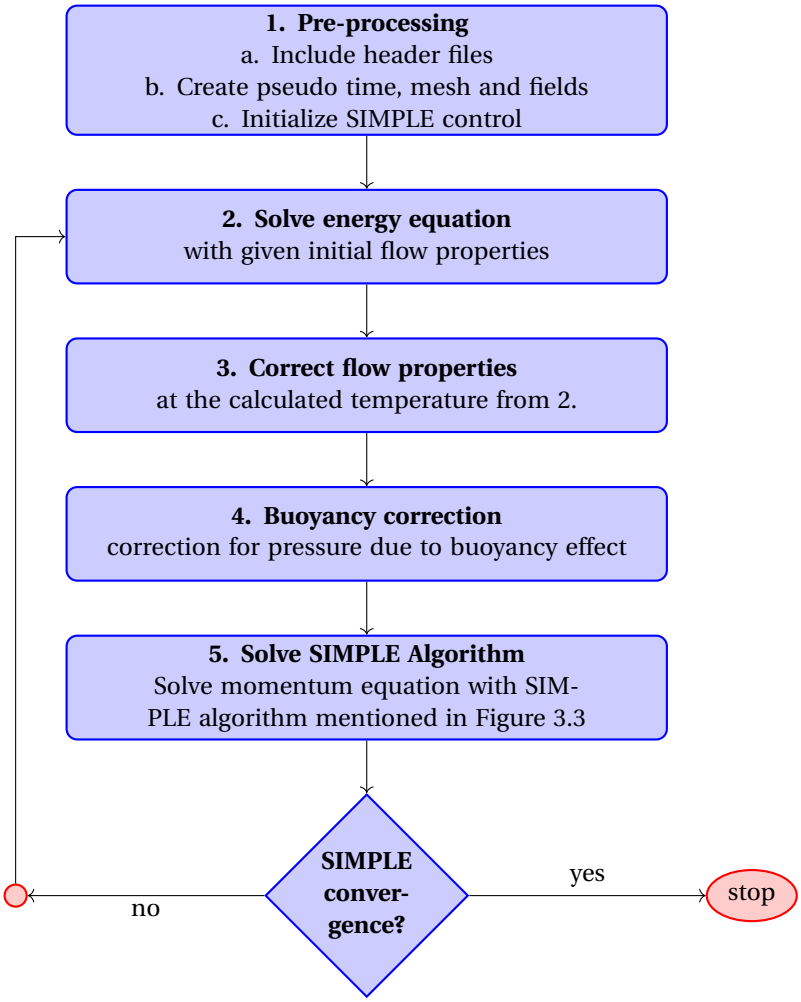


Figure 3.7: buoyantSimpleFoam Algorithm

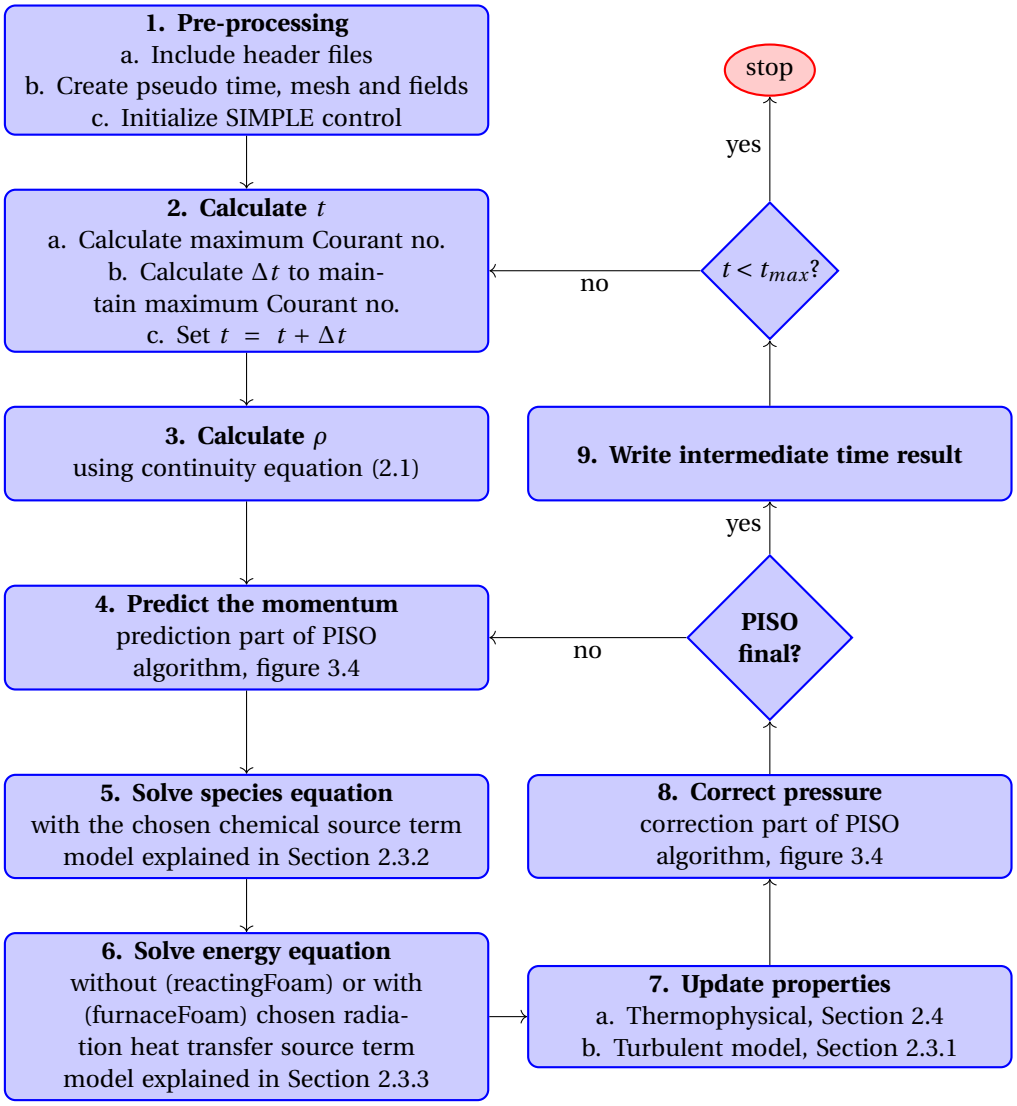


Figure 3.8: reactingFoam/furnaceFoam Algorithm

reactingFoam and furnaceFoam
Solver reactingFoam is built-in solver to calculate turbulent combustion problem. The solver uses PIMPLE algorithm, a combination between PISO and SIMPLE, and needs to be calculated in transient mode. User-defined solver furnaceFoam is an extension of reactingFoam with an addition of radiation source term for energy balance equation. This solver is developed by Ali Kadar in OpenFOAM 2.3.0 [14] and adapted for compatibility with the newer version of OpenFOAM 1606+. The solver also being used for execution time comparison experiment detailed in Chapter 4, which compares the computational time of given case with different complexity. The algorithm for reactingFoam/furnaceFoam is given in Figure 3.8

3.3.4. OpenFOAM Utilities

Several built-in OpenFOAM utilities are used in the present work, including mesh generator *blockMesh* and residual monitor *foamJob* and *foamLog*. Derived boundary conditions such as input-output condition and wall functions are also being used in several cases. As a reference for following chapters, this subsection elaborates several utilities especially derived boundary conditions which involve calculation.

Boundary Condition *zeroGradient*
This boundary condition imposes zero flux of arbitrary variable ϕ.

$$\nabla\phi = 0 \tag{3.15}$$

Commonly find in symmetric plane, wall, or outlet of the domain. This boundary condition assures the continuous value of variable ϕ is retained. Special cases such as zero gradient for temperature boundary indicates adiabatic wall is considered in the system.

Boundary Condition *inputOutput*
This type of boundary condition is usually used at the output of the domain. The value of arbitrary variable ϕ at the prescribed boundary depends on the direction of its flux. This allows to assuring one directional flux calculated at the domain, either only coming into the system (input) or going out from the system (output). The boundary condition is commonly used for velocity, pressure, and chemical species to retain conservation of the variable.

$$\phi \begin{cases} \nabla\phi = 0 & : \text{if} \quad \nabla\phi \geq 0 \\ \phi = c & : \text{if} \quad \nabla\phi < 0 \end{cases} \tag{3.16}$$

Boundary Conditions for Turbulence Scalars
Cases with turbulent models need a specification of turbulence scalars, for example k and ϵ in standard $k-\epsilon$ model, at the boundary patches. The values of these scalars at boundary patches can be chosen arbitrarily or using calculated initial guess from flow and domain properties. OpenFOAM utilities allow the calculation of turbulence scalars from these properties. As mentioned in [30], turbulence scalars at boundaries can be calculated as:

$$k = \frac{2}{3}\left(U_{ref} I\right)^2 \quad \epsilon = C_\mu^{3/4} \frac{k^{3/2}}{l} \quad l = 0.07L \tag{3.17}$$

where U_{ref} is the reference velocity, commonly taken as the inlet velocity; I is turbulent intensity, usually taken as a function of Reynolds number $I = 0.16Re^{-1/8}$; and l is mixing length taken as a function of characteristic length L of the equipment, or equivalent pipe diameter.

Wall Functions for Turbulence Scalars
The behavior of turbulence near the wall is different than at the middle of the flow since at the wall shear stress is higher and hence the eddies are smaller. Near the wall, the flow resembles behavior more like a laminar flow and shifting to more like turbulent flow as gaining distance from the wall. This changing behavior is captured as a wall function, where the velocity fluctuation is related to the distance from the wall. Without a wall function, this shifting behavior can not be captured. Therefore, a *wall function* is needed to calculate turbulence scalars near the wall. OpenFOAM has built-in wall functions for calculating k, ϵ, and ν_t. The calculation is derived from the equation in [30], for $k-\epsilon$:

$$u^+ = \frac{U}{u_\tau} = \frac{1}{\kappa}\ln(Ey^+) \quad k = \frac{u_\tau^2}{\sqrt{C_\mu}} \quad \epsilon = \frac{u_\tau^3}{\kappa y} \tag{3.18}$$

where von Karman's constant $\kappa = 0.41$, wall roughness $E = 9.8$ for a smooth wall, u_τ is friction velocity, y^+ is non-dimensional wall distance, and y is coordinate direction normal from the wall. While for ν_t:

$$\nu_t = \nu\left(\frac{\kappa y^+}{\ln(Ey^+)} - 1\right) \tag{3.19}$$

Wall Functions for Convective Heat Transfer
In modeling of combustion with high temperature, loss of heat to the wall shall be considered. The loss occurs as convective heat transfer to the wall. OpenFOAM allows the definition of wall heat transfer boundary by prescribing constant wall temperature and the thermal diffusivity of the wall material. This wall boundary condition is derived from OpenFOAM's *mixed boundary condition* which allows Robin type boundary condition to be applied. The wall heat transfer boundary condition is derived from:

$$T_w = f T_\infty + \left(1 - f\right) T_c, \quad f = \left(1 + \frac{k_c \delta}{\alpha_w}\right)^{-1} \tag{3.20}$$

where T_w is the temperature at the wall, T_∞ is the outer atmospheric temperature, T_c is the temperature at the center of cell next to the wall patch, k_c is effective thermal conductivity of the fluid at the wall calculated from the turbulent fluid properties library, α_w is wall thermal diffusivity, and δ is the distance between wall patch to cell center.

Boundary Condition for Radiation

Since all cases in the present work use P1 model for radiation, boundary condition for total radiation intensity G is needed. OpenFOAM can calculate the Marshak boundary condition as mentioned in equation (2.78). The implementation of this boundary condition in OpenFOAM is similar to the conductive heat transfer boundary condition, using the Robin type boundary condition.

4

Study of Computational Performance of OpenFOAM

4.1. Introduction

This chapter will elaborate the study of the computational performance of OpenFOAM. The study is carried out by analyzing the effect of adding more complexity to the solver, start from a non-reacting laminar flow, non-reacting turbulent flow, reacting flow, and an addition of radiative heat transfer. This section's objective is to know which part of solver's algorithm contributes the most of the computational cost. After knowing the crucial contributor, several attempts to improve the computational performance is made. This work is done as stepping stone for the future project so that the issue of computational performance can be handled by various strategies.

4.2. Analysis of Computational Cost Contributor

4.2.1. Simulation Setup

The experiment to compare the computational time of different cases is carried out by using a single solver, the user-defined *furnaceFoam* explained in Section 3.3.3 and turning on/off several models related to physical phenomena in each case. The motivation of doing this practice is to minimize the contribution of the different algorithm from other case-specific solvers and get the resulting comparison as purely from the complexity of the model. The solver is adapted to accommodate profiling using *gprof*, in order to get data about each computational process. The actual execution time is also being monitored and compared to see the practical comparison between cases.

4.2.2. Geometrical Mesh

To achieve a fast result, a simplified *pseudo-3-Dimensional* mesh of rotary kiln is used. This mesh features simplified boundary *patches*, representing the inlets: fuel, primary, and secondary air; walls; and the outlet. The mesh contains 4800 hexahedral cells constitute a structured mesh. The diameter of the kiln is 2.4 m with the axial length of 20.0 m. The burner sits in the radial center of the kiln, with a width of 0.5 m and length of 3.0 m long. Fuel inlet is considered as the central 0.2 m a width of the burner while the rest becomes the primary air inlet. The secondary air is located 0.2 m below the top wall and has width of 0.2 m wide. The graphical representation of the mesh is shown in Figure 4.1.

The colors of the patches represent the boundary patches:

- Red: the simplified inlet nozzle of primary air (PAir)
- Green: the simplified inlet nozzle of fuel (Fuel)
- Blue: empty faces, compensation of OpenFOAM pseudo-3D handling
- Yellow, Magenta, and Teal: the kiln walls (Wall)

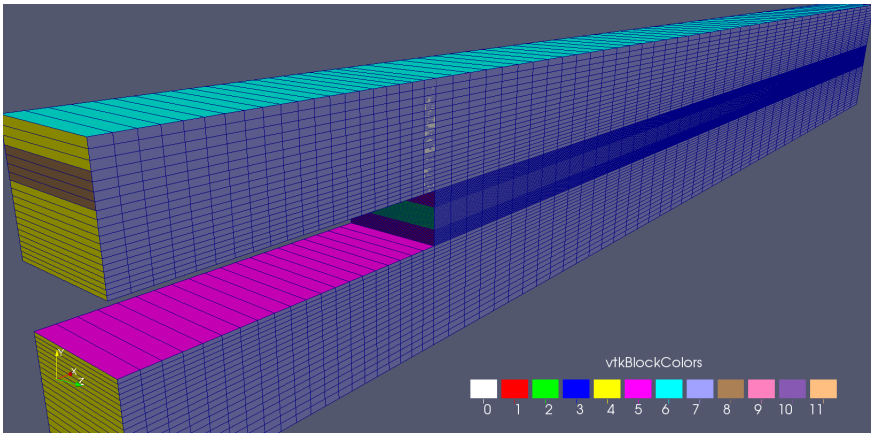


Figure 4.1: Pseudo-3D Mesh

- Violet (not visible in the picture): the outlet of the kiln (Outlet)

- Brown: the simplified inlet nozzle of secondary air (SAir)

4.2.3. Initial and Boundary Conditions

All cases using same initial and boundary conditions. Fuel inlet is taken as pure methane while air contains 23% mass of oxygen and the rest is nitrogen. Initial velocity for fuel, primary and secondary air inlet is treated as equal, 0.2 m/s in kiln's axis direction with a temperature of 500^oC $(773K)$. The initial condition of the kiln is assumed as room condition (298 K, 100 kPa) air. The system is assumed to be adiabatic where the heat can only escape at the outlet, not the wall. The complete initial and boundary conditions are given in Table 4.1.

Table 4.1: Initial and Boundary Conditions

	p(atm)	U (m/s)	T(K)	k	ϵ	G	Y_{CH_4}	Y_{O_2}	Y_{CO_2}	Y_{H_2O}	Y_{N_2}
ic	10^5	(0 0 0)	298	8.64×10^{-5}	0.00169	m	0.00	0.23	0 .00	0.00	0.77
Fuel	zG	(0.2 0 0)	773	8.64×10^{-5}	0.00169	m	1.00	0.00	0.00	0.00	0.00
Pair	zG	(0.2 0 0)	773	8.64×10^{-5}	0.00169	m	0.00	0.23	0.00	0.00	0.77
Sair	zG	(0.2 0 0)	773	8.64×10^{-5}	0.00169	m	0.00	0.23	0.00	0.00	0.77
Outlet	tP	piOV	iO	iO	iO	m	iO	iO	iO	iO	iO
Wall	zG	(0 0 0)	zG	wFk	wFe	m	zG	zG	zG	zG	zG

where abbreviation *ic* indicates initial condition, *m* denotes Marshak boundary condition (2.78), *zG* is zero gradient boundary condition as mentioned in page 33, *iO* is inlet-outlet boundary condition explained in page 34, *piOV* is special inlet-outlet boundary condition for velocity which calculates velocity based on value of *tP* total pressure, *wFk* and *wFe* is wall function for k and ϵ respectively as given on page 34.

4.2.4. Physical and Thermophysical Model

By default, the furnaceFoam treats the fluid as compressible fluid based on compressibility ψ (subsection 2.4.1). `JANAF` (2.80) and *Sutherland* (2.81) equation is used to calculate heat capacity and transport properties of the fluid. Other than the laminar case, all cases used *realizable* $k-\epsilon$ model (subsection 2.3.1) for turbulence modeling. Single step global reaction with infinitely fast chemistry model (subsection 2.3.2) is considered for combustion model, and P1 radiation (subsection 2.3.3) is used as radiation model. Four cases with different settings are simulated with active models for each case are given in Table 4.2. Positive symbol (+) means the model is active, while negative symbol (−) means the model is inactive for the given case.

Table 4.2: Active Models for Studied Cases

	Realizable $k-\epsilon$	Infinitely Fast Chemistry	P1 Radiation
Laminar	−	−	−
Turbulent	+	−	−
Reaction	+	+	−
Radiation	+	+	+

4.2.5. Simulation Control

The simulation is done in a transient state for 500 s simulation time. The time step for each iteration is adjusted automatically by the solver to maintain Courant number maximum at 0.4. The actual CPU time is measured using *foamJob* and *foamLog* command. Differencing schemes for all terms in the discretized equation and linear solver used are the same for four cases (subsection 3.3.2). Time discretization is using Implicit Euler (subsection 3.2.4) while all other terms are discretized using *Gaussian linear scheme*. Symmetric matrices such as pressure field p and total radiation intensity G is solved using *PCG* linear solver while asymmetric matrices as velocity u, enthalpy h, and turbulent scalars are calculated using *PBiCG* linear solver.

4.2.6. Result and Discussion

All cases are executed with single CPU core. The initial hypothesis is the CPU time will increase with the complexity of the case. It means, a case with more models active will needs more CPU time than the others. This hypothesis is assumed by simply considering more model to be resolved, more equation needs to be solved, hence more calculation by CPU is needed. The CPU times for all cases are plotted in Figure 4.2 while total CPU time is tabulated in Table 4.3.

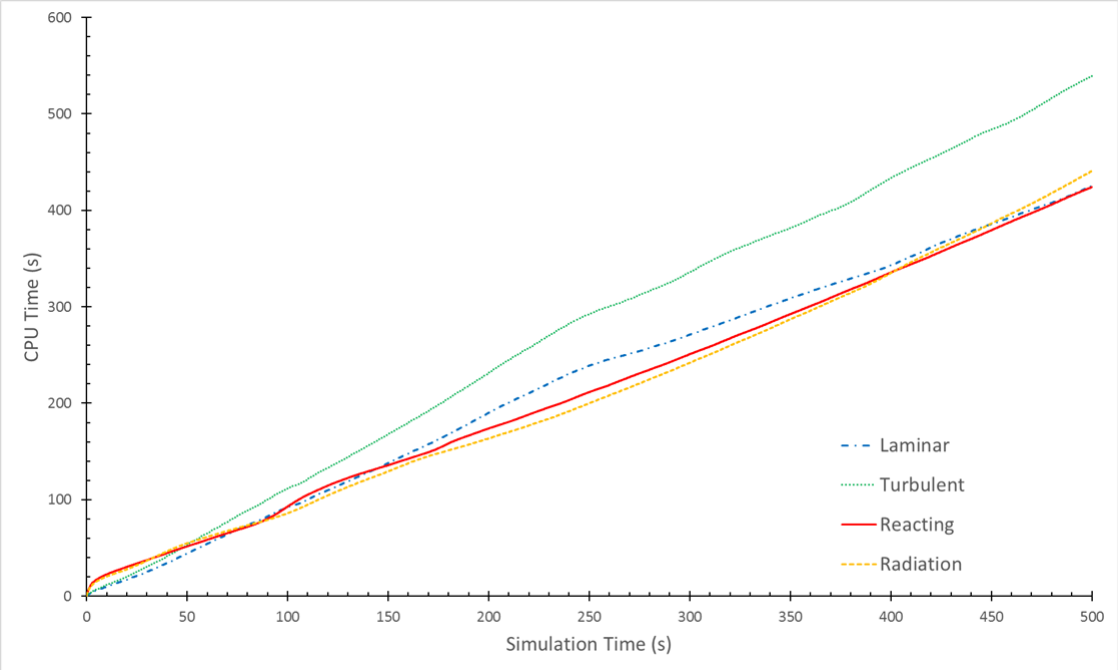


Figure 4.2: CPU Time Comparison

Table 4.3: Total CPU Time Comparison

Case	Total CPU Time (s)	Number of Time Steps	Average Time Step (s)
Laminar	425.80	14142	0.0354
Turbulent	538.96	14564	0.0343
Reaction	424.22	9655	0.0518
Radiation	440.83	9478	0.0528

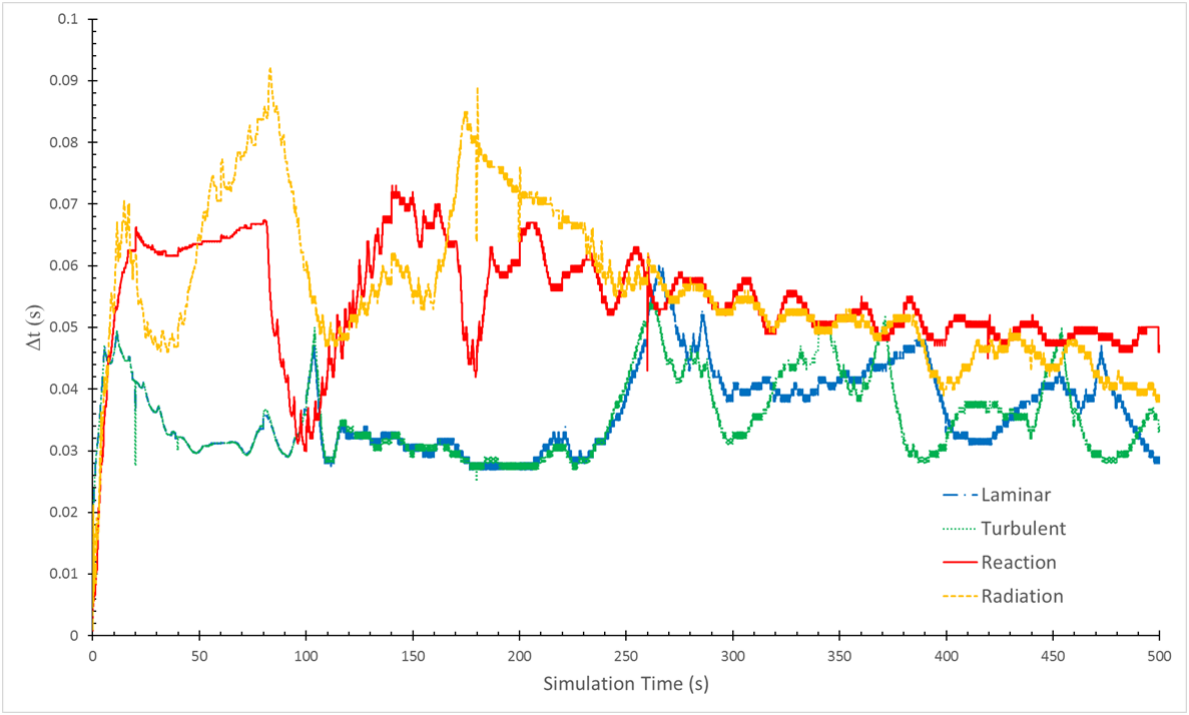


Figure 4.3: Time Step Evolution

The initial hypothesis is not proven by looking at the total CPU time result, where the case with only turbulent model, without reaction and radiation, takes highest CPU time. However, CPU time for radiation case is larger than reaction only case as predicted. The cause of this behavior is seen by looking to the number of time steps the case needed to finish the simulation, the second column of Table 4.3. Since the time step is automatically adjusted to keep maximum Courant number of 0.4, the time step is highly affected by flow velocity within the control volume cell. Courant number Co is defined as:

$$Co = \Delta t \sum_{j=1}^{n} \frac{u_j}{\Delta x_j} \tag{4.1}$$

where Δt, u_j, and Δx_j refer to time step, velocity field at j direction, and length of control volume at j direction, and since *pseudo-3D* geometry is considered, the number of dimension $n = 3$. The Courant number represent how the fluid is moving through the computational cells. If the Courant number is ≤ 1, the fluid particles move from one cell to another within at most one time step. If it is > 1, the fluid particles moves through two or more cells at each time step and can affect convergence negatively. The evolution of time step is plotted in Figure 4.3 and the average time step is given in the third column of Table 4.3.

In the laminar and turbulent cases, where no reaction is involved, the velocity component only depends on the turbulence phenomena of the flow and the mixing of the chemical species. The difference of average time step between these cases is due to resolving the effect of turbulence to species mixture which changes the transport properties of the fluid. In the reaction case, turbulent combustion gives additional energy to create more turbulence within the domain, creating more turbulent stresses, hence makes more rotation in the flow. Because the flow has more rotation, the possibility of the fluid particle passing more than one cell at one time step is decreased, even at a larger time step than the turbulence only flow. Therefore, the larger average time step is found in the reaction case compared to laminar and turbulent. In the radiation case, another physical phenomenon causes the rotation: buoyancy. An additional term of radiation results higher difference in the enthalpy, and hence temperature field. This creates higher buoyancy effect than the reaction case which creates even more rotation.

The CPU time behavior is analyzed deeper by looking at the profiling using *gprof* utility. This utility allows monitoring of CPU time used by each step in calculation algorithm and the first three result is given in Table

4.4. The % *time* refers to the percentage contribution of this process to the overall CPU time; *self-seconds* indicates the total amount of CPU time taken by this process, while *calls* gives the number of time the function is called [10]. All cases have similar three highest contributors to CPU time: operator (/ =): the division operation, operator (=): the equal operation, and *surfaceIntegrate* operator which refers to Gaussian divergence theorem (3.3) to create volume field (diffusion term) from the surface field (diffusive flux). These three operators take roughly 40% of computation time per $0.01s$ sample with same distribution. Other significant up to 1% computational time for all cases are similar, only the rank is slightly different. None of those significant operations directly mention specific operation of calculating turbulence scalars or chemical reaction. An operation explicitly mentioning correction of diffusivity due to turbulence is found in the laminar and turbulent case, but not in reacting and radiating case. However, the total CPU time contribution of this operation is only 0.02% and no call time recorded.

4.2.7. Conclusion from Computational Cost Contribution Analysis

From this findings, it is concluded that the addition of complexity of the physics, by adding more modeled phenomena, does not directly impact CPU time due to more calculation. The most significant contribution of CPU time comes from the calculation of time step from given maximum Courant number, which is affected by the physics occurs within the simulation. The highest contributor of CPU time is therefore the calculation of the flow itself. From the algorithm shown in figure 3.8, we can see that the heaviest task of flow calculation is in the pressure-velocity coupling iterative process. From the gprof analysis, surface integration operation is found to be the most expensive process. Since surface integration operation is related to the solving of governing equation by finite volume method, additional work as attempt to improve the computational performance is required.

Table 4.4: *gprof* Result

Laminar	% time	self seconds	calls
operator / =	20.36	12.23	28284
operator =	10.00	6.01	8315437
surface integral	8.69	5.22	84852
Turbulent	**% time**	**self seconds**	**calls**
operator / =	20.40	14.94	29128
operator =	11.83	8.67	10151042
surface integral	8.26	6.05	101948
Reaction	**% time**	**self seconds**	**calls**
operator / =	20.74	10.82	19310
operator =	9.78	5.11	6970845
surface integral	7.90	4.12	67585
Radiation	**% time**	**self seconds**	**calls**
operator / =	21.19	11.11	18956
operator =	9.85	5.17	7430687
surface integral	7.34	3.85	66346

4.3. Improvement of Computational Performance

4.3.1. Comparison Between Diagonal Incomplete Cholesky (*DIC*) and Generalized Geometric-Algebraic Multi-Grid (*GAMG*) Pre-conditioner

The first attempt to improve the computational performance is by changing the numerical solver used in the calculation to a more advanced solver. To minimize the complexity of the case and allows careful hand-written cross-checked can be done, the test cases are simplified to a steady laplacian cases. In these cases, the transport equation for a scalar ϕ that should be solved is

$$\frac{\partial}{\partial t}(\phi) = \nabla \cdot \left(\alpha \nabla \phi\right) \tag{4.2}$$

where α is a transport constant.

The simulations are done with meshes provided by *cfMesh*, which is based on published paper by Elattar and will be used in chapter 6. Two meshes identified as *coarse mesh* and *fine mesh* are utilized in the simulations. The *coarse mesh* contains 150 778 cells while *fine mesh* has 1 602 362 cells. The scalar ϕ is initially set as 0 at all of the internal cells. One boundary patch for ϕ is fixed at value of 1 while the other boundary patches are set to be *zeroGradient*. The transport constant α is set to be 4×10^{-5}. These values are arbitrary since the goals of the simulation are to get the system matrices and see the performance of the built-in OpenFOAM numerical solver.

The case mentioned in the beginning of this chapter uses *Diagonal Incomplete-Cholesky (DIC)* solver as pre-condition to a *Conjugate Gradient (CG)* solver. This *DIC* solver is considered as *naive*, since it only utilizes single level of modification to the matrix. A more advanced method which is available in OpenFOAM is *Generalized Geometric-Algebraic Multi-Grid (GAMG)* method. This method utilizes multi-level of matrix modification to minimize the cost of iterative solving process. In theory, *GAMG*-pre-conditioned-*CG* can reach convergence in single time-step calculation faster than *DIC*-pre-conditioned-*CG*. However, due to its complex modification to the system matrix, *GAMG* may arise an issue in a multi-core parallel computation. In this type of computation, a communication between processors is needed when processing the matrix. If this communication can't be handled efficiently, the speed that should be gained from using *GAMG* method will be bottle-necked by this communication. Test cases using two meshes mentioned earlier are constructed with exactly same problem setup. Each case is solved using *CG* solver with pre-conditioner of either *DIC* or *GAMG*. A non-orthogonal correction of 2 is considered for all cases. A variation of number of processors is also tested with 1, 2, or 4 processors. For the test case, the time integration scheme prescribed in the *fvScheme* file is set as *steadyState*, which means an iterative dummy time steps of 1 second per time step is taken. The solver is run for 1000 iterative time steps and performance comparison between two pre-conditioners are then analyzed. Since a non-orthogonal correction of 2 is considered, each case will do 3000 calculations in total.

The CPU time for both pre-conditioner to compute the cases are summarized in table 4.5. It can be seen from the table that *GAMG*-pre-conditioned *CG* takes more CPU time than *DIC* one to complete 1000 iterative time step steps. This is not confirming the former hypothesis that changing the pre-conditioner to *GAMG* can boost the performance of the solver. However, the residual at iterative time step 1000 from solver with *GAMG* is much lower at order of magnitude of 10^{-7} than the solver with *DIC* at order of magnitude of 10^{-2}. This hints that if the calculation is done for steady case with prescribed convergence criteria, solver with *GAMG* pre-conditioner will reach the convergence faster than *DIC*.

A strange behavior of *GAMG*-pre-conditioned *CG* is observed in the simulation using coarse mesh with two processors. The CPU time is enormous compared to the simulation using one or four processors. By looking at the log file, it is found that several time step needs more iteration than the others, at 1001 iteration. This indicates the solver has not reached the convergence criteria before moving on to the next iterative time step. The problem of non-converged solution is also found for calculation with 4 processors, however the number of non-converged iteration is very small compared to 2 processors calculation. The number of non-converged calculation for *GAMG* method for all cases are summarized in the last row of table 4.5.

There are two suspected cause of this problem: the usage of non-orthogonal correction and the usage of multiple processors. Another test case without non-orthogonal correction is simulated for the *coarse mesh* and *pseudo-3D mesh* with one and two processors. The number of non-converged simulation is summarized in table 4.6. From the shown value, it can be seen that there is no correlation between the non-orthogonal correction or the usage of multiple processors with the non-converged calculation by *GAMG* solver. The root cause of the problem is yet to be found and deeper learning to the OpenFOAM source code for *GAMG* solver shall be considered. However, due to the time limitation of the project, this strategy remains as a suggestion for future works.

In the end, changing the pre-conditioner from *DIC* to *GAMG* does not improve the computational performance for prescribed iteration steps. However, this setting can be beneficial for cases with prescribed convergence criteria since *CG* with *GAMG* pre-conditioner can get lower residual value than *DIC* pre-conditioner.

Table 4.5: CPU Time Comparison for *DIC* and *GAMG* Pre-conditioner with Non-Orthogonal Correction of 2(s)

Mesh Type	Coarse			Fine		
Number of Processor	1	2	4	1	2	4
DIC	716.07	199.37	93.41	7182.79	11580.6	15549.4
GAMG	6730.65	148823	2457.62	76073.2	76032.4	282526.6
Number of Non-converged Solution for GAMG	0	2324	7	2	20	430

Table 4.6: Number of Non-Converged Calculation by *GAMG*

Mesh Type	Coarse		Pesudo-3D	
Number of Processor	1	2	1	2
With 2 non-orthogonal correction	0	2324	106	58
Without non-orthogonal correction	0	772	30	19

4.3.2. Utilization of Another Solver

Another attempt to improve the computational is done by looking for another numerical routine to be compared with the performance of OpenFOAM's built in numerical solver. To do the comparison, an extraction of system matrices needs to be done. The previously mentioned laplacian case in is considered for this experiment. The matrices are extracted using *gdbOF* tools [11]. This process allows an extraction of system matrices for one iterative time step from OpenFOAM laplacian case. The resulting matrices are a sparse matrix constructed from the discretization of transport equation and a source vector from prescribed boundary conditions. These matrices is then fed to a computational solver such as Matlab's *backslash* command and more advanced solver called PETSc. An initial insight suggests that the extraction of the system matrices is successful and can be used in Matlab and PETSc. A plan to compare required CPU time to resolve one iterative time step by OpenFOAM, Matlab, and PETSc is made. However, due to the limitation of the author to PETSc solver, this plan is reserved as suggestion for future works.

5

Simplified *Pseudo-3-Dimensional* Simulation

5.1. Introduction

More than 1.5 million cells are included in full 3-dimensional rotary kiln mesh inherited from the previous project. This enormous value contributes to the very large computational cost, hence the simulation result can not be obtained quickly. To reduce the computational time, a simplified approach is taken by creating a *pseudo-3D* mesh which reflects an axial slice of full 3D mesh. This simplification comes with a cost on physical accuracy, since the third axis, the z-axis is omitted in the calculation. Turbulence is a three-dimensional phenomenon and hence reducing one dimensional axis highly affects the result. However, this approach allows fast simulation and analysis, which is important to get the idea of the reliability of the solver.

5.2. Simulation Setup

This experiment focuses on the reliability of the solver to imposed variable change. The changing variables are the availability of secondary air inlet, the inlet temperature of the fuel and primary air inlet, the usage of detailed chemistry, and the inclusion of radiation model. A logical hypothesis of the resulting profile is assumed based on underlying physics applied to the system. The *pseudo-3D* mesh used in previous chapter 4 is being used in this experiment. The result of the simulation is expected to confirm with the hypothetical assumption.

The first variable which is varied is the availability of secondary air inlet. This inlet is located on the top part of the rotary kiln, allowing additional blow of air enters the system. Without this inlet, the rotary kiln is symmetric along the axial direction. Hence, the flow profile inside the rotary kiln without secondary air shall be symmetric also. The addition of the secondary air inlet shall create disturbance in symmetric flow, pushing the flow to the bottom due to additional blow from the top.

The inlet temperature is the next changing variable. The increase of inlet temperature should also increase the maximum temperature within the calculation domain. For global single step methane combustion, it is known that the enthalpy change due to combustion is:

$$\Delta_c H^o = -890.0 kJ/mol = -55.49 MW/kg \tag{5.1}$$

Considering the mass flow rate imposed into the system, it is estimated that the maximum temperature which can be reached is $\approx 2200K$ if the initial temperature of air within the kiln and all inlet temperatures are set as room temperature $298K$. The increase of inlet temperature will not only provide higher energy inlet but also increase the viscosity and hence the thermal conductivity of the fluid. Therefore, an increase in maximum temperature and broader hot profile is expected from increasing inlet temperature case.

Two steps reaction for methane combustion implies two important factor which contributes to temperature profile:

- Although from *Hess' Law*, the total enthalpy change of a chemical reaction will remain the same regardless the reaction mechanism path, the fact that the second step is equilibrium leads to incomplete consumption of carbon monoxide to carbon dioxide. Hence, the resulting heat generated in the second step is less than theoretical value which leads to less total enthalpy change due to combustion
- Additional chemical species, carbon monoxide, in the mixture increase the bulk thermal diffusivity of the fluid. This shall result in a broader hot region in the profile.

lastly, as mentioned in section 2.3.3, the radiation term will induce higher heat loss and hence, lower the maximum temperature inside the system. A broader but cooler hot region is expected to appear in the profile due to absorption, emission, and scattering of radiative heat.

The settings for cases discussed in this chapter is summarized in Table 5.1. The positive symbol (+) indicates the case is using the setup, while negative symbol (−) means that the setup is not applied to the case.

Table 5.1: Summary of Cases

	Secondary air inlet	$T_{inlet} = 500K$	2-steps Chemistry	Radiation
1.	−	−	−	−
2.	+	−	−	−
3.	+	+	−	−
4.	+	−	+	−
5.	+	−	+	+

5.2.1. Geometrical Mesh

The initial geometry used for all cases within this chapter is the same with the *pseudo-3D* geometry used in chapter 4. However, during the experiment, it is found that the very simplified geometry leads to nonphysical behavior of the kiln. Modification of the geometry is then imposed in order to achieve more realistic behavior.

5.2.2. Initial and Boundary Conditions

The initial condition of the system is the same with the setup for the experiment in chapter 4 with an addition of carbon monoxide field for 2-steps chemistry in case 4 and 5 which is set to be zero. The inlet pressure and velocity are also the same, but the inlet temperature is set as $300K$ instead of $773K$ for all cases except case 3. In case 3, the inlet temperature of fuel and primary air is set to $500K$. Boundary conditions at the outlet and the walls for all fields are the same, with addition of carbon monoxide field which is set to be zero at all inlet, outlet boundary at the outlet, and zero gradient at the wall.

5.2.3. Physical and Thermophysical Model

The simulation is considered as turbulent combustion with the standard $k-\epsilon$ model (subsection 2.3.1) and infinitely fast chemistry model (subsection 2.3.2) for single global reaction mechanism or diffusion multicomponent (subsection 2.3.2) for two-step reaction mechanism. Radiation is modeled using P1 (subsection 2.3.3). The mixture is considered as ideal compressible gas based on compressibility ψ (subsection 2.4.1) with `JANAF` (2.80) and *Sutherland* (2.81) for calculation of bulk properties.

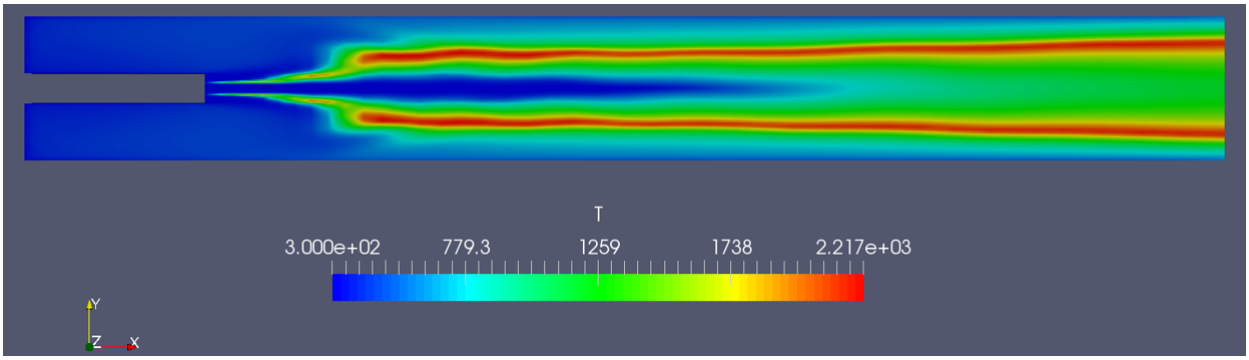
5.2.4. Simulation Control

The simulation is done in a transient mode for 500 s with adjustable time step based on *Courant number* as done in chapter 4. The schemes and linear solvers used are also similar with chapter 4: Implicit Euler for time derivative and *Gaussian linear* for other terms. The combination of *DIC* pre-conditioned *PCG* and *DILU* pre-conditioned *PBiCG* are also used for symmetric and asymmetric matrices linear solver.

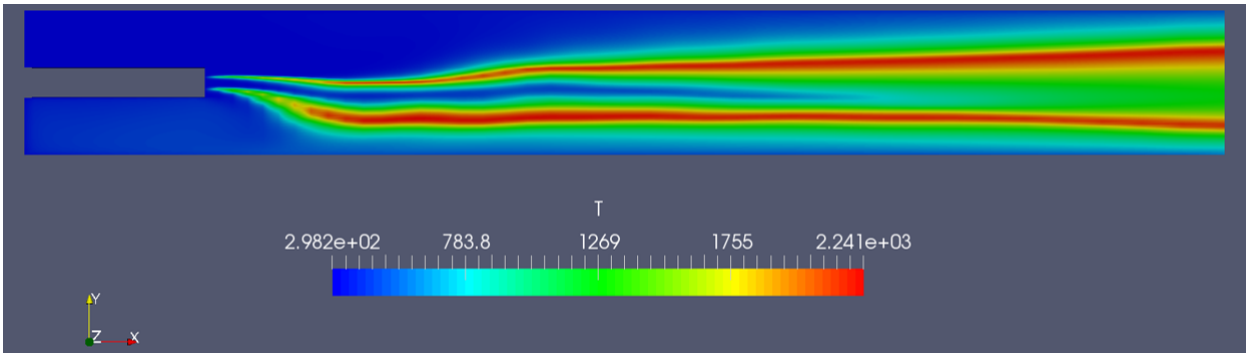
5.3. Effect of Changing Variables

5.3.1. Effect of Secondary Air Inlet

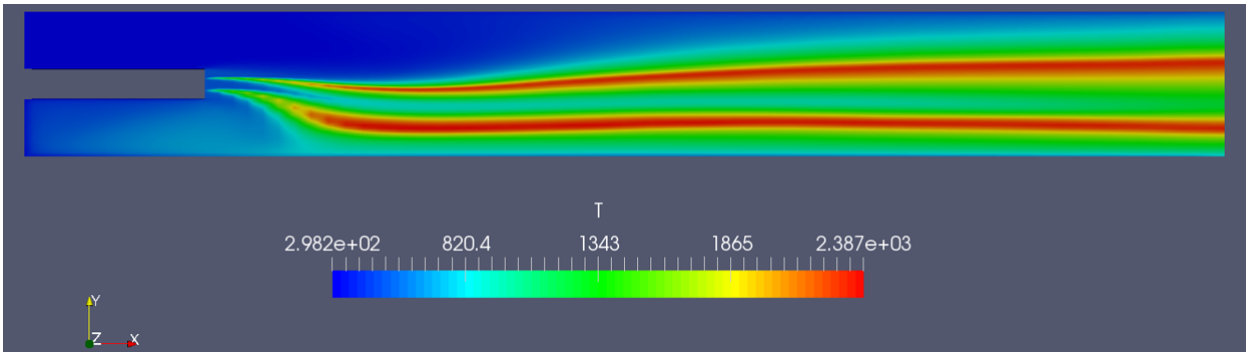
As predicted before, the temperature profile resulted from kiln without secondary inlet appears to be symmetrical along the axis. The maximum temperature measured is $2217K$ which in accordance with predicted



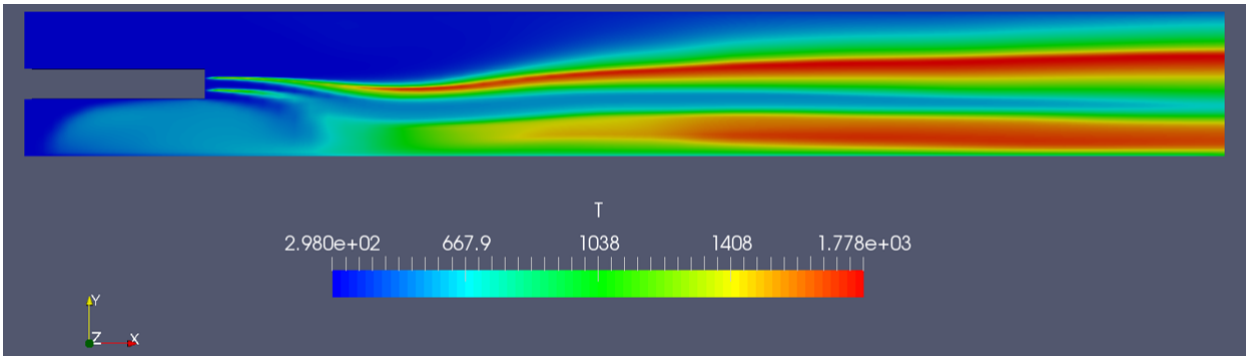
(a) *Case 1:* Without Secondary Air Inlet



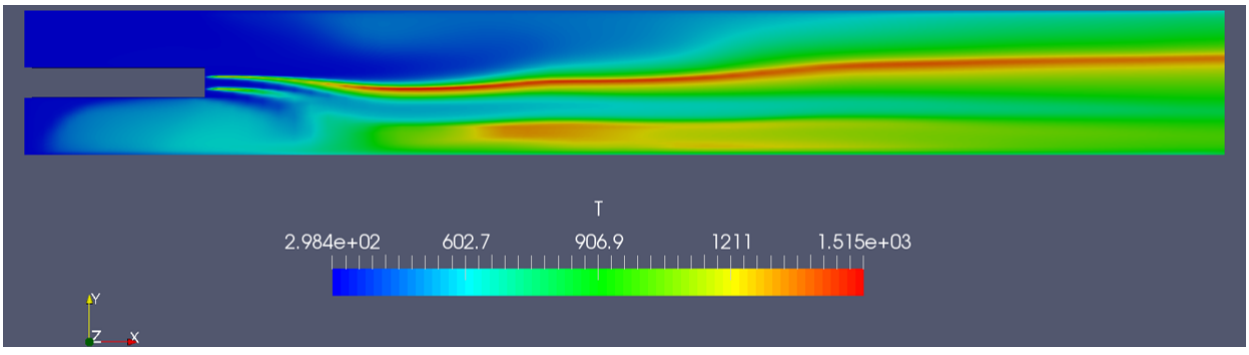
(b) *Case 2:* With Secondary Air Inlet



(c) *Case 3:* $T_{inlet} = 500K$



(d) *Case 4:* 2-Steps Mechanism



(e) *Case 5:* With Radiation

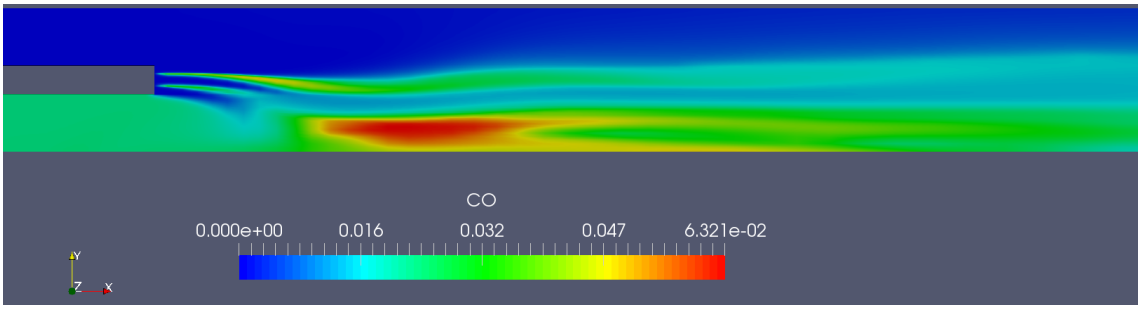


Figure 5.2: Y_{CO} Profile in 2-Steps Mechanism Case

the maximum temperature for the global single step combustion mechanism. The addition of secondary air inlet creates disturbance in the symmetry, pushing the flow downwards. The maximum temperature observed is slightly increased to $2241K$ which can be caused by more reaction occurs due to increased mixing by the secondary air blows. The kiln with secondary air inlet has higher turbulence in the upper part of the kiln and has more fresh oxygen entering the system, allowing more mass of methane burnt and creating more heat. The temperature profile of kiln for *case 1*: without secondary inlet and *case 2*: with secondary inlet is shown in Figure 5.1a and 5.1b respectively.

5.3.2. Effect of Inlet Temperature

The maximum temperature reached in this *case 3*: is $2387K$, $146K$ higher than reached by *case 2*. Although the difference in inlet temperature between *case 2* and *case 3* is $200K$, the maximum temperature difference is not. However, it is seen that the hotter regions in *case 3* are broader than in *case 2*, as depicted by Figure 5.1c. The $\sim 820.4K$ range is seen at the bottom left of the kiln, below the burner inlet, which does not appear in Figure 5.1b. The temperature along the axis for *case 3* is also higher, at $\sim 950K$ compared to $\sim 450K$. It can be concluded that the effect of thermal diffusivity change is dominant compared to the increasing inlet temperature itself.

5.3.3. Effect of Detailed Reaction Mechanism

As seen in Figure 5.1d, the maximum temperature reached by *case 4* is $1778K$, $463K$ lower than the *case 2*. The broad region below the burner is seen in the result of *case 4*. However, the green region next to it indicates that less heat is generated there, compared to the red region next to it. This phenomenon is due to incomplete combustion of methane which produces carbon monoxide occurs without further reaction to carbon dioxide. This incomplete combustion release less heat than complete combustion. This phenomenon is depicted in Figure 5.2 where the high concentration of carbon monoxide region resembles the green temperature region in Figure 5.1d.

5.3.4. Effect of Radiation

Figure 5.1e shows that the maximum temperature for *case 5* is reduced even more to $1515K$. As predicted, the radiation reduces the maximum temperature but broaden the heat region, which is depicted as more uniform temperature profile compared to other figures.

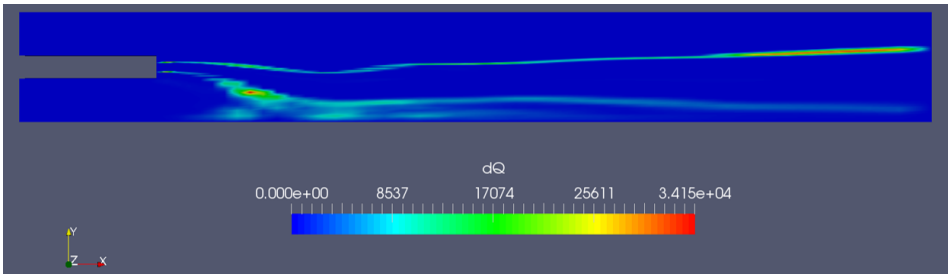
5.3.5. Conclusion on Effect of Changing Variables

The resulting temperature profile shows behavior as predicted by hypothesis in section 5.2. However, the heat profile shows unrealistic *tailing* feature of the hot region. Real-life combustion commonly has fire-front as jet near the burner which has very high temperature compared to the other region. The cause of this behavior is discussed further in the next section.

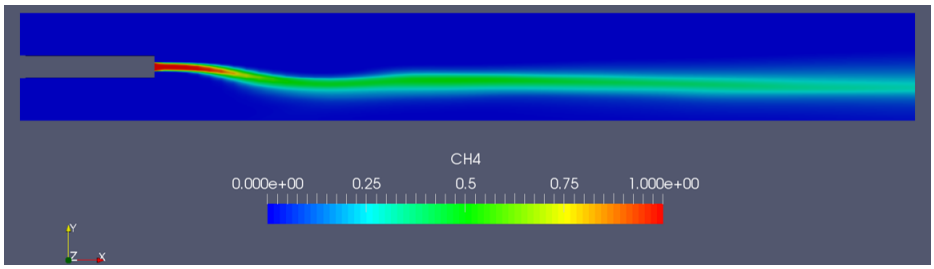
5.4. Geometry Modification

5.4.1. Hypothesis of *Tailing* Hot Region

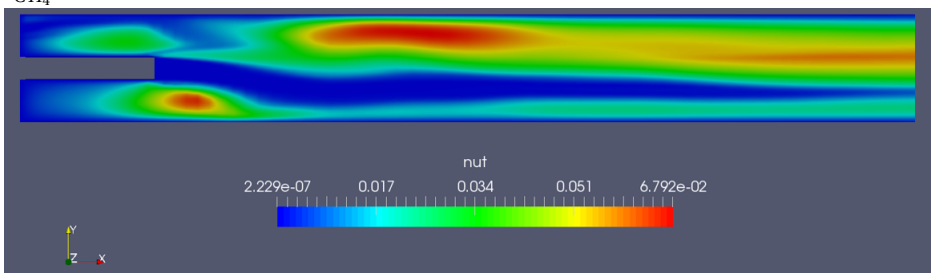
The highest temperature in the domain is related to generated heat from the reaction, since the inlet temperature is far lower than $\sim 2200K$, without generated heat, the temperature will not reach that point. Therefore, the red-hot region shall be the region where the reaction occurs. The profile of energy source term from re-



(a) Profile of Chemical Source Term in *Case 5*



(b) Profile of Y_{CH_4} in *Case 5*



(c) Profile of ν_t in *Case 5*

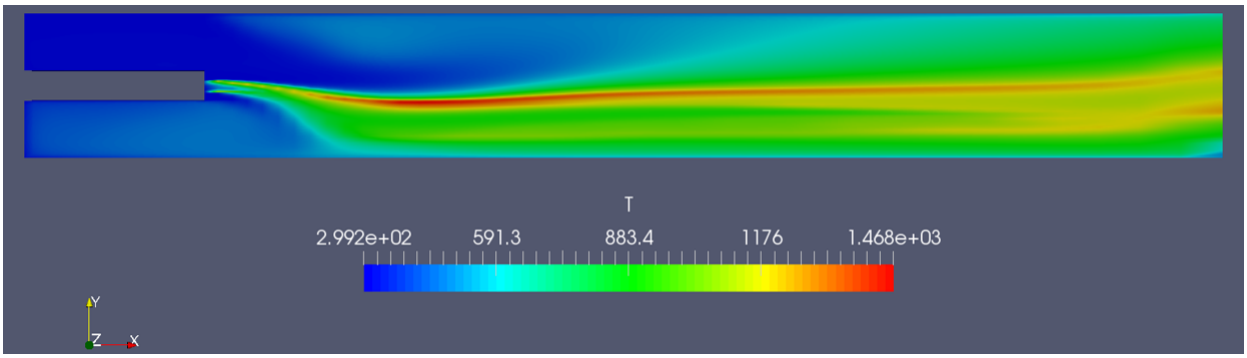
Figure 5.3: Profile of Other Variables in *Case 5*

action in *case 5*, $\dot{\omega}_{chem}$ which is denoted as dQ variable by OpenFOAM, is plotted and shown in Figure 5.3a.

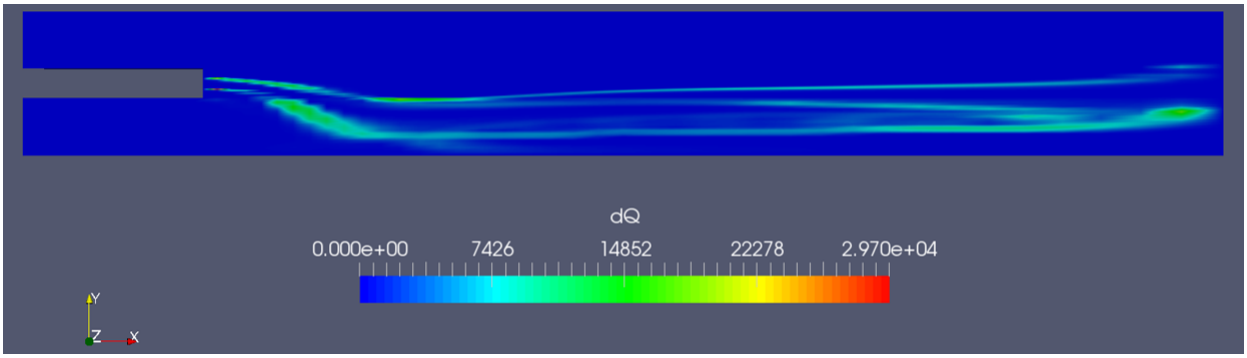
From the figure, it is seen that the reaction occurs in small reaction layer along the axis reaching the outlet. This is not depicting real-life combustion in a rotary kiln. The reaction shall occur only near the burner with a thick layer and disappear after few distances from the burner. This is due to the methane has been consumed by the air in the kiln, creating a high-temperature jet flame. The phenomena shown in the present work indicates that methane is still available along the axis and reach the outlet. As shown in Figure 5.3b, methane is indeed still available along the axis, indicated by high concentration in the green region. This means, oxygen is not mixed well with the methane, and hence only small layer of methane is consumed by the reaction. This phenomenon is well depicted from the profile of turbulent viscosity ν_t shown in Figure 5.3c. Blue region which is connected to the green region of methane concentration indicates that there is no mixing in the methane trail. Mixing only occurs between the air in the red region, which does not contribute to the combustion itself.

5.4.2. Separation of Fuel Inlet

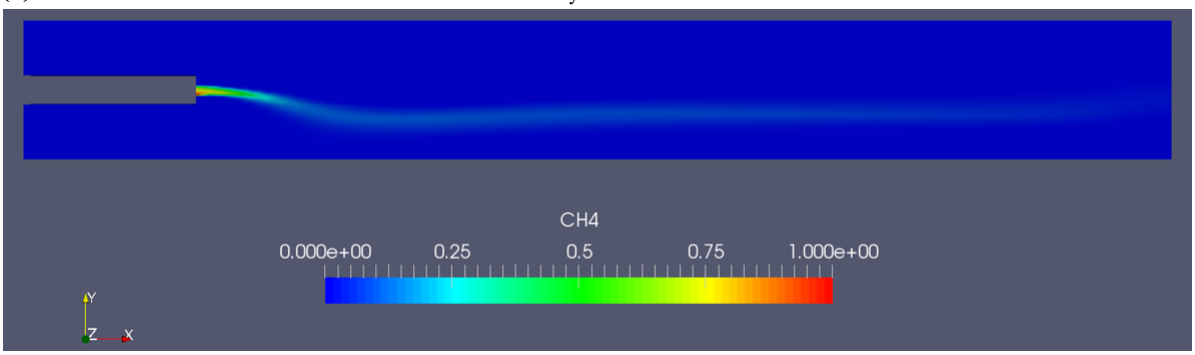
The methane profile shown here is the result of oversimplified geometry, where the fuel inlet is assumed as a $0.2m$ wide hole in the burner. The actual fuel inlet is much smaller than the assumed geometry, only several centimeters wide. The geometry is then modified to include wall between two fuel inlets, located in the middle of current fuel inlet patch and span for $0.1m$ wide, creating two $0.05m$ wide fuel inlet in its top and bottom side. The simulation similar to *case 5* is then done in this modified geometry. The resulting profile of temperature, chemical heat source, methane concentration, and turbulent viscosity profile is then plotted in Figure 5.4



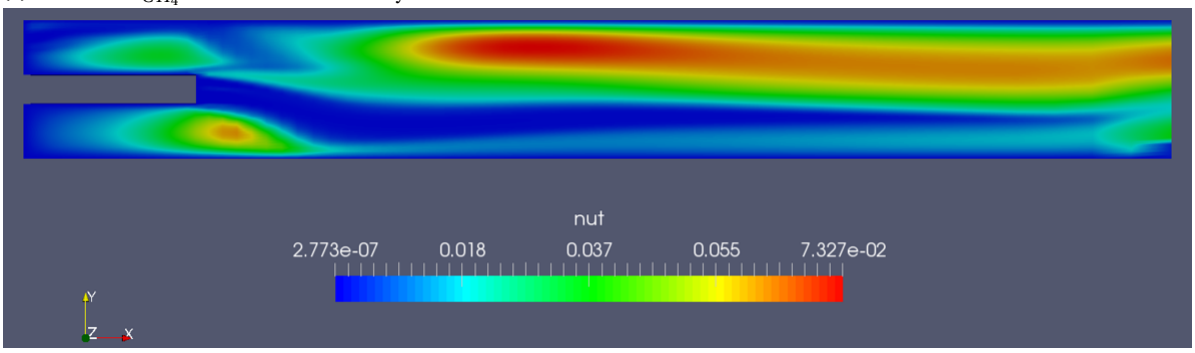
(a) Profile of Temperature in Modified Geometry



(b) Profile of Chemical Source Term in Modified Geometry



(c) Profile of Y_{CH_4} in Modified Geometry



(d) Profile of v_t in Modified Geometry

Figure 5.4: Profile of Variables in Modified Geometry

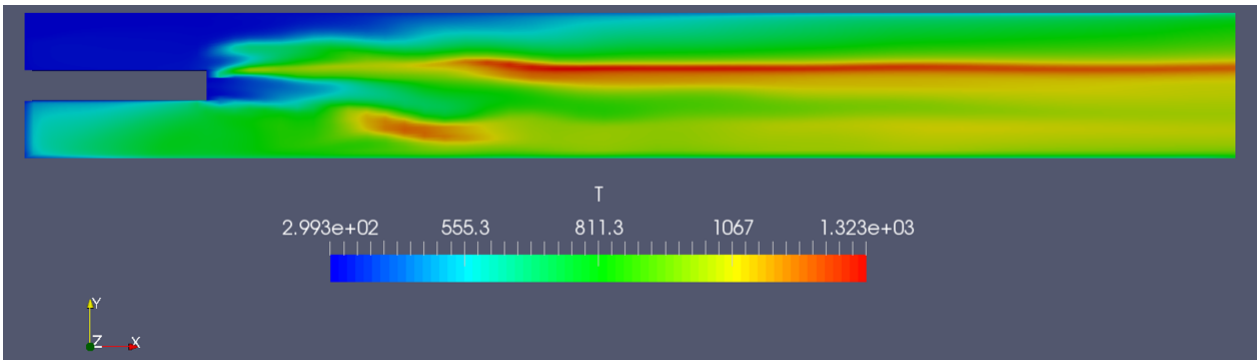
As shown in Figure 5.4c, the concentration of methane is drastically dropped along the axis, diminishing green region in the previous simulation. However, the mixing between methane and oxygen is still not enough, depicted by the profile of turbulent viscosity in Figure 5.4d, resulting in still a thin layer of the reaction zone (Figure 5.4b) and hence still tailing temperature profile (Figure 5.4a). Although the introduction of wall between the inlet and decrease the fuel inlet diameter reduce tailing methane drastically, another modification to the simulation is needed in order to achieve higher mixing between methane and oxygen.

5.4.3. Introduction of Fuel Inlet Angle

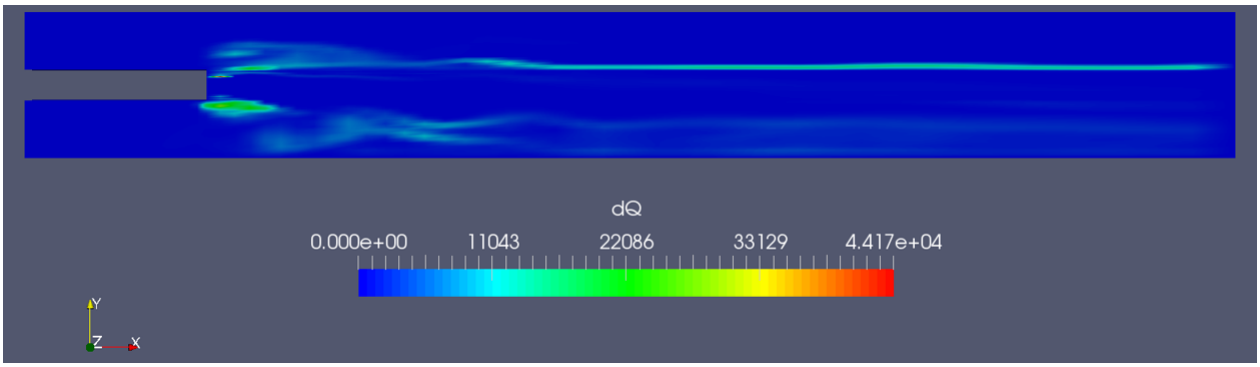
Another modification is introduced to the experiment, not by further changing the geometry, but by making the fuel inlet has angle from the axis. This approach is taken since some rotary kilns have this feature in their burner. A 30^o angle towards the wall, upward for the upper fuel inlet and downward for the lower fuel inlet, is applied to fuel inlet velocity. The resulting profile shows higher mixing near the burner inlet as shown in Figure 5.5d leads to even smaller trace of methane after several distance from the burner (Figure 5.5c). However, the reaction layer is still thin (Figure 5.5b), resulting tailing behavior of hot region remains to be seen in Figure 5.5a. This result might be caused by not enough primary air is supplied for methane combustion and hence, oxygen from secondary air inlet is needed to complete the combustion. This analysis is based on more reaction layer appears in upper part of the axis compared to the bottom in Figure 5.5b. Modification of air supply can be considered, but the result from *pseudo-3D* simulation has provided enough factor to be considered in full 3D simulation, including this modification to the primary air supply.

5.5. Conclusion from *Pseudo-3D Simulation*

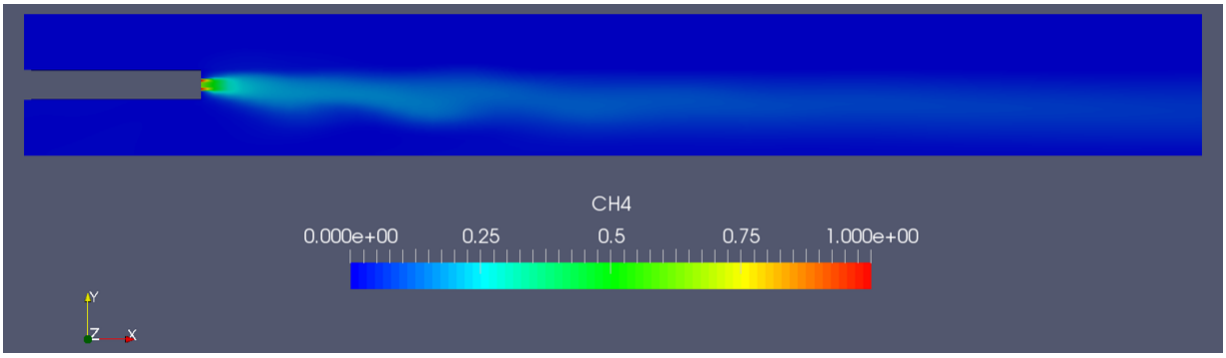
The combustion solver is shown its reliability on capturing the effect of changing variables shown in section 5.3. The resulting temperature profiles from the simulations are in accordance with the hypothesized behavior. However, the unrealistic tailing feature of the hot region is found and might be caused by oversimplification of the geometry, resulting on unconsumed methane up to the outlet of the kiln. An effort to adapt geometry to be more realistic is approached by introducing separating wall between two smaller fuel inlets and angle of inlet velocity. These modification shows a reduction of unreacted methane concentration but still unable to diminish the tailing profile of the hot region. Another possibility cause for this behavior is the insufficient primary air supply to the system, which will be considered in the full 3D simulation.



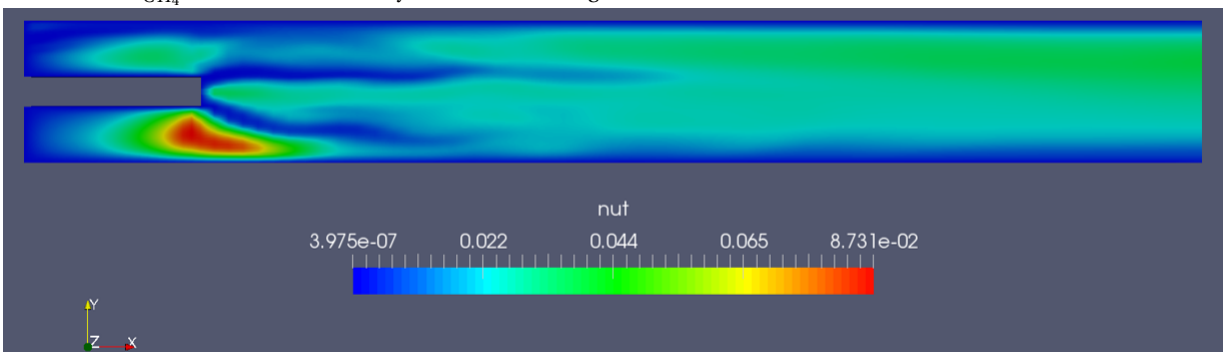
(a) Profile of Temperature in Modified Geometry with Fuel Inlet Angle



(b) Profile of Chemical Source Term in Modified Geometry with Fuel Inlet Angle



(c) Profile of Y_{CH_4} in Modified Geometry with Fuel Inlet Angle



(d) Profile of v_t in Modified Geometry with Fuel Inlet Angle

Figure 5.5: Profile of Variables in Modified Geometry with Fuel Inlet Angle

6

Reconstruction of Published Data

6.1. Introduction

Another approach to test the reliability of the solver is by reconstructing published data and compare generated a result with that. Elattar and Specht [9] has a recent publication in non-premixed turbulent combustion in an industrial kiln, similar to the present work. The main differences of the present work with the published paper other than the operating condition are the geometry and the CFD software. The geometry features axisymmetric cylindrical kiln without small secondary air inlet as the present work considers, allowing the creation of smaller mesh as calculation domain instead of treating full geometry as the domain. Elattar used commercial software ANSYS Fluent as their CFD toolbox which gives the opportunity to test the reliability of the open source solver. However, the computational power provided by DIAM in the laboratory hinders the reconstruction of precise work by Elattar. The published paper by Elattar used 40 MW burner power of methane combustion which injected to a very small nozzle, resulting on high-velocity inlet. As seen in chapter 4, OpenFOAM solver uses adjustable time step based on maximum Courant number in the cell. The high velocity used by the published paper is then raising two conflicting issues:

- High velocity means high Reynolds number, and hence high turbulence. The length of cells Δx_j shall be small enough to capture turbulence modeled within the control volume. However, smaller cells mean larger matrices to be resolved and hence more expensive computational cost.
- To keep Courant number below the maximum, the time step Δt shall be small, which contributes to more time step iterations and hence longer CPU time as seen in chapter 4

To alter this issue, the present work considers lower burner power of 1% and 10% of the published paper, allowing faster simulation and analysis. The result, however, is not directly comparable as the variables are not exactly the same. Albeit, similar features in the profile of fields is expected as the present work models consider same physical phenomena with the published paper.

Another approach to test the reliability of the software is by comparison with work of *Dr. Marco Talice* [3] [2], an expert in Computational Fluid Dynamics and has its own code which had been proven in industrial use. His solver, however, can only resolve the flow dynamics without reaction contribution, and hence adapted case from Elattar's paper is considered.

This chapter contains two separated works as attempts to reproduce published data by Elattar in combustion modeling and comparison test with proven software in buoyant flow modeling.

The two cases considered in this chapter is based on adapted operating condition of work by Elattar [9]. The published paper mentioned the fuel burner power of 40 MW and 10 MW with pure methane is considered in the fuel nozzle inlet. An air to fuel equivalent ratio $\lambda = 1.12$ with variations of primary to secondary air ratio α are taken as the value to determine the amount of air within the system. As mentioned before, the present work will use 1% of the fuel burner power with a variation of α of 0.1, 0.5, and 1.0 are taken for the combustion case. For the buoyant flow case, burner power of 1% and 10% are considered but only taking $\alpha = 0.1$ for the

air ratio.

The present work for combustion tried to use physical models as similar as possible with Elattar's work, although several models are needed to be adapted to the current limitation of OpenFOAM. The turbulence model for both the present work and Ellatar's paper is Realizable $k-\epsilon$ (subsection 2.3.1). P1 model (subsection 2.3.3) is taken as the radiation model for both works. However, Ellatar's work is using PRESTO and SIMPLE in the pressure-velocity field coupling algorithm, while the present work is using PISO algorithm which is built-in to the reactingFoam solver. Furthermore, probability distribution function (PDF) of non-premixed combustion is taken as combustion model in Elattar's paper while infinitely fast chemistry (subsection 2.3.2) is chosen as the combustion model for the present work. The last two discrepancies are due to the limitation of OpenFOAM support for the used model in Elattar's work. This can be seen as the chance of comparing the reliability of open source software to a commercial one.

For the buoyant flow case, both incompressible and compressible flow assumption are tried to see how much discrepancy arise from physical simplification in the incompressible solver. The Spalart-Allmaras turbulence model (subsection 2.3.1) is used. The axial profiles of temperature and velocity magnitude are used as the benchmark of the open source solver performance compared to the proven software.

6.2. Buoyant Flow Validation with Code by *Dr. Talice*

6.2.1. Geometrical Mesh

The geometry for this case is created based on the definition of *Burner A* in Elattar's paper. This geometry features axisymmetrical cylinder with one fuel inlet and one primary air annulus. The cylinder has a diameter of 4 m and spans for 40 m long. The burner has a fuel inlet with 50 mm diameter surrounded by primary air annulus with 100 mm inside diameter (ID) and 200 mm outside diameter (OD). The open space between primary and secondary air inlet is not mentioned explicitly in the paper but assumed as 100 mm wide. This burner located 2 m from the cylinder inlet face, but the generated mesh create the boundary patch parallel to the cylinder face to minimize the numerical effect of inlet condition [9]. The rest of inlet face of the cylinder is treated as a secondary air inlet and the opposite side is taken as the outlet patch.

Since the kiln is axisymmetric, calculation of partial domain of the kiln using *cyclic* boundary condition is possible. The present work uses 1/8 radial region of the whole kiln as the calculation domain, constituting 45^o part of the cylinder. This partition is also being used by Elattar in his work. However, Dr. Talice uses 1/2 of the cylinder as his computational domain. The present work domain is created with OpenFOAM utility *blockMesh*, creating structured mesh with 175 500 cells compared to 122 324 unstructured cells used by Elattar. Dr. Talice used $\sim 1 million$ cells in his domain. The experiment for 10% burner power requires finer mesh, and a refinement for the mesh up to 491 400 cells is done to accommodate the higher velocity in this case. Figure 6.1 depicted the structured mesh used in the present work.

6.2.2. Initial and Boundary Conditions

The boundary conditions for this case are calculated either with the burner power of 0.4 MW, (1% of Elattar's condition) or 4 MW, (10% of Elattar's condition). These burner power, combined with primary to secondary air mass ratio $\alpha = 0.1$ and equivalent air to fuel ratio $\lambda = 1.12$ are translated to mass flow of methane fuel, primary air, and secondary air. These mass flows is converted to volumetric flow rate using density of the respective fluid at prescribed inlet temperature, 20^oC for burner inlet and 250^oC for secondary air inlet. The inlet velocity of the three inlets are then calculated using their inlet area and assumed to be only in x-direction, the cylinder axis. Although this case assumes air only as the fluid, the boundary conditions are still based on the velocity calculated above, not the mass flow rate. The initial guess for turbulent viscosity ν_t is taken as 50% of the bulk viscosity $\nu = 1.8 \times 10^{-5} m^2/s$

6.2.3. Physical and Thermophysical Model

The fluid is assumed as air only, without defining any species in the thermophysical model. The transport properties of the fluid is calculated using *Sutherland* model with constant Prandtl number $Pr = 0.73$ to calculate the thermal diffusivity α. Constant thermal capacity of $1004.5 J/K$ and ideal gas behavior is considered.

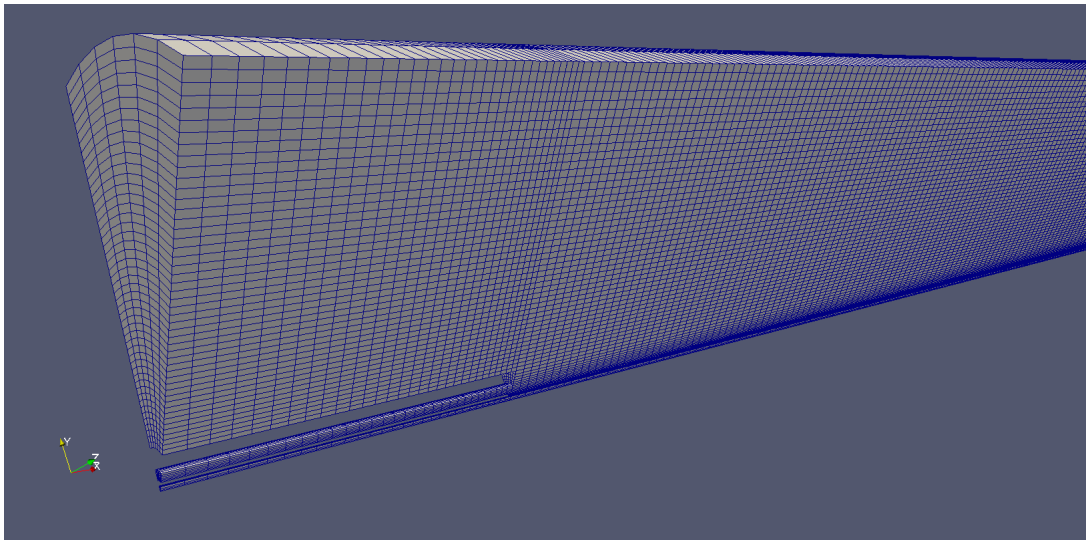


Figure 6.1: Reconstruction of Elattar's Mesh

6.2.4. Solver Setup

The simulation is done in steady state condition, hence the accumulation which contains derivation in time is considered to be zero. Two solvers are considered for the experiment, buoyantSimpleFoam 3.3.3 which solve buoyancy driven flow in compressible fluid, and buoyantBoussinesqSimpleFoam which is simplified version of buoyantSimpleFoam where the fluid is considered incompressible. These solvers compute continuity (2.1), momentum (2.8), and energy balance (2.20) without reaction and radiation source term. In the incompressible solver buoyantBoussinesqSimpleFoam however, the energy balance equation is simplified, by using temperature scalar transport instead of enthalpy:

$$\underbrace{\frac{\partial}{\partial x_j}\left(u_j T\right)}_{\text{convection}} - \underbrace{\frac{\partial}{\partial x_j}\left[\alpha \frac{\partial T}{\partial x_j}\right]}_{\text{diffusion}} = 0 \tag{6.1}$$

The buoyancy effect to the pressure field on momentum equation is also simplified with Boussinesq approximation [7]:

$$\underbrace{\frac{\partial}{\partial x_j}\left(u_i u_j\right)}_{\text{convection}} - \underbrace{\frac{\partial}{\partial x_j}\left[\nu \frac{\partial u_j}{\partial x_j}\right]}_{\text{diffusion}} = \underbrace{-\frac{1}{\rho}\frac{\partial}{\partial x_i} p}_{\text{pressure}} - \underbrace{g_i \beta \left(T - T_{ref}\right)}_{\text{buoyancy}} \tag{6.2}$$

where g_i is gravity acceleration in direction i, while $T_ref = 300K$ and β refers to reference temperature and thermal expansion coefficient from Boussinesq approximation:

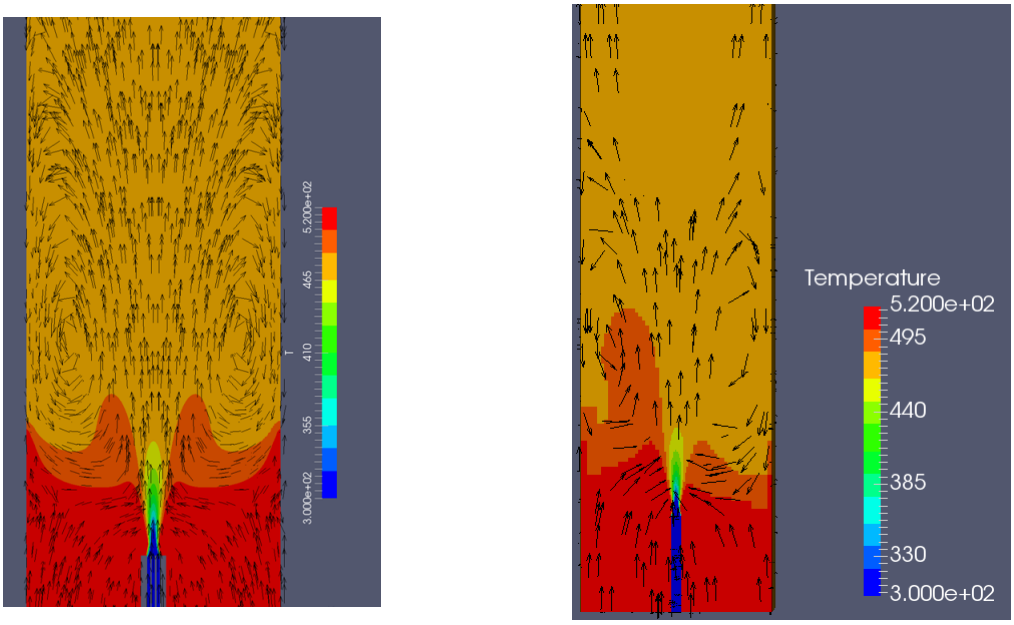
$$\beta = -\frac{1}{\rho}\frac{\partial \rho}{\partial T} = 3 \times 10^{-3} \tag{6.3}$$

The initial setup for differencing schemes used for all cases in this experiment are *Gaussian linear* for diffusive terms and *TVD* for convective terms. *GAMG* is used as linear solver for pressure field, *Gauss-Seidel* for Spalart-Allmaras variable $\tilde{\nu}$, while *PBiCG* is used for all other fields such as enthalpy, velocity, or temperature. However, in higher velocity cases, this setting is changed to compensate divergence in the calculation.

6.2.5. Result and Discussion

The simulation result for burner power of 0.4 MW, referred to *V100 case*, from the author and Dr. Talice is discussed first. From quick estimation using mass balance, the steady state temperature at the outlet should reach:

$$T_{steady} = \frac{\dot{m}_{fuel} \times T_{fuel} + \dot{m}_{pair} \times T_p air + \dot{m}_{sair} \times T_{sair}}{\dot{m}_{total}} \approx 480K \tag{6.4}$$



(a) Present Work

(b) Dr. Talice

Figure 6.2: Axial Slice of Kiln

Figure 6.2 shows the axial slice of the kiln from a converged simulation done by the author using compressible solver and simulation by Dr. Talice. Similarities in the velocity field and temperature profile can already be seen. In the converged steady state, all region after the burner is heated to around $480K$ as estimated before. To obtain more detailed comparison, the profile of temperature and velocity magnitude along the central axis of the kiln is plotted in Figure 6.3. The index *literature*, *compressible*, and *incompressible* refer to result by Dr. Talice, the result using compressible solver *buoyantSimpleFoam*, and result using incompressible solver *buoyantBoussinesqSimpleFoam*. The plot shows highly similar profile between result by the author using compressible solver and results from Dr. Talice in both temperature and velocity profile. A slight discrepancy between the two results is found in velocity magnitude at the burner pipe, the first two meters of axial distance. This difference is caused by two factors: the mesh and the given boundary condition. Dr. Talice used a finer mesh which can capture more turbulence, and hence more fluctuation of the velocity, than the author. Dr. also imposed constant zero value for Spalart-Allmaras variable at the burner wall while the author imposed wall function of turbulent viscosity (subsection 3.3.4). It can be seen that the velocity magnitude profile from the author is smoother than the Dr. Talice's profile due to less turbulence is captured. Other than this first two meters, the profile is almost similar, even the gradient of both temperature and velocity magnitude is very close. The incompressible solver, however, produces a very different result with two others, since it simplifies the effect of buoyancy and diminishes the fluctuation in diffusivity due to density change. It overestimates the thermal diffusivity value leading to higher thermal diffusion and hence higher steady state temperature at the outlet. This shows that the compressible solver from OpenFOAM is reliable enough to be compared with the proven solver. To increase the confidence level, another simulation using 4 MW burner power, referred to *V10 case*, is done by both the author and Dr. Talice.

The result for *V10 case* resembles the *V100 case* where the compressible solver achieve very similar result with Dr. Talice's simulation while the incompressible solver still overestimates the temperature profile. It is then concluded that the compressible solver is reliable enough to handle buoyancy-driven flow in the kiln. During the *V10 case*, since the velocity is higher than *V100 case*, a problem occurs which cause divergence in the computation. The error indicates that the calculation of density ρ contains a negative value which is not physical. This value is calculated at the fuel inlet, where the velocity is very high due to its small nozzle diameter. It is then found that the gain of this divergence is due to used linear solver in the simulation. A modification of relaxation factor for ρ calculation is then made, and the convergence is able to achieved.

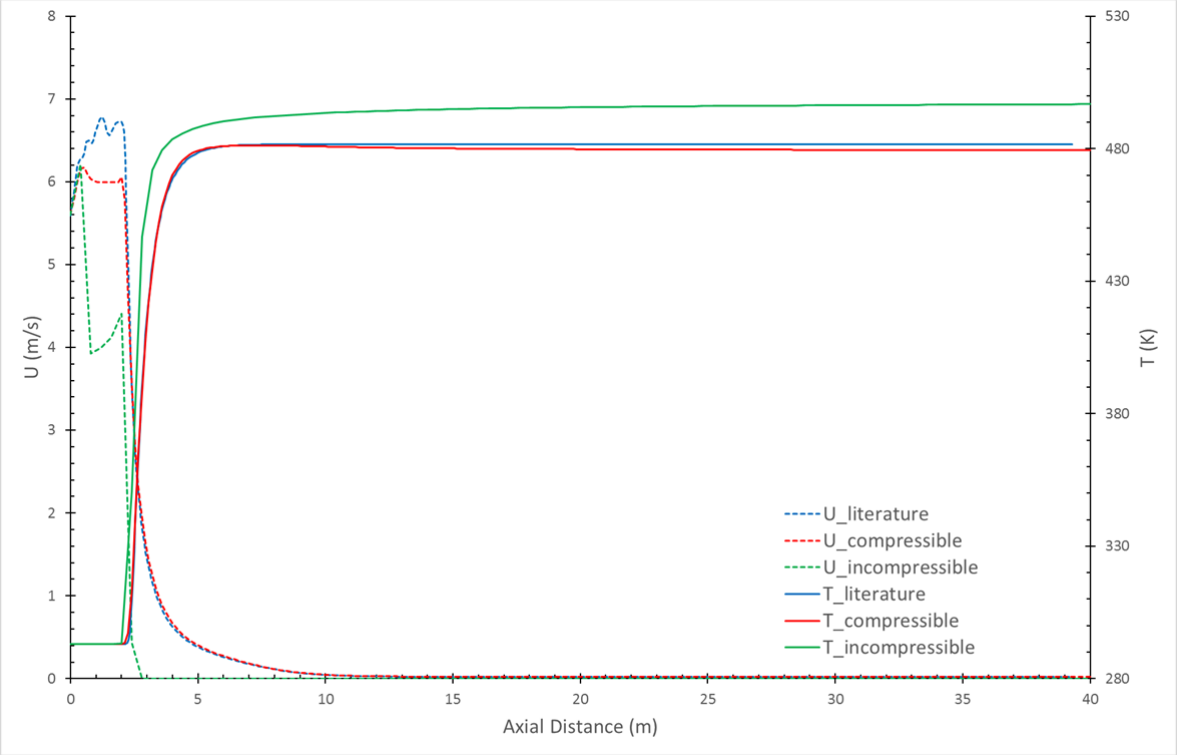


Figure 6.3: Plot of Temperature and Velocity Magnitude at Central Axis of Kiln for V100 Case

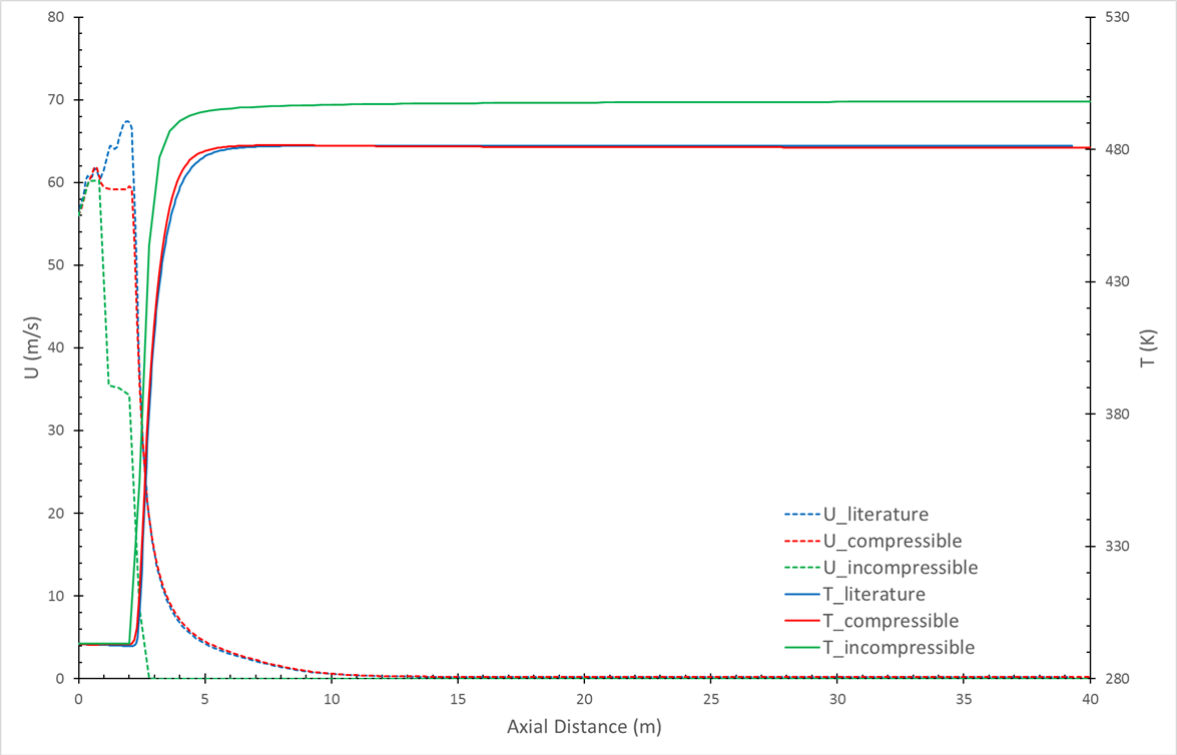


Figure 6.4: Plot of Temperature and Velocitu Magnitude at Central Axis of Kiln for V10 Case

6.2.6. Reconstruction with 40 MW Burner Power

Another case for real burner power 40 MW, which is then referred as *V case*, is planned. However, this case has more difficulties than the two previous cases, since the velocity reaches the value of $559 m/s$ which is more than the speed of sound at the prescribed temperature and pressure. In the other word, the condition for real burner power case is *supersonic*. This high velocity imposes the threat of non-physical calculation, especially using Spalart-Allmaras model for turbulence which is originally developed for *subsonic* cases. As hypothesized, the simulation of *V case* using compressible solver and same solver setting with previous cases results stuck in a non-convergence loop. Several attempts of altering the setting and geometrical mesh are executed to achieve converged solution.

Density Field Relaxation Factor Modification

The error emerged in the usage of the previous setting is observed to be caused by exploding value of ρ. This phenomenon is related to the calculation of temperature and pressure field in the flow. Due to very high velocity in a narrow channel, a high pressure gradient is observed in the fuel inlet region. This high pressure gradient leads to unstable density calculation at the region which blows the simulation up. One alternative solution for this issue is by modification to the relaxation factor for density field up to 10^{-6}. This modification provides more stable calculation, but since the relaxation factor is too low, the convergence is not obtained. This is denoted by a high residual value of pressure and continuity equation. If the relaxation factor is increased, the unstable calculation of density reappears leading to the same error as the previous setting.

Boundary Condition Modification

Dr. Talice suggests imposing fixed value boundary condition for pressure at the inlets instead of zero gradient. The previous boundary condition only imposed fixed velocity value and calculates pressure field based on the momentum balance. This imposed fixed value boundary is expected to increase the stability of calculation especially at the narrow fuel inlet, where high pressure gradient is observed. The simulation remains unstable and indicates error arose due to the calculation of turbulent viscosity ν_t at the wall using wall function. The wall function utilizes calculation of dimensionless wall distance y^+ which is calculated as a function of the velocity near the wall. Modification to the imposed turbulent viscosity field at the boundary is made. The boundary condition is changed to be fixed value of zero instead of wall function boundary condition. This suggestion from Dr. Talice accounts for the ability of Spalart-Allmaras model which includes calculation of the turbulent viscosity near the wall by the model itself. However, non-converged simulation still occurs even after the boundary condition modification.

Geometrical Mesh Modification

Another suspect that is considered to give accounts to unstable simulation is the mesh quality. Due to the limitation of the author who can only create a structured mesh by using the *blockMesh* utility from OpenFOAM, the quality of the mesh could not be guaranteed to be sufficient for the current simulation. This suspicion arose after observing high pressure gradient at the narrow channel near the fuel inlet. Several new meshes are provided by professional CFD company *Creative Fields Mesh, cfMesh* ranging from a coarse mesh which contains around 150000 cells to a fine mesh which has around one million cells. The meshes feature local refinement at the fuel inlet up to mixing zone. The coarse mesh is shown in figure 6.5 while the finer mesh is shown in figure 6.6. The usage of the new mesh improves the pressure field calculation, indicated by no high pressure gradient is found in the region near fuel inlet shown in figure 6.7. However, despite being able to overcome high pressure gradient problem, the convergence is still not achieved. The error found mentioned a problem with calculation of the thermodynamics, which is related again to the density of the fluid.
A modification in the shape of fuel inlet had also been made. This modification allows Dr. Talice to get a converged result shown in figure 6.8. The modification is done by shortening the fuel inlet. This is done because the problem of instability appears in the region of fuel inlet. High pressure gradient as found before indicates that the fluid is trapped in the fuel inlet channel. By making the fuel inlet shorter, the fluid can escape from the narrow channel and interacts with the fluid inside the kiln body. The resulting pressure field from Dr. Talice simulation as shown in figure 6.8b indicates that indeed the high pressure gradient is still found near the burner outlet area. However, since the inlet pipe is short, the flow is able to escape this region. Several modifications of fuel inlet channel had been tried, from halving the length, up to omit the fuel inlet entirely, allowing the fuel inlet patch to be put directly to the kiln body. A mesh provided by the PhD student in our department, El Abbassi which features fine mesh without fuel inlet pipe shown in figure 6.9 has also being used. However, the modification still creates the same error in the calculation of density.

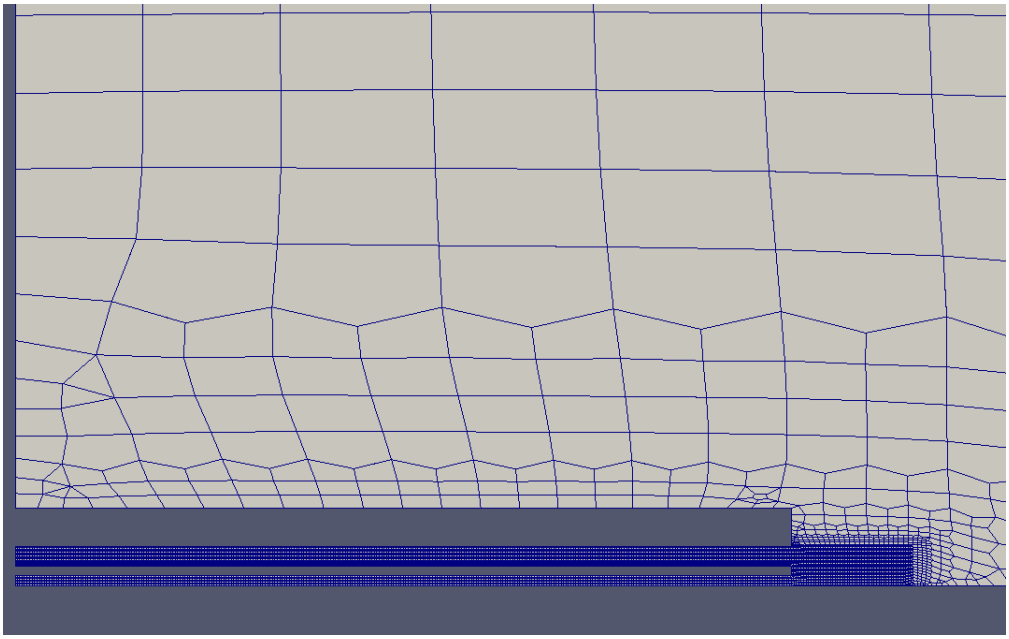


Figure 6.5: Coarse Mesh Provided by cfMesh

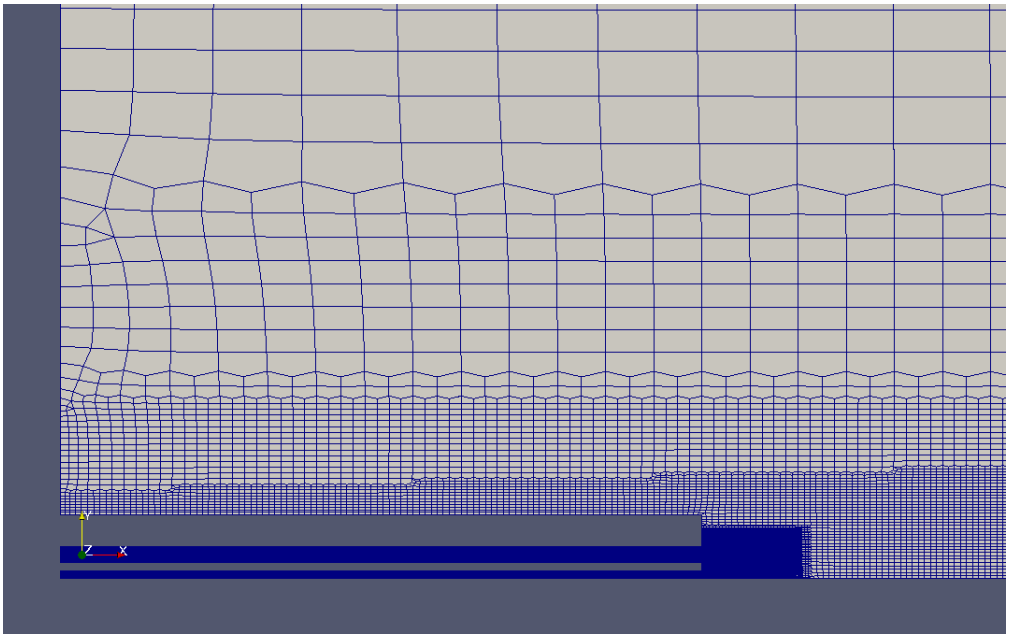
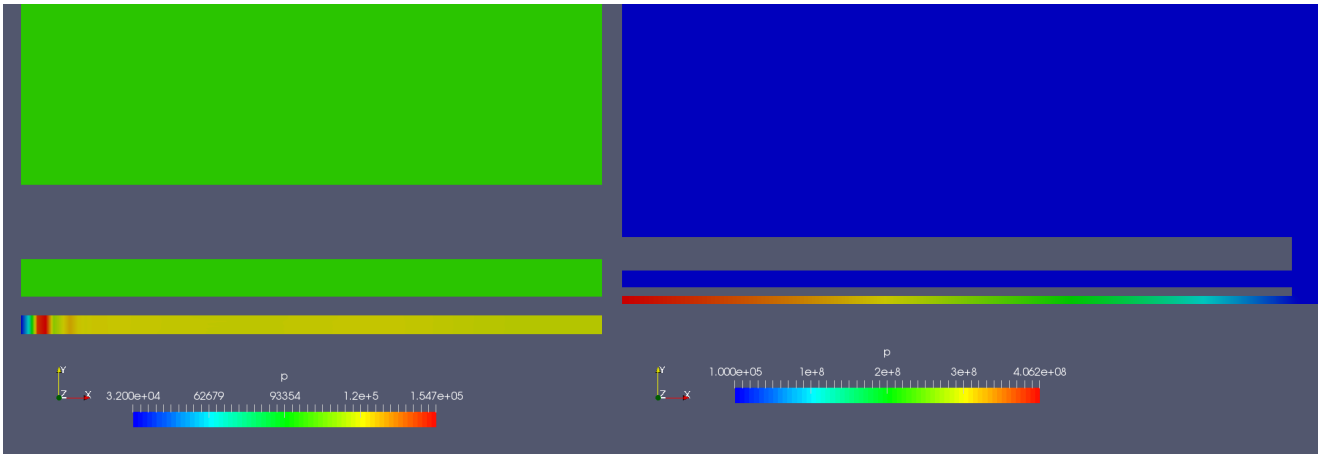
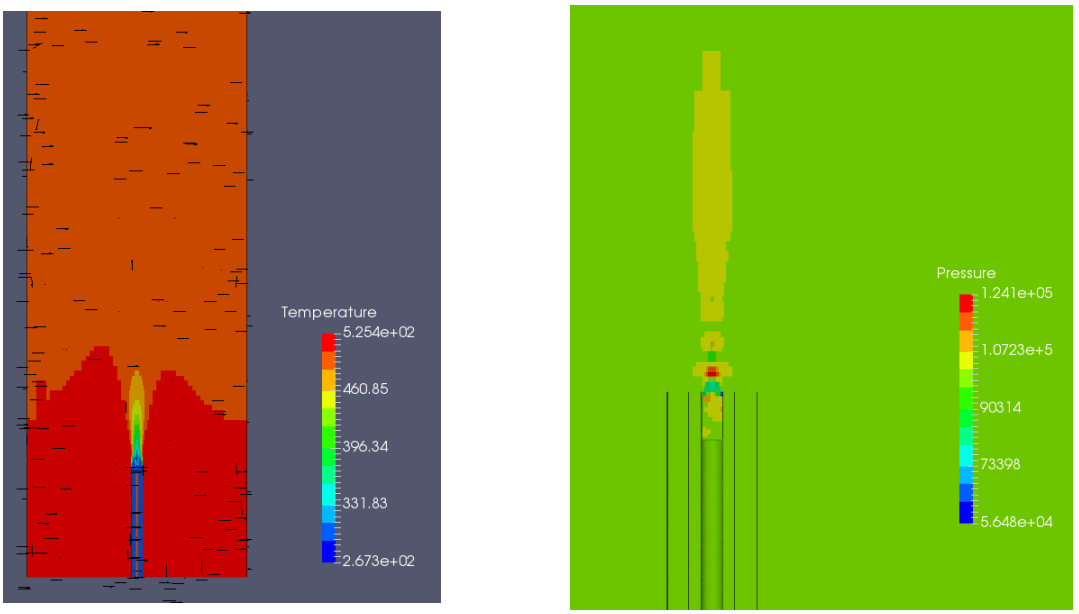


Figure 6.6: Fine Mesh Provided by cfMesh



(a) Mesh by Author (b) Mesh by cfMesh

Figure 6.7: Pressure Field in Two Different Meshes



(a) Temperature Profile (b) Pressure Profile

Figure 6.8: Result of *V case* by Dr. Talice After Modifying the Fuel Inlet

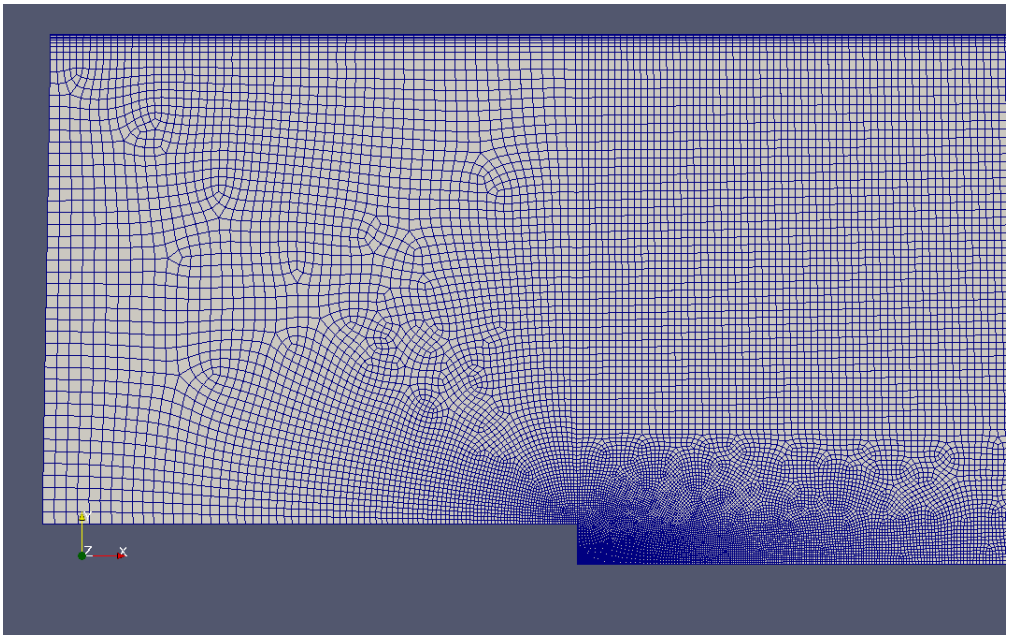


Figure 6.9: Fine Mesh Provided by El Abbassi

Numerical Solver Modification

Another attempt to get stable simulation is by changing the numerical solver from *GAMG* to more primitive *PCG* with *DIC* preconditioner for the pressure field calculation. This choice of numerical solver combined with lowering relaxation factor to 10^{-6} allows calculation for 1500 iteration, but without any indication of convergence. The residual for pressure field and continuity is still big, in order of magnitude of 10^{-2} which is far from the convergence criteria set at 10^{-6}. When the relaxation factor is increased, the instability reappears, leads to another error. Other attempts such as changing the preconditioner from *DIC* to *GAMG* combined with *PCG* solver and set the interpolation scheme as *upwind* instead of *linear* had also been tried, but no improvement is observed.

OpenFOAM Solver Modification

Another tried attempt is by changing the solver from *buoyantSimpleFoam* to *rhoSimpleFoam*. This is another solver which evaluate compressible flow which accounts the energy balance and using the *SIMPLE* algorithm in the process. The differences between this solver and *buoyantSimpleFoam* are in the calculation of compressibility and the treatment of density calculation. The *buoyantSimpleFoam* calculates compressibility based on the density by set the *thermophysicalProperties* file in the *constant* folder to *hePsiThermo* instead of *heRhoThermo*. This setting allows the calculation of density directly in the simulation instead of referring from the thermophysical library based on the pressure and temperature field value. It also allows explicit bounding of density in the setting of numerical solver within the *fvSolution* file, which is expected to be able to overcome the error caused by density calculation. This attempt is taken based on findings in the error that cite the inability to calculate fluid properties from the thermophysical library. This attempt also fails to achieve converged solution for *V case*.

An attempt to use modified *reactingFoam* by turning off the chemistry and combustion model is also executed. The attempt is made since an early insight from turbulent combustion simulation with high velocity using *reactingFoam* does not give an error even though the time step in resolving transient term is very low at order of magnitude of 10^{-9}. It is also known that *reactingFoam* is basically a compresible solver but with addition of chemical species balance. This solver is also a transient solver by nature and expected to be more robust in handling the high velocity case. Therefore, by turning off the combustion model, this solver can act similar to *buoyantSimpleFoam* and *rhoSimpleFoam* but with transient behavior. However, the error still appears after several time steps. The *reactingFoam* is not stable enough to handle the *V case* even with its transient behavior.

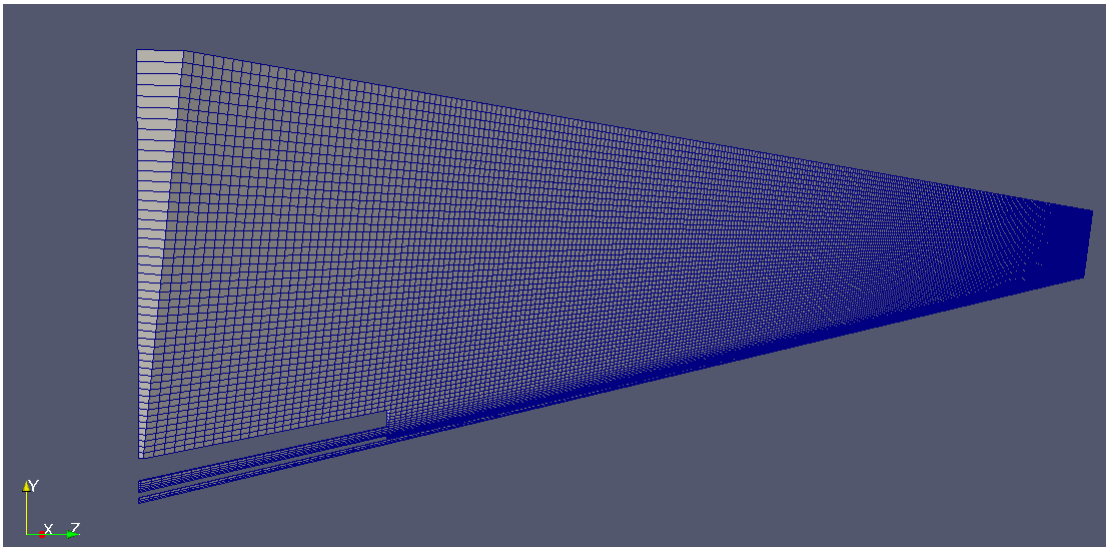


Figure 6.10: Mesh for Combustion Case

***Mach* Number Modification**

Based on the failure from modification mentioned above, the last attempt to know the root cause of the problem is by changing the *Mach* number of the flow. It is to see what value is the maximum *Mach* number which can be tolerated by the solver. Simulations with *Mach* number of 0.4, 0.8, and 1.2 are then carried using *buoyantSimpleFoam*. The solver can reach the convergence for the first two cases with *Mach* number of 0.4 and 0.8, but the error again arises when the *Mach* number is increased above 1. Similar test is also carried out using *rhoSimpleFoam* with tutorial case provided by OpenFOAM. The case with *Mach* number of 0.63 is converged successfully, but when the *Mach* number is doubled to above 1, the solver creates an error. Therefore, a solid proof had shown that indeed the limitation is buried within the provided solver. The solver that had been tried could not handle supersonic condition.

6.2.7. Conclusion from Buoyant Flow Validation

The compressible OpenFOAM solver, *buoyantSimpleFoam*, is found to be reliable to reproduce simulation by Dr. Talice in the subsonic condition. However, the solver is not applicable for the supersonic condition due to the instability of fluid properties calculation. Various attempt to modify the numerical contribution of case configuration such as geometrical mesh and numerical solver modification fails to achieve converged solution. This suggests that modification to thermophysical models, such as turbulence model, or adapting the transport equation to allow supersonic flow calculation need to be considered. Another strategy is using different solver that is developed for supersonic flow calculation. It is found that OpenFOAM provides transient solver called *sonicFoam* which is mentioned can deal with compressible supersonic flow and *LTSreactingFoam* which is suitable for local time stepping of reactive compressible flow. Due to the limitation of project time frame, this suggestion is made for the future project.

6.3. Combustion Modeling as Reconstruction of Case by Elattar

6.3.1. Geometrical Mesh

The mesh for current case is also based on *Burner A* in Elattar's paper. However, instead of using 45^o part of the kiln, the present work used 7.5^o part of the kiln, allowing only one grid in the radial direction to be created. This approach is taken to minimize the computational cost to achieve a fast result. The mesh contains 22 410 cells with the finer grid is present near the fuel inlet. The mesh is shown in Figure 6.10.

6.3.2. Initial and Boundary Conditions

The boundary condition is translation of Elattar's burner power comparison case, but using 0.4 MW burner power instead of 40 MW or 10 MW with air to fuel ratio $\lambda = 1.12$. The variation of primary to secondary air ratio (α) is taken as either $0.1, 0.5,$ or1.0 to analyze the effect of this ratio to the temperature and velocity profile. All inlets are set to have temperature of 20^oC with zero gradient pressure. Turbulent intensity is taken

as 0.10 for the air inlets and 0.05 for the fuel inlet. Turbulent length scale for calculation of ϵ at the inlets is taken as $0.038d_h$, where d_h is the diameter of the kiln. Boundary condition at the outlet is set as total pressure with fixed value of $1 \times 10^5 Pa$ and allows velocity to be calculated based on the pressure field at the outlet. The walls are set to have wall heat transfer with outside temperature of $300K$ and thermal diffusivity of $20m^2/s$. The calculation of turbulent scalar at the wall is supplied by the wall functions provided in section 3.3.4.

6.3.3. Solver Setup

The simulation is done in transient mode for $1000s$ simulation time. The scheme for time derivative is *Implicit Euler*. *Gaussian linear* is used for diffusive terms while *TVD* is defined for convective terms. Symmetric matrices such as pressure p and total radiation intensity G field is solved using *PCG* with *DIC* pre-conditioner while all other asymmetric matrices such as velocity u_j, enthalpy h, and turbulent scalars are solved using *PBiCG* with *DILU* pre-conditioner.

6.3.4. Result and Discussion

Discrepancy in the result of present work and Elattar's paper is expected due to the difference in the burner power. In his paper, Elattar showed that lowering burner power inlet results in smaller primary jet momentum due to smaller inlet velocity. This phenomenon affects the aerodynamics and shape of the flame. Due to lack of momentum, the recirculation of the flow is less, hence lower mixing is observed. This prediction is shown in simulation of $\alpha = 0.1$ with figure 6.11a, where recilculation in the flow is not observed, unlike the result shown in Elattar's work. The lack of recirculation makes the flame is not pushed downward to the kiln axis, creating a flame jet inclined upward to the hotter flow region due to buoyancy. This flame shape affects axial temperature profile of $\alpha = 0.1$ simulation shown in figure 6.12 where the peak temperature of $\alpha = 0.1$ simulation at kiln axis is shown to be less than other two values of α, which contradicts the result from Elattar's work. This lower peak temperature is not because of lower energy emitted by the flame, but due to the shape of the flame. As seen in comparison between figures 6.11a, 6.11b, and 6.11c, the peak temperature of these three configuration are about the same. The difference is the occurrence of recirculation in the flow, which pushes the flame towards the kiln. In $\alpha = 0.5$ and $\alpha = 1.0$ simulation, primary jet momentum is increased because more air comes from primary air rather than secondary air. This primary jet momentum is sufficient to create recirculation in the flow as shown by the velocity profile depicted by black arrows.

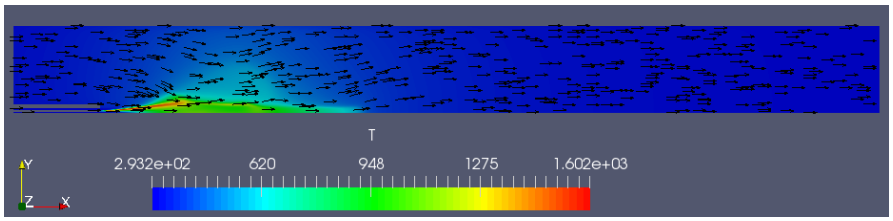
The temperature profile at the wall in $\alpha = 0.5$ and $\alpha = 1.0$ is highly affected by the recirculation of the flow as shown in figure 6.11 and 6.12. The peak temperature is located at the location of recirculation. This recirculation brings hotter fluid from the flame towards the wall before bouncing back to the flame front. In the other hand, when primary air is lower as in $\alpha = 0.1$, the peak temperature of the wall location is correlated to the location of valley of kiln axis temperature profile. This is because of the flame in this configuration inclines upward, creating buoyancy driven flow at the tip of flame. This buoyancy pushes hotter air towards the wall, marked by the peak temperature at the wall.

6.3.5. Conclusion from Combustion Modeling

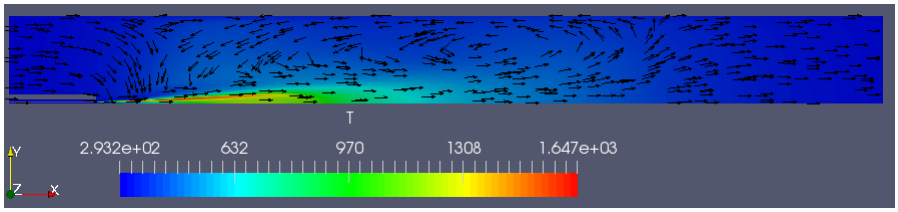
Although the result of the present work is not exact resemblance of Elattar's work, the resulting profile agrees with the underlying physical phenomena elaborated in Elattar's paper. The burner power difference between the two works resulting different flame shape and hence axial profile of the temperature. However, the explanation about recirculating flow and its relation to the burner power difference in Elattar's work applies to the present work's result. Therefore, it can be concluded that the current solver used in the present work can reliably reconstruct the physical phenomena simulated with commercial software by Elattar.

6.4. Conclusion from Reconstruction of Published Data

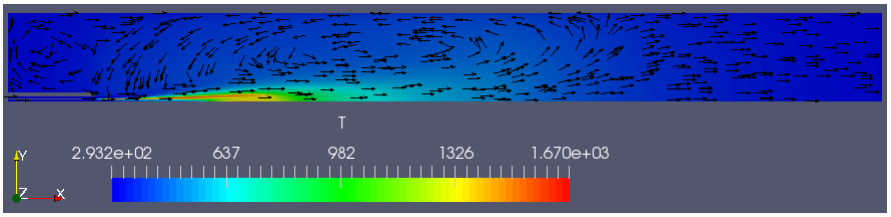
Based on the two cases done in this chapter, it can be concluded that OpenFOAM still has room for improvement to be as robust and reliable as commercial software. Its limitation in computational power and stability forced the present work to adapt the case to lower burner power to achieve faster and stable simulation. Furthermore, the limitation of supersonic simulation as found in the *V case* of buoyant flow validation suggests that further consideration in thermophysical model and transport equation adaptation is needed. However, the consistency of OpenFOAM solver in the result in low burner power case in both buoyant flow and turbulent combustion with the available published data signs a promising application of this open source CFD toolbox.



(a) $\alpha = 0.1$



(b) $\alpha = 0.5$



(c) $\alpha = 1.0$

Figure 6.11: Temperature and Velocity Profile

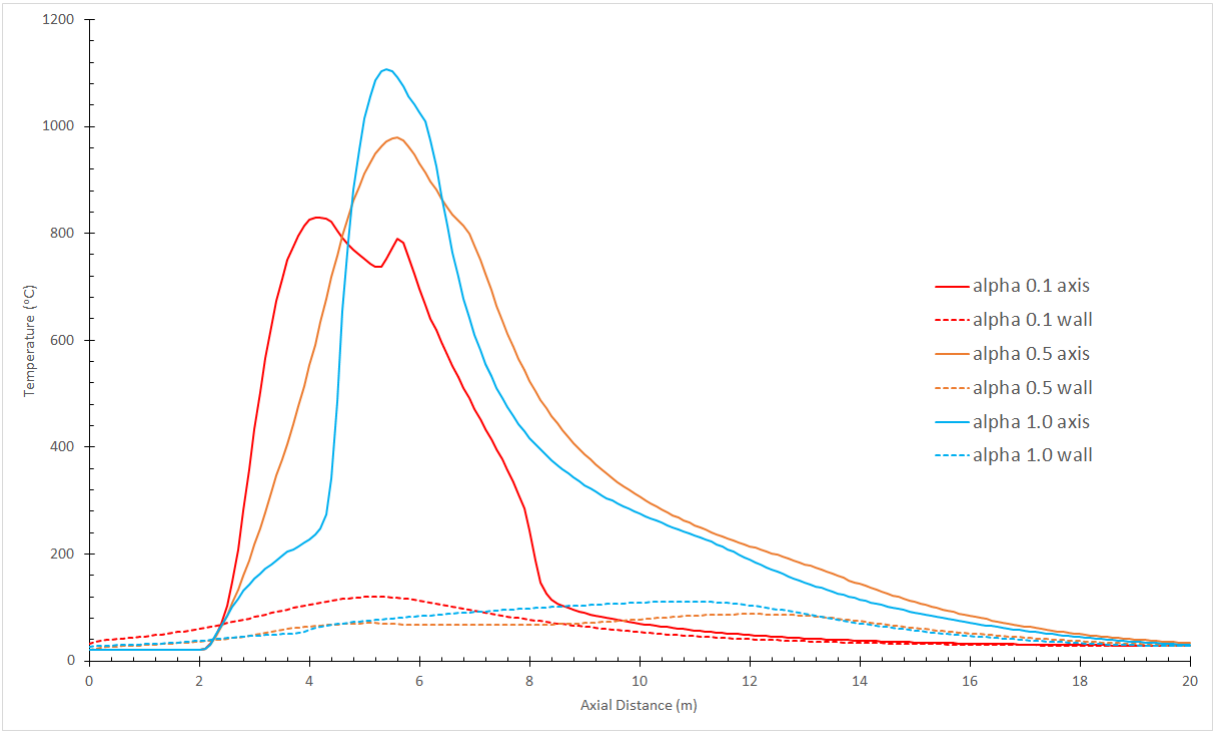


Figure 6.12: Axial Temperature Profile at Kiln Axis and Wall

7

Conclusions and Recommendations

The present work is done as continuation of previous work by Kadar and Pisaroni while providing a stepping stone for future work towards complete simulation of rotary kiln. The results presented in previous chapters are able to fulfill the objective of the present work mentioned in the first part of this paper with some remarks. This chapter summarizes the final conclusions and recommendations gathered from the present work.

7.1. Conclusions

The study to identify the highest contribution of computational cost to improve computational performance of OpenFOAM solver concludes that:

- The complexity of the model such as adding turbulence, reaction, and radiation model does not impact the computational performance directly.
- The highest contributor for the *furnaceFoam* solver developed by Ali Kadar is the calculation of the flow, which is dominated by the pressure-velocity coupling iteration.
- The computational performance for prescribed number of iteration is not improved by changing the pre-conditioner for *Conjugate Gradient (CG)* method from *Diagonal Incomplete-Cholesky (DIC)* to *Generalized Geometric-Algebraic Multi-Grid (GAMG)* as the later method needs more CPU time for calculation.
- The performance to achieve convergence criteria is improved by using *GAMG* as pre-conditioner for *CG* compared to *DIC*.

The test cases with simplified geometry and variation of input variable confirm that:

- The developed solver by Ali Kadar can capture the effect of input variable change as hypothesized from underlying physics. The changes include the input temperature, a detailed reaction mechanism, and inclusion of radiation model.
- The geometrical mesh highly affects the profile resulted from the simulation, as the simplified geometry fails to capture the physical shape of the flame.
- The unreal profile of tailing hot region in the simplified geometry is caused by poor mixing between fuel and oxidizer and can be improved by detailing the geometrical mesh.

The accuracy of OpenFOAM compared to commercial software and the reliability of the developed solver is tested by reconstructing published data with some modification. The results confirm that:

- An OpenFOAM built-in solver *buoyantSimpleFoam* can reliably reproduce subsonic compressible flow simulation in rotary kiln geometry by Dr. Marco Talice using his own commercial software.
- The *buoyantSimpleFoam* solver and another compressible solver, *rhoSimpleFoam* fails to compute supersonic compressible flow in rotary kiln geometry.

- The developed combustion solver *furnaceFoam* is not powerful enough to reproduce exact simulation done by Elattar of turbulent combustion in symmetric rotary kiln, as the time steps is very small.
- The *furnaceFoam* can reliably reproduce modified Elattar's case by reducing the burner power.
- The resulting simulation of *furnaceFoam* confirms the theory mentioned in Elattar's paper on the effect of jet momentum to the shape of flame and the profile of temperature.

In general, the developed solver *furnaceFoam* is proven to be reliable in capturing physical phenomena of turbulent combustion in rotary kiln. This suggest that the usage of open source toolbox such as OpenFOAM has promising benefit for analyzing industrial problem. However, the computational performance of the solver still has room for improvement to be powerful enough to handle large problem such as detailed three dimensional geometry of complex industrial rotary kiln.

7.2. Recommendations

During the execution of the present work, several limitations are found to halt the progress of the project. These limitations arise recommendation for future works which can take the project further.

- The OpenFOAM code which handles the communication between processors in parallel simulation needs to be analyzed further, since it is suspected to limit the effectiveness of *GAMG* solver.
- The usage of another advanced solver such as PETSc can be used as comparison for the computational performance of OpenFOAM solver given that the system matrices are able to be extracted from the OpenFOAM cases.
- A more detailed and realistic rotary kiln geometry needs to be considered in order to achieve better representation of flame shape in a turbulent combustion case.
- A new addition to OpenFOAM combustion model is found in the latest version of OpenFOAM-v1612+ which includes the Eddy-break-up model. This model can be used in future work in order to minimize computational effort of resolving the chemical reaction.
- Some alternative solvers for handling supersonic compressible flow such as *sonicFoam* can be analyzed further and combined with *furnaceFoam* to handle compressible or combustion cases with high *Mach* number as in case from Elattar.
- A general improvement of computational performance of the OpenFOAM solvers is needed in order to resolve detailed three dimensional geometry of industrial rotary kiln.
- A usage of more powerful computational arsenal can be considered for simulation of detailed three dimensional geometry of industrial rotary kiln.

As closing remarks, the usage of OpenFOAM in industrial application has a lot of promises, yet a lot of limitation. The license-free status of this toolbox can be beneficial for industrial application in minimizing the project cost. An active community of OpenFOAM users and developers also assures the improvement of this toolbox to be more robust and practical for industrial application.

A

furnaceFoam Source Code

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2015 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Application
    reactingFoam

Group
    grpCombustionSolvers

Description
    Solver for combustion with chemical reactions.

\*---------------------------------------------------------------------------*/

#include "fvCFD.H"
#include "turbulentFluidThermoModel.H"
#include "psiCombustionModel.H"
#include "multivariateScheme.H"
#include "pimpleControl.H"
#include "fvOptions.H"
#include "localEulerDdtScheme.H"
#include "fvcSmooth.H"
#include "radiationModel.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{
    #include "setRootCase.H"
```

```
50      #include "createTime.H"
51      #include "createMesh.H"
52
53      pimpleControl pimple(mesh);
54
55      #include "createTimeControls.H"
56      #include "createRDeltaT.H"
57      #include "initContinuityErrs.H"
58      #include "createFields.H"
59      #include "createMRF.H"
60      #include "createFvOptions.H"
61
62      #include "createRadiationModel.H"
63
64      turbulence->validate();
65
66      if (!LTS)
67      {
68          #include "compressibleCourantNo.H"
69          #include "setInitialDeltaT.H"
70      }
71
72      // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
73
74      Info<< "\nStarting time loop\n" << endl;
75
76      while (runTime.run())
77      {
78          #include "readTimeControls.H"
79
80          if (LTS)
81          {
82              #include "setRDeltaT.H"
83          }
84          else
85          {
86              #include "compressibleCourantNo.H"
87              #include "setDeltaT.H"
88          }
89
90          runTime++;
91
92          Info<< "Time = " << runTime.timeName() << nl << endl;
93
94          #include "rhoEqn.H"
95
96          while (pimple.loop())
97          {
98              // #include "UEqn.H"
99        // Solve the Momentum equation
100       MRF.correctBoundaryVelocity(U);
101
102       tmp<fvVectorMatrix> tUEqn
103       (
104          fvm::ddt(rho, U) + fvm::div(phi, U)
105        + MRF.DDt(rho, U)
106        + turbulence->divDevRhoReff(U)
107         ==
108          fvOptions(rho, U)
109       );
110       fvVectorMatrix& UEqn = tUEqn.ref();
111
112       UEqn.relax();
113
114       fvOptions.constrain(UEqn);
115
116       if (pimple.momentumPredictor())
117       {
118          solve(UEqn == -fvc::grad(p));
119
120          fvOptions.correct(U);
```

```
121         K = 0.5*magSqr(U);
122     }
123         // #include "YEqn.H"
124     // Solve the Chemical Species equation
125     tmp<fv::convectionScheme<scalar>> mvConvection
126     (
127         fv::convectionScheme<scalar>::New
128         (
129             mesh,
130             fields,
131             phi,
132             mesh.divScheme("div(phi,Yi_h)")
133         )
134     );
135 
136     {
137         reaction->correct();
138         dQ = reaction->dQ();
139         label inertIndex = -1;
140         volScalarField Yt(0.0*Y[0]);
141 
142         forAll(Y, i)
143         {
144             if (Y[i].name() != inertSpecie)
145             {
146                 volScalarField& Yi = Y[i];
147 
148                 fvScalarMatrix YiEqn
149                 (
150                     fvm::ddt(rho, Yi)
151                   + mvConvection->fvmDiv(phi, Yi)
152                   - fvm::laplacian(turbulence->muEff(), Yi)
153                  ==
154                     reaction->R(Yi)
155                   + fvOptions(rho, Yi)
156                 );
157 
158                 YiEqn.relax();
159 
160                 fvOptions.constrain(YiEqn);
161 
162                 YiEqn.solve(mesh.solver("Yi"));
163 
164                 fvOptions.correct(Yi);
165 
166                 Yi.max(0.0);
167                 Yt += Yi;
168             }
169             else
170             {
171                 inertIndex = i;
172             }
173         }
174 
175         Y[inertIndex] = scalar(1) - Yt;
176         Y[inertIndex].max(0.0);
177     }
178         // #include "EEqn.H"
179     // Solve the Energy Balance equation
180     {
181         volScalarField& he = thermo.he();
182 
183         fvScalarMatrix EEqn
184         (
185             fvm::ddt(rho, he) + mvConvection->fvmDiv(phi, he)
186           + fvc::ddt(rho, K) + fvc::div(phi, K)
187           + (
188                 he.name() == "e"
189               ? fvc::div
190                 (
191                     fvc::absolute(phi/fvc::interpolate(rho), U),
```

```
                p,
                "div(phiv,p)"
            )
          : -dpdt
        )
      - fvm::laplacian(turbulence->alphaEff(), he)
     ==
      reaction->Sh()
        + radiation->Sh(thermo)
        + fvOptions(rho, he)
      );

      EEqn.relax();

      fvOptions.constrain(EEqn);

      EEqn.solve();

      fvOptions.correct(he);

      thermo.correct();
      radiation->correct();

      Info<< "min/max(T) = "
          << min(T).value() << ", " << max(T).value() << endl;
  }

      // --- Pressure corrector loop
      while (pimple.correct())
      {
          if (pimple.consistent())
          {
              // #include "pcEqn.H"
      // Solve corrector step of PIMPLE algorithm
      rho = thermo.rho();
      rho = max(rho, rhoMin);
      rho = min(rho, rhoMax);
      rho.relax();

      volScalarField rAU(1.0/UEqn.A());
      volScalarField rAtU(1.0/(1.0/rAU - UEqn.H1()));
      volVectorField HbyA(constrainHbyA(rAU*UEqn.H(), U, p));

      if (pimple.nCorrPISO() <= 1)
      {
          tUEqn.clear();
      }

      if (pimple.transonic())
      {
          surfaceScalarField phid
          (
              "phid",
              fvc::interpolate(psi)
              *(
                  fvc::flux(HbyA)
                + fvc::interpolate(rho*rAU)*fvc::ddtCorr(rho, U, phi)
                  /fvc::interpolate(rho)
              )
          );

          MRF.makeRelative(fvc::interpolate(psi), phid);

          surfaceScalarField phic
          (
              "phic",
              fvc::interpolate(rho*(rAtU - rAU))*fvc::snGrad(p)*mesh.magSf()
          );

          HbyA -= (rAU - rAtU)*fvc::grad(p);
```

```
263             volScalarField rhorAtU("rhorAtU", rho*rAtU);
264
265             while (pimple.correctNonOrthogonal())
266             {
267                 fvScalarMatrix pEqn
268                 (
269                     fvm::ddt(psi, p)
270                   + fvm::div(phid, p)
271                   + fvc::div(phic)
272                   - fvm::laplacian(rhorAtU, p)
273                  ==
274                     fvOptions(psi, p, rho.name())
275                 );
276
277                 pEqn.solve(mesh.solver(p.select(pimple.finalInnerIter())));
278
279                 if (pimple.finalNonOrthogonalIter())
280                 {
281                     phi == phic + pEqn.flux();
282                 }
283             }
284         }
285         else
286         {
287             surfaceScalarField phiHbyA
288             (
289                 "phiHbyA",
290                 (
291                     fvc::flux(rho*HbyA)
292                   + fvc::interpolate(rho*rAU)*fvc::ddtCorr(rho, U, phi)
293                 )
294             );
295
296             MRF.makeRelative(fvc::interpolate(rho), phiHbyA);
297
298             phiHbyA += fvc::interpolate(rho*(rAtU - rAU))*fvc::snGrad(p)*mesh.magSf();
299             HbyA -= (rAU - rAtU)*fvc::grad(p);
300
301             volScalarField rhorAtU("rhorAtU", rho*rAtU);
302
303             // Update the pressure BCs to ensure flux consistency
304             constrainPressure(p, rho, U, phiHbyA, rhorAtU, MRF);
305
306             while (pimple.correctNonOrthogonal())
307             {
308                 fvScalarMatrix pEqn
309                 (
310                     fvm::ddt(psi, p)
311                   + fvc::div(phiHbyA)
312                   - fvm::laplacian(rhorAtU, p)
313                  ==
314                     fvOptions(psi, p, rho.name())
315                 );
316
317                 pEqn.solve(mesh.solver(p.select(pimple.finalInnerIter())));
318
319                 if (pimple.finalNonOrthogonalIter())
320                 {
321                     phi = phiHbyA + pEqn.flux();
322                 }
323             }
324         }
325
326         #include "rhoEqn.H"
327         #include "compressibleContinuityErrs.H"
328
329         // Explicitly relax pressure for momentum corrector
330         p.relax();
331
332         U = HbyA - rAtU*fvc::grad(p);
333         U.correctBoundaryConditions();
```

```
334         fvOptions.correct(U);
335         K = 0.5*magSqr(U);
336
337         if (thermo.dpdt())
338         {
339            dpdt = fvc::ddt(p);
340         }
341
342         // Recalculate density from the relaxed pressure
343         rho = thermo.rho();
344         rho = max(rho, rhoMin);
345         rho = min(rho, rhoMax);
346
347         if (!pimple.transonic())
348         {
349            rho.relax();
350         }
351
352         Info<< "rho max/min : " << max(rho).value() << " " << min(rho).value() << endl;
353               }
354               else
355               {
356                   // #include "pEqn.H"
357         // Solve Pressure equation
358         rho = thermo.rho();
359         rho = max(rho, rhoMin);
360         rho = min(rho, rhoMax);
361         rho.relax();
362
363         volScalarField rAU(1.0/UEqn.A());
364         surfaceScalarField rhorAUf("rhorAUf", fvc::interpolate(rho*rAU));
365         volVectorField HbyA(constrainHbyA(rAU*UEqn.H(), U, p));
366
367         if (pimple.nCorrPISO() <= 1)
368         {
369            tUEqn.clear();
370         }
371
372         if (pimple.transonic())
373         {
374            surfaceScalarField phid
375            (
376               "phid",
377               fvc::interpolate(psi)
378               *(
379                  fvc::flux(HbyA)
380                + rhorAUf*fvc::ddtCorr(rho, U, phi)/fvc::interpolate(rho)
381               )
382            );
383
384            MRF.makeRelative(fvc::interpolate(psi), phid);
385
386            while (pimple.correctNonOrthogonal())
387            {
388               fvScalarMatrix pEqn
389               (
390                  fvm::ddt(psi, p)
391                + fvm::div(phid, p)
392                - fvm::laplacian(rhorAUf, p)
393                ==
394                  fvOptions(psi, p, rho.name())
395               );
396
397               pEqn.solve(mesh.solver(p.select(pimple.finalInnerIter())));
398
399               if (pimple.finalNonOrthogonalIter())
400               {
401                  phi == pEqn.flux();
402               }
403            }
404         }
```

```
405         else
406         {
407            surfaceScalarField phiHbyA
408            (
409               "phiHbyA",
410               (
411                  fvc::flux(rho*HbyA)
412                + rhorAUf*fvc::ddtCorr(rho, U, phi)
413               )
414            );
415
416            MRF.makeRelative(fvc::interpolate(rho), phiHbyA);
417
418            // Update the pressure BCs to ensure flux consistency
419            constrainPressure(p, rho, U, phiHbyA, rhorAUf, MRF);
420
421            while (pimple.correctNonOrthogonal())
422            {
423               fvScalarMatrix pEqn
424               (
425                  fvm::ddt(psi, p)
426                + fvc::div(phiHbyA)
427                - fvm::laplacian(rhorAUf, p)
428                ==
429                  fvOptions(psi, p, rho.name())
430               );
431
432               pEqn.solve(mesh.solver(p.select(pimple.finalInnerIter())));
433
434               if (pimple.finalNonOrthogonalIter())
435               {
436                  phi = phiHbyA + pEqn.flux();
437               }
438            }
439         }
440
441         #include "rhoEqn.H"
442         #include "compressibleContinuityErrs.H"
443
444         // Explicitly relax pressure for momentum corrector
445         p.relax();
446
447         // Recalculate density from the relaxed pressure
448         rho = thermo.rho();
449         rho = max(rho, rhoMin);
450         rho = min(rho, rhoMax);
451         rho.relax();
452         Info<< "rho max/min : " << max(rho).value()
453            << " " << min(rho).value() << endl;
454
455         U = HbyA - rAU*fvc::grad(p);
456         U.correctBoundaryConditions();
457         fvOptions.correct(U);
458         K = 0.5*magSqr(U);
459
460         if (thermo.dpdt())
461         {
462            dpdt = fvc::ddt(p);
463         }
464      }
465         }
466
467         if (pimple.turbCorr())
468         {
469             turbulence->correct();
470         }
471       }
472
473       runTime.write();
474
475       Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
```

```
476             << "  ClockTime = " << runTime.elapsedClockTime() << " s"
477             << nl << endl;
478     }
479
480     Info<< "End\n" << endl;
481
482     return 0;
483 }
484
485
486 // ************************************************************************* //
```

Bibliography

[1] P. V. Barr, J. K. Brimacombe, and A. P. Watkinson. A heat-transfer model for the rotary kiln: Part I. pilot kiln trials. *Metallurgical Transactions B*, 20(3):391–402, 1989. ISSN 03602141. doi: 10.1007/BF02696991.

[2] T Belmrabet, M Talice, G Delussu, and S Hanchi. An implicit parallel fully compressible roe based solver for subsonic and supersonic reacting flows. In *Computational Fluid Dynamics 2008*, pages 167–172. Springer, 2009.

[3] T Belmrabet, M Talice, G Delussu, M Mulas, and S Hanchi. Combustion analysis using roe's scheme and the spalart-allmaras model. *AIAA journal*, 47(11):2726–2737, 2009.

[4] J Bibrzycki and T Poinsot. Reduced chemical kinetic mechanisms for methane combustion in O 2 / N 2 and O 2 / CO 2 atmosphere. *Working note ECCOMET WN/CFD/10/17, CERFACS*, 2010. doi: 10.1016/j.combustflame.2007.10.022.

[5] A. A. Boateng and P. V. Barr. A thermal model for the rotary kiln including heat transfer within the bed. *International Journal of Heat and Mass Transfer*, 39(10):2131–2147, 1996. ISSN 00179310. doi: 10.1016/0017-9310(95)00272-3.

[6] A.A. Boateng. *Rotary Kilns: Transport Phenomena and Transport Processes*. Elsevier Science, 2015. ISBN 9780128038536. URL `https://books.google.nl/books?id=0DodCAAAQBAJ`.

[7] William M. Deen. *Analysis of Transport Phenomena*. Oxford University Press, New York, 2 edition, 2013. ISBN 9780199740253.

[8] Guy Dumont and Pierre R Bélanger. Steady-State Study of a Titanium Dioxide Rotary Kiln. *Industrial & Engineering Chemistry Process Design and Development*, 17(2):107–114, 1978. ISSN 0196-4305. doi: 10.1021/i260066a001. URL `http://dx.doi.org/10.1021/i260066a001{%}5Cninternal-pdf://212/i260066a001.html`.

[9] Hassan F. Elattar, Eckehard Specht, Ali Fouda, and Abdullah S. Bin-Mahfouz. Study of Parameters Influencing Fluid Flow and Wall Hot Spots in Rotary Kilns using CFD. *Canadian Journal of Chemical Engineering*, 94(2):355–367, 2016. ISSN 1939019X. doi: 10.1002/cjce.22392.

[10] Jay Fenlason and Richard Stallman. Interpreting gprof's output, November 1998. URL `https://ftp.gnu.org/old-gnu/Manuals/gprof-2.9.1/html_chapter/gprof_5.html`.

[11] Juan Marcelo Gimenez and Santiago Marquez Damian. Contrib gdbof, August 2016. URL `https://openfoamwiki.net/index.php/Contrib_gdbOF`.

[12] K Hanjalic, Kenjeres S, Tummers M.J., and H.J.J Jonker. *Analysis and Modelling of Physical Transport Phenomena*. VSSD, Delft, The Netherlands, 2 edition, 2009. ISBN 9065621652.

[13] FLUENT Inc. 12.4.3 realizable k-epsilon model, September 2006. URL `https://www.sharcnet.ca/Software/Fluent6/html/ug/node480.htm`.

[14] Ali Hussain Kadar. Modelling Turbulent Non-Premixed Combustion in Industrial Furnaces, 2015. URL `http://repository.tudelft.nl/.`

[15] Brunel University London. Numerical simulations of turbulent combustion and combustion engines, 2017. URL `http://www.brunel.ac.uk/cedps/mechanical-aerospace-civil-engineering/mecheng/research/capf/tcce`.

[16] OpenCFD Ltd. Openfoam® v3.0+: New solver and physical modelling functionality, January 2016. URL `http://www.openfoam.com/version-v3.0+/solvers-and-physics.php`.

[17] OpenCFD Ltd. Openfoam c++ source code guide, January 2016. URL http://openfoam.com/documentation/cpp-guide/html/dir_2ea871e4defbdbd60808750cae643c63.html.

[18] OpenCFD Ltd. Openfoam user guide - chapter 1: Introduction, January 2016. URL http://openfoam.com/documentation/user-guide/userch1.php#x3-20001.

[19] OpenCFD Ltd. Openfoam user guide - chapter 6.2: Numerical schemes, January 2016. URL http://openfoam.com/documentation/user-guide/fvSchemes.php#x23-820006.2.

[20] F. Marias, H. Roustan, and A. Pichat. Modelling of a rotary kiln for the pyrolysis of aluminium waste. *Chemical Engineering Science*, 60(16):4609–4622, 2005. ISSN 00092509. doi: 10.1016/j.ces.2005.03.025.

[21] Vincent Meyer, Alexander Pisch, Karri Penttilä, and Pertti Koukkari. Computation of steady state thermochemistry in rotary kilns: Application to the cement clinker manufacturing process. *Chemical Engineering Research and Design*, 115, Part B:335 – 347, 2016. ISSN 0263-8762. doi: http://dx.doi.org/10.1016/j.cherd.2016.08.007. URL http://www.sciencedirect.com/science/article/pii/S0263876216302234. 10th European Congress of Chemical Engineering.

[22] K Mujumdar and V Ranade. Simulation of Rotary Cement Kilns Using a One-Dimensional Model. *Chemical Engineering Research and Design*, 84(March):165–177, 2006. ISSN 02638762. doi: 10.1205/cherd.04193. URL http://linkinghub.elsevier.com/retrieve/pii/S026387620672874X.

[23] CFD Online. Diffusion term, January 2010. URL https://www.cfd-online.com/Wiki/Discretization_of_the_diffusion_term.

[24] CFD Online. Combustion, March 2011. URL https://www.cfd-online.com/Wiki/Combustion#The_Damk.C3.B6hler_Number.

[25] CFD Online. Approximation schemes for convective term - structured grids - common, October 2016. URL https://www.cfd-online.com/Wiki/Approximation_Schemes_for_convective_term_-_structured_grids_-_Common.

[26] Michele Pisaroni, Rudi Sadi, and Domenico Lahaye. Counteracting ring formation in rotary kilns. *Journal of Mathematics in Industry*, 2(1):3, 2012.

[27] Dirk Roekaerts. Turbulent reacting flow lecture note - basic equations of reacting flow, February 2016.

[28] T H Shih, W W Liou, A Shabbir, Z Yang, and J Zhu. A New K-epsilon Eddy Viscosity Model for High Reynolds Number Turbulent Flows: Model Development and Validation. *Computer \& Fluids*, 24 (August):227–238, 1995. ISSN 00457930. doi: 10.1016/0045-7930(94)00032-T.

[29] Konrad S. Stadler, Jan Poland, and Eduardo Gallestey. Model predictive control of a rotary cement kiln. *Control Engineering Practice*, 19(1):1 – 9, 2011. ISSN 0967-0661. doi: http://dx.doi.org/10.1016/j.conengprac.2010.08.004. URL http://www.sciencedirect.com/science/article/pii/S0967066110001851.

[30] H. K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics*, volume 1. 2007. ISBN 9780131274983.