Detonation Modeling in OpenFOAM Using Adaptive Mesh Refinement

by

Duncan A. McGough

B.S., University of Colorado Boulder

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Smead Department of Aerospace Engineering Sciences

2020

This thesis entitled:
Detonation Modeling in OpenFOAM Using Adaptive Mesh Refinement
written by Duncan A. McGough
has been approved for the Smead Department of Aerospace Engineering Sciences

_____
Dr. Peter Hamlington

_____
Dr. John Evans

_____
Dr. Kenneth Jansen

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

McGough, Duncan A. (M.S, Aerospace Engineering)

Detonation Modeling in OpenFOAM Using Adaptive Mesh Refinement

Thesis directed by Dr. Peter Hamlington

This thesis contains research performed on detonation modeling in OpenFOAM, an open-source computational fluid dynamics toolbox. Several existing solvers are tested for their ability to capture shocks and model reactions needed for detonations. It is shown that the central-upwind-scheme-based solvers better capture sharp flow features with shocks, a necessity for high-speed reacting flow simulations. Automatic time stepping algorithms are examined, tracking both the central and acoustic Courant numbers. Lower than typical Courant numbers for supersonic nonreacting flows are found to be required for solution convergence and noise reduction. The pre-exponential factor in the Arrhenius rate equation is explored to determine the sensitivity on the solution as well as compare to published values for single-step Arrhenius hydrogen-air combustion. Static one- and two-dimensional meshes were compared, and separate convergence criteria for resolving the detonation wave shape as a whole as well as fine detonation structure were seen. Adaptive mesh refinement (AMR), a type of runtime-active mesh generation that refines the mesh in regions of the domain that need it, is explored to determine if AMR can be used to decrease computational cost of detonation modeling for PDEs and RDEs within OpenFOAM. The normalized pressure gradient is tracked and used to direct where adaptive meshing should occur, and different values of normalized pressure gradient for when adaptive meshing should refine and unrefine the computational domain are examined. Effects of refinement level and the number of buffer layers between refined and unrefined layers are explored in context of both solution accuracy as well as computational cost. It was found that good reproduction of two-dimensional detonation waves formed with static computational grids can be done with adaptive meshing routines in OpenFOAM. Computational cost reductions of over 96% were seen with AMR.

Acknowledgements

For all the continual support, patience, guidance throughout my work on this thesis, I'd like to thank Dr. Hamlington. I appreciate his willingness to guide me through my Master's degree research, and without it this would not have been possible.

In addition, I'd like to thank the students in the Turbulent Energy Systems Laboratory (TESLa): Caelan Lapointe and Colin Towery. Their patience throughout my journey of knowledge through areas they were familiar was admirable. Without their assistance, I would have progressed much more slowly.

I'd like to thank my mentor during my early time at SpaceX, Dr. Diego Arias. His guidance and advice led me to pursuing this thesis work throughout my graduate education.

Lastly, I'd like to thank those closest to me, my parents Julie and Jeff as well as my dearest Vismaya. Their love and support was invaluable.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

1.1 Motivation

Many challenges associated with modern combustion systems, such as improving the efficiency and performance of internal combustion engines, have been the focus of considerable research for the past several decades. This focus has resulted in improved engine performance, efficiency, and reliability. However, internal combustion engines are just one area within the much broader field of study of reacting flows, which can include topics as diverse as wildfires, supernovae, explosions, ocean biogeochemistry, and aircraft or rocket propulsion.

Considering aircraft and rocket propulsion, in particular, modeling of these systems is made exceedingly difficult by the reactive and potentially high-Mach nature of the turbulent flows involved in these applications. Computational fluid dynamics (CFD) is often used to model these flows, where the system domain is discretized using a computational grid, on which flow equations such as the Navier-Stokes equations are solved. Alternative to CFD is modeling the fluid particles themselves discretely, like what is done with some gas dynamics simulations. For reacting flows and other standard fluid problems, this is prohibitively expensive or impossible to model computationally, as there are billions of particles to track, as well as their interactions. Instead, the fluid can be modeled using continuum mechanics, where the behavior of the fluid is modeled as a continuous volume. This greatly simplifies how many computations must be performed. For CFD that utilizes continuum mechanics of

high-speed turbulent reacting flows, this grid must be very fine in order to maintain accurate solutions of the governing equations. The addition of chemical reactions further complicates the calculation and can substantially increase the time required for a solution. As a result, CFD of high-speed turbulent reacting flows is computationally expensive and the use of cost-reducing methods, such as subgrid-scale modeling for large eddy simulations (LES), can substantially reduce physical accuracy.

These challenges are all particularly relevant to a growing area of interest within aircraft and rocket propulsion; namely, the use of detonation engines to achieve high efficiency and high speed propulsion. Two main subtypes of these engines exist: the pulsed detonation engine (PDE) and the rotating detonation engine (RDE). Both PDEs and RDEs are attractive since, in principle, they can theoretically produce higher efficiencies than traditional propulsive devices such as turbofans and some rocket engines. Both of these propulsion technologies rely on the ignition of a detonation wave (i.e., a supersonic flame front) to provide the means of propulsion. The detonation wave itself is usually quite thin and is accompanied by high pressures and temperatures.

With such fast and physically small flow features, detonations require very fine grids in order to be accurately captured. This adds to the computational expense of turbulent reacting flow simulations relevant to PDEs and RDEs. A potential solution to this problem is to adaptively refine the grid (mesh), allowing for a finer meshes in locations where high resolutions are more critical, such as around a detonation wave, and a less refined mesh in areas of the domain that do not have complex flow features. By doing this, the computer is only solving the fluid equations on grid points where necessary, so computational resources are deployed as efficiently as possible. The resulting procedure is called adaptive mesh refinement (AMR).

Because PDEs and RDEs have strong detonation waves within their domains, they are ideal candidates for the efficiency and performance gains enabled by AMR. With AMR, PDE and RDE computational research can progress more efficiently, since accurate flow-field

solutions can be obtained with less computational expense. The present MS thesis is focused, in particular, on developing an AMR capability for the simulation of detonation waves found in PDEs and RDEs.

1.2 Objectives and Scope

The primary objective of this thesis is to implement and test AMR techniques for simulations of detonation waves found in PDEs and RDEs using the open-source computational fluid dynamics toolbox known as OpenFOAM [1]. Compared to an in-house CFD code use previously for detonation engine research [2], OpenFOAM has several benefits, including increased geometric flexibility, an input structure that is nearly identical for most solvers, reduced modeling time when comparing solvers for similar cases, and documentation is more widely-available.

The cases studied in this thesis demonstrate the shock-capturing accuracy of the solver, as well as the simulation of one- and two-dimensional detonations. Three-dimensional detonations with AMR were tested, but are not the primary focus and presented higher computational cost for studying AMR parameters. Another objective of this thesis work is to build on previous work [2] and develop the detonation modeling techniques with AMR in OpenFOAM such that the Turbulence and Energy Systems Laboratory (TESLa) at the University of Colorado, Boulder can use them for more in-depth and important research, such as the optimization of geometrically-complex RDE systems. As such, the focus of this thesis is centered on solver methods, exploration of solver features, AMR configuration sensitivites, and building a framework that can serve as the basis for further research.

1.3 Relevant Previous Work

In order to be able to do the research outlined in this document, many resources were utilized and many others have layed the path in which this work was built on.

The primary work this research was built on was done by Towery *et al.* [2]. The work

done in this paper examined both PDE or detonation tube computational modeling as well as RDE modeling. The numerical code used was developed in-house by TESLa, written in Fortran and was massively scalable in parallel. However, this code lacked the geometric and generic flexibility of OpenFOAM, hence the research done in this document. This paper examined numerical methods for large eddy simulation (LES) of PDEs and RDEs in two and three dimensions. Reynolds averaged turbulence models were also compared to the higher resolution LES simulation results. This research simulated the reactive Euler equations, and the flow fields were thus inviscid. Both single and two-step reactions kinetics were used in the simulation of detonations in comparison to the single step kinetics used in this document. AMR was not utilized for simulation.

In 2013 Schwer and Kailasanath [3] examined the effects of different hydrocarbon fuels on RDEs as well as as their numerical methods for simulating RDEs and detonation waves in tubes. Their tube detonation simulations were performed using a flux-corrrected-transport (FCT) algorithm scaled across hundreds of cores in parallel. Their work showed that RDEs are not affected by thermodynamic differences in hydrocarbon fuels in comparison to hydrogen fuel and that ideal detonation cycle theory could be used for baseline performance. In order to reach conclusions about RDE performance, studies were done on PDE and detonation tube modeling to verify the validity of their solution methods, much like the work done in this document. Detonation tube numerical results were compared to Chapman[4]-Jouguet[5] (CJ) theory and detonation cellular structure was also produced.

Marcantoni *et al.* [6] created `rhoCentralRfFoam`, a non-AMR solver, by combining Kurganov central schemes that model compressible flow from `rhoCentralFoam` [7] with reaction modeling capability. Various mass fraction compositions were tested in air as well as chemistry validity spanning into multi-reaction chemical models. They selected a maximum Courant number of 0.25 for the one-dimensional planar detonation tests. `rhoCentralRfFoam` was found to agree with CJ theory as well as other numerical codes such as CEA [8].

Dinh *et al.* [9] developed an in-house proprietary OpenFOAM code for modeling deto-

nations. This code had the ability to use AMR and was run on a single processor. The AMR parameters used for tracking refinement were normalized reaction heat as well as normalized velocity gradient at an interval of every 10 computational time steps. Together, both thresholds had to be met in order for AMR to occur. Comparisons with analytical theory such as CJ theory or experimental results were not performed in this work, and research done was more exploratory to determine the applicability of the solver.

Additionally in 2019, Kim and Kim [10] used an OpenFOAM solver with central-upwind schemes to study hydrogen-air detonation within confined spaces. Arrhenius rate modeling with seven-step chemistry component reactions were utilized. Similar numerical schemes for solving the inviscid reactive Euler equations were used in this work compared to the solvers in this document. Shock tube comparison was done for solver validation as well as further comparison to confined detonation cases, and a CFL number of 0.1 was prescribed. It was found that their solver was capable of capturing shock structures and fluid modeling to match theory as well being able to replicate accurate confined hydrogen-air detonations.

Li *et al.* [11] used AMR with the parallelized AMROC code to simulate cellular detonation diffraction. Cellular detonation structure diffracting can be seen when emerging from a confined space into a larger one. The inviscid Euler equations with two-step Arrhenius rate chemistry were modeled. They were successful in simulating detonation cells that were comparable to experimental results. The AMR used in the simulations was fine enough to resolve fine detonation structure, and adaptively refined the mesh into smaller half-squares using the Berger-Oliger AMR algorithm [12].

In 1985, Berger and Jameson [13] used an AMR method in the transonic domain while solving the steady Euler equations. In this paper, more refined and less refined cell interfaces are examined as well as the boundary conditions between refined and unrefined regions. Unlike specific parameter-tracking AMR such as the methods used in this thesis, the method utilized in this paper involved time stepping forward across the grid and then considering each grid point for large residual error. Grid points with more error were refined

further.

Then, in 1989, Berger and Colella [14] used the adaptive meshing routines shown in 1985 to simulate shock hydrodynamics. The AMR is automatic and centered around the solving of two-dimensional hyperbolic conservation laws. These routines consisted of 3000 lines of Fortran. This paper also focused on minimizing CPU and memory overhead of the adaptive meshing, something that was noticed in this thesis to be non-negligible.

Subsequently, Deiterding [15] took the AMR routines from Berger and Colella to simulate multi-dimensional detonation structures. The simulations modeled detonations with the inviscid Euler equations, with recursively-iterated adaptive meshing. Parallelization and distributed computing is developed and implemented. Detonations are simulated with hydrogen and oxygen, and cellular structures are computed in two and three dimensions. Grid resolutions achieved were finer than previously published results at the time.

Henshaw and Schwendeman developed a method [16] for simulating reacting flow inside complex geometries using a block-structured adaptive mesh refinement and overlapping grids. The reactive Euler equations were modeled along with stiff reaction models and an ideal equation of state. The focus was extending the typical AMR techniques of the time to overset grid methods in a curvilinear domain. The detonation examined was in an expanding channel, and the reaction chemistry was single-step with Arrhenius rate reaction modeling.

Yi *et al.* [17] explored three-dimensional numerical simulations of rotating detonations in an annular chamber in 2009. Overall flow field structure is studied under continuous fuel injection into the combustion chamber. The code used for simulation was developed by the authors and modeled the reactive Euler equations discretized with the finite volume method (FVM). The governing equations utilized a MUSCL-based scheme [18] and temporal terms discretized with second-order techniques. The flow field was also examined in two dimensions, and overall reaction modeling was single-step. Previous one- and two-dimensional RDE models were compared to three-dimensional results.

Kessler *et al.* [19] discuss cell structures in methane-air mixture detonations. High-

resolution numerical simulations showed multi-level cellular detonation structures. Transverse waves trailing the detonation wavefront created the structure for several configurations of detonation cells. These cells were qualitatively matched with experimental smoke foil results.

Together, these papers and other similar work have laid the framework for adaptive meshing of detonation modeling in OpenFOAM.

1.4 Thesis Organization

This document is broken up into several chapters. The next chapter discusses the fundamental governing equations of the flows studied here. It introduces various numerical solution techniques, as well as how they are implemented in order to approximate a solution to the governing equations. This chapter also discusses OpenFOAM as a whole, as well as specific details as to how detonations were modeled with the toolbox. Some initial failures are shown here and solutions or alternatives used to overcome the failures. The following chapter compares different solver parameters used to model detonations and how the variation of the parameters affects the solution. Different levels of accuracy are discussed as well as well as generic results and comparisons overall. Conclusions and directions for future work are provided at the end.

Chapter 2

Description of AMR-Based Detonation Solver in OpenFOAM

2.1 Overview

OpenFOAM [1] is a C++ toolbox that is a collection of various programs that is primarily used to solve computational fluid dynamics problems. Unlike many commercial CFD codes available, OpenFOAM does not use a GUI for problem setup and instead relies on the user to provide the toolbox a variety of text-based input files in which it interprets and uses for deriving a solution to a particular problem of interest. This is a blessing in disguise, as it allows for very clear control over what OpenFOAM does. Since the source is open, many additional tools and solvers have been developed for OpenFOAM, making it very flexible. Some commercial CFD tools have been based on OpenFOAM, such as the cloud-based CFD platform with a GUI, SimScale [20], the desktop software simFlow [21], and RapidCFD [22], an OpenFOAM toolbox ported to Nvidia CUDA [23] for GPU-based solving.

The OpenFOAM solvers tested during this research include:

- **rhoReactingFoam**: a solver included with the OpenFOAM installation, in which the provided description calls it a "[s]olver for combustion with chemical reactions using density-based thermodynamics package."

- **rhoCentralFoam**: a solver included with the OpenFOAM installation, in which the provided description calls it a "Density-based compressible flow solver based on

central-upwind schemes of Kurganov and Tadmor with support for mesh-motion and topology changes." Originally developed by Greenshields *et. al.* [7].

- **rhoReactingCentralFoam**: a solver developed by Caelan Lapointe at the University of Colorado Boulder, combining `rhoReactingFoam` and `rhoCentralFoam` together as well adaptive mesh refinement support. Originally based on the work of Nakul [24].

In addition to these solvers, several additional tools outside the standard OpenFOAM installation were used. This includes `funkySetFields`, included with `swak4Foam` [25]. `funkySetFields` is a tool much like the OpenFOAM-included `setFields` utility, which allows the user to define set values of variables within certain regions of the computational domain (e.g. allows the user to set a cube within a domain to a certain temperature which is different than what is set globally). `funkySetFields` extends this functionality, allowing for time and position-dependent defining of variables, such that an initial condition such as a temperature ramp, or randomization of temperature within the domain is possible.

2.2 Governing Equations

In order to model detonations, the problem can be broken up into smaller pieces and solved with CFD. The OpenFOAM computational toolbox will utilize the finite volume method (FVM) to solve the set of partial differential equations describing the fluid flow. In the FVM, the computational domain is broken into multiple smaller control volumes, known as cells. Each cell contains a set of thermo-fluid properties at the centroid. The differential equations are then integrated over this control volume in order to obtain a solution at each cell. The solution is interpolated between cell centroids to obtain a solution throughout the domain.

The solution for the fluid problem explored in this thesis work is approximated through density, momentum, energy, and species equations. The solver used to approximate these

equations in a discretized computational domain, `rhoReactingCentralFoam`, utilizes the central-upwind schemes developed by Kurganov and Tadmor [26]. The density equation solved by `rhoReactingCentralFoam` is

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \tag{2.1}$$

the momentum equation is

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) + \nabla p - \mu \nabla^2 \boldsymbol{u} = \boldsymbol{0}\,, \tag{2.2}$$

the energy equation is

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot [(\rho E + p)\,\boldsymbol{u}] - \alpha \nabla^2 e = \dot{q}\,, \tag{2.3}$$

and the species equation is

$$\frac{\partial \rho Y_i}{\partial t} + \nabla \cdot (\rho Y_i \boldsymbol{u}) - \mu \nabla^2 Y_i = \dot{\omega}_i\,, \tag{2.4}$$

where

$$\dot{q} = \sum_{i=1}^{N} \dot{\omega}_i \Delta h^0_{f,i}\,, \tag{2.5}$$

is the heat flux, $\dot{\omega}_i$ is the species source reaction rate, ρ is the density, $\boldsymbol{u}$ is the fluid velocity vector, Y_i is the mass fraction of the ith species, E is the total energy, p is the pressure, μ is the dynamic viscosity, e is the internal energy, α is the thermal diffusivity and $\Delta h^0_{f,i}$ is the species formation enthalpy.

These equations [27], known as the reactive Navier-Stokes equations [28], are solved using OpenFOAM's differential equation time step integrators. It should be noted that additional terms to these equations exist within `rhoReactingCentralFoam`, but are omitted here as they are used for turbulence modeling. Turbulence is not modeled in this work, and all sub-grid scale turbulence structures are averaged out numerically, making the simulations performed akin to implicit large-eddy simulations (LES). Much of the simulated detonation work performed previously has been with the reactive Euler equations, omitting fluid viscosity.

Different equation solvers can be selected for different quantities of interest. For quantities like velocity, internal energy, and species, the preconditioned biconjugate gradient stabilized method [29] is used (`PBiCGStab` in OpenFOAM). For solvers like this, preconditioners can be selected as well as solution tolerance and relative tolerance. Preconditioners are computational operations done on the solution matrix in order to help prepare them for numerical iterative solving such that less iterations are required. The system is closed when energy is expanded and an equation of state is added. Total energy [27] can be written as

$$E = h - \frac{p}{\rho} + \frac{1}{2}\left(\boldsymbol{u}\cdot\boldsymbol{u}\right) , \tag{2.6}$$

where the total summed enthalpy is written [27] as

$$h = \sum_{i=1}^{N} h_{s,i} Y_i \, , \tag{2.7}$$

and the species total enthalpy [27] is given by

$$h_i = \Delta h^0_{f,i} + h_{s,i} \, , \tag{2.8}$$

with the sensible enthalpy for the ith species expressed as

$$h_{s,i} = \int_{T_0}^{T} C_{p,i} \mathrm{d}T \, , \tag{2.9}$$

Here $C_{p,i}$ is the specific heat for the ith species, T is the temperature, and T_0 is an initial, or reference, temperature. The equation of state is expressed as

$$p = \rho R T \, , \tag{2.10}$$

where $R = R_u/W$, with R_u the universal gas constant and W is the mixture molecular weight. Transport coefficients were represented using the Sutherland model [30], given as [31]:

$$\mu = \frac{A_s \sqrt{T_s}}{1 + \frac{T_s}{T}} \, , \tag{2.11}$$

where μ is the dynamic viscosity, A_s is the Sutherland coefficient, and T_s is the Sutherland temperature.

Table 2.1: Species mass fraction initial condition

Species	Mass Fraction
H_2	0.02851
H_2O	0
N_2	0.745
O_2	0.226
Total	0.99951

Since detonation modeling consists of both a fluids problem as well as chemical problem, chemical reactions must also be considered. Like a fire, a detonation requires a fuel and an oxidizer. Typically, detonations in experimental and computational modeling will utilize gaseous hydrogen or methane fuels mixed with air or pure oxygen oxidizers. For the computational modeling here, hydrogen-air was the primary respective fuel-oxidizer combination utilized for exploring detonation modeling. This was due to the highly reactive nature of hydrogen as well as previous computational and experimental modeling resources available. The reaction used [27] for the hydrogen detonation modeling here is

$$2\,H_2 + 2\,(O_2 + 3.76\,N_2) \longrightarrow 2\,H_2O + 7.52\,N_2$$

This is a stoichiometric reaction between diatomic hydrogen gas and air. When converted to mass fraction Y for OpenFOAM [6], it becomes Table 2.1.

Other trace elements present in air are not modeled here, and this could be a topic of further study within the context of OpenFOAM detonation modeling with AMR. Additionally, a single-step reaction was modeled as opposed to several reactions together to model hydrogen-air detonations. While more steps are certainly more realistic, adding further reaction complexity was found to significantly increase computational expense. It was decided that gaining a better baseline understanding of how AMR could improve modeling efficiency in OpenFOAM was the focus here and thus further work on increasing the reaction complexity and accuracy was not performed.

The chemical reaction rate can be modeled in OpenFOAM with the Arrhenius equation

[32] which takes the form [33]

$$\dot{\omega}_i = AT^{\beta} \exp\left(\frac{-E_a}{RT}\right), \tag{2.12}$$

where A is the pre-exponential factor, β is the temperature exponent, and E_a is the activation energy. OpenFOAM requests `A`, `beta`, and `Ta`. We can write $Ta = \frac{Ea}{R}$. While in later sections the effect of A is explored on the solution, the values landed on for numerical simulation in this work are:

$$A = 1.4 \times 10^{13}\ \mathrm{m^3 mol^{-1} s^{-1}}, \qquad Ta = 12996\ \mathrm{K}, \qquad \beta = 0\,. \tag{2.13}$$

In addition, the specific gas constant used is $R = 368.9$ J/Kg-K. It is seen in Chapter 3.3 that the values used are reasonably close to Chapman-Jouguet theory as well as other published values [2, 34].

2.3 Numerical Methods

Within OpenFOAM, a user can choose how to numerically solve the stiff set of differential equations that model reacting flows. Thermodynamic variables such as specific heat can be selected to be temperature-dependent using polynomial approximations such as those given by the JANAF tables [35], such that $C_{p,i} = C_{p,i}(T)$. No significant increase in complexity of the solution was noticed, so JANAF settings were used for temperature-dependent thermodynamics. The energy equation solution variable can also be selected, such that either sensible enthalpy or sensible internal energy is solved, and we selected sensible internal energy for modeling detonations with `rhoReactingCentralFoam`, since less numerical error was noticed in the solution, as compared to solutions using the sensible enthalpy. The numerical solution of the reacting Navier-Stokes equations in OpenFOAM occurs with several different settings. Different numerical schemes can be applied separately to variables of interest, and can be seen in Table 2.2.

In Table 2.2, Gauss linear specifies two things [31]. The "Gauss" portion specifies that OpenFOAM should use a standard finite volume Gaussian integration [36] with inter-

Table 2.2: OpenFOAM finite volume numerical schemes applied during solving

Term	OpenFOAM Variable	Numerical Scheme
Flux Scheme	`fluxScheme`	Kurganov
Time Scheme	`ddtSchemes`	Euler
Gradient Schemes	`gradSchemes`	Gauss linear
Divergence Schemes	`divSchemes`	none by default
	`div(tauMC)`	linear
	`div(phi,`specie`)`	van Leer
Laplacian Schemes	`laplacianSchemes`	Gauss linear uncorrected
Interpolation Schemes	`interpolationSchemes`	default linear
	`reconstruct(rho)`	Minmod
	`reconstruct(U)`	MinmodV
	`reconstruct(T)`	Minmod
	`reconstruct(Yi)`	Minmod
Surface Normal Gradient Schemes	`snGradSchemes`	uncorrected

polation of values of interest between face centers and cell centers. The "linear" portion tells OpenFOAM that piecewise linear interpolation should be used for the interpolation. It should be noted that the "uncorrected" term is a flag that tells OpenFOAM that the orthogonal component of a flow variable with respect to a cell face should not undergo a non-orthogonal correction, since structured grids are used in this research. "Corrected" is more useful for engineering geometries where the mesh faces are not always aligned with a Cartesian coordinate system.

The next important scheme is the van Leer [18] flux limiting scheme that limits the solution gradient near regions such as shocks. Cell interpolation with this method is piecewise linear, but the slopes are limited. This scheme requires use of the linear Godunov scheme [37] that assumes a discontinuous solution on ever finite volume cell face. Like the van Leer scheme, Minmod [38] is a slope limiter, but uses a different set of criteria for the limiting. Time integration is performed using the Euler scheme [39], which is a transient and first-order implicit scheme. This scheme relies on knowledge of the derivative and therefore both the current and subsequent solution in order to give an approximation of the solution at the next time step. By knowing the slope of the solution, we can guess the solution at the next

time step by tracing out where it should be by this rate of change. The last numerical scheme used is the Kurganov scheme [26], a central difference scheme by Kurganov and Tadmor that is second order and high resolution.

Once the numerical schemes are defined, the system of equations needs to be solved. The mass, momentum, energy, and species equations are combined together in matrix linear form to be solved ($\boldsymbol{Ax} = \boldsymbol{b}$). OpenFOAM allows the user to alter the solver settings for each solution variable [31]. The velocity, internal energy, and species mass fraction utilize the PBiCGStab solver mentioned earlier. This is an iterative linear solver that uses a preconditioner to help accelerate convergence of the solution. Computationally, there are two matrix-vector multiplications, six parallel reductions, and a twice-applied preconditioning step that occurs every time step. Upon startup there is an initial matrix-vector multiplication and a single parallel reduction. The preconditioner used for enthalpy and species is known as DILU, or Diagonal-based Incomplete LU [40]. This preconditioner performs an incomplete LU factorization [41] of the solution matrix and reduces the solver sensitivity to errors or changes in input. The preconditioner for velocity is DIC, or Diagonal-based Incomplete Cholesky [42]. This preconditioner performs an incomplete Cholesky factorization of the solution matrix to precondition the solution matrix. Density is solved using the `diagonal` solver, a simple solve by a solution matrix inversion. Parameters used for numerical solving can be seen in Table 2.3.

Table 2.3: Finite volume numerical solvers and preconditioners used for solution variables

Variable	Solver	Parameter	Value
`e`, `Yi`	`PBiCGStab`	Preconditioner	`DILU`
		Tolerance	`1e-17`
		Relative tolerance	0
`U`	`PBiCGStab`	Preconditioner	`DIC`
		Tolerance	`1e-15`
		Relative tolerance	0
`rho`	`diagonal`		

The chemistry uses a separate solver than the rest of the reacting flow solution variables. The settings for the chemistry solver are found in `constant/chemistryProperties`. The directory structure of OpenFOAM is discussed in greater detail in the next section. The chemistry is solved before the flow variables, with a separate initial time step. For detonation modeling presented here, the chemistry ordinary differential equation (ODE) is solved using the Runge-Kutta-Fehlberg method [43], known to OpenFOAM as RKF45.

Together, the numerical schemes, numerical solvers, and chemistry solver combine to solve the reactive Euler equations as `rhoReactingCentralFoam` within OpenFOAM. These solution techniques can then be applied to flow problems such as detonations.

2.4 Code Organization

In order to simulate a fluids or combustion problem in OpenFOAM, the directory structure in which the simulation is to be run needs to be correct. Each OpenFOAM simulation contains a directory `0/` which holds initial conditions for the domain as well as boundary conditions [31]. Inside this directory are files named like the quantities they represent, such as `p`, `T`, and `U`. `funkySetFields` edits these files to apply the special initial conditions. As the simulation progresses, more folders representing timesteps will appear with data saved for that timestep. The folder name will represent the specific timestep exactly.

The next important directory is `constant/`. This directory holds physical quantities as well as mesh information once it is formed. For reacting flows, some files of interest within this folder may be:

- **chemistryProperties**: a file that contains definitions for which chemistry solver is used as well as tolerances.
- **combustionProperties**: contains the combustion model (laminar, or turbulence model) definition.

- **dynamicMeshDict**: defines adaptive mesh refinement options. One can define which variable is tracked during adaptive meshing as well as how often and refined the meshing is.

- **reactions**: contains all the reactions that occur during the simulation. The active species are defined here as well as the rate model (such as Arrhenius).

- **thermo.compressibleGas**: included are the thermodynamic properties such as temperature-dependent specific heat coefficients, viscosity coefficients, and molecular weights of the species that are included within the chemistry models.

- **thermophysicalProperties**: defines things such as the equation of state, inert species, locations of `reactions` and `thermo.compressibleGas`, if the mixture is reacting, if the thermodynamic properties are temperature-dependent, and how the energy equation is defined (e.g. one can have the model focused on sensible enthalpy or sensible internal energy).

- **turbulenceProperties**: defines turbulence modeling for the flow in general.

The last necessary directory is `system/`. This directory contains files that will define some generic solver tolerances as well mesh setup and simulation setup. Specifically, it could contain:

- **blockMeshDict**: defines the mesh. A user will input dimensions and how the hex mesh will be partitioned. This could also be placed within `constant/polyMesh/`.

- **controlDict**: the overarching control file for OpenFOAM. This contains start and end times, time increments, write-to-file frequency, save file compression and format, and adaptive time stepping parameters such as maximum Courant number [44]. Other options such as functions for residual tracking and libraries for functions like `funkySetFields` can be placed here.

- **decomposeParDict**: defines parallel processing domain decomposition. The number of processors as well as where domain splits should occur or algorithm definitions for automatic domain decomposition are set here.

- **setFieldsDict/funkySetFieldsDict**: defines custom field setting, as discussed earlier. Blocks of pressure or temperature ramps are defined here.

- **fvSchemes**: finite volume methods are defined here, such as Gauss linear or van Leer MUSCL schemes. It is broken into derivative, gradient, divergence, laplacian, and more, sections.

- **fvSolution**: defines the specific differential equation solvers as well as their tolerances. This can also contain smoothers and other solver-specific settings.

- any files that define sampling for post processing, such as line plots through a region in the domain.

Together, these directories comprise the OpenFOAM directory structure. A user then will run the solver of interest within the directory containing these subdirectories, and the solver will look within these subdirectories for the files just discussed and apply them to the solution process. Pre-processing and post-processing steps are also run from this directory, such as creating the mesh from `blockMesh`, placing special initial conditions with `funkySetFields`, or obtaining results at a timestep with a line plot with `postProcess`. Results can also be viewed graphically with `paraview` or `paraFoam`.

Line plots seen in the upcoming sections were sampled with the OpenFOAM utility `postProcess` with the `lineUniform` type, using 1000 sampling points interpolated from cell centers. The line progresses from the edge of the initiation of the detonation to the end of the detonation tube. This data is then read into the Julia programming language for plotting. The colorful surface plots seen in this chapter were produced with ParaView[45], a graphical interface for post-processing data of many types.

At this stage, some modeling notation should be introduced to simplify some things. Initial and static grid resolutions will take the form "Nx-Ny-Nz" with N representing the number of evenly-spaced cells that will fill the dimension, and x/y/z representing the dimension. A 500-40-3 grid represents 500 cell divisions in x, 40 divisions in y, and 3 divisions in z. This notation will be used later to quickly distinguish between resolutions and dimensionality of a mesh. Since OpenFOAM does not strictly have a one-dimensional or two-dimensional operating mode, three dimensions must always be realized. This is not a problem however, as a single cell can be used to represent a flattened dimension, with special boundary conditions for the faces that are in the "dimensionless" plane. For example, a two-dimensional geometry would be represented with a 500-40-1 mesh, and a one-dimensional geometry with a 500-1-1 mesh.

2.5 Domain Setup

The geometry used for this research is seen in Figure 2.1. The so-called "inlet" (even though fluid is not actively moving through this face) is the region where the detonation is initiated. The inlet is a solid adiabatic wall with velocity slip conditions such that the fluid can slide across it without effects from viscosity. This was chosen such that just the detonation itself can be studied in a pure form. Other walls are like this, such as the walls that make up the sides. It should be noted that the sides located in the x-y planes ("front and back") change their type depending on the simulation. Since OpenFOAM does not have a two-dimensional mesh mode, for static mesh two-dimensional simulations these faces are set as a special "empty" patch type in order to tell OpenFOAM that it is a two-dimensional simulation taking place. For three-dimensional simulations the front and back faces are identical to the x-z "top and bottom" planes, which like the inlet are velocity-slip adiabatic walls. Two-dimensional AMR runs also use the three-dimensional wall setup here as the oct-tree cell splitting algorithm for mesh refinement is inherently three-dimensional even if we are simulating a two-dimensional domain. Due to this, two-dimensional simulations are

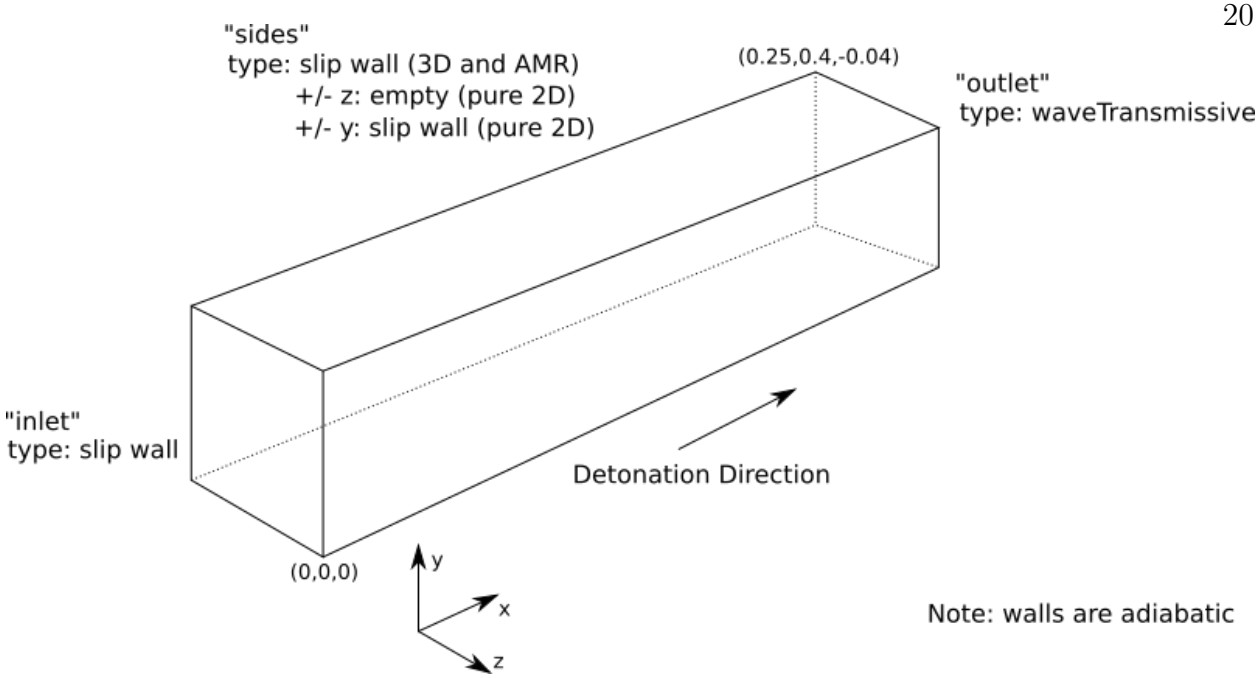


Figure 2.1: Geometry and domain setup with boundary conditions

actually pseudo-two-dimensional.

The exit plane of the domain utilizes a "waveTransmissive" boundary condition. This is a special boundary condition that allows for waves such as shock waves to not reflect at the boundary while expelling gases [46]. A traditional OpenFOAM pressure outlet was seen to reflect the shockwave coupled with the detonation wave during testing of various boundary conditions.

The last required conditions to set are the initial conditions. The initial conditions for the domain are set at 300 K and 1 atm. An ignition region is initialized at a higher pressure and temperature at the start of the detonation tube and is discussed further in Section 2.9.

2.6 Parallel Computing

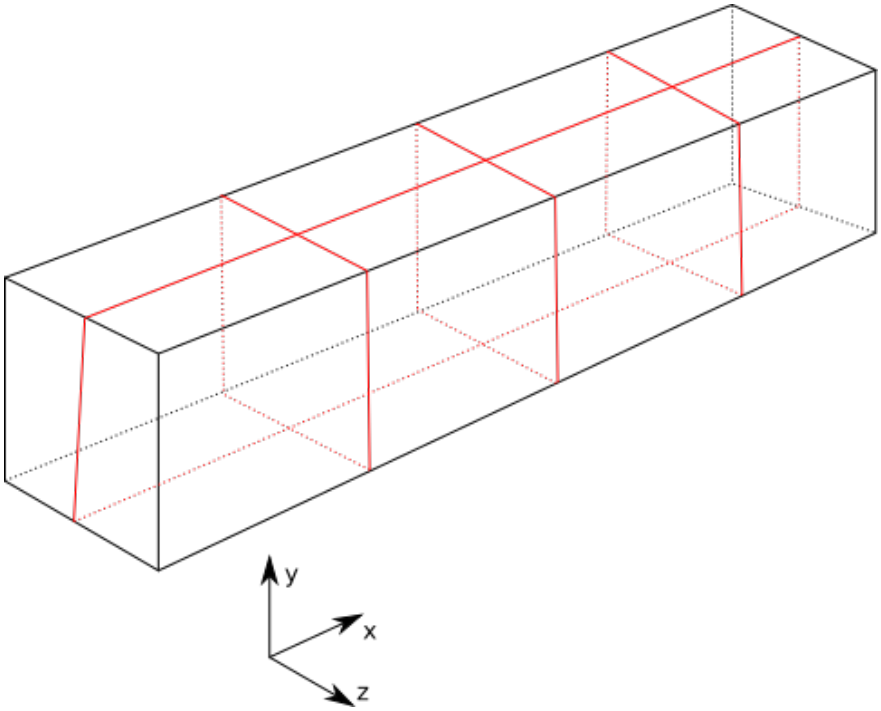
Included with the standard installation of OpenFOAM are solvers that are capable being scaled to make use of multiple processors [31]. Solvers like `rhoReactingCentralFoam` include this functionality. OpenFOAM's parallel processing is done through Message Passing

Interface (MPI) [47], where several instances of a solver are run simultaneously and MPI passes information between the solvers. Each solver running in parallel computes on a different portion of the domain.

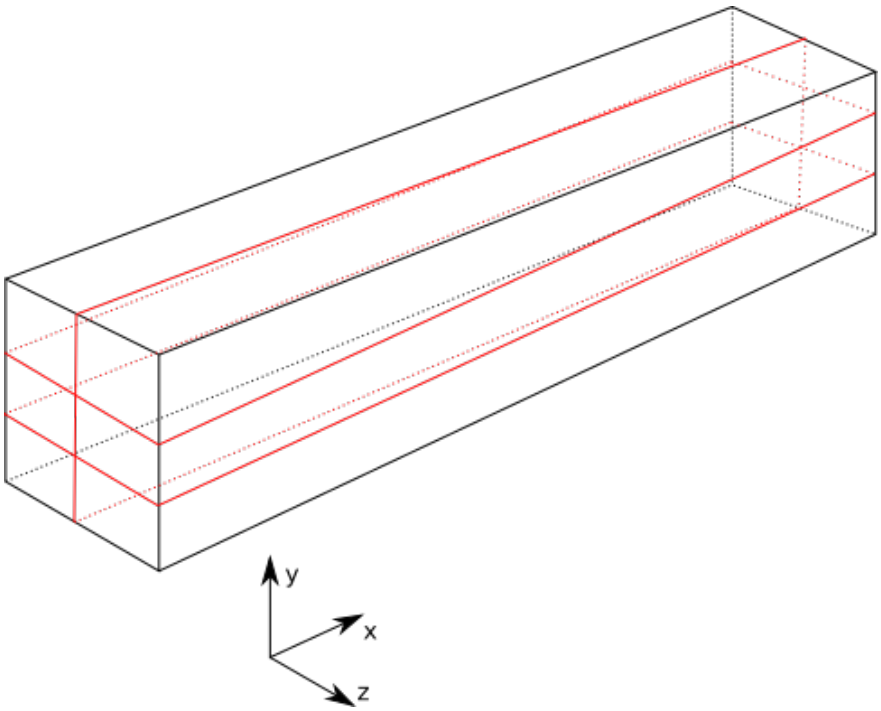
In order to be able to run a solver in parallel, the domain must be decomposed into smaller pieces first. This is done with the `decomposePar` utility, which takes user input from the `decomposeParDict` file. Within this file, a user dictates how many times the domain is broken up, and how or where the domain is broken up. Several different decomposition methods can be utilized:

- **simple**: this is the most common method used for decomposition. The domain is split into chunks based on direction, and the user specifies how many splits there should be in each direction.
- **hierarchical**: similar to the `simple` method, but instead an ordering to the domain splitting is used
- **scotch**: this method does not require geometric input from the user but rather it attempts to minimize boundaries between processors. Processor weighting can be set.
- **manual**: cells are manually allocated to each processor here.

In the researched performed here, the `simple` method for domain decomposition was utilized due to the simplicity. Since detonations have small regions of high pressure, temperature, and velocity, some care must be taken with respect to how the domain is decomposed, especially when considering AMR. If one were to decompose the domain into several pieces along the length of the tube, some processors would be working harder than other processors as the detonation passes into their respective domain chunks. We load balance the processors by not splitting the domain along the length of the detonation tube, and instead across it. This is shown in Figure 2.2.



(a) Domain decomposed into typical chunks, bad for detonation and AMR load balancing



(b) Domain decomposed into long chunks, better for detonation and AMR load balancing

Figure 2.2: Example domain decomposition techniques for parallel computing

Using long slivers seen in Figure 2.2b instead of a typical aspect ratio consistent decomposition in Figure 2.2a allows for every processor to always have a small portion of the detonation in its domain. Otherwise, some processors would be waiting on other processors that have the detonation in them or heavy mesh refinement regions from the AMR routines.

2.7 Adaptive Mesh Refinement

Adaptive mesh refinement is the process of refining or unrefining the computational mesh based on a tracked parameter of interest at a specific point in time. As seen in some of the previous work discussed, some AMR routines in previous research track the solution residual and refine the mesh based on high residual errors. In OpenFOAM, AMR can track any variable exported in the solver source code, and is available with several standard solvers such as `interFoam`, and has been implemented into `rhoReactingCentralFoam` by Caelan Lapointe. The AMR in this solver functions by looking within a mesh cell at a quantity of interest at a specific time interval, and deciding based on the value of that quantity of interest whether or not to refine the mesh in that cell. If the quantity is within the refinement bounds set by the user, the AMR routine will split the cell with an octree splitting method as shown in Figure 2.3. Additionally, the AMR can unrefine the mesh if the quantity of interest is outside the refinement bounds set by the user. The user settings are located within `constant/dynamicMeshDict`, and several options exist for the user to tweak. Some of the most important ones include:

- `dynamicFvMesh`: allows for the user to turn on and off the dynamic mesh refinement.
- `refineInterval`: how often to do mesh refinement in terms of time step iteration. For example, a value of 2 here would mean that the AMR would look to refine or unrefine every other time step.
- `field`: the variable of interest to track and refine based off of. If set as `T`, AMR would check the temperature field values.

- `lowerRefineLevel`: the lower bound for the AMR `field` value tracking to refine upon. If the solution is higher than this bound during a `refineInterval`, the cell will be refined. If the value is lower, it will not refine.

- `upperRefineLevel`: the upper bound for the AMR `field` value tracking to refine upon. If the solution is lower than this during a `refineInterval`, the cell will be refined. If the value is higher, it will not refine.

- `unrefineLevel`: the value of `field` that will trigger unrefinement. If the value is less than this quantity, the AMR will unrefine cells.

- `nBufferLayers`: how additional layers of cells will be refined around the cell that triggered a refinement.

- `maxRefinement`: how many additional times cells can be recursively refined within an iteration.

- `maxCells`: the maximum number of cells in the computational domain that can be reached before adaptive mesh refinement is no longer performed. Note that this is not strictly followed but rather an estimate provided to the solver, since the octree splitting can potentially increase number of cells beyond this value at a specific iteration.

An exploration on how different selections in these parameters affects solution quality was performed in this thesis. Results will be shown in later chapter sections. An important aspect to note when utilizing `rhoReactingCentralFoam`'s AMR is that the routine is inherently three-dimensional, due to the octree splitting method in Figure 2.3. Due to this, one-dimensional simulations become pseudo one-dimensional, and two-dimensional simulations become pseudo two-dimensional, since cells will be added in all dimensions when refinement occurs. For detonation modeling in areas like detonation tubes, PDEs, and RDEs, this is

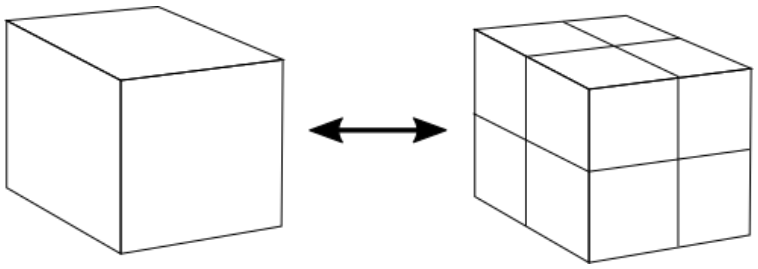


Figure 2.3: Adaptive mesh refinement octree splitting method

acceptable as the velocity in other dimensions tends to be very small in comparison to the velocity in the detonation direction.

Within `dynamicMeshDict`, two main parameters can be used to guide the adaptive mesh refinement regionally to the target static mesh resolution. `maxCells` stops the adaptive meshing from splitting and refining cells further once a `maxCells` number of cells is reached in total throughout the domain. `maxRefinement` tells the adaptive meshing routines how many further times per iteration it can split and refine cells, allowing for very fine cell resolution in small regions. The necessary number of refinement levels in a detonation simulation to reach target static cell resolution can be described with

$$\texttt{maxRefinement} \geq \log_2 \left(\frac{N_{\text{static target}}}{N_{\text{AMR base}}} \right) \tag{2.14}$$

where $N_{\text{static target}}$ is the static mesh target resolution and $N_{\text{AMR base}}$ is the unrefined base mesh resolution provided to the AMR simulation. For example, if our target static mesh had square cells and 5000 cells in length and the mesh we would like to use AMR with has 250 cells in length, this means that we need to use $\texttt{maxRefinement} \geq 5$ in order to reach the static mesh resolution. This does not guarantee that the simulation will be exactly as accurate as the static mesh resolution however, as other parameters such as the number of buffer cells, maximum number of cells, and more importantly the tracked field for refinement all change how the mesh will be refined. Note that the progression from refined cell to cell is also fairly smooth, such that a single octree touches each other. If a region needs to have lots of refinement, one must take this into account and provide the appropriate buffer

layers with `nBufferLayers` to allow the AMR to smoothly transition the mesh in time for the sensitive region to have an appropriate refinement level. The next parameter for mesh control discussed is `maxCells`. The maximum number of cells can be estimated by considering the region where static mesh refinement needs to be active. By adding the number of cells in this region in the refined static mesh resolution to that of the base AMR resolution, an estimate for `maxCells` can be determined. Note that if more buffer layers are utilized, `maxCells` needs to be increased.

In Figure 2.4, an example AMR detonation simulation is shown. The black boxes are the cell edges, and the adaptive meshing around the detonation wave can be clearly seen. Transition and buffer layers are visualized as well, as AMR moves from one layer to the next spatially.

2.8 Initial Meshing

While the primary topic of interest in this work is adaptive meshing, a primary static mesh is required in order for the adaptive mesh refinement routines within OpenFOAM to refine from. Structured meshes are utilized, with hexahedrons comprising the mesh. Meshes used for static mesh and initial AMR simulations can be seen in Figure 2.5. The meshes displayed utilize 125 partitions along detonation "tube" and 20 partitions in width, depending on two or three dimensionality in order to be visible for the reader. Real meshes used in simulation are much more fine in order to capture the thin detonation and shock region, but these resolutions are feasible for adaptive meshing, as seen in the results. Meshes used in this research were created with the OpenFOAM utility `blockMesh`, which generates structured hex meshes based on use input of "blocks" which define rectangular prisms that `blockMesh` then uses to form faces and patches for boundary conditions as well as to partition into the mesh.

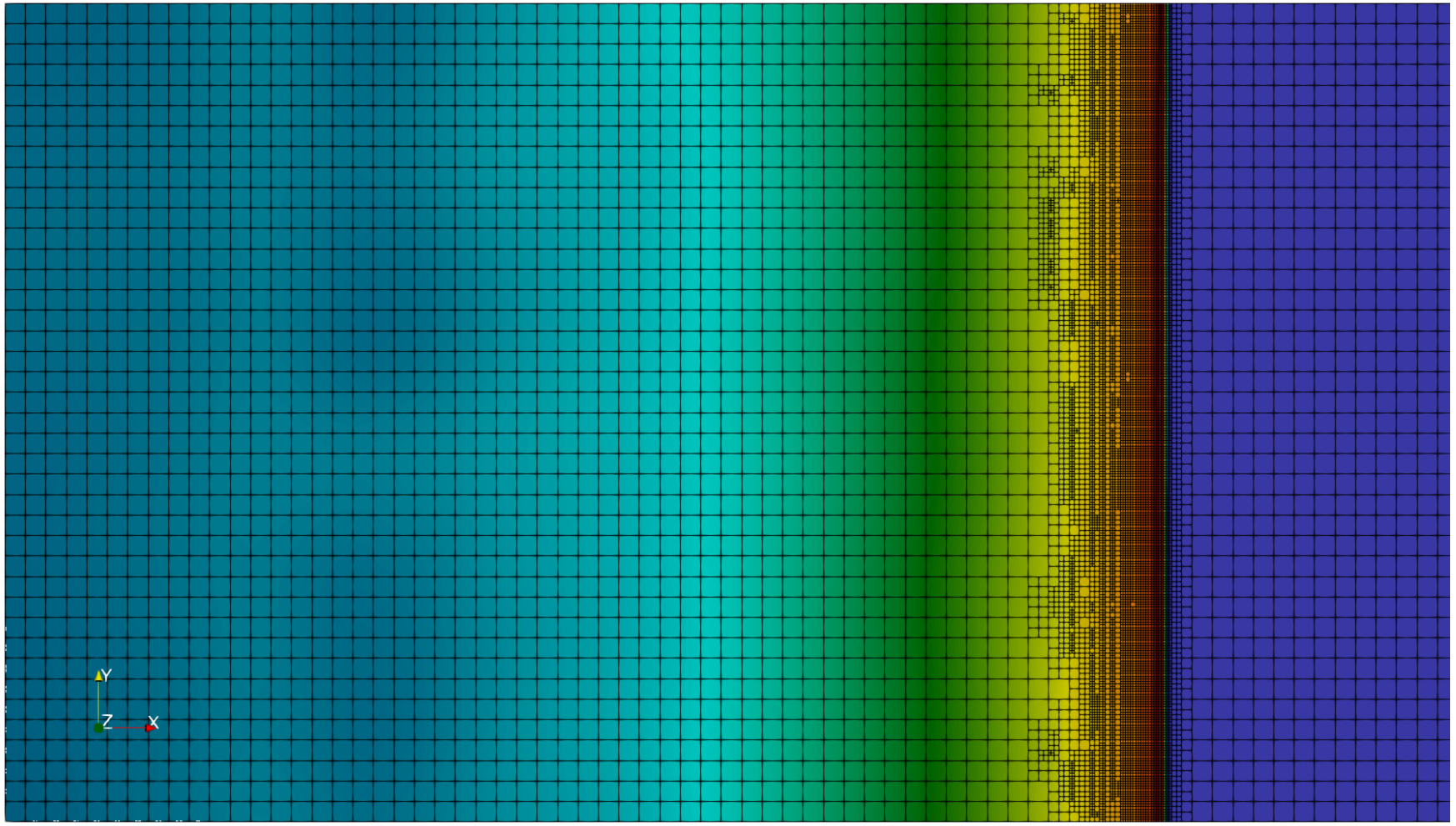
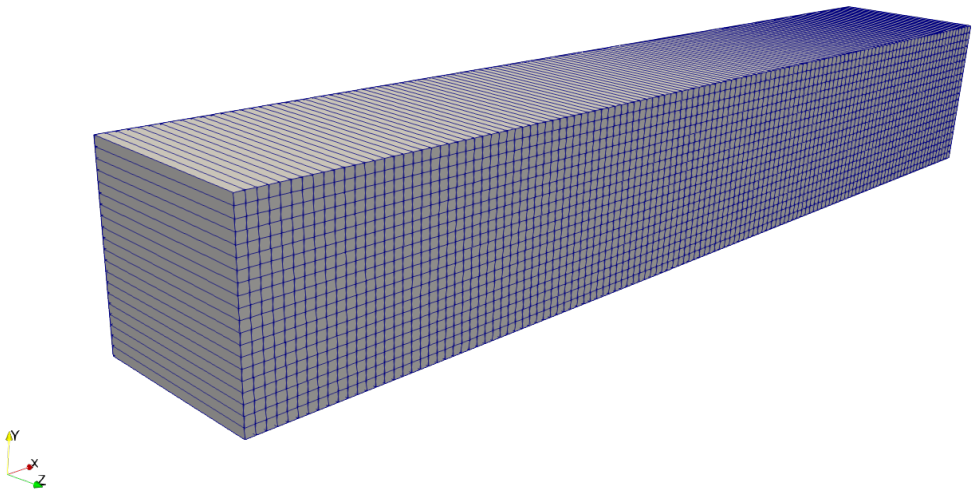
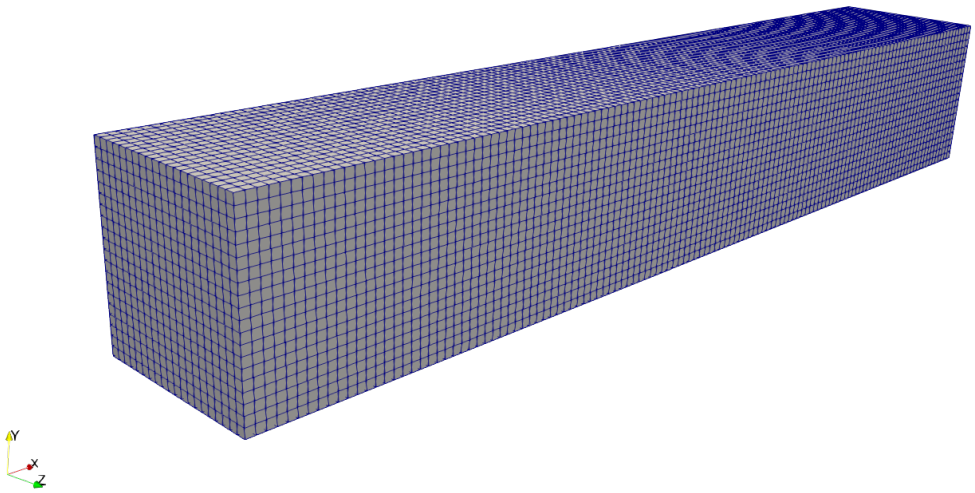


Figure 2.4: Three-level adaptive mesh refinement over a pressure field surface contour, with the detonation wave traveling from the -x wall to +x exit



(a) Two-dimensional mesh



(b) Three-dimensional mesh

Figure 2.5: Static meshes used in OpenFOAM for detonation modeling, at an exaggerated unrefined resolution for example purposes

2.9 Detonation Initiation

Once the solver have had parameters set and the domain is prepared, a method to initiate a detonation must be applied. The initiation of a detonation is seemingly trickier computationally than it is experimentally. There are several methods to do this. Typically, a region of high temperature and pressure gas is instantiated at one end of the tube, such that the fuel and oxidizer ignites within the tube. If the pressure and temperature conditions within this ignition region are high enough, detonation will occur. If they are not, it is possible to instead just ignite the gas and send a decoupled shockwave down the tube in front of the flame. This is not a detonation and undesirable. This decoupling was seen for certain reaction rate coefficients, explored in 3.3.

In addition to the thermodynamic conditions that the ignition region is in, the ignition gas itself can be varied. Some researchers have ignited detonation computationally using a block of high temperature and pressure helium [6]. Towery has used an ignition block of high temperature and pressure stoichiometric reactants [2] as well as a high temperature and pressure gradient [48] ranging from higher pressure and temperature near the ignition wall down to the initial condition of the domain. Ignition will be discussed further in 3.1. All the methods aforementioned can be used to ignite a detonation, but have different effects on the stability of the detonation as well as setup complexity.

2.10 Initial Solution Attempts

To begin on the problem of modeling detonations in OpenFOAM, the OpenFOAM installation-included tutorials folder as well as the geometric and initial condition setup of located in a related TESLa paper by Towery *et al.* [2] was utilized to begin the work towards an OpenFOAM model of a linear detonation tube/PDE.

Firstly, the detonation of methane fuel and oxygen oxidizer without inert nitrogen filler was explored within a 2D detonation tube with the solver `rhoReactingFoam`. A region

of high temperature methane and oxygen on the left hand side was used to initiate the detonation. In the PDE and detonation simulations presented in this paper, stoichiometric fuel and oxidizer was premixed and available thoughout the domain. A detonation was successfully created with this, but the accuracy and stability were still untested. In order to check that the boundary conditions being applied were correct, a small detonation cell of high pressure and temperature was set in the bottom left corner of the tube. As the simulation progressed, correct wave reflection was seen on the sides of the tube as well as some wave transmissive behavior from the exit was seen (Figure 2.6). It was noticed that the exit boundary condition was expelling gas in a rectangular region long before the detonation wave(s) reached the exit, so some tweaking was done on this `waveTransmissive` boundary condition in order to make it more accurate. The boundary condition option accepts inputs such as ratio of specific heats, expected flow velocity, far-field conditions and distance from exit plane, and names of some variables such as the condition to track at the exit as well as some flux variables. Thus, some knowledge of the flow is required in order to optimize a smooth exit of supersonic waves.

The next step with `rhoReactingFoam` was to start progressing towards testable detonation results. To do this, a change from methane to hydrogen for the fuel was selected. This fuel change was done to better match papers such as those being produced by TESLa as

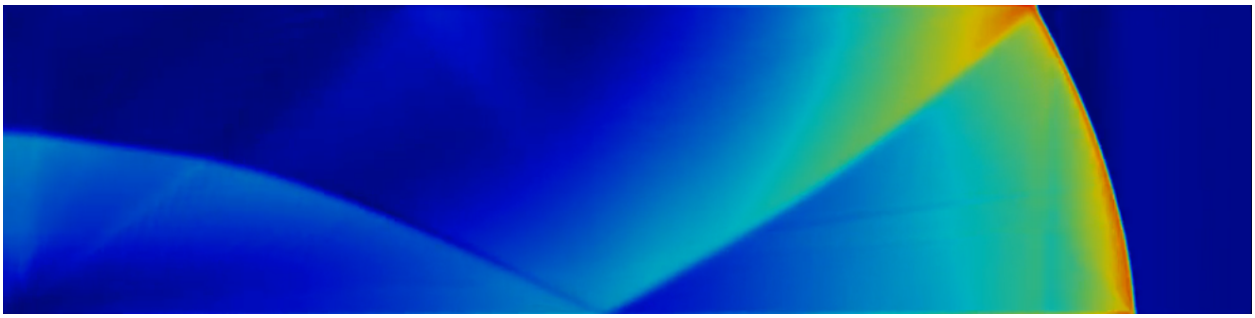


Figure 2.6: Initial methane and oxygen detonation boundary condition test with corner detonation. Velocity magnitude is plotted here without scale to just check the solver and boundary conditions for modeling potential. Detonation was initiated in the lower left corner.

well as the general experimental and computational detonation modeling community. The exact geometric detonation tube was taken from Towery *et al.* [2] for comparison as well as the initial detonation size and thermodynamic conditions. Additionally, inert nitrogen was added into the tube to transition from hydrogen-oxygen to hydrogen-air. In accordance with trying to double check whether results made sense with Towery *et al.* [2], the Arrhenius rate equation [32] was used for reaction rates. An issue that was problematic starting this research was that the documentation for the units within the `reactions` file were conflicting. Much testing was performed at this stage in an attempt to determine the units that OpenFOAM uses for the reactions given in this file, especially regarding the pre-exponential factor. While some ground was gained on this, a shift of focus towards the solver was done in order to determine if `rhoReactingFoam` was appropriate for capturing shocks.

The Sod shock tube problem was utilized to determine if the OpenFOAM solvers used were capturing shocks accurately. This problem typically consists of placing a region of higher pressure and density fluid in one region and a lower pressure and density fluid in another adjacent region. Typically the regions are rectangular and share a face, both comprising a large cube. When the simulation starts, a shock will form due to the immediate differential in pressure and density from one region to the next. This problem cannot be verified perfectly experimentally, but it can be solved analytically, making this a good method to test numerical CFD solvers for compressible flow accuracy as well as their ability to capture shocks. The results for `rhoReactingFoam` matched published analytical results as well as a `rhoPimpleFoam` case that was matched to the analytical results. Due to this, it was thought that `rhoReactingFoam` would be a good candidate for detonation modeling.

The next stage consisted of testing different mesh refinement levels, before AMR was utilized. The domain will be expanded on in later chapters, but the domain considered for the initial work is 0.5 meters in length and 0.04 meters square in width. Different static mesh resolutions were tested, such as 5000-1-1, 5000-3-3, 1000-80-1, 1000-3-1, and 10000-2-1. Additionally, some different time steps were tested. At this stage, it was found that there

was significant noise and instability in the detonation wave solution, such that consideration of a different solver was advisable. Example results for problems can be seen in Figure 2.7. We next tried moving towards OpenFOAM solvers utilizing central-upwind schemes of Kurganov and Tadmor, as they should perform better for detonations due to their design around compressible flow.

2.11 Final Solver

The final solver settled on for this research was `rhoReactingCentralFoam`. This is a solver combined together by Caelan Lapointe from `rhoReactingFoam` and `rhoCentralFoam` and the testing and validation of it is part of this thesis work. First, `rhoReactingCentralFoam` was tested to see if it was accurately capturing shocks like its sister solver, `rhoCentralFoam`. OpenFOAM includes a validated shock tube case utilizing `rhoCentralFoam`, so the results of this were compared to `rhoReactingCentralFoam` results in Figure 2.8. It can be seen that the validation case and `rhoReactingCentralFoam` match precisely, and this was expected. Better numerical stability and less noise was seen in the solutions through this change. Additionally, AMR was implemented and ready for testing. All further detonation simulations were performed using this solver. Once the solver was selected, mesh resolutions, solver parameters, and AMR parameters could be examined in greater detail.

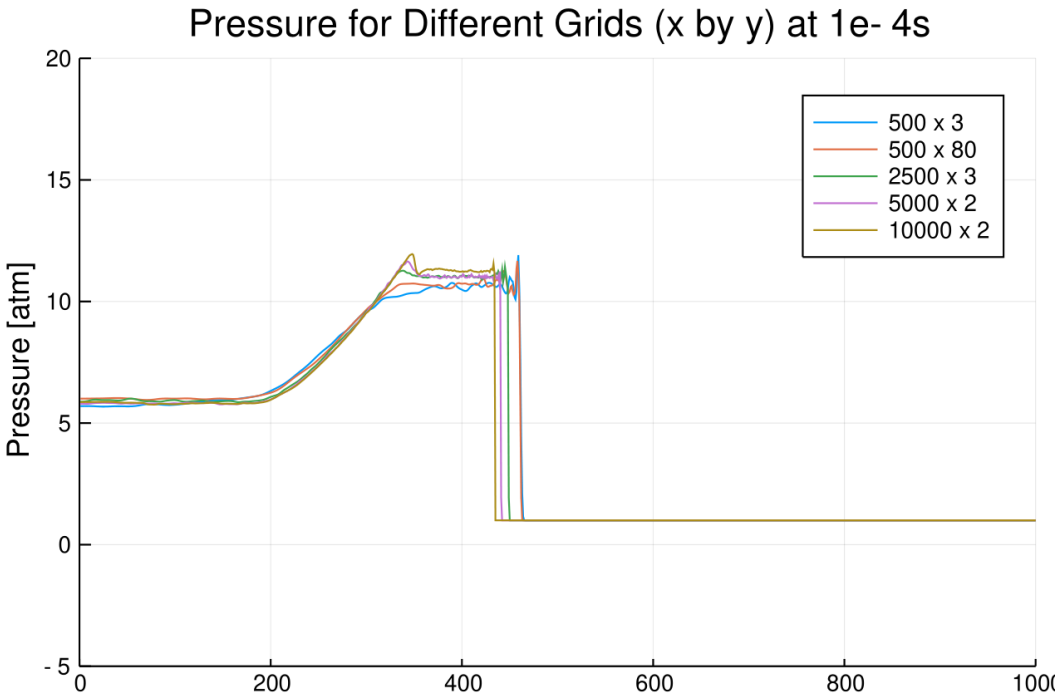


Figure 2.7: Noise and instability in solution and shock capturing problems using the `rhoReactingFoam` solver

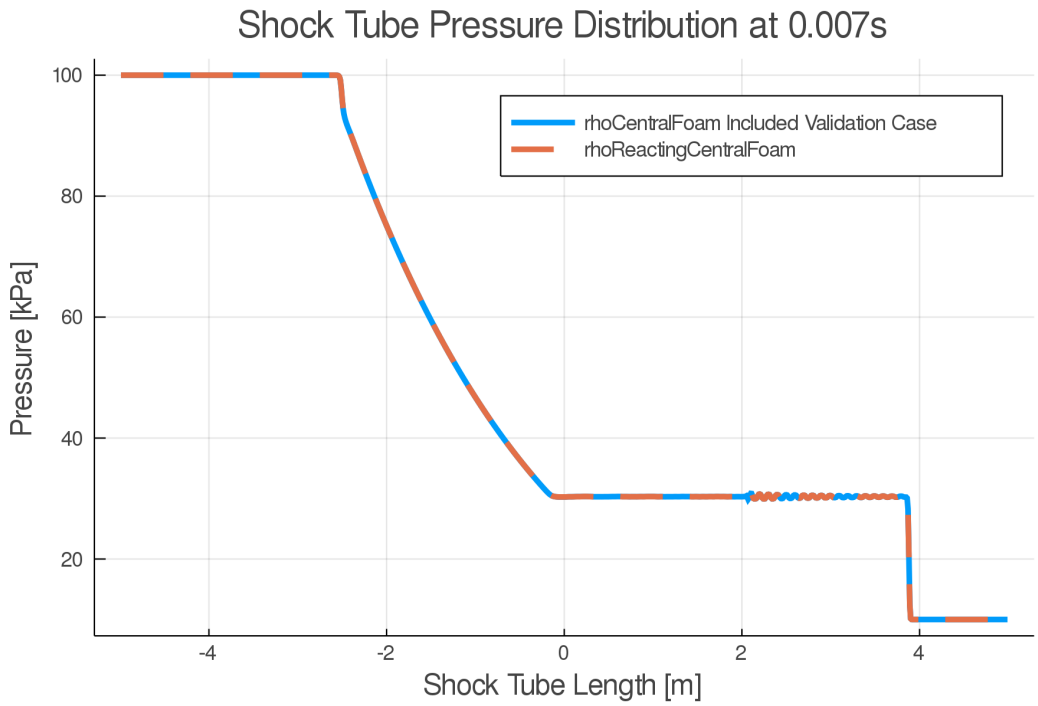


Figure 2.8: Shock tube validated test case included with OpenFOAM compared to hybrid solver

Chapter 3

Solver Parameter Testing

Now that `rhoReactingCentralFoam` has been selected for use, the solver can be tested further to determine how different selections in solver parameters alter the solution. Parameters that were tested are the Arrhenius equations's pre-exponential factor exponent, the Courant [44] number and time step variation, static mesh resolution, and adaptive mesh refinement variation. Sensitivities to certain solver parameters as well as mesh resolution on the solution were be explored.

3.1 Ignition Tests

The first necessary set of tests before solver and AMR parameters can be explored were those testing detonation ignition methods. Different ignition techniques can be used to start a detonation. Both the block ignition and gradient ignition methods were examined.

3.1.1 Block Ignition

The block ignition method with high pressure and temperature stoichiometric reactants was utilized initially to build off the work done in [2]. A rectangular region spanning from the ignition wall (y-z plane at $x = 0$ m) to $x = 0.001$ m was set with a different initial condition than the rest of the domain using the `setFields` utility. The ignition block region contained high pressure and temperature reactants at 3000 K and 20 atm, with the rest of the domain at 300 K and 1 atm. This method is seen in Figure 3.1.

Figure 3.1: Block ignition method, seen on left side of domain, at t = 0 s

Ultimately, this ignition approach was not pursued further due to instability in detonation initiation due to the very large gradient in conditions between the block ignition region and the rest of the domain. For both very high and very low grid resolutions, detonation would sometimes not occur and instead the shockwave would decouple from the flame reaction front. A better approach shown by Towery *et. al.* [48] is to use a pressure and temperature gradient ranging from the wall to the domain, smoothly transitioning the ignition region to the domain.

3.1.2 Gradient Ignition

Using the theory shown in Towery *et al.* [48], we developed an initial gradient ignition condition to guarantee a detonation across mesh resolutions. This gradient ignition spanned 0.01 m from the leftmost wall, ranging from 1200 K at the wall to the domain temperature of 300 K. The tool `setFields` cannot set fields for initial conditions with functions, so `funkySetFields` was applied for the gradient condition. This can be seen in Figure 3.2. After transitioning to this ignition method, detonations were less noisy and more consistent in their ignition. The next stage towards matching experimental results and other compu-

Figure 3.2: Gradient ignition method, seen on left side of domain, at t = 0 s

tational detonation modeling was to attempt to model detonation cells.

3.2 Cellular Detonation Modeling

Comparison of cellular detonation structure is one method to validate numerical solvers with experimental results. Maximum pressure and density plots across all time steps for every cell can be plotted to emulate experimental "smoke foil" results. The smoke foil is a piece of smooth metal placed flat against a wall in a detonation tube chamber, in which smoke particles have been evenly distributed (usually by a fire) upon the surface of the foil. When a detonation passes over the smoke foil, the density and pressure differences will leave fish scale-like patterns upon the foil.

In order to better match experimental conditions and form detonation cells, the temperature gradient initial condition was also expanded to include a randomized 240 K temperature increase throughout the domain seen in Figure 3.3. This was chosen as it is 20% of the maximum initial ignition temperature, and should provide enough randomization to instantiate detonation cells. The randomization of temperature from +0 K to +240 K is uniform and not Gaussian in distribution. With the randomization of temperature throughout

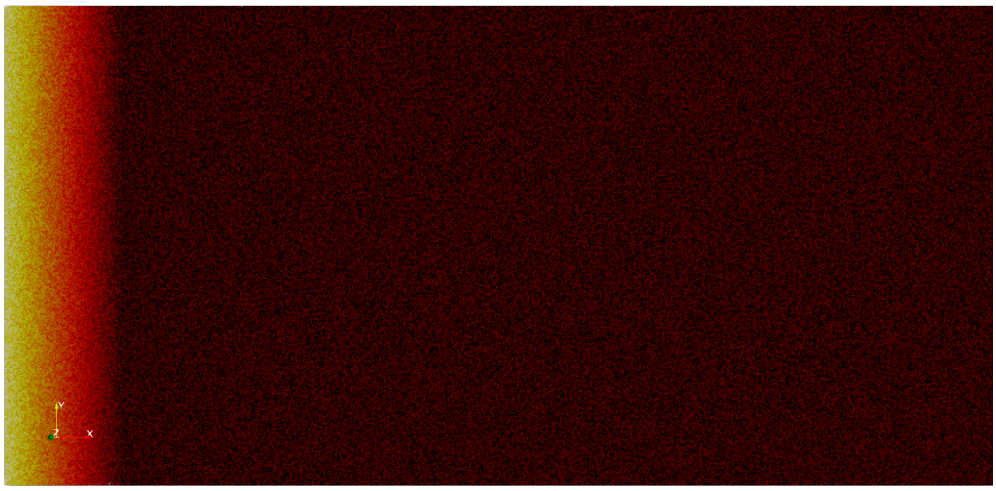


Figure 3.3: Gradient ignition method with randomized temperature distribution throughout domain, seen on left side of domain, at t = 0 s

the domain, the maximum value of density across all time steps for each cell was recorded and plotted as a surface plot. For a 2500-400-1 mesh, this is seen in Figure 3.4.

From this figure, we deduced that detonation cells are possible but never initiated with this grid resolution. The distinct cellular structure is not seen here, and further work with more accurate multi-step reaction mechanisms may have more success in bringing light to these features for detonations within `rhoReactingCentralFoam`. Note that the field was not

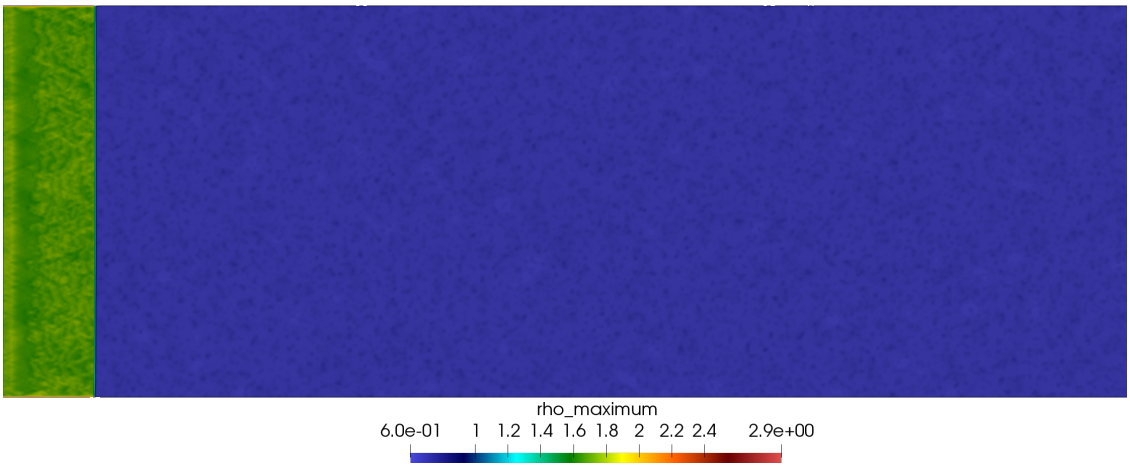


Figure 3.4: Surface plot of maximum density tracked across all time steps for each cell, emulating a "smoke foil"

tracked for very long throughout the simulation due to the extreme data storage requirements for such thermodynamic field tracking. When ParaView is utilized to access the temporal statistics on a flow variable, very small time steps are required in order for the temporal statistical plots to not appear "choppy" and broken up. For the plot here, save files were required every 1×10^{-8} s in order to be continuous and smooth. With one million cells per time step, this quickly scales. However, detonation cells would be visible at this stage in the simulation as the detonation wave has stabilized. Other factors that may assist the formation of detonation cells are mesh resolution and the strength of the randomization parameters used throughout the flow field. When utilizing AMR, care must be used when attempting to model detonation cells since un-refinement behind the detonation wave smooths the solution considerably and flame structure as well as detonation cell formation may be eliminated. It is likely best to start with a static mesh and formulate the detonation cells to compare to experimental results, and then use the cell-structure-matched static mesh as a target for tuning AMR parameters. Different ranges of randomization throughout the flow field will also alter how the detonation front progresses and forms detonation cells. Another benefit of the addition of slight noise in the background domain is the ability to determine when the mesh resolution is refined enough to begin resolving fine detonation structure. As seen in the high refinement two-dimensional static mesh detonation results later in this chapter, the fine variation in the detonation wave itself is seen and therefore the threshold for resolving such structures is known.

A pressure distribution field of the static 10000-1600-1 mesh was taken, to see if detonation cell evidence was present. Note that due to computational requirements, only a pressure field at a single time step was visualized, in comparison to the maximum pressure or density for each cell over all time steps. Figure 3.5 is the result. Note that the detonation cells are fairly small, and it is likely that with more complex chemistry models that their sizing will be more realistic.

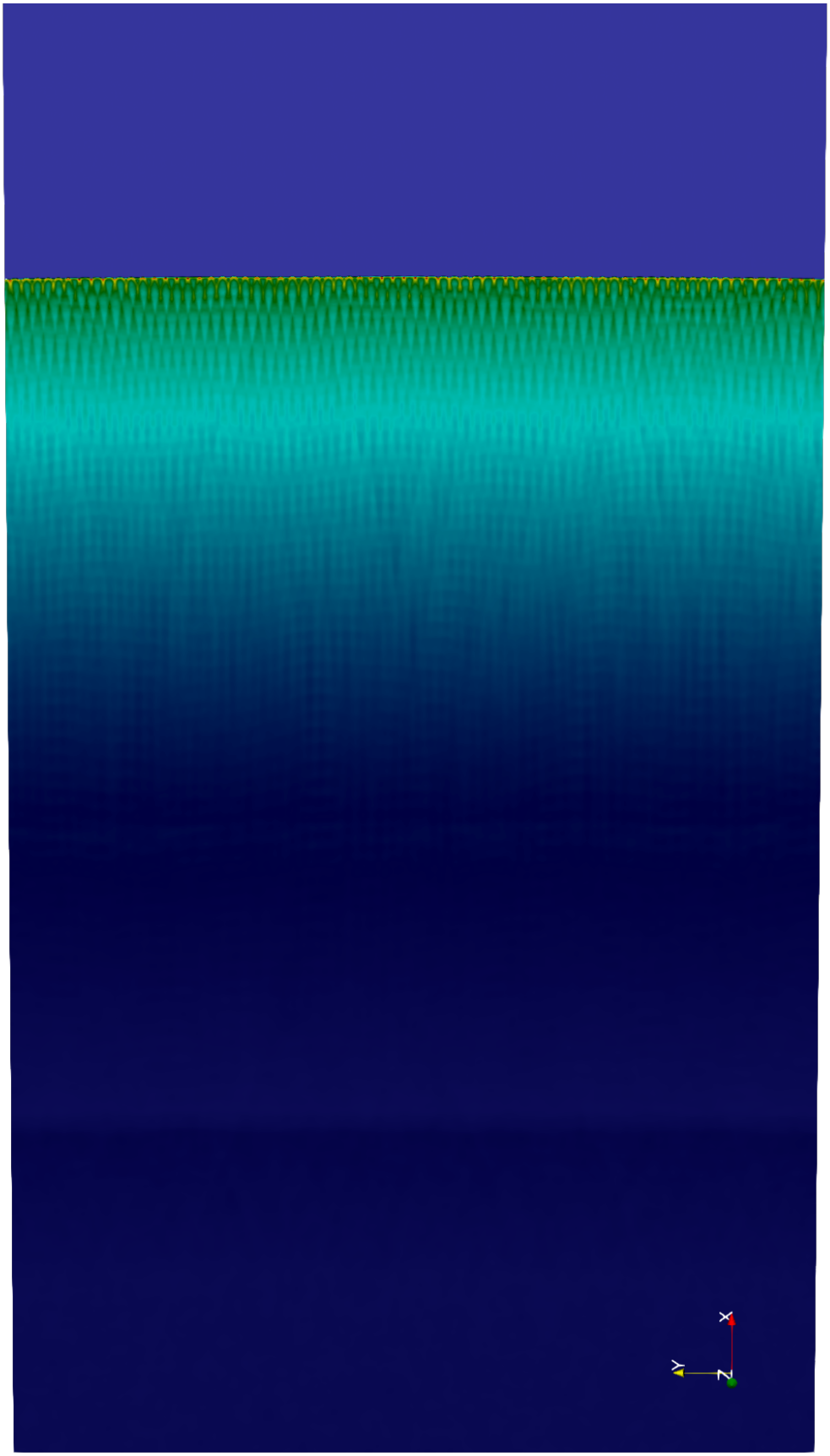


Figure 3.5: Static 10000-1600-1 mesh with detonation cell formation

3.3 Arrhenius Pre-exponential Factor

In order to better characterize reaction sensitivity and compare to other published results, testing was performed on the units of the pre-exponential factor for the Arrhenius equation in the `reactions` file. Since the geometry, boundary, and initial conditions are all matched to the setup for the detonation tube in Towery *et al.* [2], we will use the results there as a generic target to determine the order of magnitude of the exponent on the pre-exponential factor. The exponent was swept between 10^{11} through 10^{17}. The results are seen in Figures 3.6, 3.7, and 3.8.

From these figures, we can see the importance of having an accurate order of magnitude on the exponent for results. As the exponent was increased beyond the power of 12, plateaus emerge in the results for all thermodynamic variables. Under the power of 12, there is a large jump in solution, and a transition from between detonation states, seen in Figure 3.9. The differences in these states is due to the coupling of the shockwave and the reactions (flame). The grouping of plots with higher pressure values have the reactions coupled with the shock, and the smaller pressure value grouping has the reaction occurring more slowly and trailing behind the shock. This is more evident in Figures 3.10 and 3.11 where we see that there is still a shockwave present in the "lower" group, but it is clear by Figure 3.11 that the reaction process is happening much later than the shockwave instead right at the shock front. Thus presence of a shockwave in this scenario is not due to the sustained reaction, and was triggered by high pressure (and temperature) region used to initialize detonations in the simulation.

The exponent in this region was explored further since plateaus in thermodynamic properties values are not physical for real-world detonations. Note that the velocity plotted in Figure 3.8 is a fluid velocity, not a wave velocity. When instead we swept over a smaller interval, with the Chapman-Jouguet (CJ) targets from Towery *et al.* [2] plotted, we can see that there is a very abrupt transition in detonation stability (Figures 3.10 and 3.11).

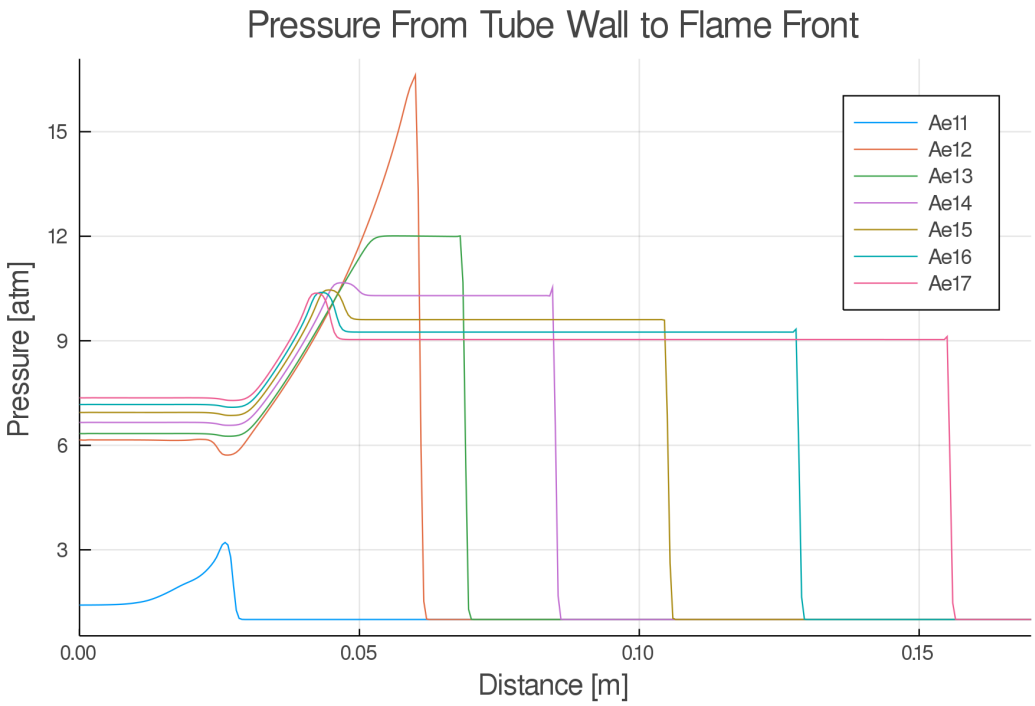


Figure 3.6: Pressure distribution in detonation tube for pre-exponential factor exponent sweep test

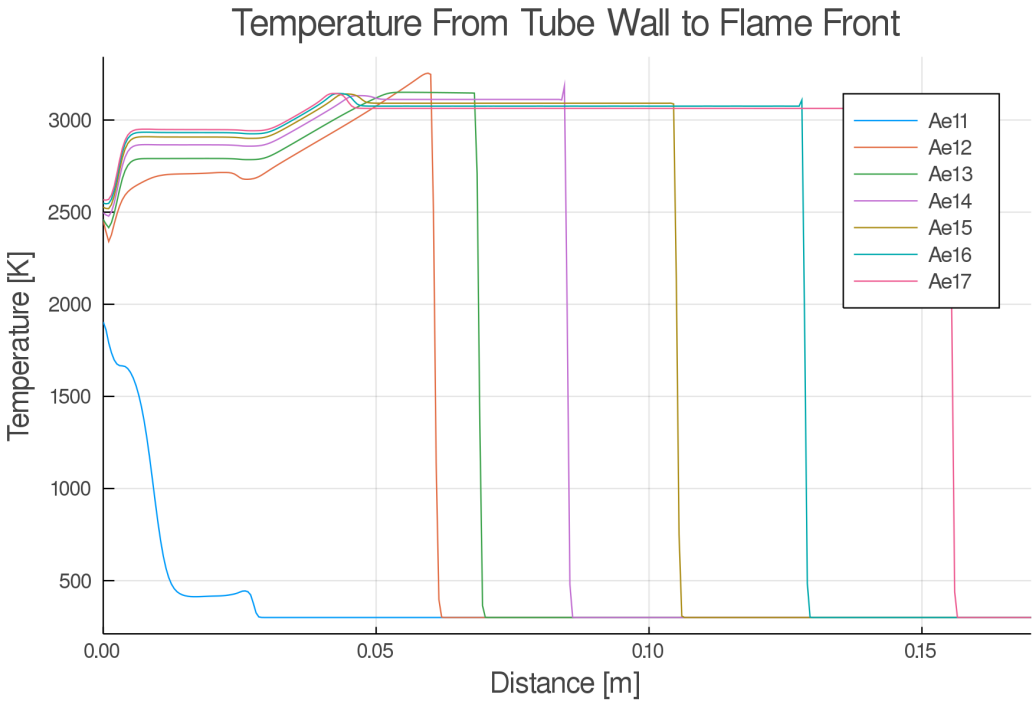


Figure 3.7: Temperature distribution in detonation tube for pre-exponential factor exponent sweep test

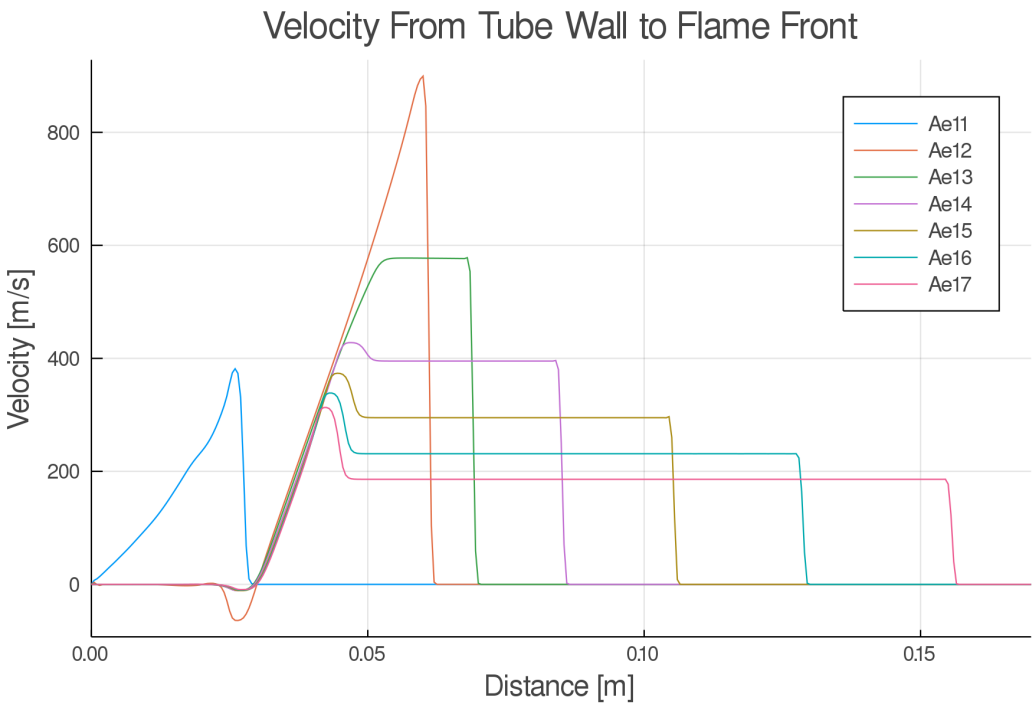


Figure 3.8: Velocity distribution in detonation tube for pre-exponential factor exponent sweep test

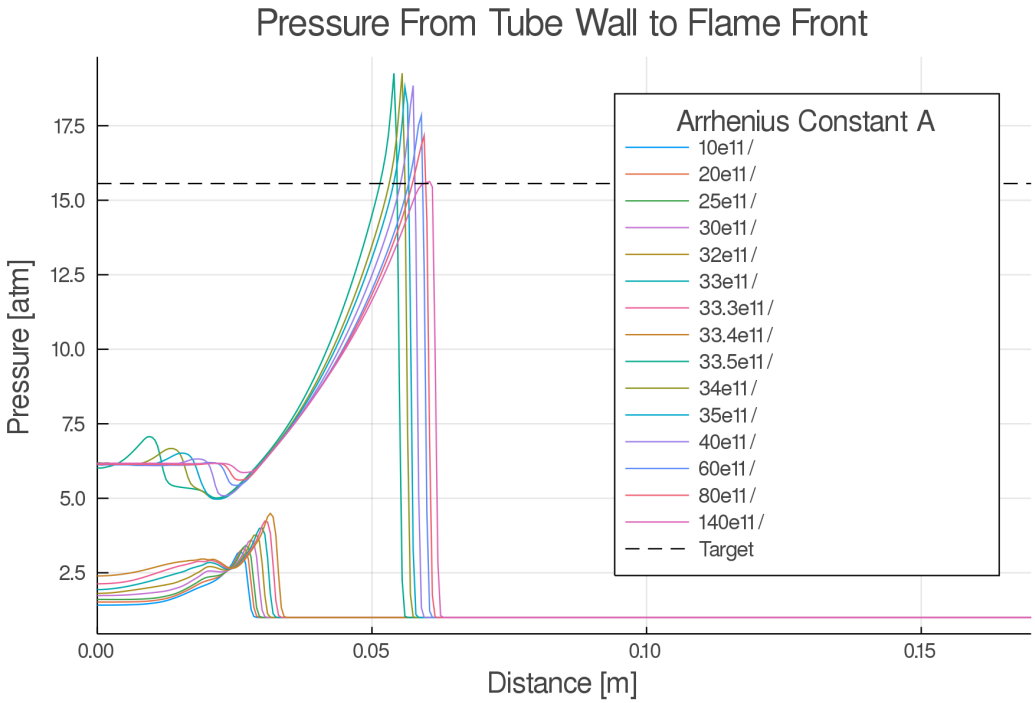


Figure 3.9: Pressure distribution in detonation tube for expanded and refined pre-exponential factor exponent sweep test, showing detonation shock-reaction coupling transition. See Figures 3.10 and 3.11 for a filtered version.

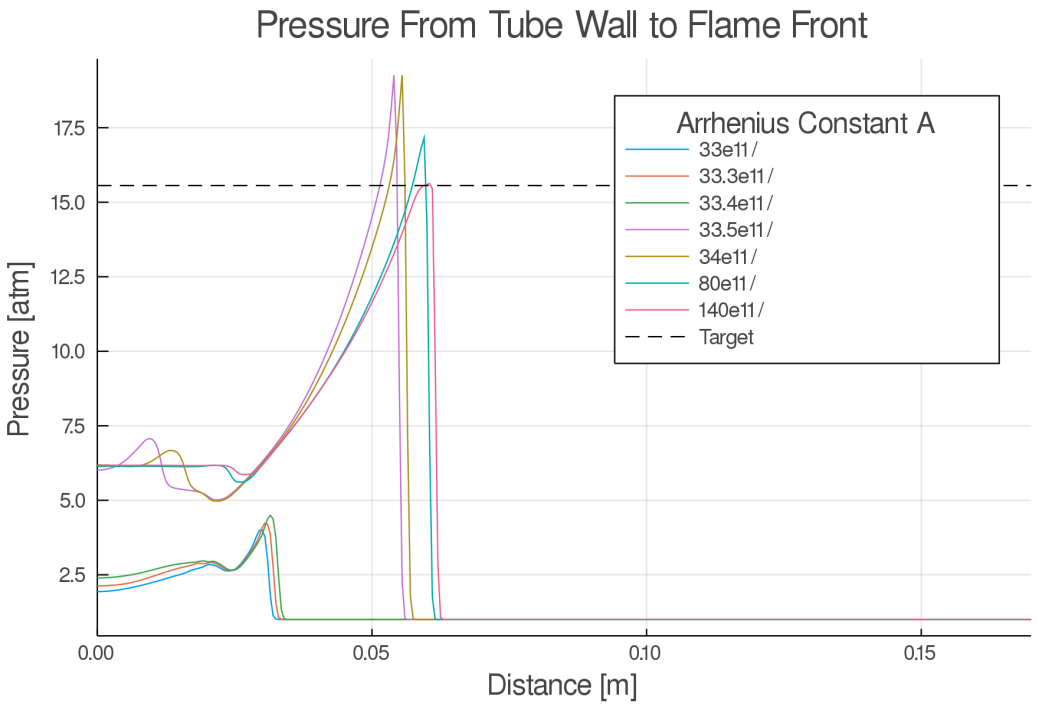


Figure 3.10: Pressure distribution in detonation tube for refined pre-exponential factor exponent sweep test, filtered from Figure 3.9 and refined from Figure 3.6

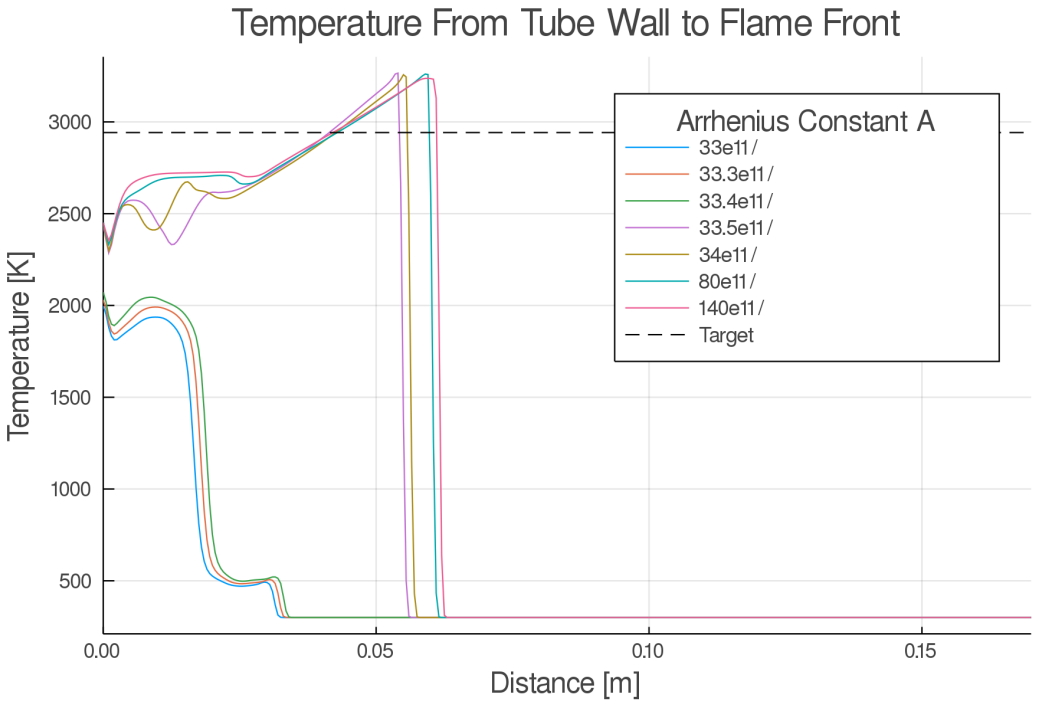


Figure 3.11: Temperature distribution in detonation tube for refined pre-exponential factor exponent sweep test, refined from Figure 3.7

With these tests, 10^{13} was selected as an appropriate order of magnitude as it not only agreed with the results seen in Towery *et al.* [2], but also with the value given for a single-step hydrogen-oxygen detonation in other published works [34].

3.4 Time Step Variation

One of the important tests was to determine how large the time step of the simulation could be taken. In OpenFOAM, one has several options for how the time step should be selected. By default, time steps are fixed, and set within `controlDict`. Setting different values for how large the time step should be will affect the stability and solution of the simulation. Larger time steps will be more unstable, increasing the Courant number and sometimes leading towards the inability for solutions at a time step to converge. If we consider a fluid particle, the faster it moves, the more distance it can cover in a unit of time. If it is moving very quickly, large time steps mean this fluid particle could have traversed very far, leading to uncertainty in the solution. In a more general sense, we consider information and how it progresses throughout the gridded domain. For Courant numbers larger than one, the fluid is moving through more than one grid point per time step, and this is very difficult for the time integrators solving the flow equations to assess accurately. The logical choice would be to always select a very small time step to ensure a more stable and accurate solution. However, with decreased time step intervals comes increased computational cost, as the computer must solve the fluid equations much more often to arrive at a specific point in time. Instead, a smarter approach can be taken, where the time step is allowed to vary based on the Courant number.

Inside `controlDict` for `rhoReactingCentralFoam`, we can set a maximum for both the typically-used fluid velocity Central Courant number as well as the acoustic (wave speed) Courant number. Each iteration, the solver will check the value of both these Courant numbers, and decrease the time step accordingly if the Courant number maximum is exceeded. In an effort to best determine how large the Courant number and therefore the time step can

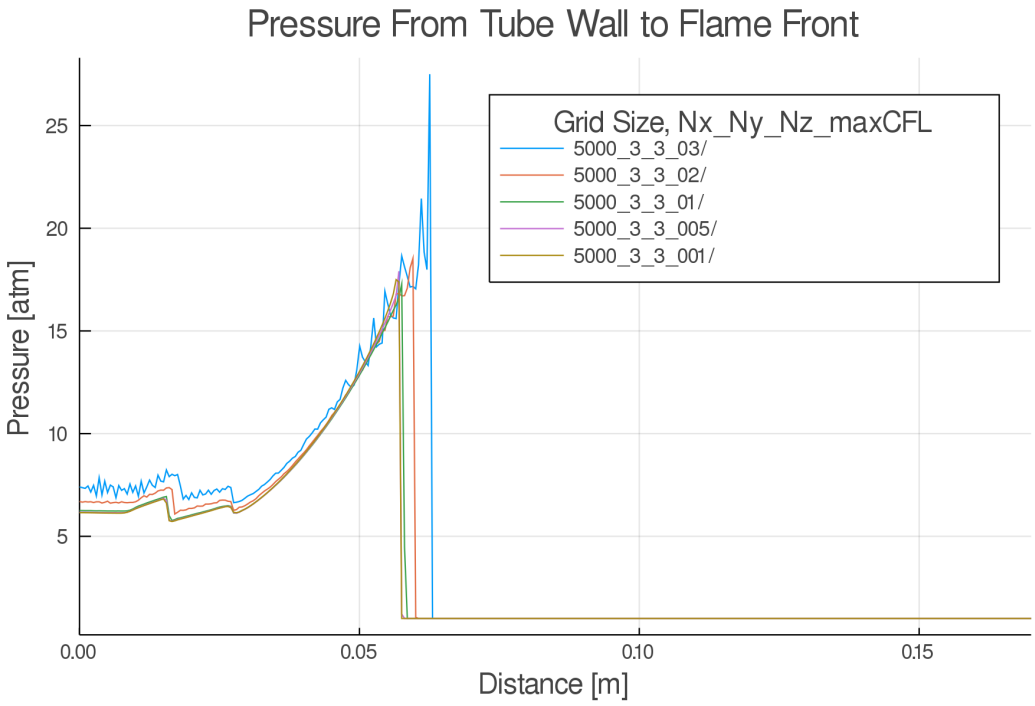
be, the maximum Courant number was set and swept across for various values. For these tests, a domain size of $(x, y, z) = (0.5, 0.04, 0.04)$ meters was used with a 5000-3-3 mesh. This is pseudo-one-dimensional setup. Several maximum Courant (CFL) [44] numbers were swept across, and the results at 3×10^{-5}s are plotted in Figures 3.12a, 3.12b, and 3.12c. Note that the notation used in the plot omits the decimal point for the maximum CFL number, so `5000_3_3_005` represents a 5000-3-3 mesh with a maximum CFL of 0.05.

From Figures 3.12a, 3.12b, and 3.12c, it is seen that there is decreased noise in the solution as well as convergence as the time step and therefore maximum CFL is decreased. More evident in the temperature plot 3.12b, noise and instability is especially clear in the detonation simulations with maximum CFLs of 0.2 and 0.3. The CFL of 0.1 was decided to be acceptably converged due to a small comparative error, seen in Table 3.1. Note that while the CFL of 0.2 has a smaller velocity error than the CFL of 0.1, it is a much noisier and unstable solution, so the values themselves have further uncertainty. The selection of a maximum CFL of 0.1 for detonations in OpenFOAM also matches with the work done by Ajaero [49].

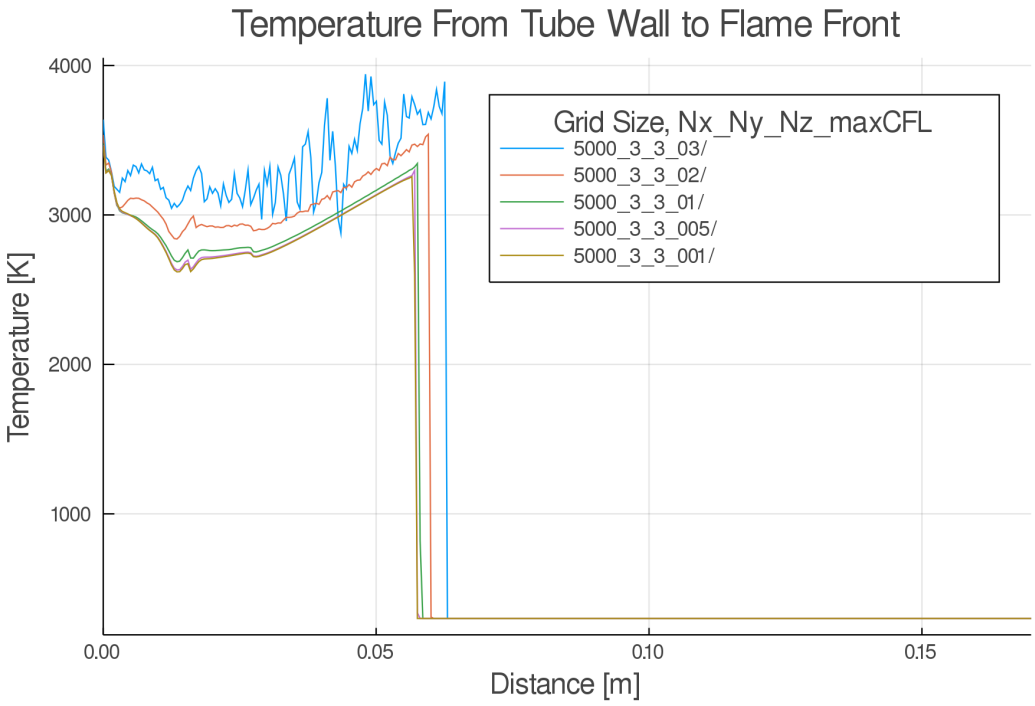
Higher CFL numbers of 0.4 to 0.8 were also attempted. These were unsuccessful as the solution was extremely unstable or never converged at these CFLs. Typically, high CFL numbers cause more ringing in the solution as seen in Figure 3.12 for the CFL of 0.3. However, it is seen in this research as well as supporting reacting flow work [49, 10, 6] that lower CFL numbers are required for solution convergence.

Table 3.1: Max CFL errors compared to CFL = 0.01

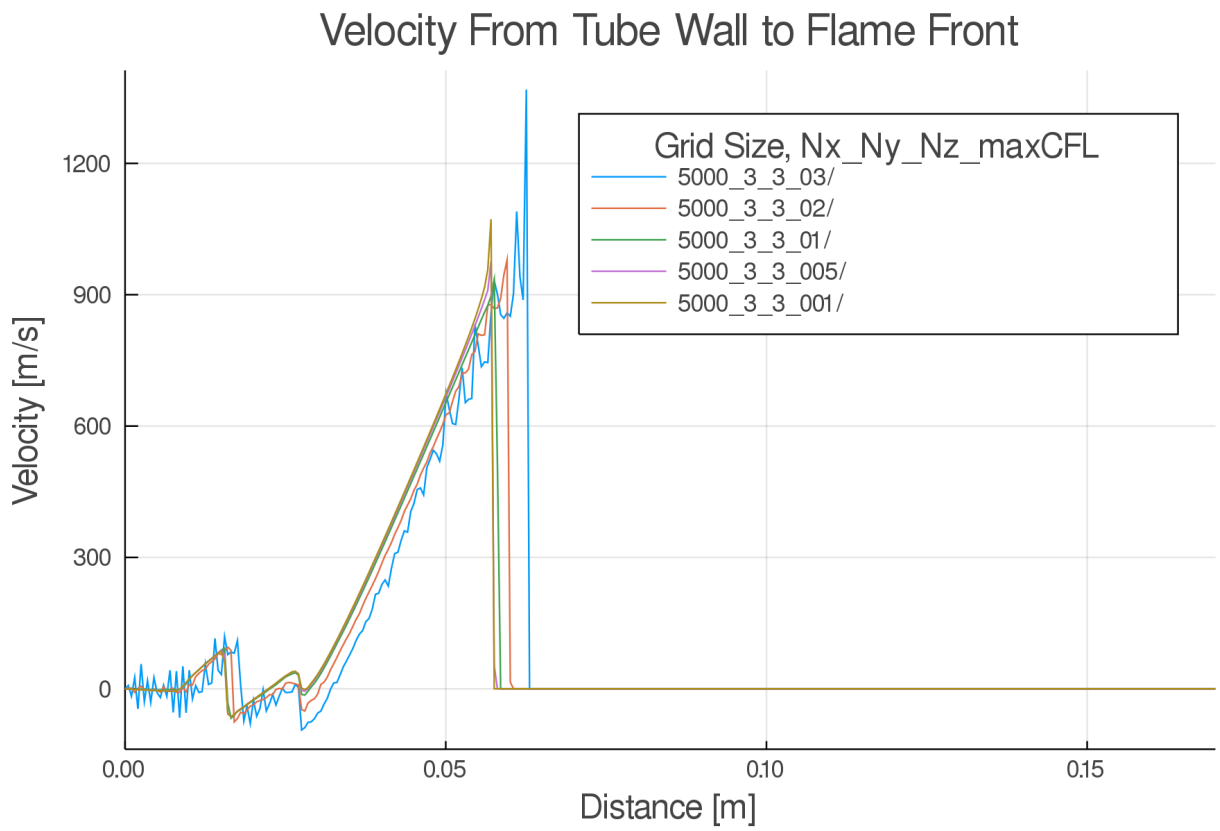
Max CFL	Pressure Error (%)	Temperature Error (%)	Velocity Error (%)
0.3	57.18	19.54	27.62
0.2	5.89	8.72	8.66
0.1	1.28	2.71	12.92
0.05	2.34	1.29	8.96
0.01	0	0	0



(a) Pressure distribution



(b) Temperature distribution



(c) Fluid velocity distribution

Figure 3.12: Max CFL sweep test

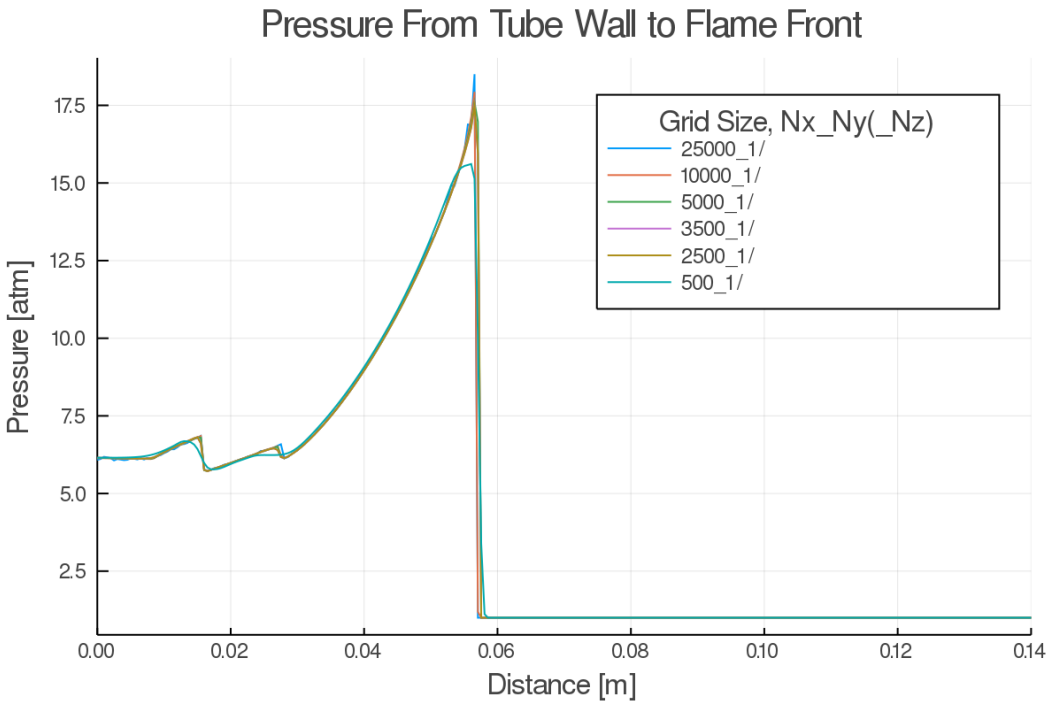
3.5 Static Mesh Variation

In order to determine the performance gains, accuracy, and other comparative parameters between mesh resolutions, baseline meshes must be created. These meshes must not change adaptively in the simulation in order to provide clear comparisons. Additionally, the static meshes themselves should be compared with each other to determine when mesh convergence occurs. Once a mesh resolution is determined, the AMR parameters can be tuned in an attempt to match the static mesh resolution in the small region surrounding the detonation wave.

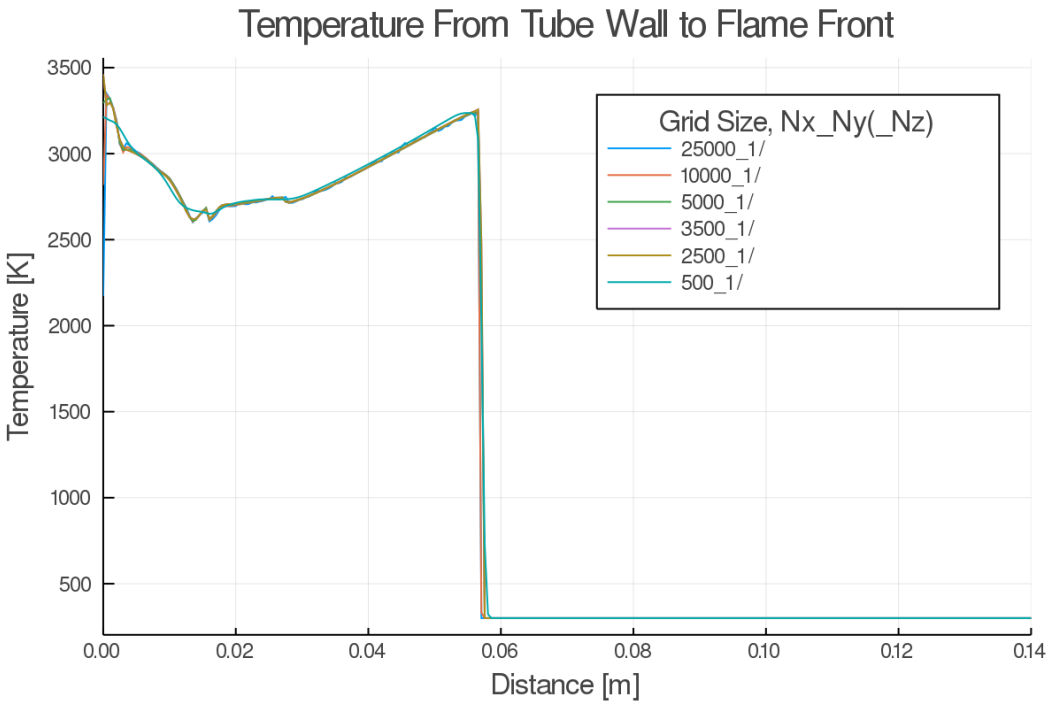
3.5.1 One-dimensional

The first set of static mesh comparisons performed is between one-dimensional detonation tube simulations, seen in Figure 3.13. These meshes only have resolution along the direction of the tube, which allows for capturing the detonation wave as simply as possible. However, one-dimensional detonation tube simulations are not optimal for generic detonation modeling as the cellular detonation structure seen in experimental detonation modeling cannot be captured with a single dimension. The cellular detonation cells can be used to infer velocity and compare numerical simulation with experiments, so one-dimensional detonations must utilize other methods to compare with experiments or just be utilized for tuning numerical methods themselves.

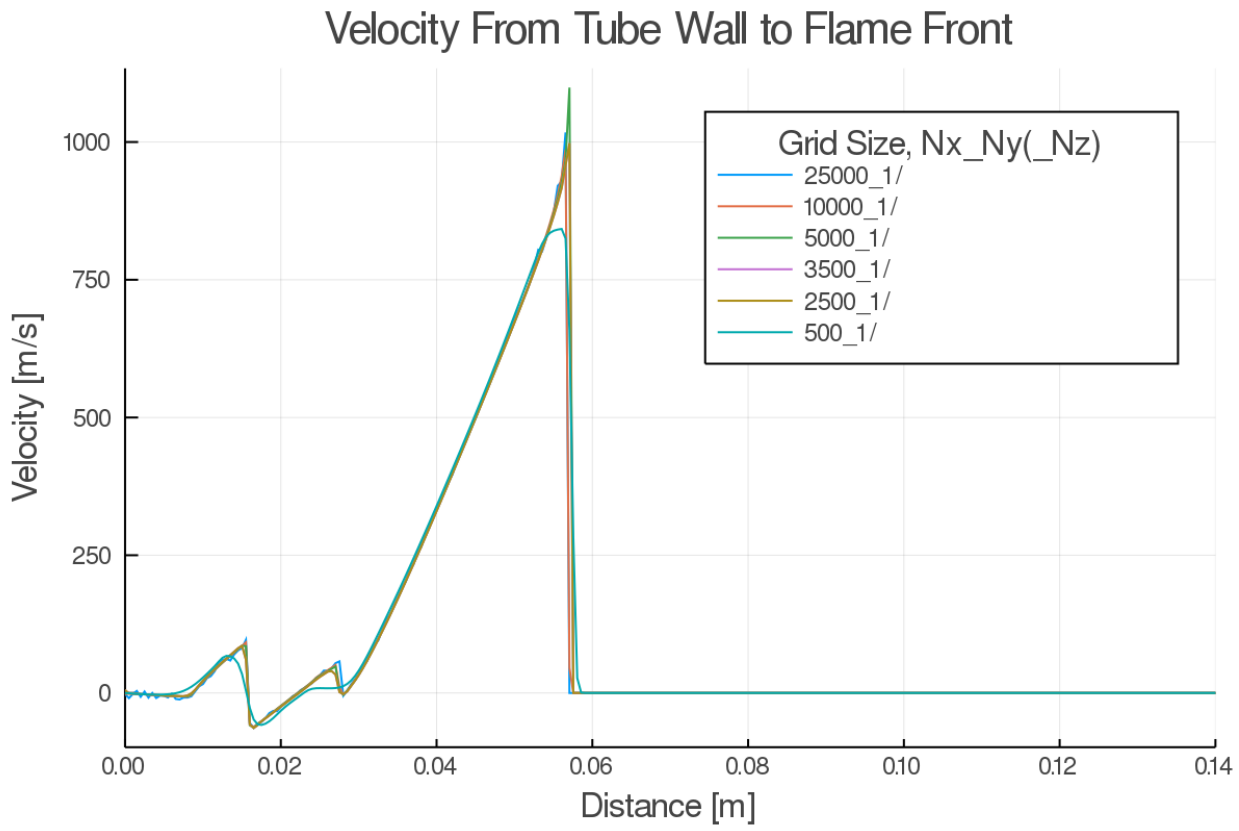
Various static one-dimensional meshes were compared and can be seen in Figure 3.13. These one-dimensional static meshes are actually twice as long, as initially the detonation simulations were performed using a domain that was twice as long, at a half meter in length. This long mesh was unnecessarily large and computationally expensive. The mesh partition values written in the figure are divided in half to match the meshes and convention used in subsequent analysis runs, with a quarter meter long domain. We see that the 500-1-1 mesh has not converged yet to a solution, and that the 2500-1-1 mesh has converged.



(a) Pressure plot



(b) Temperature plot



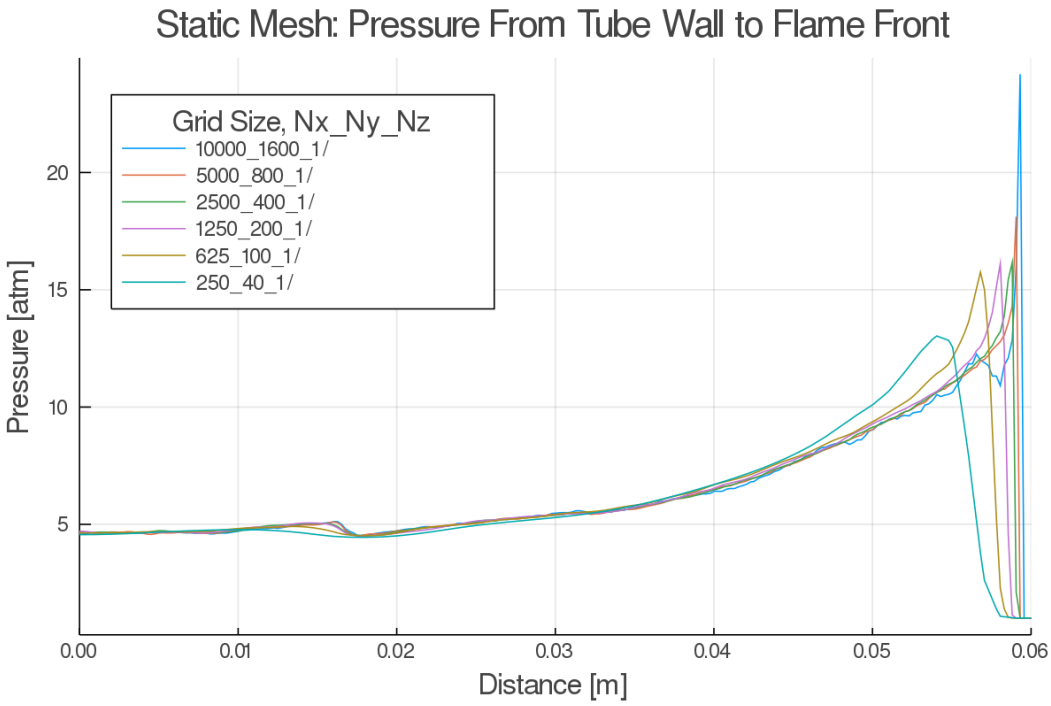
(c) Fluid Velocity plot

Figure 3.13: One-dimensional static meshes used in detonation tube simulations

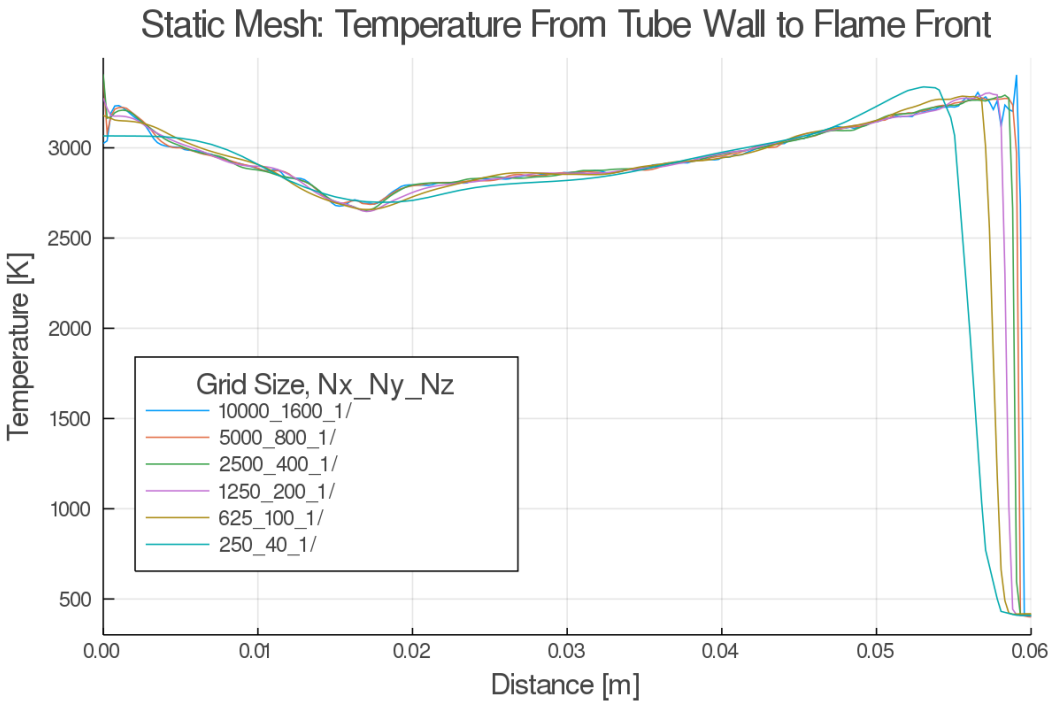
After this point, the difference between the meshes lies in their ability to capture finer flow effects such as detonation cells (not possible in one dimension), the transition from a singular detonation front to the combination of multiple smaller detonation fronts (also not possible in one dimension), and the increase in magnitude of the von Neumann pressure wave spike. The von Neumann spike comes from ZND detonation model theory, where Zel'dovich[50], von Neumann[51], and Doring[52] independently published research on a detonation model theory that utilizes an infinitesimally thin shock wave. Knowing that the spike is what changes within the converged mesh simulations allows for intelligent choices upon what sorts of AMR and static resolutions to choose both in research and engineering design. If an individual is designing a part that can withstand very fast high pressure fluctuations, then resolving the mesh up until the von Neumann spike appears will be sufficient for their design purposes. Otherwise, further mesh resolution is required in order to resolve the extremely thin pressure spike shock region. In regards to AMR, this translates to either further refinement around the spike region or just focusing refinement on the detonation wave and following region as a whole in order to resolve turbulence details or fluctuations in the field from randomization and reflected shocks.

3.5.2 Two-dimensional

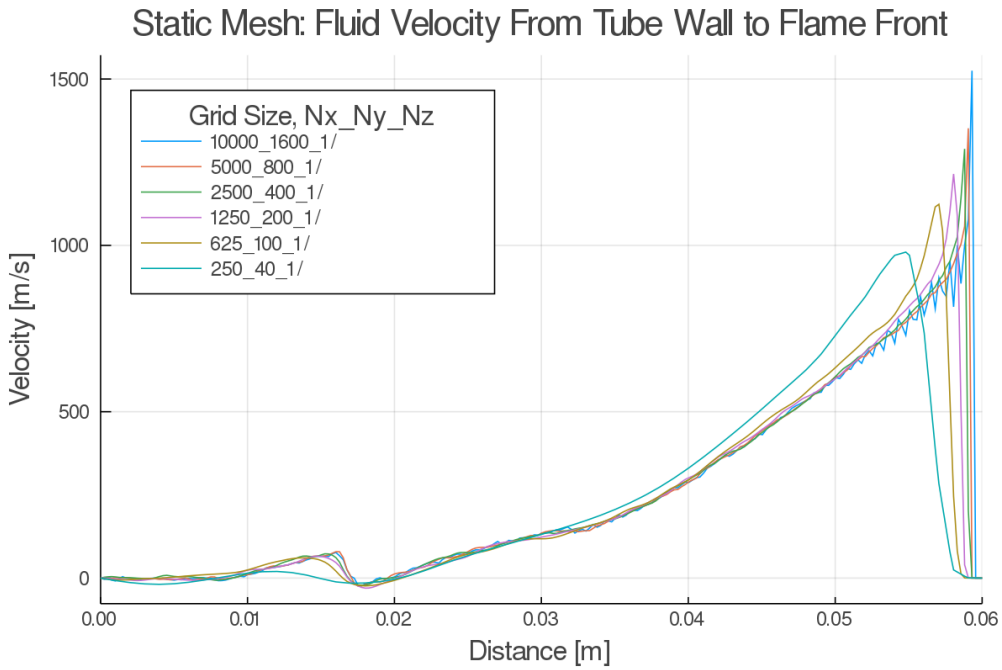
In addition to the static and dynamic one-dimensional detonation tube simulations performed, two-dimensional simulations were also run. These can be seen in Figure 3.14. For two-dimensional simulations, the mesh was kept square. A smaller mesh resolution than the one-dimensional simulations were used due to computational expense. The 10000-1600-1 mesh contains 16 million cells at a 25 micron physical resolution, and was computed across twenty 3.6 GHz AMD Threadripper cores at a computational time of over 150 core-hours. Note that this high mesh resolution does not have the smoothness seen in more unrefined simulation results, seen clearly in Figures 3.14a and 3.14c. This is due to the mesh capturing the fine shock and reaction structures described by ZND theory mentioned in Section 3.5.1.



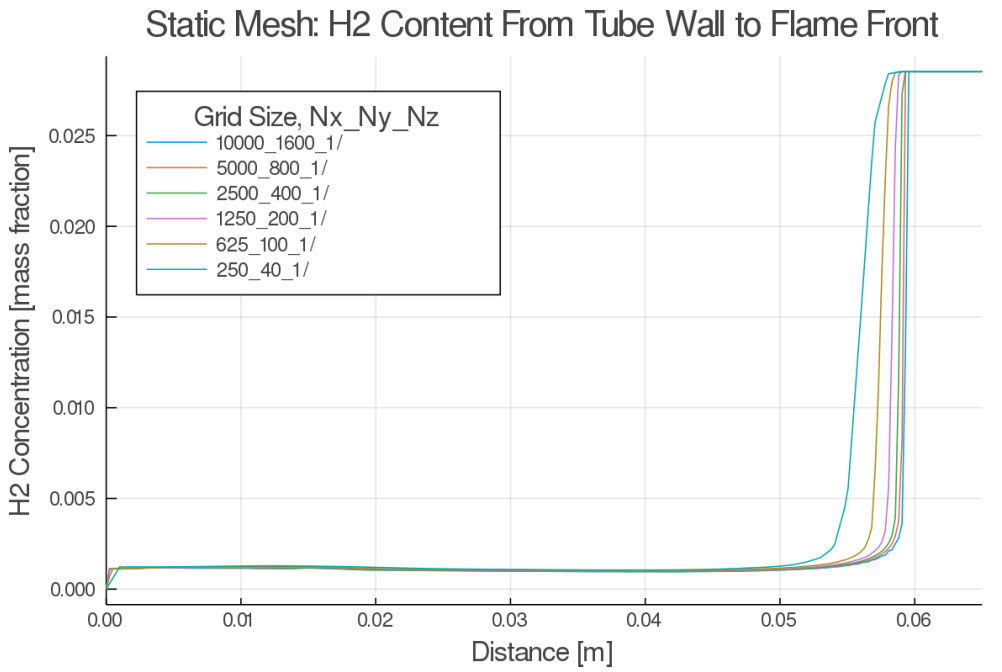
(a) Pressure plot



(b) Temperature plot



(c) Fluid velocity plot



(d) H_2 mass fraction plot

Figure 3.14: Two-dimensional static meshes used in detonation tube simulations

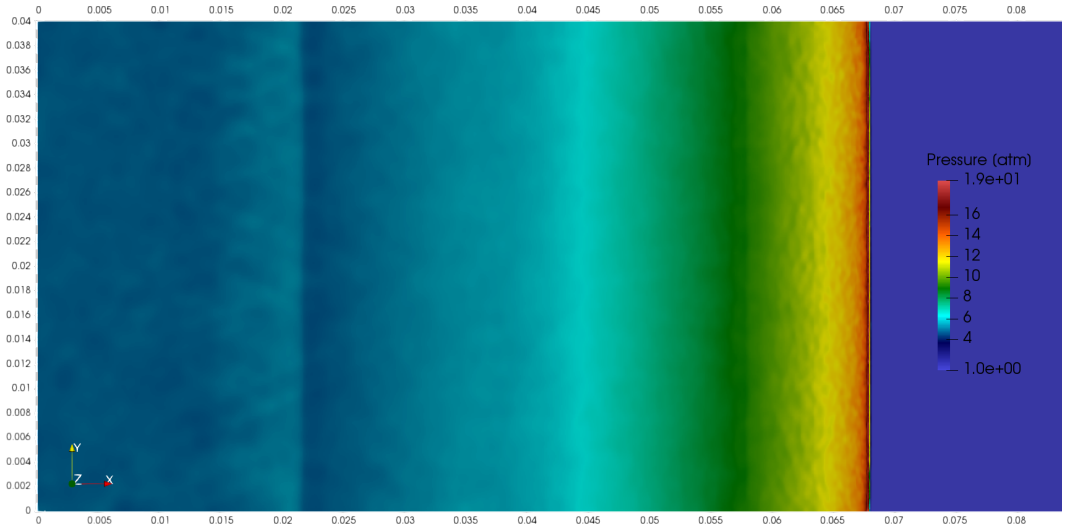
The fluctuations in the temperature field begin to resolve at this resolution.

Example surface plots for the static 2500-400-1 mesh are plotted in Figure 3.15. The plots seen in this figure represent the flow field results of a detonation wave traveling from the left wall to the right, at $t = 3.45 \times 10^{-5}$ s.

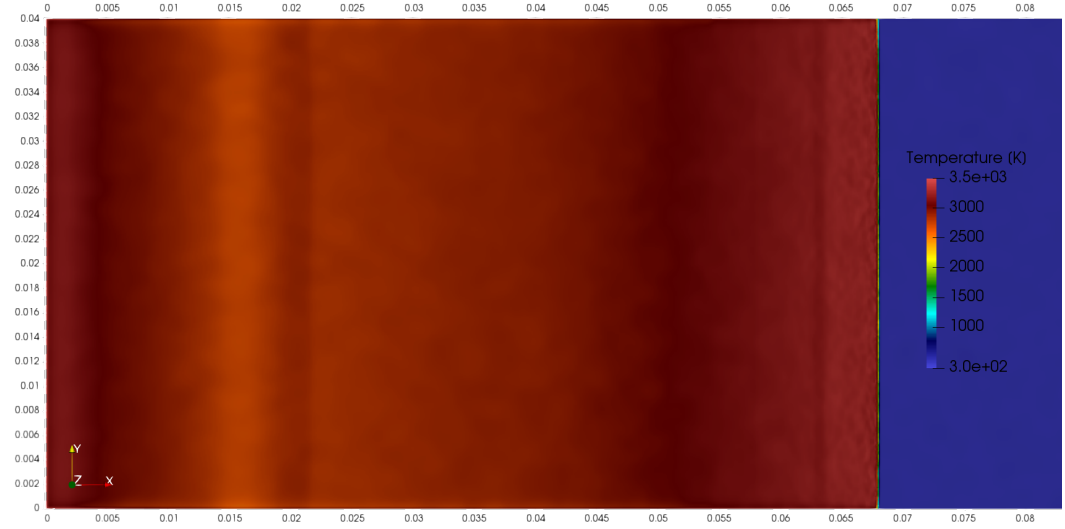
3.6 Adaptive Mesh Refinement and Static Comparisons

Since static mesh resolutions have been compared and are available for baseline comparisons, AMR was then added. As mentioned in Section 3.5, the adaptive mesh refinement parameters can be tuned to try and match the fine static mesh resolutions in important areas. Detonation waves are very thin with extremely high pressures, temperatures, and velocities within this small region. This makes them an ideal candidate for adaptive meshing. To validate the AMR routines, comparison with the baseline static meshes is required to determine how the flow field changes. The AMR settings were tweaked and changed over a range of values and compared to static meshes in order to determine if the AMR can practically replace static meshes for detonation modeling in OpenFOAM as well as attempt to characterize how each change in AMR parameter affects the solution. For the following comparison cases, it was all performed in two dimensions. This allows for the AMR to start delivering noticeable performance gains as cell counts can often quickly increase into the millions.

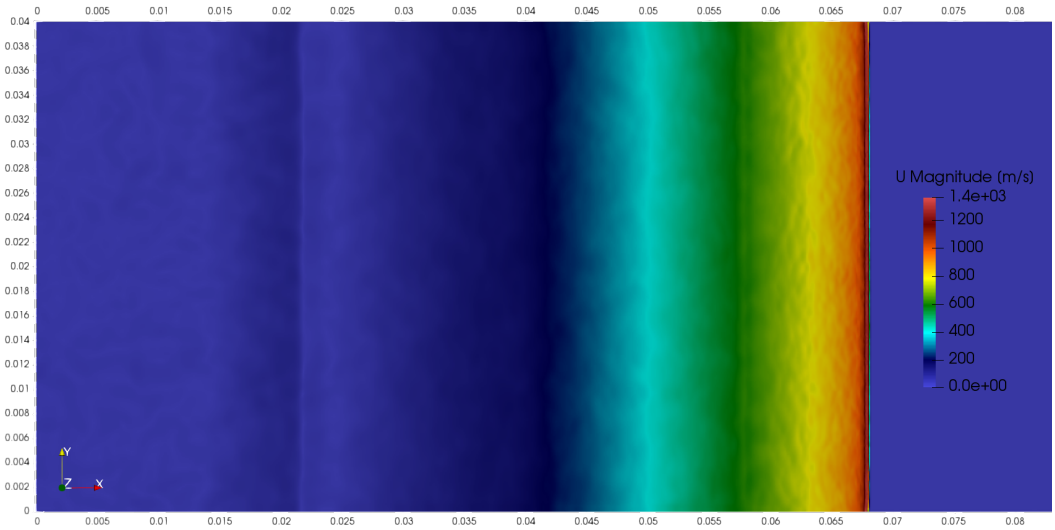
The primary variable used to track when the AMR should be applied was the normalized gradient of pressure, $||\nabla(p)||$. Detonations and shock waves will have very large pressure gradients, so this is a logical selection. By normalizing this quantity, the values selected for the bounds are always consistent across any similar simulation and the setup is easier. Other variables can also be selected for refinement tracking variables, such as pressure, temperature, and velocity. The reasons that these were not selected as primary tracked variables was due to the rest of the flow field. Pressure itself could change significantly between different simulations, so normalized pressure is probably a better choice. Even if normalized pressure



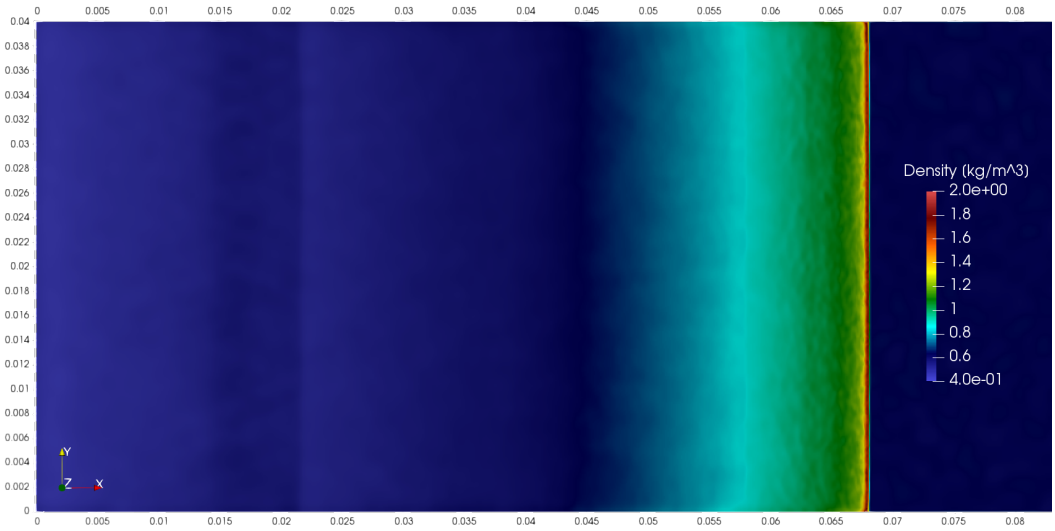
(a) Pressure plot



(b) Temperature plot



(c) Fluid velocity magnitude plot



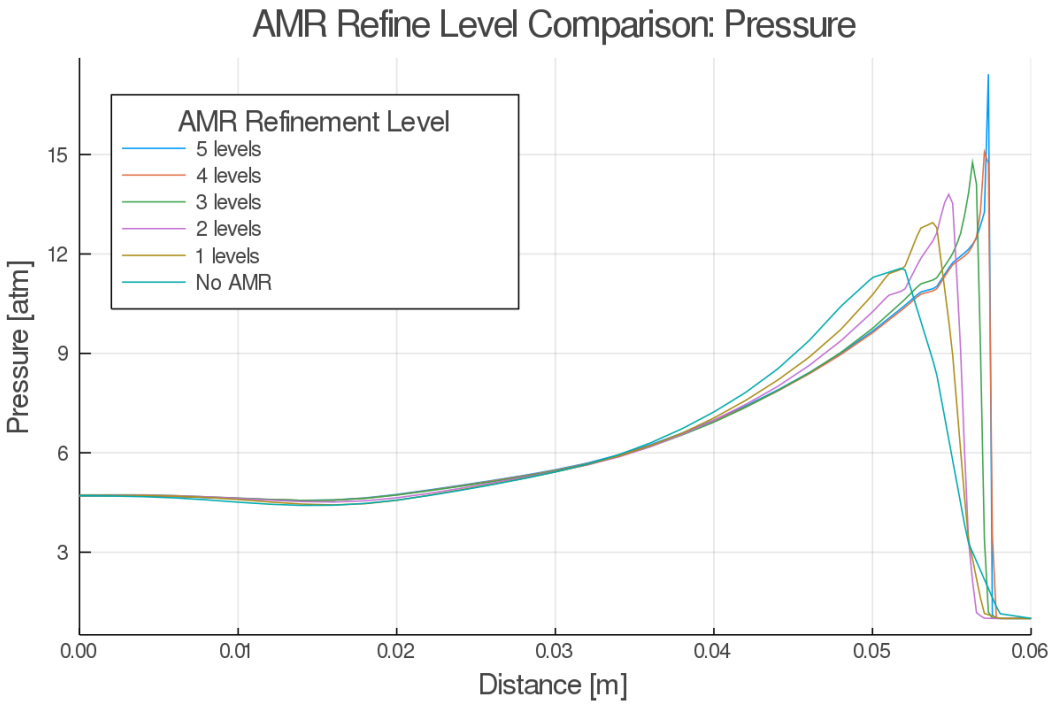
(d) Density plot

Figure 3.15: Two-dimensional detonation wave surface plots for 2500-400-1 mesh at $t = 3.45 \times 10^{-5}$ s

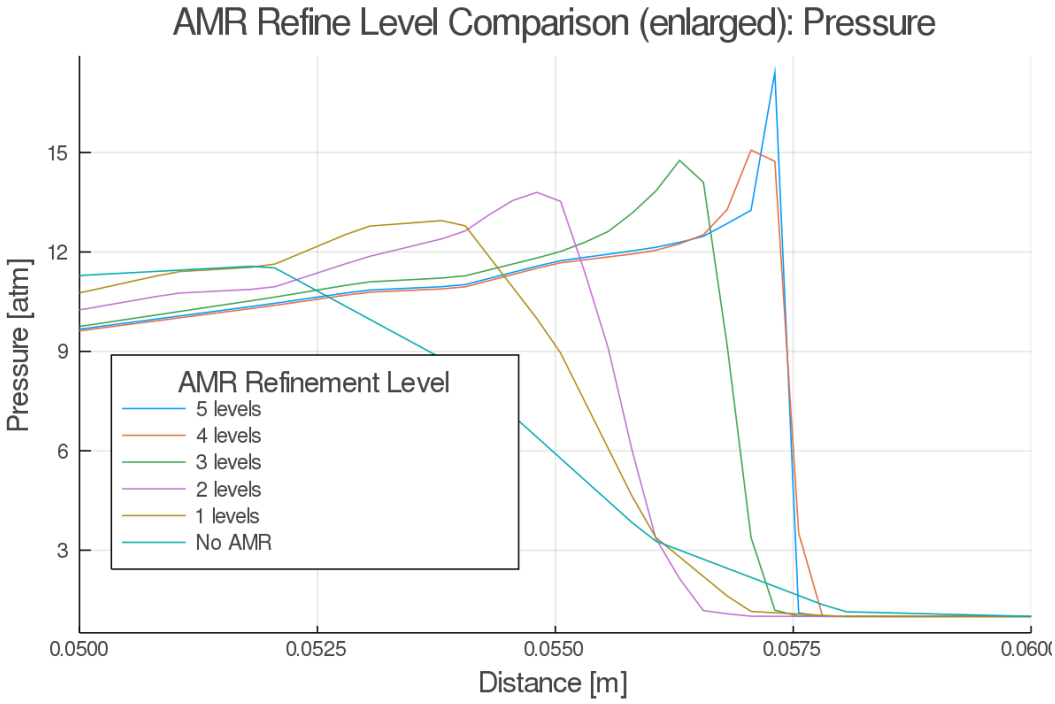
is chosen, a portion of the field can remain at high pressure (i.e. in the burning region after the detonation). The pressure gradient waveform is much sharper and thinner than pressure, so this provides a better small region for the AMR to focus on. Temperature is fairly consistent across the products and even increases back towards the detonation initiation region, so tracking temperature itself around the wave only would be very difficult. Velocity and normalized velocity may be appropriate, but only for initiated detonations as the ignition region has no flow field velocity and therefore the important initial refinement of the flow field as the detonation wave is forming will not occur due to the lack of velocity before the reactants ignite and detonate. Exploration in combining parameters together to track or different tracking parameters altogether is a topic for further research.

3.6.1 AMR Refine Level Comparisons

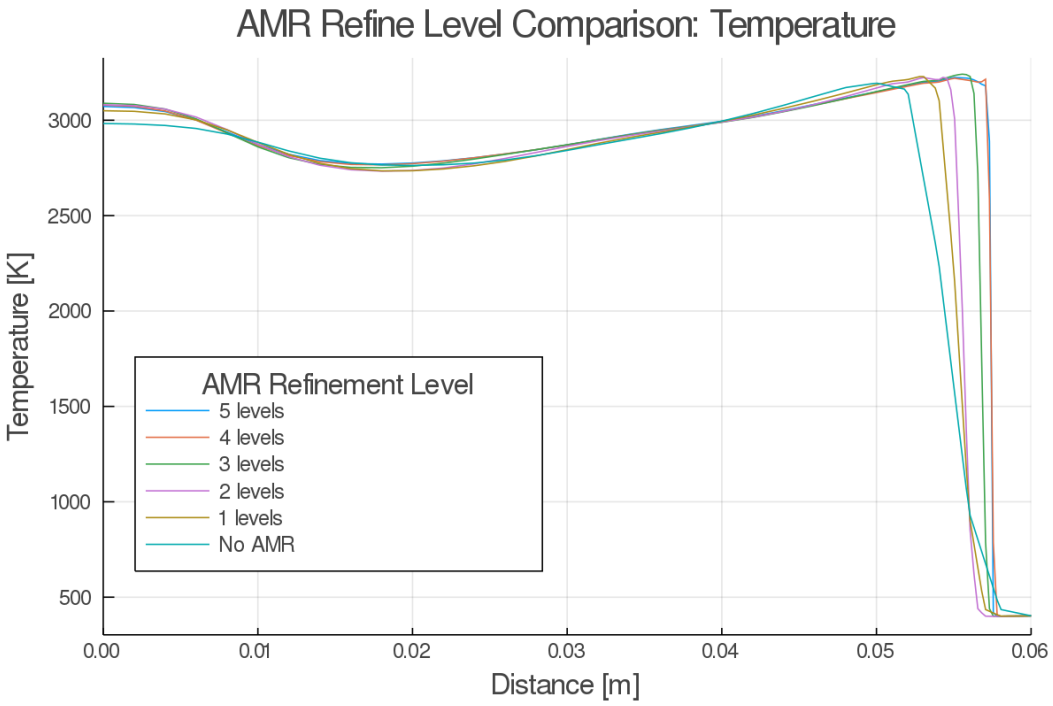
One of the more important parameters to test is the number of times the adaptive meshing routines refine each cell, or the refinement level. A 125-20-1 base mesh was used to test each refinement level using the normalized pressure gradient as the tracked variable with three buffer layers. The results, taken at 3×10^{-5} s, are plotted in Figure 3.16. As expected, as the refinement level increases, so does the accuracy of the detonation wave. For this particular base mesh, it can be seen that the solution starts to converge around 4 levels of refinement and after the von Neumann pressure spike begins to resolve. For different base resolutions and simulation setups, this may change. It can also be seen that each increase in refinement level brings the solution roughly as close to a potential "perfect" static mesh resolution by half the previous x-distance, with the increase in pressure solution being exponential due to the resolving of the von Neumann spike. Further refinement levels are similarly as expensive, being exponential in cell growth, seen in Table 3.2. The exponential cell growth was also reflected in the time the simulations took to complete. The level 5 refinement has nearly the same peak pressure value as the 5000-800-1 simulation, with a slight spatial offset and 3,771,393 fewer cells. Further tweaking with tracked variable bounds



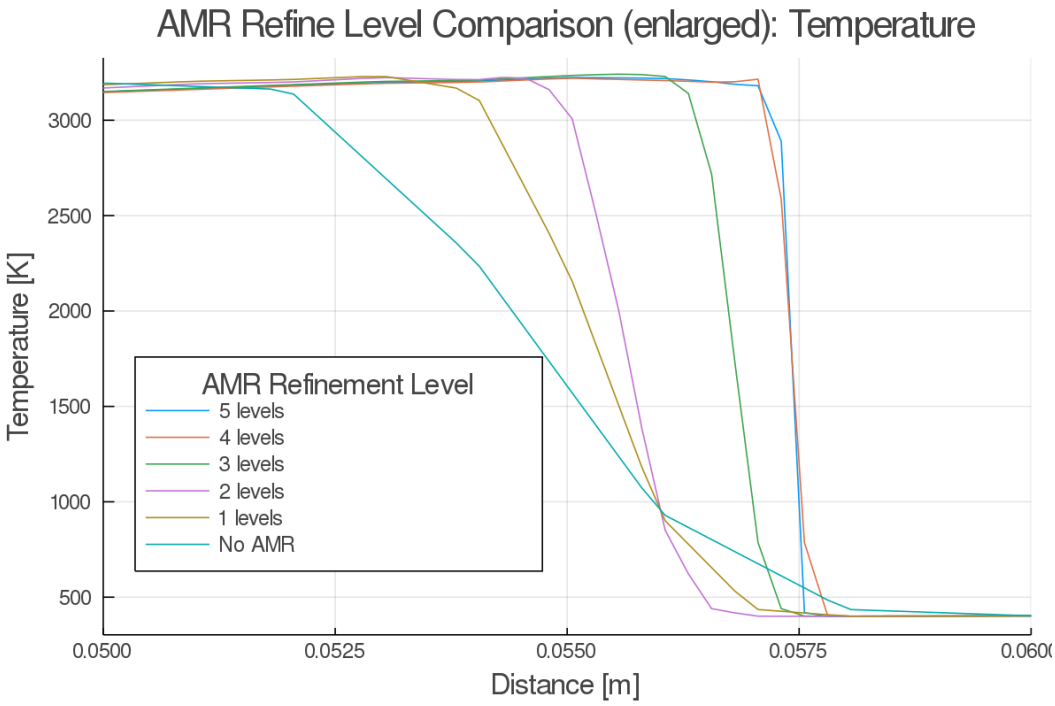
(a) Pressure plot



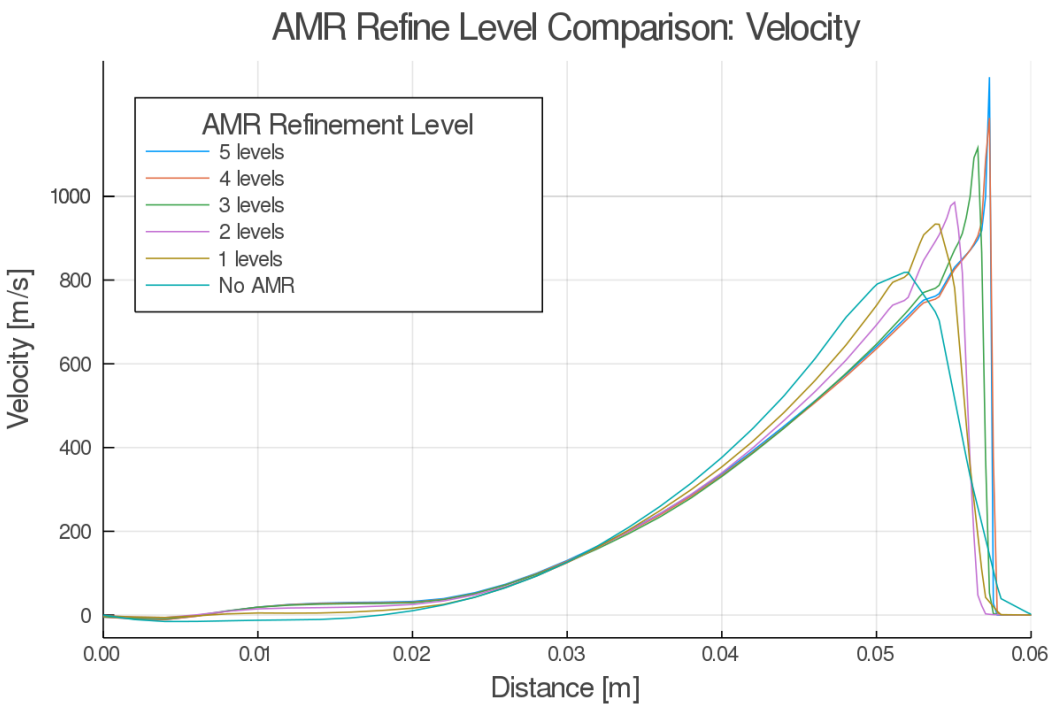
(b) Pressure plot, enlarged



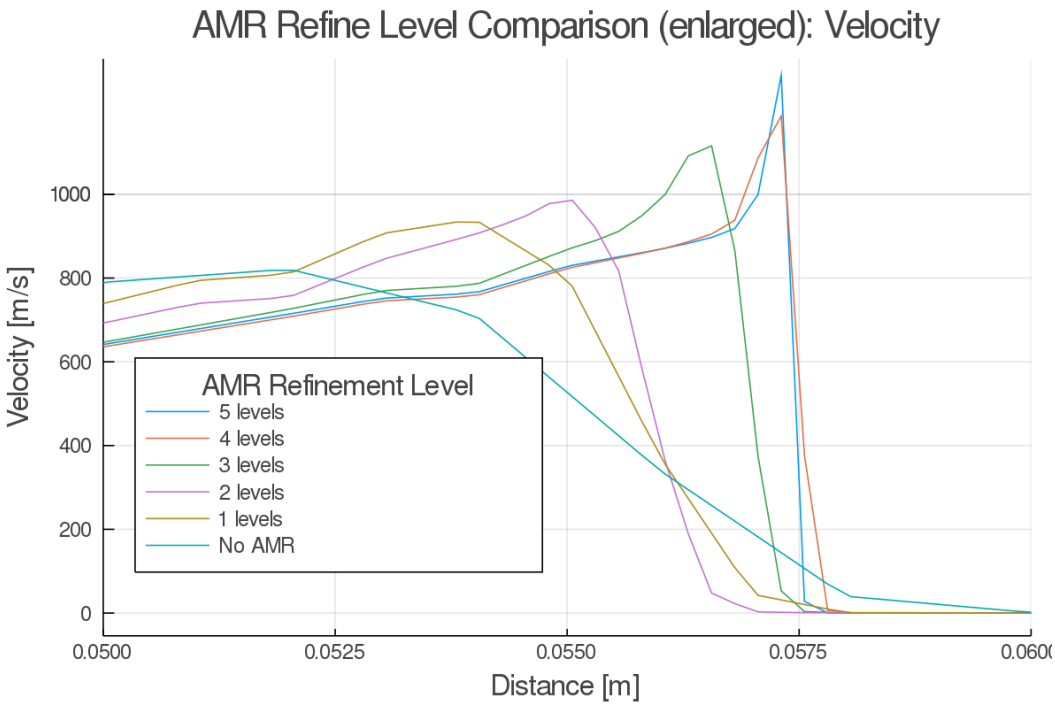
(c) Temperature plot



(d) Temperature plot, enlarged



(e) Fluid velocity plot



(f) Fluid velocity plot, enlarged

Figure 3.16: Two-dimensional detonation wave AMR results for a 125-20-1 base mesh with 3 buffer layers, tracking $||\nabla(p)||$ between 0.2 and 1 at 3e-5 s, sweeping refinement levels

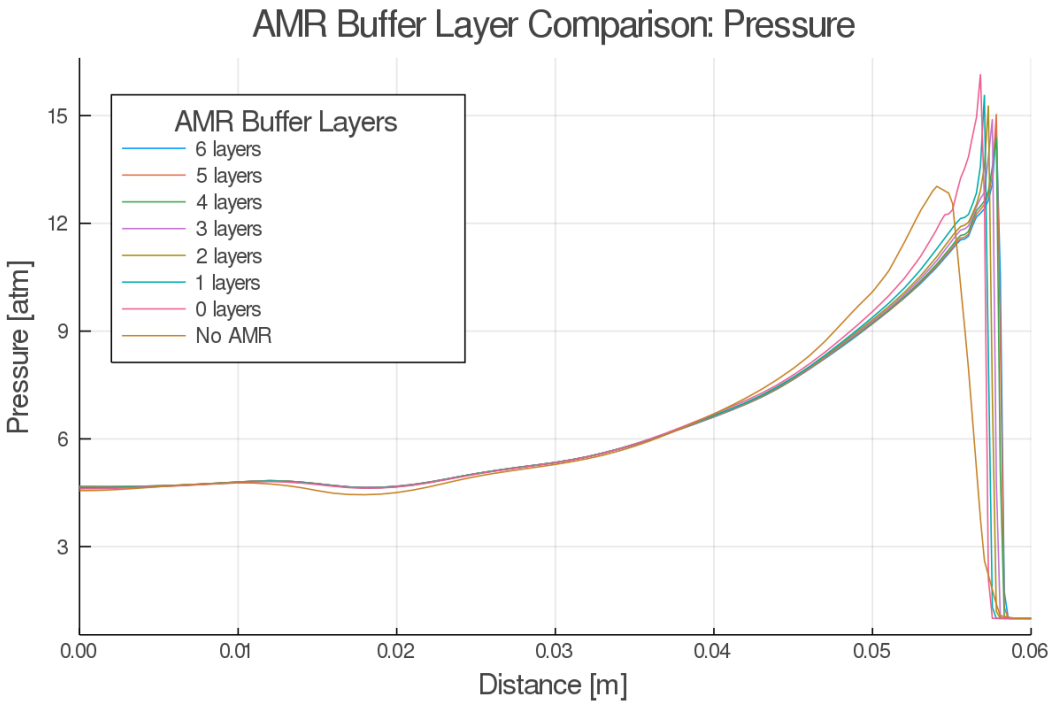
as well as buffer layers may bring the detonation wave closer to that static mesh simulation. After 4 and more refinement levels (e.g. 4000-640-1 mesh) the von Neumann spike begins to resolve, which agrees with the static 2D mesh results.

Table 3.2: Cells for each refinement level for the 125-20-1 simulation at $t = 3 \times 10^{-5}s$ in Figure 3.16

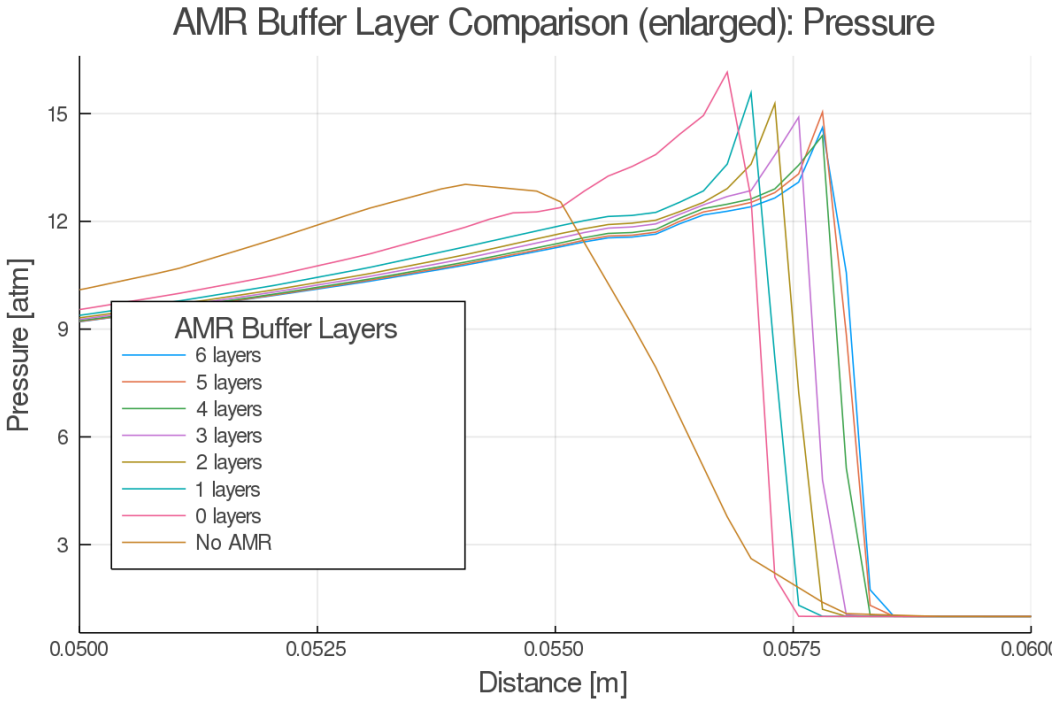
Refinement Level	Cells
5	228,607
4	54,678
3	13,028
2	5,300
1	2,920
None	2500

3.6.2 AMR Buffer Layer Comparisons

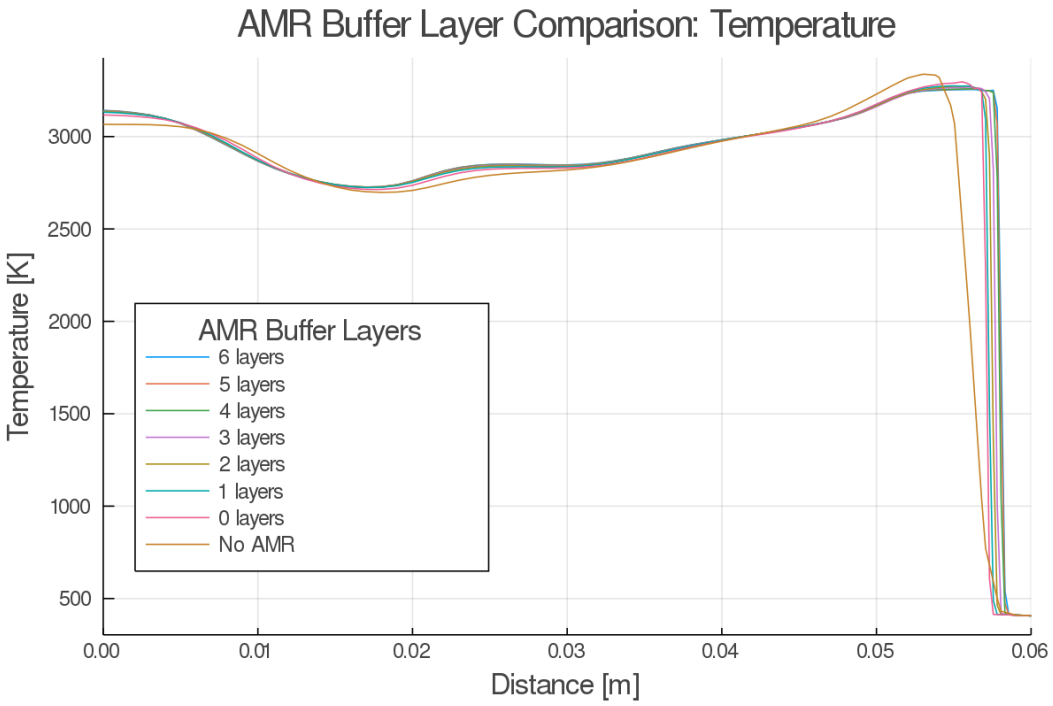
Within the AMR settings, different numbers of buffer layers between the refinement layer and the rest of the flow can be set. A 250-40-1 base mesh was set, with 3 refinement levels tracking the normalized pressure gradient $||\nabla(p)||$ from 0.2 to 1. Different number of buffer layers were tested and plotted at $t = 3 \times 10^{-5}$ s in Figure 3.17. From the figure, it can be seen that the pressure solution generally decreases and the wave propagates further right, around 0.2 mm per layer, as more buffer layers are added. When 4 buffer layers are reached, the buffer layers have "converged" with their overall affect on the solution. Due to this, we can conclude that for PDEs and detonation tube simulations 4 buffer layers are sufficient and significant improvements in solution accuracy are not worth the increased cell count which could be used for additional refinement layers or an increased AMR refinement range. That said, the subjective timing of adding additional buffer layers was not as significant as additional refinement levels. The maximum temperature and velocity seen do not change beyond using AMR or not, but the location of the detonation wave does as with the rest of the solution variables. When a user is attempting to move the AMR-simulated detonation wave closer to static results while tuning AMR parameters, adding additional buffer layers



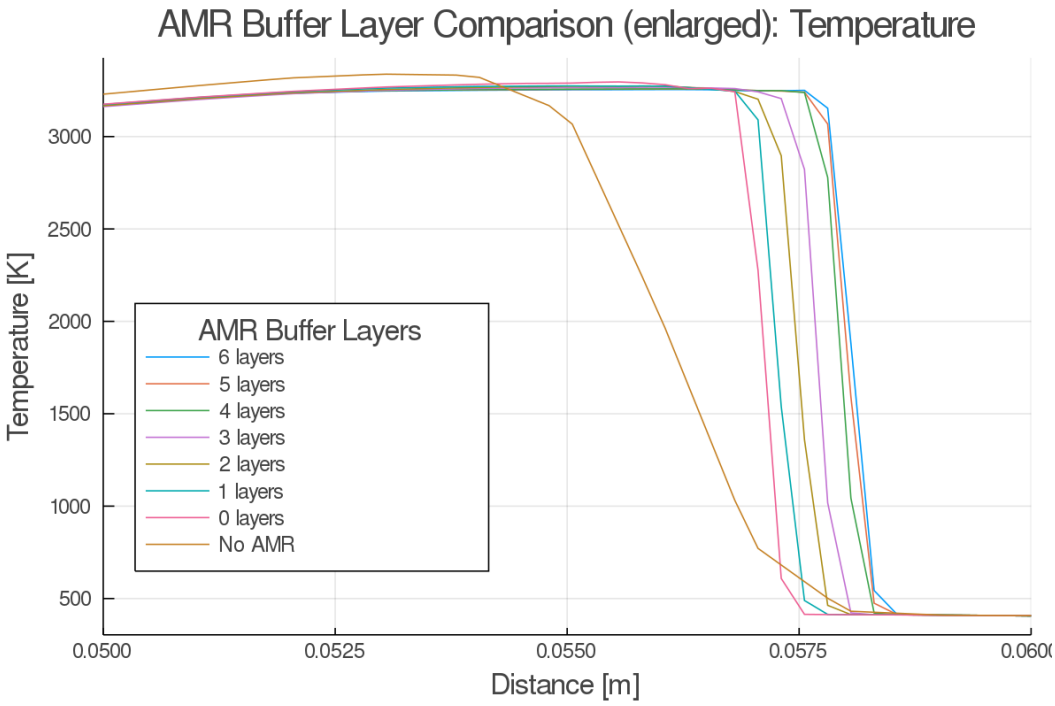
(a) Pressure plot



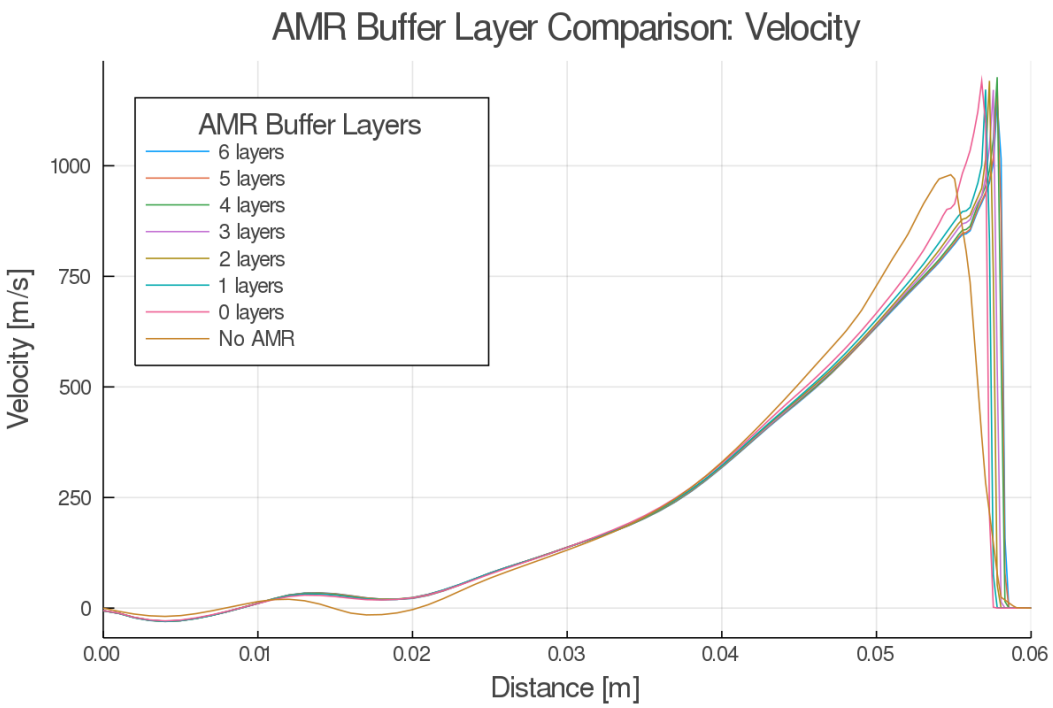
(b) Pressure plot, enlarged



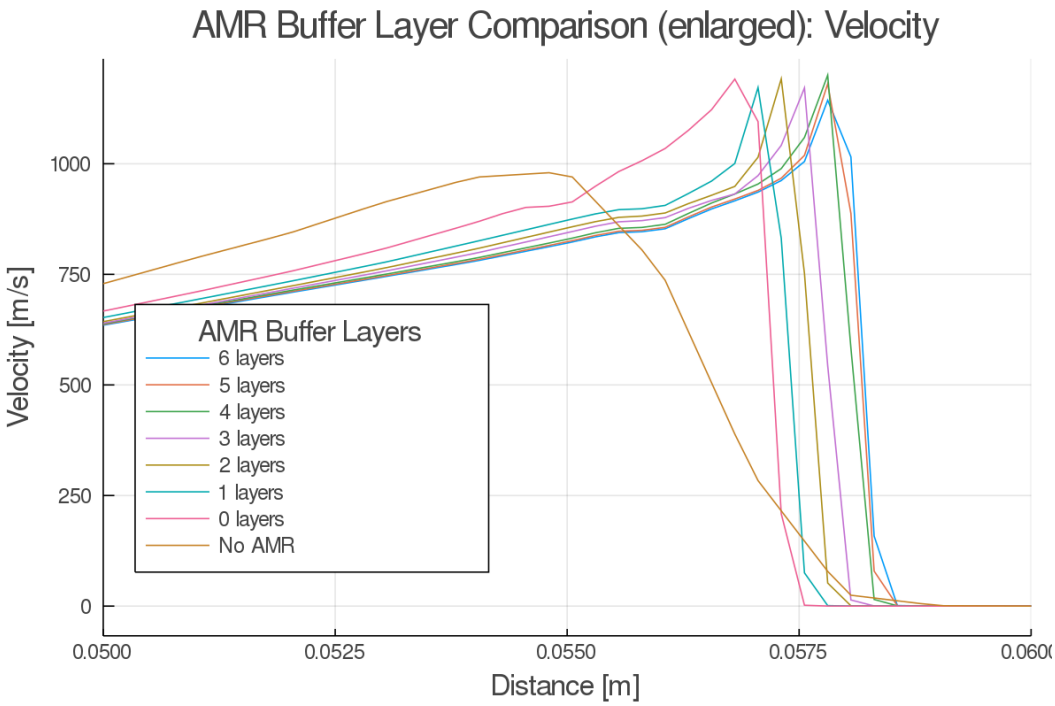
(c) Temperature plot



(d) Temperature plot, enlarged



(e) Fluid velocity plot



(f) Fluid velocity plot, enlarged

Figure 3.17: Two-dimensional detonation wave AMR results for a 250-40-1 base mesh with 3 refinement levels, tracking $||\nabla(p)||$ between 0.2 and 1 at 3e-5 s, sweeping buffer layers

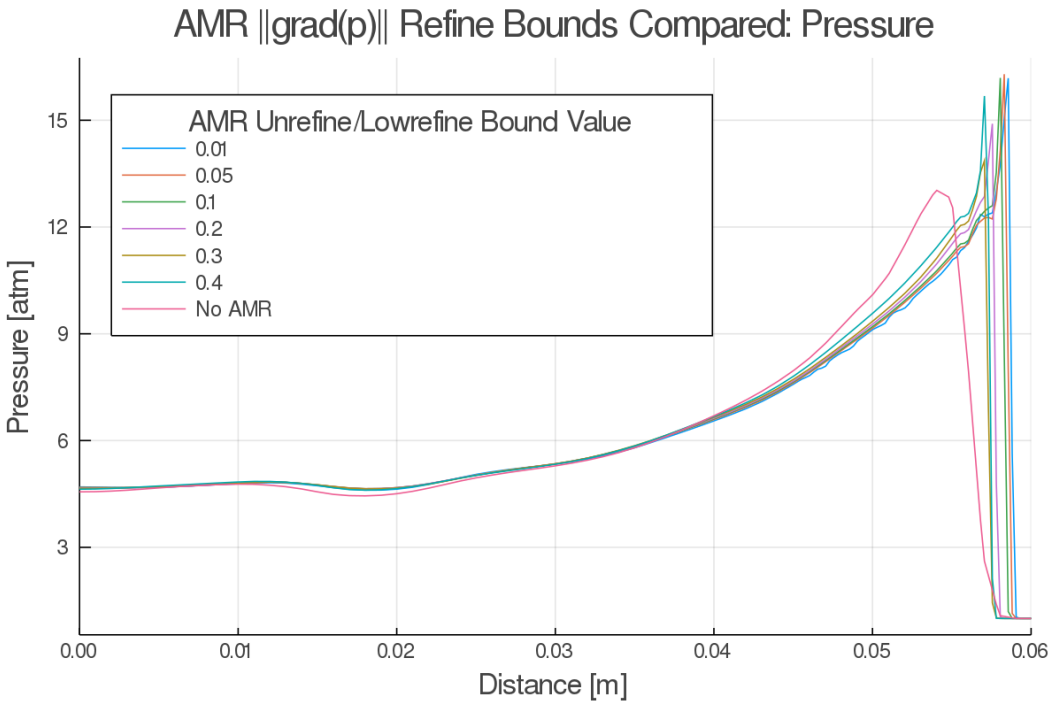
up to 5 layers will move the AMR-simulated detonation wave close to static simulation results at minimal additional cost. Table 3.3 displays the cell counts for each additional AMR buffer layer for the simulations tested in Figure 3.17. For each additional buffer layer, there is roughly a linear increase in cells, which tracks with the time the simulations took to complete.

Table 3.3: Cells for each refinement level for the 250-40-1 simulation at $t = 3 \times 10^{-5}s$ in Figure 3.17

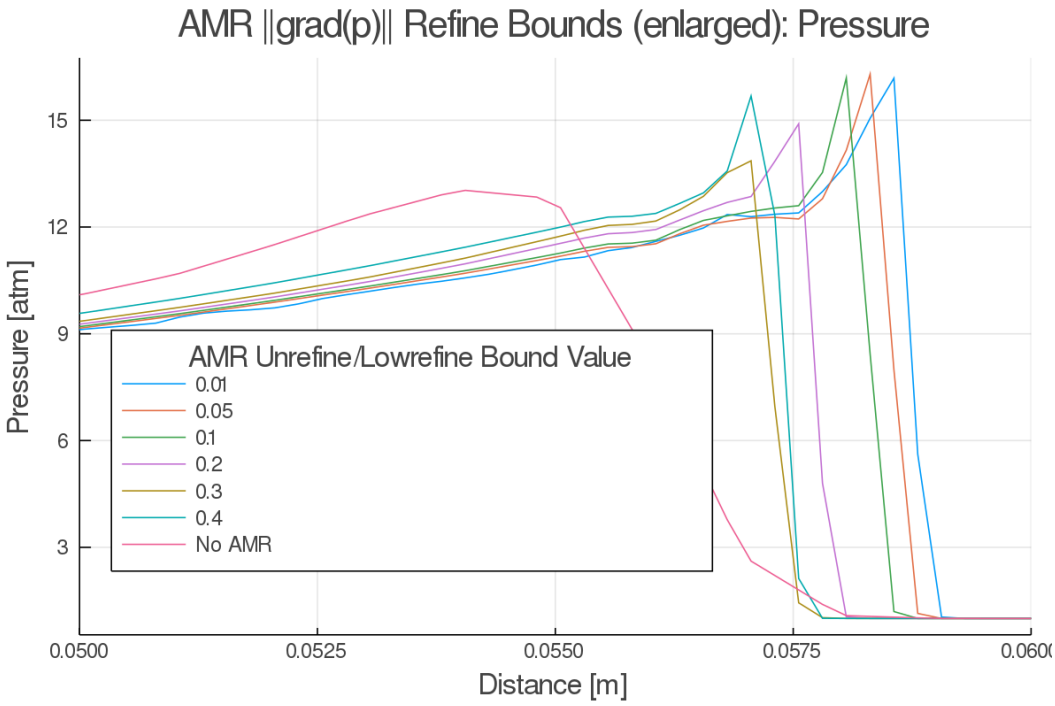
Buffer Layers	Cells
6	37,405
5	34,353
4	32,960
3	37,678
2	16,173
1	10,840
None	10,000

3.6.3 AMR Normalized Pressure Gradient Refinement Range Comparisons

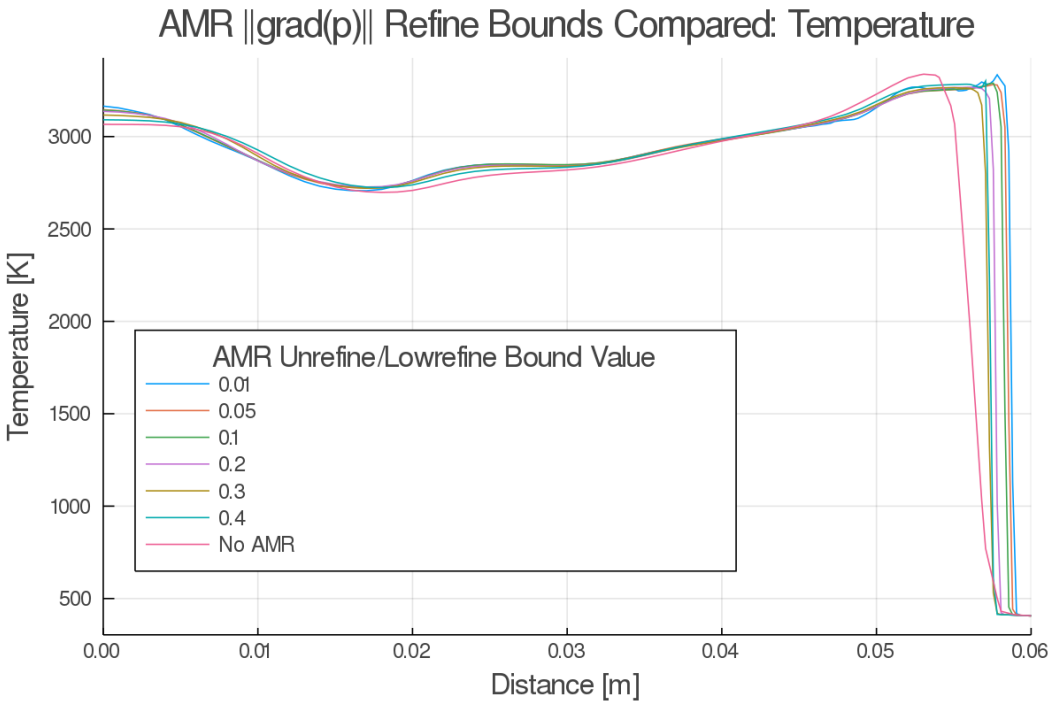
In addition to the levels of refinement and buffer layers, the actual range in which the normalized pressure gradient is refined over can be altered. The upper bound of the normalized pressure gradient tracking variable was kept at 1, with the lower refinement bound and the unrefine bound kept identical and swept across a range of values. As a reminder, if the normalized pressure gradient is between 1 and the lower bound, adaptive meshing will be active within the cell. If the value is lower than the unrefinement bound, unrefinement of the cell until the base resolution will occur. This was plotted at 3×10^{-5} s in Figure 3.18. The static mesh was unchanged from the previous sensitivity plots and the number of refinement levels as well as buffer layers were both 3. It is seen in the figures that by decreasing the refinement bound, the solution increases in accuracy. Pressure values increase the most initially when decreasing the lower bound, up until a lower bound of around 0.1. From here, only further wave propagation is what changes from iteration to iteration.



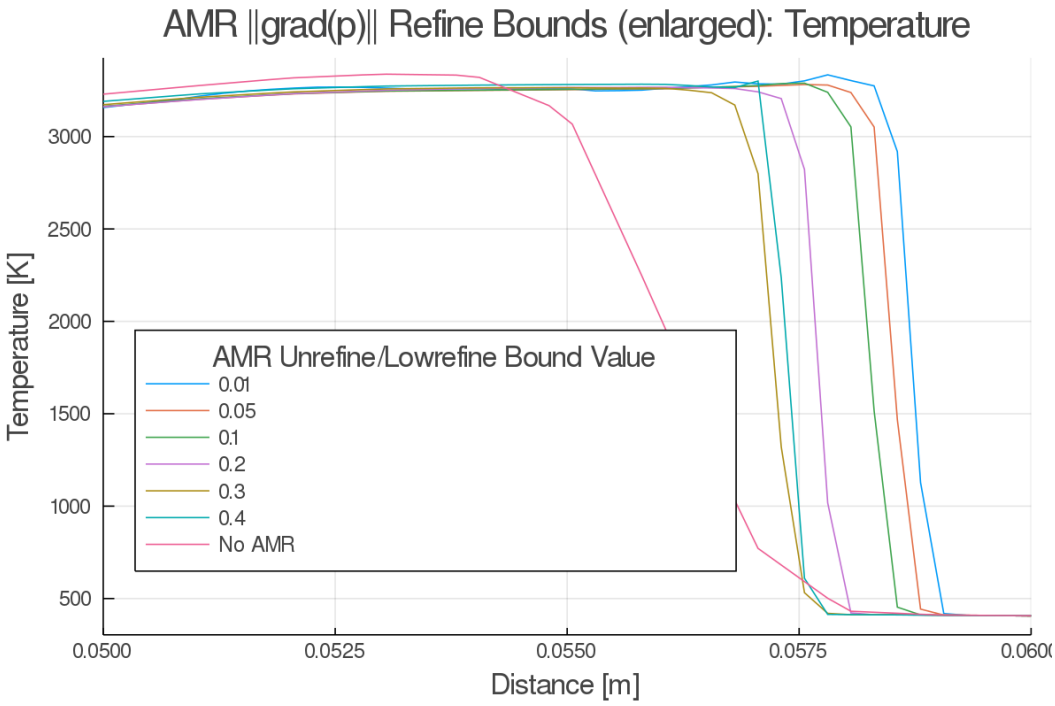
(a) Pressure plot



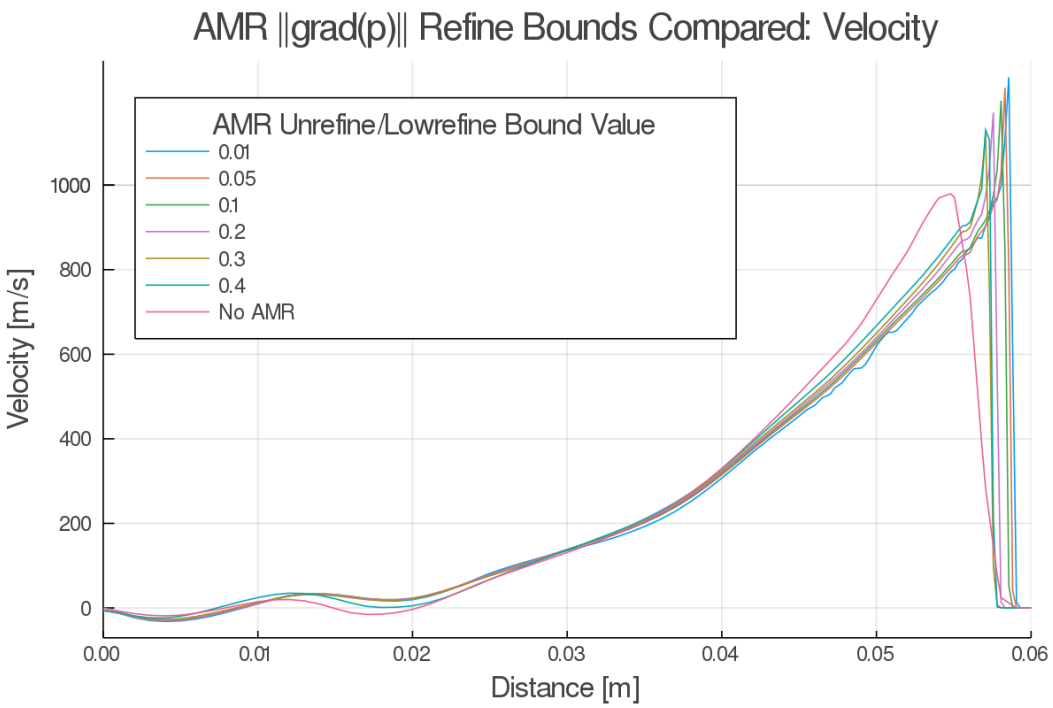
(b) Pressure plot, enlarged



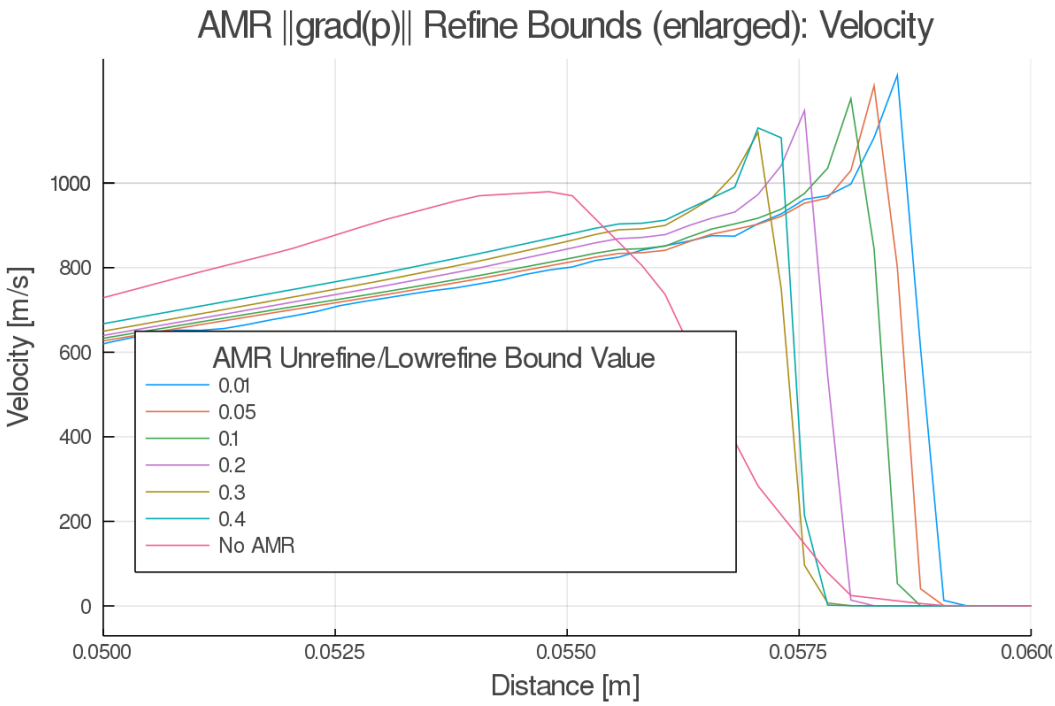
(c) Temperature plot



(d) Temperature plot, enlarged



(e) Fluid velocity plot



(f) Fluid velocity plot, enlarged

Figure 3.18: Two-dimensional detonation wave AMR results for a 250-40-1 base mesh with 3 refinement levels and 3 buffer layers, tracking $||\nabla(p)||$, sweeping refinement bounds

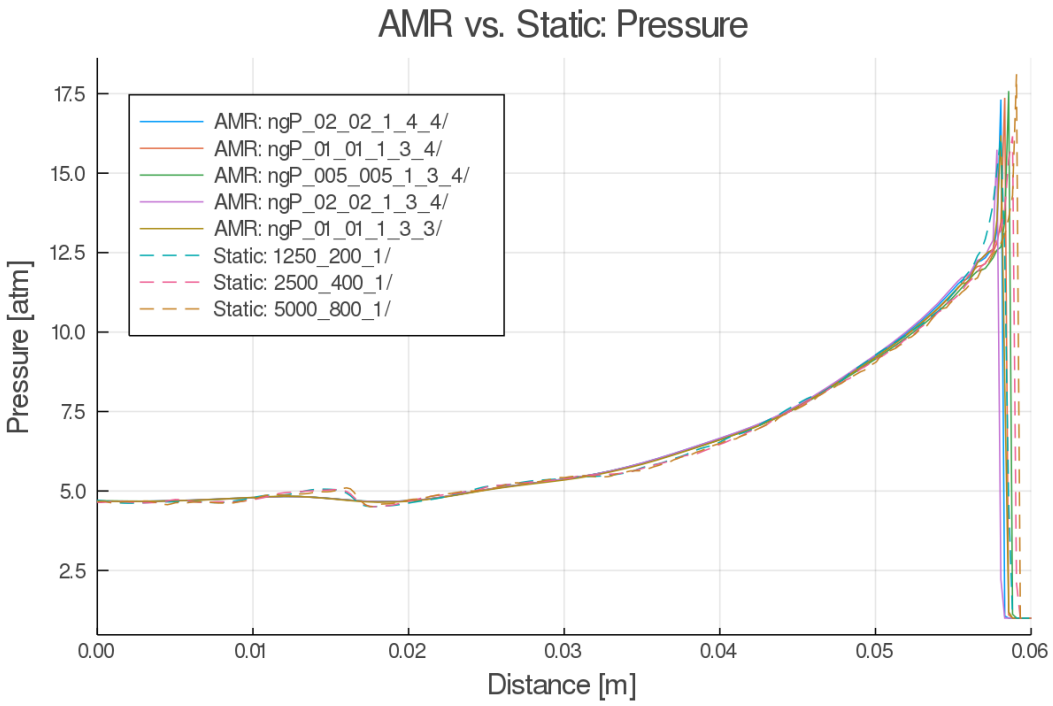
Due to this, 0.1 is the lower bound value that should be selected for PDE and detonation tube modeling if pressure is of interest while keeping the cell count as low as possible for cheaper computation. After this value, the detonation wave location is the parameter that will be affected by further decreases in lower bound of the normalized pressure gradient. If accurate wave location is also of interest (keep in mind that the lowest three refinement bounds, i.e. the most accurate, span around 0.6 mm), then further expansion of the AMR range can be done. Table 3.4 contains the cell counts for each lower refinement/unrefinement bound simulation performed in Figure 3.18. It is seen that for this base mesh resolution, a lower bound for the normalized pressure gradient of 0.05 is when the cell count begins to increase considerably. The increase is exponential, so normalized pressure gradient bounds below 0.05 will be more expensive than they are worth.

Table 3.4: Cells for each lower refinement/unrefinement bound for the 250-40-1 simulation at $t = 3 \times 10^{-5} s$ in Figure 3.18

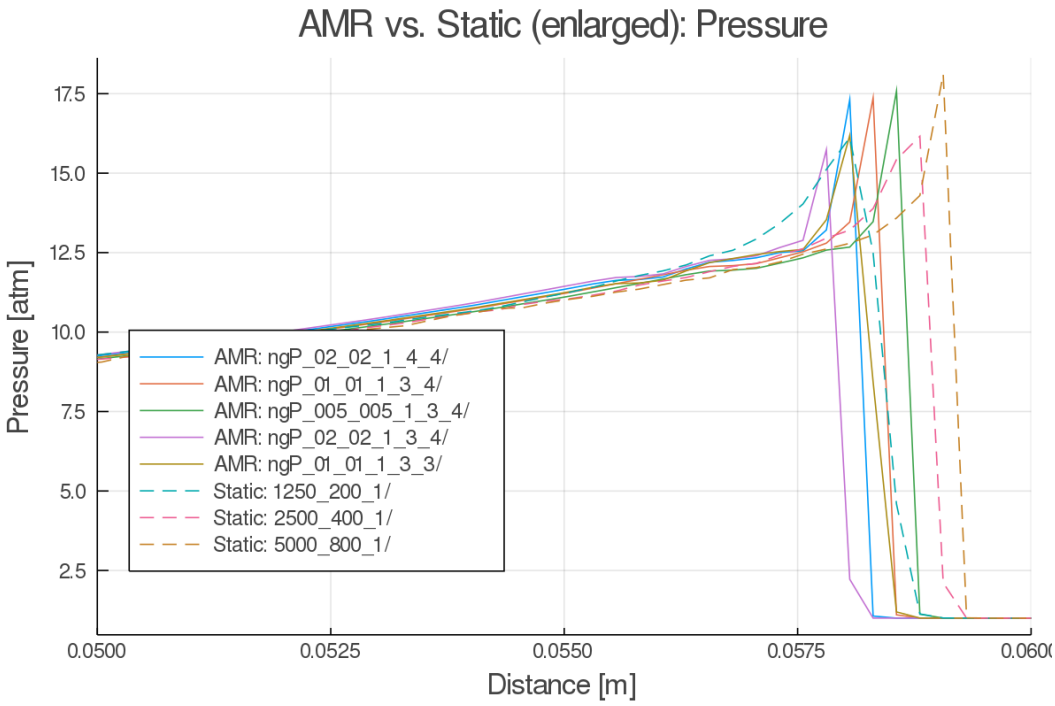
Lower refinement/unrefinement bound	Cells
0.01	139,136
0.05	39,799
0.1	38,343
0.2	37,678
0.3	37,692
0.4	36,355
No AMR	10,000

3.6.4 AMR and Static Comparison

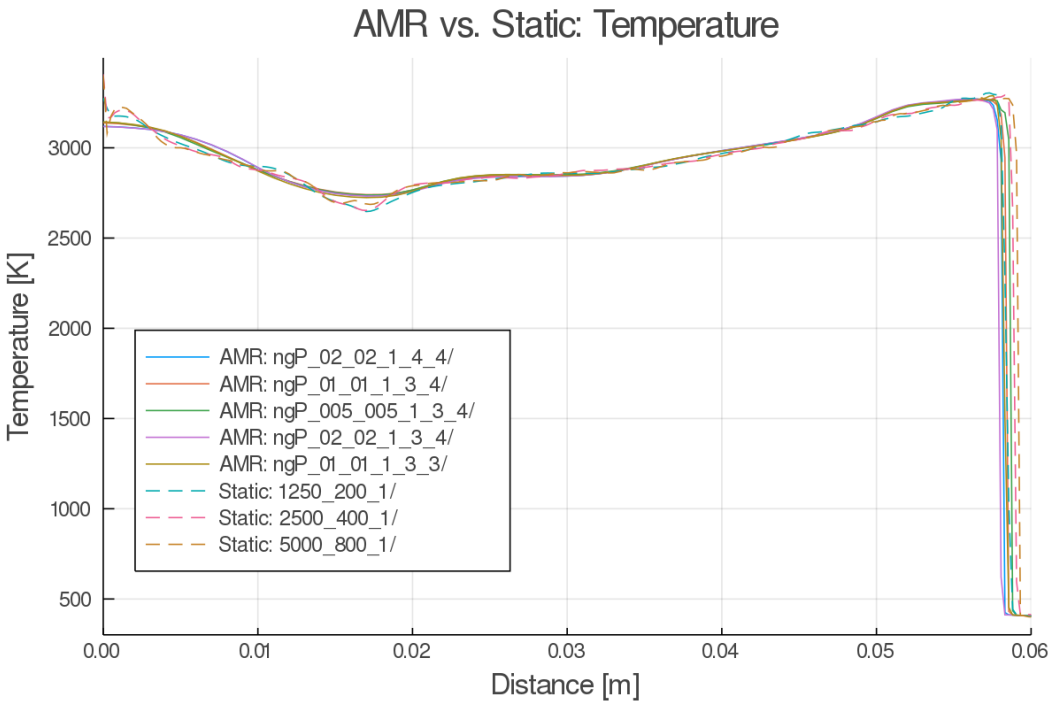
Various AMR runs were performed, and the best can be found plotted in Figure 3.19 along with various comparable static mesh resolutions. Note that due to space, the AMR legend was condensed down and in the following format: [AMR-tracking variable]-[unrefinement level]-[low bound for refinement]-[upper bound for refinement]-[number of buffer layers]-[number of refinement levels]. Cell counts can be found in Table 3.5. From these plots, a few observations can be made. A very good fit exists between the AMR run with 0.1 for



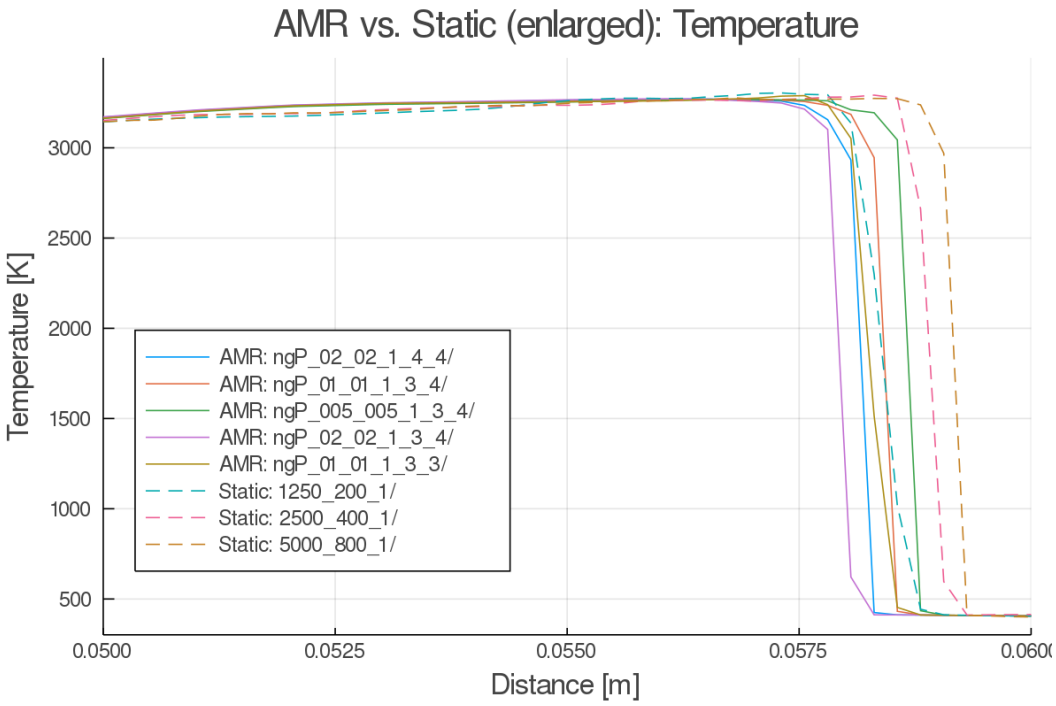
(a) Pressure plot



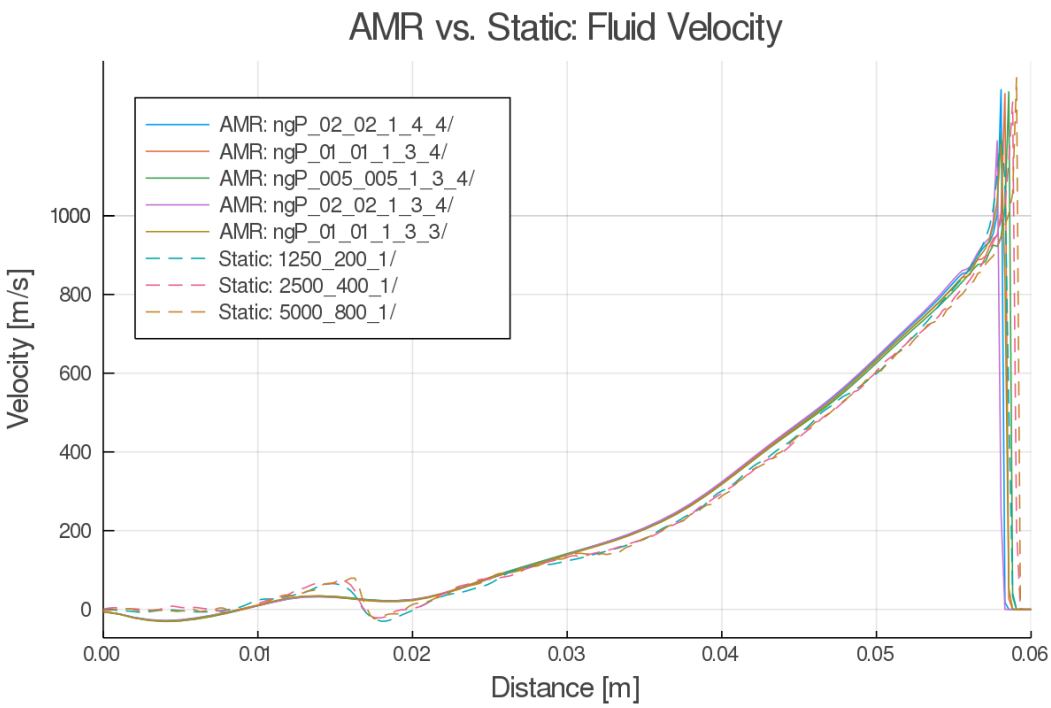
(b) Pressure plot, enlarged



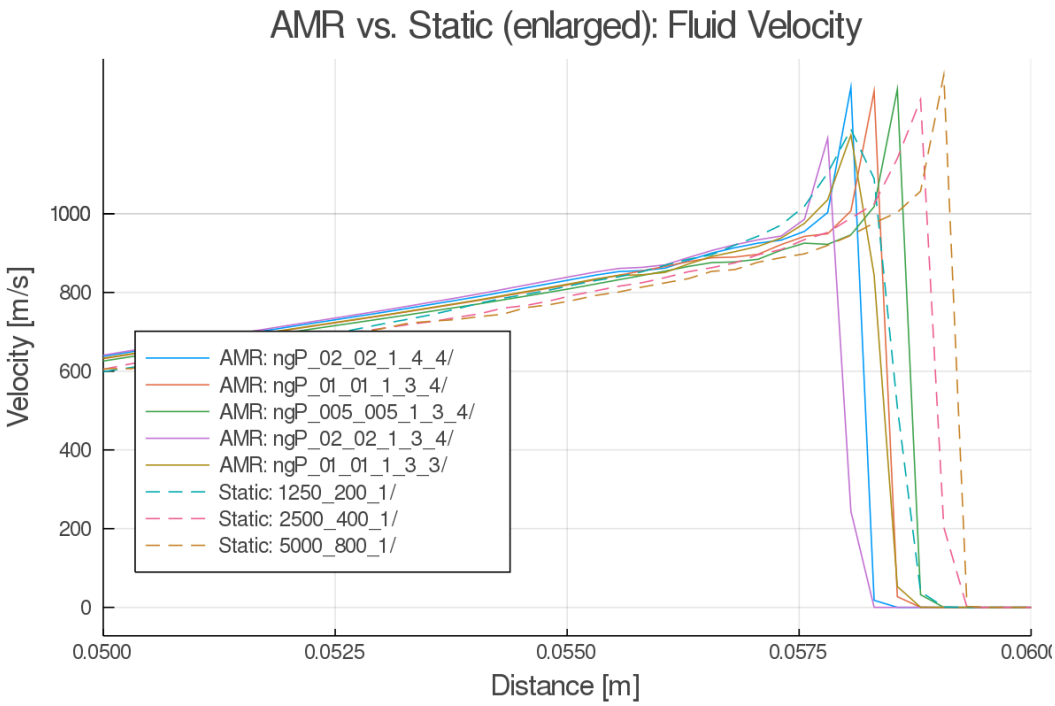
(c) Temperature plot



(d) Temperature plot, enlarged



(e) Fluid velocity plot



(f) Fluid velocity plot, enlarged

Figure 3.19: Best two-dimensional detonation wave AMR and static mesh results compared, tracking $||\nabla(p)||$. AMR legend: field-unrefine-lowrefine-upperrefine-bufferlayers-refinelevels

Table 3.5: AMR and Static Mesh Cell Count Comparisons for Figure 3.19, tracking normalized pressure gradient

Mesh	Type	Unrefine/ lowrefine	Upper refine	Buffer layers	Refine levels	Cell Count
1250-200-1	Static	-	-	-	-	250,000
2500-400-1	Static	-	-	-	-	1,000,000
5000-800-1	Static	-	-	-	-	4,000,000
250-40-1	AMR	0.2	1	4	4	148,950
250-40-1	AMR	0.1	1	3	4	149,762
250-40-1	AMR	0.05	1	3	4	160,570
250-40-1	AMR	0.2	1	3	4	146,710
250-40-1	AMR	0.1	1	3	3	38,343

both the unrefine and low refine bound with 3 buffer layers and 3 refinement levels (ngP-01-01-1-3-3) and the static mesh of 1250-200-1. There is an 85% reduction in cell count between this static mesh and AMR simulation, while exactly reproducing the peak thermodynamic conditions and representing the rest of the detonation profile quite well.

If instead we'd like to model the position of the 1250-200-1 static mesh detonation while having more accurate thermodynamic peak temperatures more akin to the 5000-800-1 static mesh, the ngP-02-02-1-4-4 achieves this with 148,950 cells, which is 40% smaller than the 1250-200-1 static mesh and 96% smaller than the 5000-800-1 static mesh. A use case where this may be preferable is when you'd like to make sure you're modeling a certain grid resolution entirely, but having a more accurate peak solution would also aid design. Another use scenario of this AMR parameter style is when the position of the detonation wave itself is of less concern but the peak solution values need to still be resolved.

An important comparison also lies between the ngP-01-01-1-3-3 (solid brown) just mentioned and ngP-02-02-1-3-4 (solid purple) with higher refinement bounds but a single further refinement level AMR runs. While the solid purple simulation has a higher refinement level, the lower normalized pressure gradient bound allows for more cells to be captured around the detonation and model it more accurately while being 3.8 times smaller in terms of number of cells.

Chapter 4

Summary

4.1 Project Summary

In this thesis, the computational fluid dynamics toolbox OpenFOAM was used to simulate detonation waves within a square tube. Different solvers were tested until the hybrid solver `rhoReactingCentralFoam` was decided as best due to the use of the Kurganov and Tadmor central schemes which reproduce compressible, high speed flow with good accuracy. Different ignition methods were tested, and gradient ignition was found to produce results with the least noise. Next, the chemistry was tested for sensitivity and then matched to Chapman-Jouguet targets and the automatic time stepping algorithm was tested to determine how large the CFL number could be allowed to increase before numerical ringing and noise reduced solution accuracy. A value was found that agreed with similar detonation modeling published works. Static computational meshes were then compared to determine the required mesh resolution to resolve shock and detonation structures. The threshold for resolving finer flow field structures as well as the von Neumann spike was determined, as well as the threshold for overall convergence and detonation wave shape integrity. Detonation cells were shown to be able to form with the solver. Adaptive meshing parameters were then explored, namely refinement levels, buffer layers, and refinement/unrefinement bounds. It was shown that refinement levels and the lower refinement bound are exponential in expense and the buffer layers are linear in expense. No more than 4 buffer layers led to a significant increase in solution accuracy, and a normalized pressure gradient lower bound smaller than

0.05 would significantly increase computational cost. Refinement levels started to converge and reproduce the von Neumann spike for 4 and more levels of refinement for the 250-40-1 mesh, which corresponds to a mesh resolution of 4000-640-1 mesh, where the von Neumann spike began to resolve in the static mesh. Lastly, some AMR simulations were compared to static mesh simulations. It was found that using AMR one could potentially reduce the computational expense by 85% matching a static profile nearly exactly, and reduce computational expense by over 96% by allowing the detonation spacial position to vary while still targeting peak solution values.

4.2 Next Steps

4.2.1 Areas to Improve

Increased knowledge as to the effects of AMR in three-dimensional simulations versus the use in two-dimensional simulations may be valuable. Two-dimensional simulations were the primary target of this thesis work due to a balance of feasibility in computational time as well as sufficient expense to justify and show the benefits of AMR.

An exploration into different tracking parameters other than the normalized gradient of pressure, pressure, temperature, and velocity is useful as a more efficient parameter (i.e., determining a parameter that better describes where refinement is absolutely necessary) may further reduce computational cost. While pressure, temperature, and velocity were all tested during the thesis work, tuning them was found to be more difficult than the normalized pressure gradient. Additionally, combining parameters together may produce smarter AMR "active" periods as the refinement is not free and can be potentially more expensive than static mesh cases if not carefully set up.

Better load balancing in parallel computing in OpenFOAM could use improvement. Currently, intelligent setup and awareness of the detonation itself is required to ensure that the AMR does not offset considerable computational load onto one processor or another as

the detonation moves through the domain. For detonation tubes, this is not a problem as long decompositions can be used to balance the processor load. However, for simulations of RDEs or rocket engine combustion chambers that can be inherently unpredictable, smart domain partitioning is difficult. Like adaptive meshing, an adaptive domain decomposition may assist with improving load balancing in these scenarios.

Further characterization as to how the base mesh affects the solution should be done. While this is seemingly obvious as increased base resolutions will help give a better solution, the AMR can unrefine back to the base resolution, so if large turbulent structures or other larger structures want to be resolved still after adaptive unrefinement, the base resolution needs to be considered.

The AMR routines in parallel can be unstable. This does not seem to affect the solution, but it is not feasible to "babysit" a simulation and restart it if it crashes. The crashes seem to be due to AMR cells not getting communicated correctly with MPI, leading to a disproportionate number of cell faces shared at domain-decomposed boundaries where the MPI communication is occurring. Further work here is needed, and it will likely improve as development continues on the AMR routines.

4.2.2 Future Work

Further work in detonation modeling in the context of RDEs is the logical next step. This technology is still emerging and having better characterization of the highly chaotic flow field inside the engine will guide better engineering design and analysis towards improved propulsion technology. AMR applied to RDE simulation in OpenFOAM will reduce computational cost, especially for three-dimensional RDE simulations.

Deflagration to detonation transition modeling is another area to be explored with this solver. Characterizing this will allow the solver to be used in a wider set of scenarios, where sudden detonation of reactants is a potential concern.

Personally, I think application of AMR simulating the high-speed and chaotic envi-

ronments of rocket engine combustion chambers in OpenFOAM would be really interesting. Combustion instability within these environments is still very hard to characterize and an ongoing topic of research and development within industry.

4.3 Impact

The targeted impact area for this research is the field of detonation modeling, with consideration for future use in RDE modeling and other propulsion systems that are expensive computationally. By utilizing the computational tools developed, tested, and validated in this project, existing research in detonation modeling using the computational fluid dynamics toolbox OpenFOAM can be performed more quickly or to a greater extent. Specifically, I showed:

- The solver `rhoReactingFoam` and `rhoCentralFoam` themselves are unable to model detonations accurately, or at all;
- The solver `rhoReactingCentralFoam` is able to simulate detonations;
- How to simulate detonations in OpenFOAM and what the parameters will do to the solution;
- Detonation cells can be modeled with `rhoReactingCentralFoam`;
- Results are sensitive to Arrhenius pre-exponential factor order, and care must be taken such that the shock does not decouple from the flame;
- CFL number needs to be lower than typical high-speed flows due to the reactive nature of detonations;
- How fine PDE/detonation tube meshes must be to resolve fine detonation and shock structure;

- Certain AMR parameters have different effects computationally, and tuning certain parameters over others will optimize the overall cell count by placing refinement cells where they are truly needed;
- Adaptive mesh refinement can decrease computation cost for detonation modeling up to 96% while remaining true to the static mesh resolution results.

Bibliography

[1] H Weller, C Greenshields, M Janssens, A Heather, and S Ferraris. Cj, openfoam®-the open source computational fluid dynamics (cfd) toolbox, 2004.

[2] Colin A. Towery, Katherine M. Smith, Prateek Shrestha, Peter E. Hamlington, and Marthinus Van Schoor. Examination of Turbulent Flow Effects in Rotating Detonation Engines.

[3] Douglas Schwer and K. Kailasanath. Fluid dynamics of rotating detonation engines with hydrogen and hydrocarbon fuels. Proceedings of the Combustion Institute, 34(2):1991 – 1998, 2013.

[4] David Leonard Chapman. Vi. on the rate of explosion in gases. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 47(284):90–104, 1899.

[5] Emile Jouguet. Sur la propagation des réactions chimiques dans les gaz. J. Maths. Pure Appl., 7:347, 1905.

[6] L.F. Gutiérrez Marcantoni, J. Tamagno, and S. Elaskar. rhocentralrffoam: An openfoam solver for high speed chemically active flows – simulation of planar detonations –. Computer Physics Communications, 219:209 – 222, 2017.

[7] Christopher J Greenshields, Henry G Weller, Luca Gasparini, and Jason M Reese. Implementation of semi-discrete, non-staggered central schemes in a colocated, polyhedral, finite volume framework, for high-speed viscous flows. International journal for numerical methods in fluids, 63(1):1–21, 2010.

[8] NASA. Chemical equilibrium with applications (cea). `https://www.grc.nasa.gov/WWW/CEAWeb/`.

[9] Thien Xuan Dinh, Masatake Yoshida, and Shuichi Ishikura. Simulation of rotating detonation engine by openfoam. Sci. Tech Energetic Materials, 80(2):68 – 71, 2019.

[10] D. Kim and J. Kim. Numerical method to simulate detonative combustion of hydrogen-air mixture in a containment. Engineering Applications of Computational Fluid Mechanics, 13(1):938–953, 2019.

[11] Jian Li, Jianguo Ning, Charles B. Kiyanda, and Hoi Dick Ng. Numerical simulations of cellular detonation diffraction in a stable gaseous mixture. Propulsion and Power Research, 5(3):177 – 183, 2016.

[12] MJ Berger and J Oliger. The amr technique. J. Computat. Phys, 53:484–512, 1984.

[13] Marsha J Berger and Antony Jameson. Automatic adaptive grid refinement for the euler equations. AIAA journal, 23(4):561–568, 1985.

[14] Marsha J Berger, Phillip Colella, et al. Local adaptive mesh refinement for shock hydrodynamics. Journal of computational Physics, 82(1):64–84, 1989.

[15] Ralf Deiterding. Parallel adaptive simulation of multi-dimensional detonation structures. Dissertation. de, 2003.

[16] William D Henshaw and Donald W Schwendeman. An adaptive numerical scheme for high-speed reactive flow on overlapping grids. Journal of Computational Physics, 191(2):420–447, 2003.

[17] Tae-Hyeong Yi, Cary Turangan, Jing Lou, Piotr Wolanski, and Jan Kindracki. A Three-Dimensional Numerical Study of Rotational Detonation in an Annular Chamber.

[18] B. van Leer. Towards the ultimate conservative difference scheme, v. a second order sequel to godunov's method. Journal of Computational Physics, 32:101 – 136, 1979.

[19] D.A. Kessler, V.N. Gamezo, and E.S. Oran. Multilevel detonation cell structures in methane-air mixtures. Proceedings of the Combustion Institute, 33(2):2211 – 2218, 2011.

[20] Simscale: Cloud-based hpc simulation platform. `https://www.simscale.com/`, 2020.

[21] SIMFLOW Technologies. simflow: Computational fluid dynamics software. `https://sim-flow.com/`, 2020.

[22] SIMFLOW Technologies. Rapidcfd: Openfoam running on a gpu. `https://github.com/Atizar/RapidCFD-dev`, 2020.

[23] Nvidia. Cuda: a parallel computing platform and programming model. `https://developer.nvidia.com/cuda-zone`, 2020.

[24] Nakul. rhocentralspeciesfoam, a new solver for supersonic combustion. `https://www.cfd-online.com/Forums/openfoam-community-contributions/82851-new-solver-supersonic-combustion.html`.

[25] swak4foam. `https://openfoamwiki.net/index.php/Contrib/swak4Foam`.

[26] Alexander Kurganov and Eitan Tadmor. New high-resolution central schemes for nonlinear conservation laws and convection-diffusion equations. Journal of Computational Physics, 160:241–282, 2000.

[27] Kenneth Kuo. Principles of Combustion. Wiley, 2nd edition, 2005.

[28] George Gabriel Stokes. On the steady motion of incompressible fluides. Transactions of the Cambridge Philosophical Society, 7:439 – 453, 1842.

[29] H. A. van der Vorst. Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing, 13(2):631–644, 1992.

[30] William Sutherland. Lii. the viscosity of gases and molecular force. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 36(223):507–531, 1893.

[31] The OpenFOAM Foundation. Openfoam v7 user guide. `https://cfd.direct/openfoam/user-guide`.

[32] Svante Arrhenius. Uber die dissociationswarme und den einfluss der temperatur auf den dissociationsgrad der elektrolyte. Zeitschrift fur Physikalische Chemie, 4U(1):96 – 116, 1889.

[33] Dominik Christ. Simulating the combustion of gaseous fuels. `https://openfoamwiki.net/images/8/8a/Christ_OFW6_slides.pdf`.

[34] Jung Rai, Sudeep Parajuli, Arash Abbasi, and Sayed Hashemi. Investigation of different chemical kinetics reaction rates and its relevant modelling for h2 -o2 combustion. 05 2016.

[35] Jr. M.W. Chase, C.A. Davies, Jr. J.R. Downey, D.J. Frurip, R.A. McDonald, and A.N. Syverud. Nist-janaf thermochemical tables. `https://janaf.nist.gov/`, 1998.

[36] Carl Friedrich Gauss. Methodus nova integralium valores per approximationem inveniendi. apvd Henricvm Dieterich, 1815.

[37] S. K. Godunov. Eine Differenzenmethode für die Näherungsberechnung unstetiger Lösungen der hydrodynamischen Gleichungen. Mat. Sb., Nov. Ser., 47:271–306, 1959.

[38] Krzysztof Michalak and Carl Ollivier-Gooch. Limiters for Unstructured Higher-Order Accurate Solutions of the Euler Equations.

[39] L. Euler. Institutionum calculi integralis. Number v. 1 in Institutionum calculi integralis. imp. Acad. imp. Saènt., 1768.

[40] Yousef Saad. Iterative method for sparse linear systems. PWS publishing company, a division of international Thomson publishing Inc, 2nd edition, 1996.

[41] A. Schwarzenberg-Czerny. On matrix factorization and efficient least squares solution. Astronomy and Astrophysics Supplement,, 110:405, April 1995.

[42] G.H. Golub and C.F. Van Loan. Matrix Computations. Johns Hopkins Studies in Atlantic History & Culture. Johns Hopkins University Press, 1983.

[43] E. Fehlberg. Klassische runge-kutta-formeln vierter und niedrigerer ordnung mit schrittweiten-kontrolle und ihre anwendung auf wärmeleitungsprobleme. Computing, 6(1):61–71, 1970.

[44] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen differenzengleichungen der mathematischen physik. Mathematische Annalen, 100(1):32–74, Dec 1928.

[45] J. Ahrens, Berk Geveci, and Charles Law. Paraview: An end-user tool for large data visualization. Visualization Handbook, 01 2005.

[46] T.J Poinsot and S.K Lelef. Boundary conditions for direct simulations of compressible viscous flows. Journal of Computational Physics, 101(1):104 – 129, 1992.

[47] D W Walker. Standards for message-passing in a distributed memory environment. 1992.

[48] Colin A.Z. Towery, Alexei Y. Poludnenko, and Peter E. Hamlington. Detonation initiation by compressible turbulence thermodynamic fluctuations. Combustion and Flame, 213:172 – 183, 2020.

[49] Chibuikem Ajaero. Validation of the kurganov-tadmor/kurganov-noelle petrova scheme for rotating detonation engine simulations using openfoam. `https://spectrum.library.concordia.ca/984878/`, January 2019.

[50] Ya. B. Zel'dovich. On the theory of the propagation of detonations on gaseous system. Zhurnal Eksperimental'noi i Teoreticheskoi Fiziki, pages 542 – 568, 1940.

[51] J. von Neumann. Theory of detonation waves. progress report to the national defense research committee div. b, osrd-549 (pb 31090). John von Neumann: Collected Works, 1903–1957, pages 178 – 218, 1942.

[52] Werner Doring. On detonation processes in gases. Annalen der Physik, pages 421 – 436, 1943.

[53] Alexander Kurganov, Sebastian Noelle, and Guergana Petrova. Semidiscrete central-upwind schemes for hyperbolic conservation laws and hamilton–jacobi equations. SIAM Journal on Scientific Computing, 23(3):707–740, 2001.

[54] Nagoya University. 900-n ethelene-oxygen rotating detonation engine. `http://www.mae.nagoya-u.ac.jp/flight/research1.pdf`.