Optimisation and Analysis of Injector Geometry on a Centre-Bodiless Continuous Rotating Detonation Rocket Engine

Reuben Hann

Submitted in fulfilment of the requirements for the
Degree of Master of Sciences (By Research) In Aerospace Engineering

School of Engineering
College of Science and Engineering
University of Glasgow

September 2021

Abstract

Rotating Detonation Engines (RDEs) provide a means of improving the efficiency of combustion engines at a time when reducing emissions is paramount. The key to their operation are RDE injectors, the two main injector design methodologies being the Semi-Impinging Injector (SII) method and Pintle injector method. In this thesis, the SII method was modified to add an additional degree of freedom (DOF) perpendicular to the two DOFs present in the SII method to develop the Modified Semi-Impinging Injector (MSII) method. This was done with the goal of improving the optimisation, implementation, and performance of the injector. The MSII and SII methods were compared where it was found that the injector flows could be categorised into two stages, the mixing phase where the mixing efficiency rose rapidly and the dampening phase where the mixing efficiency value stabilise over the length of the flow. The stabilised flow was found to remain relatively constant over the DOF ranges explored. Therefore, to determine the optimal injector, given that the speed of mixing is key to RDE performance, the characteristic length measurement was developed. The characteristic length is defined as the length required to meet 63.2% of the final stabilised value, with the lower the length, the faster the mixing. It was discovered that the mixing efficiency was the most relevant performance characteristic and that applying the characteristic length to the mixing efficiency allowed the mixing speed to be measured. It was found that the MSII injectors outperformed the SII injectors in mixing speed. The MSII method was then applied to a numerically simulated RDE and compared to a comparable Pintle injector method RDE. The two injector designs were simulated using Ansys Fluent by a detailed and simplified simulation method. It was found that the Ansys software had issues simulating RDEs resulting in only short runs of the engines, where the results were inconclusive and often contradictory. It was recommended that the MSII and SII methods be first empirically validated before more numerical simulations are conducted, and that with the information currently available, that the Pintle injector method is the best currently in use.

Contents

List of Tables

List of Figures

Acknowledgements

My thanks to my supervisor Dr. Enric Grustan-Gutiérrez, and my parents for their support throughout this thesis.

Declaration

With the exception of chapter 1, which contains introductory material, all work in this thesis was carried out by the author unless otherwise explicitly stated.

Chapter 1

Introduction

As the Falcon 9's engines lit up and roared into action, lifting NASA's Robert Behnken and Douglas Hurley to the International Space Station (ISS) on 30th May 2020 as part of Demo-2, thus sounded the starting gun of a new space race. This space race, unlike that of the 1960s, is predicated on competition between corporations as opposed to a clash of economic systems. The state of the space launcher market at the beginning of the 21st century was a monopoly, with preferential treatment towards launchers such as the Space Shuttle and Soyuz which though reasonably effective and reliable, were prohibitively expensive and economically inefficient. As the Space shuttle was coming to the end of its service life, a replacement was required. The replacement program was called the Constellation Moon program, a Bush Jr era ineffective, over-scoped, and underfunded attempt of a 21st-century parody of the Apollo moon-shot program, following the same uncompetitive practises as before. Under the Obama administration, this program was cancelled and replaced with the Commercial Crew Program [31]. Built on the back of the 18th century Scottish Economist and fellow University of Glasgow alumni Adam Smith's thinking on free markets, the program injected competition and a level playing field into the sector. In Smith's book "An Inquiry into the Nature and Causes of the Wealth of Nations" it was noted over almost 300 years ago that "enjoying a sort of monopoly there [countries or companies], will often sell their goods for a better [higher] price than if exposed to the free competition" [46]. This is a timeless fact, regardless of the field of commerce be that spaceflight or the sale of grain, lacking competition increases costs for all involved, and in NASA's case, American taxpayers. Therefore, the Commercial Crew program was initiated by putting out a funding round, with a set of requirements, requiring spaceflight companies regardless of their shape or size to compete for these rounds of funding to develop the next generation of launchers and space infrastructure. This approach allowed multiple new start-ups at the time, such as Blue Origin and SpaceX to receive funding to build and complete their technologies, while also competing against each other for further funding and later contracts to launch payloads into orbit [34]. This culminated in 2020 with the Demo-2 NASA mission. The mission was the first of many things, the first launch

directly from the US to the ISS in 9 years, a milestone in the Commercial Crew Program, and the first crew ever launched into orbit by a commercial provider [11]. This landmark illustrated the fruits of the push to commercialise space and increase competition while also reducing costs. For example, on average launching a payload to low earth orbit has decreased costs on the scale of 20 times, and to the ISS by a factor of 4 [17]. This has created profitable companies such as SpaceX, Blue Origin, or United Launch Alliance (ULA) and an expanding market with increasing competition. As this market expands, competition increases, to continue to be competitive companies must innovate through the research and development of different technologies to improve the performance of their launch platforms. One of these potential technologies is rotating detonation.

1.1 Background

1.1.1 Detonation

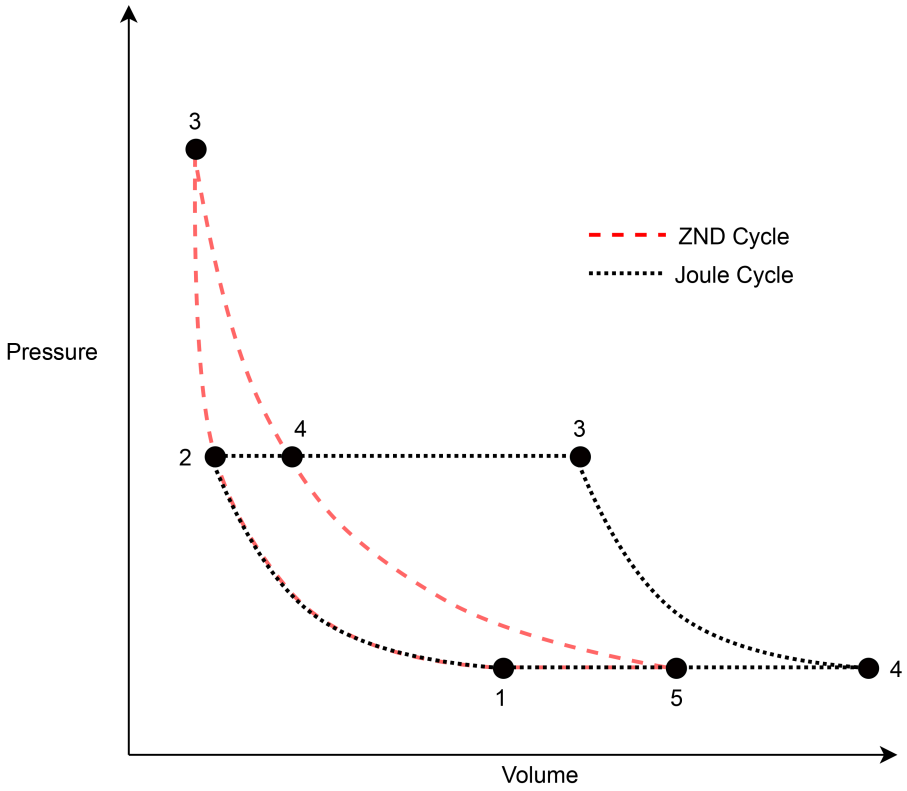


Figure 1.1: p-v Diagram comparing the ZND cycle to the Joule Cycle.

The combustion of reactants can be broken down into two types, deflagration and detonation. Deflagration is characterised by relatively slow subsonic flame speeds [12] and can be described by the Joule cycle [41]. Detonation on the other hand is characterised by fast supersonic flame

speeds and is described by the Zel'dovich-von Neumann-Doring (ZND) cycle [25]. Comparing the two cycles it can be seen that the ZND cycle has a larger pressure gain as shown in Figure 1.1 and has been found to have a higher thermodynamic efficiency due to its lower entropy increase during the cycle [41]. As detonation has higher thermodynamic efficiency, there has been an interest in using detonation to replace deflagration in combustion engines. The two are not interchangeable though, detonation comes with distinctive qualities and factors which pose unique and complex engineering challenges.

A complication with detonation is that within a gaseous atmosphere, detonation requires a transition from deflagration to detonation (DDT) to initiate. The DDT phenomenon currently does not have a mathematical description, but the general process is understood. It starts with the ignition and expansion of a laminar flame at a relatively low speed. As the flame front expands the front warps and crumples losing its smooth form increasing the flame front's area and turbulence. This turbulence causes pressure pulses which heat the reactants ahead of the flame front. As the temperature of the incoming reactants increases, the flame speed accelerates. This process repeats exponentially, until in some areas ahead of the flame front, the reactants self-ignite, causing new flame fronts which can combine to further accelerate the front until a shockwave is formed and the reaction zone behind starts driving the shockwave. The combination of the shockwave and reaction zone is called a detonation wave.

Once detonation is achieved, new complexities are found. The detonation wave is still a highly volatile and unstable process, with the detonation wave's shockwave, dissimilar to that of typical shockwaves, taking on a microscopically small cellular structure [38]. The detonation wave operates in a periodic cycle of the creation and destruction of detonation cells. The detonation cells are the scale-like soot patterns produced by the trajectories of the triple points. Triple points occur where Mach stems, powerful curved shockwaves that form on the flame-front intersect with a straight incident shockwave. The incident shockwaves form between Mach stems, with triple points at each end. As the detonation wave travels forward the Mach stems grow and expand encompassing the incident shockwave until the two triple points meet and combine to form a new Mach stem. As the Mach stems expand, the strength of the shockwave decreases until it transitions to an incident shockwave, between the two recently formed Mach stems, repeating the cycle. The detonation cell width often used as a defining parameter within the field of detonation research is found by measuring the maximum width of the triple point soot paths (i.e. detonation cells).

Another engineering complexity of detonation waves is the discontinuity within the process. As with all shockwaves, due to the large pressure, temperature and density gradients produced, they cannot be described accurately by isentropic or continuous flow equations [25]. The discontinuity disconnects the area ahead of the detonation wave from the area behind, where the area behind cannot influence the area ahead, the only continuity between the two areas are the conti-

nuity laws. The continuity laws are the continuity of mass (equation 1.1), momentum (equation 1.2), and energy (equation 1.3) as described by the Hugoniot adiabatic. In describing detonation waves and the discontinuity, the ZND model, developed in the 1940s is the most widely used. The ZND model describes a shockwave travelling through an atmosphere of well-mixed reactants, considering the shock wave to be the point of reference. The reactants pass through the shockwave and then react behind it in the combustion region. This reaction takes time, where the shorter the reaction, the higher the temperature release and the stronger the shockwave, thereby the strength of the shockwave is tied to the time of reaction. As the combustion region follows behind the shockwave, the region matches the shockwaves velocity, therefore the thickness of the combustion region is that of the velocity of the detonation wave multiplied by the time of combustion. This allows the relation of the energy release to be correlated to the velocity of the detonation wave. The energy released is not dependant on any other factors than the fuel and oxidiser used, the equivalence ratio, and initial conditions. The model was obtained by modifying the Hugoniot adiabatic, to produce the detonation adiabatic by specifying that the enthalpies H_1 and H_2 are different from the Hugoniot, due to the combustion of the reactants they are no longer functions of(P_1, V_1) and (P_2, V_2) [25]. This produces two curves as seen in Figure 1.2, the Hugoniot adiabatic denoted with the dashed line and the detonation adiabatic denoted with the continuous line. The detonation adiabatic line is higher as the temperature and pressure will be higher for the same specific volume after the combustion of the reactants. The Hugoniot adiabatic is a curve of P_2 as a function of V_2, noting that $V = \frac{1}{\rho}$, passing through (P_1, V_1) and the detonation adiabatic a curve of P_2 as a function of V_2, following equation 1.4 [25]. The Hugoniot adiabatic's description of a shockwave is listed in equation 1.5. For a shockwave to be described by the Hugoniot adiabatic, this being attributable to the detonation adiabatic as well, is that a chord passing through the curve cannot be tangent to the curve as it would violate the Hugoniot adiabatic's description of a shockwave, with $u_2 > a_2$ [25]. Given this, the minimum gradient of a chord starting at (P_1, V_1) to any position on the detonation adiabatic must not be lower than the chord a-O [25]. Obtaining the gradient of any position on the Hugoniot adiabatic curve is found with the linear equation 1.6 which with equation 1.1 allows the shows that the gradient of the line is equal to the velocity [25].

$$j = \rho_1 u_1 = \rho_2 u_2 \tag{1.1}$$

$$P_1 + \rho_1 u_1^2 = P_2 + \rho_2 u_2^2 \tag{1.2}$$

$$H_1 + \frac{u_1^2}{2} = H_2 + \frac{u_2^2}{2} \tag{1.3}$$

$$H_1 - H_2 + \frac{1}{2}(V_1 - V_2)(P_2 - P_1) = 0 \tag{1.4}$$

$$P_2 > P_1, \quad V_1 > V_2, \quad u_1 > a_1, \quad u_2 < a_1, \quad u_1 > u_2 \tag{1.5}$$

$$j^2 = \frac{P_2 - P_1}{V_1 - V_2} \tag{1.6}$$

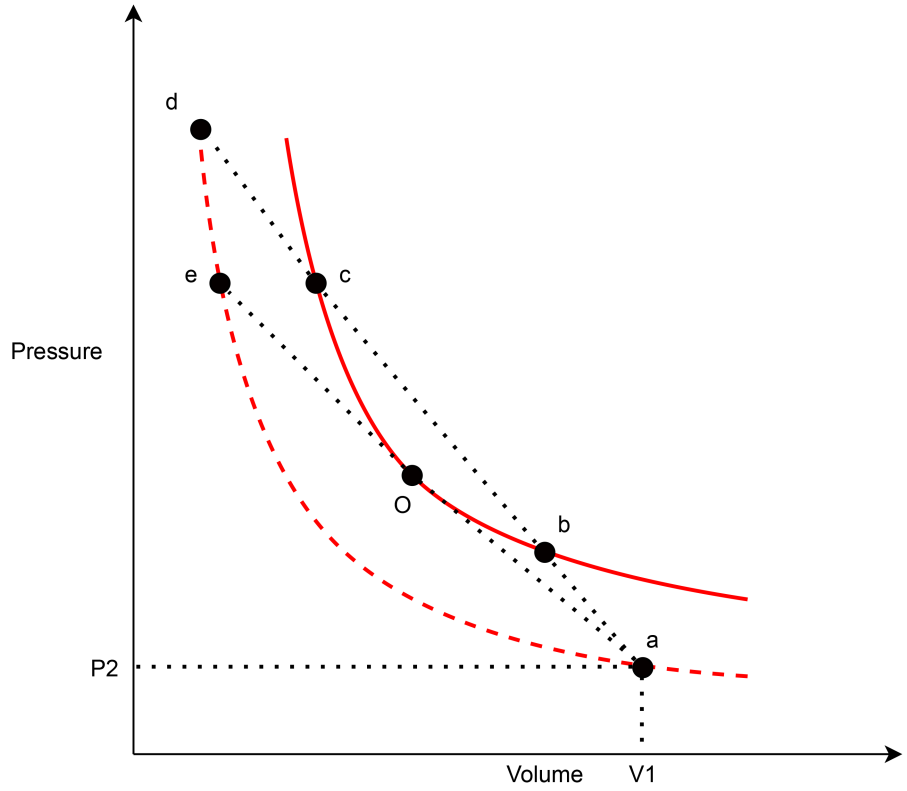


Figure 1.2: p-v diagram comparing the Hugoniot adiabatic to the detonation adiabatic.

With the adiabatic curves and chords set, the ZND cycle can be identified as shown:

1. As the reactants enter the detonation wave, they are first compressed and heated by the leading shockwave, moving from point a to d. See 1 to 3 in Figure 1.1.

2. The reactants are combusted behind the shockwave and the products then expand reducing the pressure, moving to point c. See 3 to 4 in Figure 1.1.

3. As to satisfy the continuity laws, the gradient cannot be lower than the tangent, therefore the minimum position on the detonation adiabatic curve must be O, also known as the Chapman-Jouguet point. As the volume increases the pressure drops moving to O and

then along from O to A to repeat the cycle as fresh reactants interact with the detonation wave. See 4 to 5 and then 5 to 1 in Figure 1.1.

1.1.2 Rotating Detonation Engines

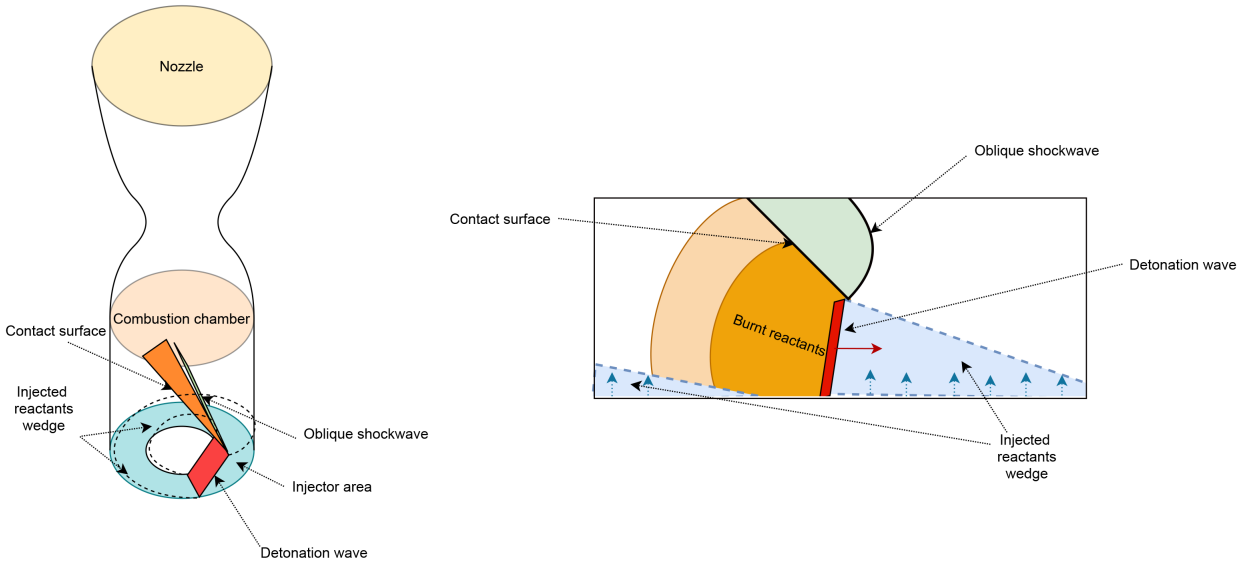


Figure 1.3: 2D representation of the structure of a rotating detonation wave

RDEs operate by utilising a detonation wave and continuously rotating it around the inside of a cylinder or annulus. By feeding in reactants perpendicular to the detonation wave from the top and ejecting out the burnt products below an RDE theoretically should be able to continuously operate so long as reactants are fed in. RDEs are comprised of the combustion chamber of which if the outer wall is unwrapped, produces the diagram shown in Figure 1.3. This is comprised of the detonation wave, which consumes the unburnt reactants wedge, which grows in height the longer the position has had since the detonation wave passed over. The detonation wave produces an oblique shockwave which alters the flow of the burnt products. A contact surface is produced between the burnt products and the fresh reactants ahead of the detonation wave, another is produced between the high temperature recently combusted products and the older, cooler products. The burnt products then are ejected by a nozzle to produce thrust. To start the engine and initialise detonation, a pre-detonator is used. With a spark igniting a stoichiometric mixture of oxygen and hydrogen in the pre-detonator, the starting deflagration flame front is created. To incite the DDT process a shchelkin spiral is used to accelerate the turbulent eddies that go on to produce the detonation wave. The detonation wave is then transplanted into the detonation chamber, by injecting it tangent to the outer wall, with the detonation wave then beginning to curve and feed on the injected reactants starting the rotating detonation process. RDE detonation waves frequently have lower detonation wave velocities than traditional detonation waves

using the same global conditions, this is due to incomplete mixing of the injected reactants and mixing of burnt and unburnt gases due to the partial ejection of the burnt products. Additionally, parasitic deflagration and a loss in momentum due to the constant curving of the detonation wave take their toll on detonation wave velocity [8].

To measure the performance of the engine, excluding the traditional measures of the thrust, specific impulse and characteristic velocity, RDEs provide another measure specific to the use of detonation within the engine, the tangential detonation velocity [1] though it is often just called the detonation velocity while referencing an RDE. The use of this measure, has the benefits of simplicity and ease of measure, requiring only a single point measuring the pressure over the run time of the engine, measuring the time between peak pressures of the passing detonation waves, which given the knowledge of the geometry of the engine, allows the detonation velocity to be found. The use of the tangential detonation velocity as a measure poses issues at large mass flow rates, as the detonation wave is typically angled forward as seen in Figure 1.3, with the direction of travel perpendicular to the detonation wave. Combining the injected reactants velocity with the detonation waves velocity produces a velocity triangle, where the unknown is the actual detonation velocity as the hypotenuse, therefore it will be larger than the tangential detonation velocity creating an error between actual detonation velocity, and the calculated one. As most research conducted to this date uses comparatively low mass flow rates, the error inherent to this issue is relatively low but will pose an issue as the technology continues to mature [1]. To allow comparison of different RDE geometries and reactants, an independent measure of performance is needed, as the detonation velocity will be different depending on the geometry of the engine, and reactants used. A theoretical method of calculating the idealised detonation velocity is the Chapman-Jouguet (C-J) method. In contrast to ZND theory, it assumes an instantaneous thermodynamic equilibrium [15] over a cycle that takes time, though this is an idealised assumption, it allows an idealised detonation velocity to be calculated, based on the initial conditions of the reactants. The C-J velocity provides a universal reference point for detonation, by dividing the detonation velocity by the C-J velocity, we can calculate the %C-J velocity, as a point to compare the performance of different RDEs.

Today RDEs that are studied typically fall into two types, hollow and annular. The original and traditional design is the annular type, first being tested in 1966 [35], with the detonation wave travelling in the channel between two cylinders. This design is effective in reaching continuous operation [47] with approximately 70-75% C-J velocities [40], though faces a number of issues implementing into commercial combustion engines. This is because annular designs require complex cooling channels to cool both the inner and outer walls [54], have low-pressure gains due to the majority of the kinetic energy being in the axial direction [45], produce large amounts of parasitic deflagration [45] and have thrust stability issues at low altitude [43], thereby restricting use to high altitude and the vacuum of space. Hollow type RDEs both solve these issues and improve performance at the same time. Taking inspiration from combustion instabilities

in rocket engines, and their similarities to detonation [3,26], as typical rocket engine combustors are cylinders in 2015 the inner cylinder was removed to produce the hollow RDE design [26]. By removing the inner cylinder hollow RDEs remove some of the complexity of the cooling system, simplifying the design. Hollow RDEs also typically reach at least 80% C-J velocities [29] with examples found to each upwards of 95% C-J velocity [49] and don't face the same thrust instabilities as annular designs, especially in conjunction with a De Laval nozzle [45].

1.2 Modelling RDEs

1.2.1 Simulations

Though the concept of a rotating detonation engine has been around since the 1960s [35], it took until the late 2000s to be more widely studied [8]. As before then studying RDEs was limited to empirical tests of which a limited amount of data could be obtained about the detonation structure and evolution of the detonation wave. Since the 1990s, parallel computing has risen as the main form of high-power computing, in combination with the exponential growth in processor computing speeds following Moore's Law. HPC's (high-power computing) has formed the foundation for computational fluid dynamics (CFD) use in research and industry to gain insights, model, and analyse all types of flows. This includes modelling combustion and reactive flows which is applicable to RDEs. Today, by making use of the finite volume method [50] RDEs have been simulated by both open-source [50] and commercial [44] CFD packages and codes. Traditionally RDEs have been simulated with the Euler equations, specifically the 2D or 3D inviscid reactive Euler equations [28]. The viscous, thermal conduction and mass diffusion effects are discarded [9, 28], with this method being found to accurately model the main structures of detonation waves [27]. These days models that consider viscous and diffusive effects are being used more, with the K-ϵ, Realisable K-ϵ, RNG K-ϵ, Reynolds Stress equation model, Shear Stress Transport (SST) K-ω [48], and Large Eddie Simulation methods [60] found to accurately model RDEs and detonation waves. Additionally, when modelling the mass diffusive effects it has been found that it has a large effect on recreating the parasitic deflagration seen in empirical results, and how the detonation wave interacts with boundary layers at the wall, though this has only been studied in annular RDEs [27]. When viscous effects are accounted for it was found that the detonation wave became more curved and with a reduced detonation velocity of around 10% [27].

These effects, fall into the same problems all CFD simulations face, that as the complexity of the simulation increases, for example using an SST K-ω model over an Inviscid Euler model, the computational cost increases. This is especially problematic when simulating RDEs, as RDEs are very complex phenomena and therefore require high computational power to run, in the authors

experience a typical 2D RDE can make use of up to 700,000 elements, whereas a 3D annular RDE upwards of 10 million for the same geometry. And the difference in accuracy is not trivial, it was found that the difference between a 2D and 3D RDE simulation led to the parasitic deflagration increasing from 10% to 30% and %C-J velocity decrease from 96% to 87% [14]. Another area in which there is a required accuracy vs computational cost analysis required is the reaction mechanism used. All RDE reaction mechanisms make use of the Arrhenius equation. This is as detonation waves are supersonic flames with slow chemical kinetics and little turbulence-flame interaction, therefore using a laminar finite-rate flame model is appropriate [48]. Depending on the accuracy of the simulation and, available computing resources and desired accuracy of the simulation, 1-step [28, 51], 2-step [53], 5-step [24] and 8-step [23, 44] reaction mechanisms are the most common. The difference between a multi-step reaction mechanism and a single step can be noticeable, with a multi-step, found to have a 0.4% deficit from the C-J velocity and the single-step having around a 7% deficit [56]. The validation of the reaction mechanism and simulation method has been achieved by the comparison of the simulation's averaged detonation wave velocity, to the 1D theoretical C-J velocity, calculated from the initial global conditions, often NASA's CEA code is used to calculate this [22]. The other method which is often combined with the prior is empirically testing the model and comparing the results [49]. While no model is perfect, RDE CFD simulations are useful and a key asset in bringing RDEs to technical viability.

1.2.2 Modes of Operation

By sensing the pressure at points along the engine, as is often done in empirical studies [18, 37, 56], the propagation modes, and operating dynamics of RDEs have been studied. Within annular RDEs the modes of detonation can be classified into 4 different types, Fast Deflagration, Unstable Detonation, Quasi-Stable Detonation and Stable Detonation with the modes transitioning along this mode spectrum as the mass flow rate is increased and the equivalence ratio approaches stoichiometric conditions [52]. Fast Deflagration occurs at very low mass flow rates or below the lean detonation equivalence ratio limit, producing a small gain in pressure, rotating around the detonation chamber. This is due to the curvature of the annular geometry and the engine being initiated tangent to the annulus, giving a tangent velocity to the deflagration flame when ignited as opposed to ignition from a spark plug which spreads equally from a single point. Fast deflagration also has flame speeds below 1000 m/s typically reaching %C-J velocities less than 50%. The frequency of the fast deflagration wave is coupled with the circumferential acoustic frequency of the annular chamber [52]. At the boundary between deflagration and detonation Unstable Detonation occurs. Alternating between the two conditions with an unstable detonation wave velocity it experiences three types of instabilities: The High-Frequency Chaotic Instability, where pressure peaks fluctuate quasi-periodically and at a frequency in the kHz range and can be identified via Fast-Fourier Transform analysis by the multiple main peaks produced,

as opposed to just one in a stable rotating detonation engine [52]. This instability occurs when the pressure waves couple to the injector pressure, which is exacerbated with unchoked injectors [52]. The Low-Frequency Bulk Mode Instability, where periodic peak pressures rise and fall in a pattern similar to a sawtooth wave at a frequency below the kHz range. This instability occurs when the flame front transitions from deflagration to detonation. As this process is not instantaneous, while in this transition the High-frequency Chaotic Instability can initiate this instability, with the Low-Frequency Bulk Mode Instability acting parasitically off of the High-frequency Chaotic Instability [52]. The Extinction and Restart Instability where the detonation wave self-extinguishes and transforms to deflagration until it re-initiates a detonation wave itself in a quasi-periodic manner. So far there is no proven explanation for this process, though probable processes have been put forward relating to non-uniform mixing [52]. Quasi-stable detonation occurs when all signs of fast deflagration have disappeared, but instabilities remain, as multi-wave detonation waves can form from a single initial detonation wave and other complex phenomena such as generating additional counter-rotating waves and the detonation wave alternating its direction of rotation [52]. Once there is a sufficient mass flow rate and equivalence ratio Stable Detonation occurs producing a stable detonation velocity and frequency [52].

As hollow RDEs remove the inner cylinder, the modes, and dynamics of annular RDEs are not entirely identical to hollow RDEs, though they are not unalike. The modes of propagation in a hollow RDE can be classified into four types, sawtooth-wave mode, single wave mode, two-dominant peak one-wave mode [16], and two-wave mode [59] with the types transitioning from one mode to another as the mass flow rate increased and equivalence ratio approached stoichiometric conditions and when the contraction ratio was increased. The sawtooth wave mode is rather analogous to the quasi-stable detonation mode in annular RDEs, with a rather low detonation velocity and pressure gain and a high instability with the ability to spontaneously extinguish itself. The sawtooth wave occurs at the lean limits of detonation giving a sawtooth-like pressure graph, unlike the single wave-mode with a larger pressure jump and more curved pressure drop closer matching ZND theory. If the conditions for detonation are improved for a rotating detonation wave in a sawtooth mode, the detonation wave can move to a more stable mode, showing areas for mode control. [37]. The Single-wave mode resembles the stable detonation mode in annular RDEs, where a stable detonation wave occurs, reaching at least 70% to C-J velocities and pressures. Most RDEs operate in this mode [16]. The two dominant peak single wave (TDPO) mode, discovered in 2017 by Zhang et al. [59] describes a mode where two similar peaks are found within the same detonation wave, this has so far only been found in hollow RDEs with nozzles, thereby leading to the conclusion that the other peak, was due to the oblique shockwave being reflected off the nozzle [59]. This mode has the same pressure and detonation velocities as the single wave-modes but differs as it occurs at better conditions for detonation, and thereby is a stronger detonation wave, which then produces a stronger oblique shockwave, making the reflected wave more noticeable in the pressure readings. The two-wave mode occurs from a

spontaneous transition from one wave to two. These produce a frequency higher than the single detonation wave, but with a much lower pressure gain, and occurs only at high contraction ratios, and close to stoichiometric conditions [37, 59]. The number of detonation waves can increase as the mass flow increases, but this effect is dependent on the injector and RDE geometry and flow [44].

Multiple waves have been found to be more stable than a single wave, in what is called a multi-wave mode (also called the two-wave mode), where there can be up to 8 detonation waves [55]. As the number of detonation waves increases the intensity and velocity of all the detonation waves decrease and if stable, all the detonation wave velocities are the same [55]. The number of waves is reliant on the timing of the injection, combustion of the reactants and the extraction of the burnt products, this is seen with Zhang et al. where as the contraction ratio increases, therefore, obstructing and slowing the extraction of the products, the mode switches from a single detonation wave to two [59]. Huang et al. also shows the effects of mixing, with closer the two injectors, spaced apart by the insertion length, thereby having better mixing and a lower wall recirculation region, the mode switches from a single wave mode to a TDPO mode [16]. Jian et al. [45] saw that as the mass flow increased the detonation wave mode improved and that with the geometry explored in the study, to achieve stable rotating detonation a larger mass flow rate was needed, this backs the assertion that the timing is key to the stable operation of RDEs, as the larger mass flow rate decreased the time of injection. The controlled changing through these modes has been empirically tested and validated by Li et al. [10]. Mathematically describing and modelling these effects in some way would go to reduce the reliance on trial and error approaches to developing and studying RDEs.

1.2.3 RDE Analogues

While CFD simulations of RDEs are effective in modelling the core dynamics, detonation wave structures and instabilities seen in empirical RDEs, these simulations are extremely computationally expensive and time-consuming. Additionally, these models do not clarify the major forces, dynamics and physics that produce the unique rotating detonation phenomenon. This is where the recent development of an RDE analogue comes into use. Koch et al. developed an RDE analogue by modifying a detonation analogue to model a self-sustaining non-linear pulse, acting within a 1-dimensional domain with periodic boundary conditions [20]. This model can approximate the rates of gain depletion (combustion process), gain recovery (injection of fresh gases), and dissipation (ejection of burnt gases) [19]. These three effects are modelled by using a simplified version of the Arrhenius equation for the combustion process, an activation function to model the injection process and a loss function to model the energy loss and dissipation for the ejection process. It should be noted before balancing these effects can lead to a stable RDE, but if unbalanced, produces the instabilities and different modes and to quote Kock et al. “the

system bifurcates." [19]. To validate and tune the model, data was gathered from an annular RDE by a high-speed camera, where it was assumed that the brightness of the light emitted was proportional to the intensity of the combustion. This data was flattened into a 1D sample of the detonation to allow comparison to the model with the parameters tuned to the results. A large number of key insights were found, that by conducting Hopf bifurcation analysis on the system, the bifurcation diagrams produced provide operability maps, for a particular engine, allowing the stability limits and the position within these limits to be found, this profound finding shows a possible direction for the development of control systems and models for RDEs. In addition, it was proved that the "multi-scale physics is unique and fundamental to the rotating detonation engine" [20] showing that there is a uniqueness to RDEs and a differentiation from traditional detonation and that the assumption of a complete discontinuity of a detonation wave is wrong in the case of an RDE. This is because of the periodic nature of RDEs and the often issue of the burnt gases, mixing with and affecting the fresh reactants, thereby breaking the discontinuity. Finally, though it should be noted that though this is not mentioned by Koch et al., the model itself poses opportunities for its use within the RDE design process as the system is a rather simple 2-component coupled partial differential equation system, if the parameters were somehow directly linked to the real world parameters directly, it could stand in and replace the need for a significant amount of empirical and CFD experiments and help direct the development towards a stable design that balanced the timing of the injection, combustion, and ejection. Furthermore, the model was only tested on an annular RDE and more research is needed to see whether it is applicable to hollow RDEs.

1.3 RDE Design

1.3.1 Design

The most pressing factor in rotating detonation propagation is the mixing and injection of reactants within the injection region [8]. Within annular RDEs the height of this injection region can be used as a parameter can be used as a stability criterion with the stable range determined ultimately by the detonation cell size in conjunction with the pressure ratios between the fuel and oxidiser and chamber [8]. The detonation cell size can be related to the initial conditions of the injected mixture i.e., temperature, pressure, equivalence ratio, and reactants used [30]. This relationship between detonation cell size and annular RDE stability and performance is still used in current research and RDE design [3]. To this day only empirical methods have been found to directly relate the initial conditions to detonation cell size, this falling within the larger issue of lacking mathematical descriptions of detonation phenomenon, such as RDEs and the DDT process [30]. Furthermore, if found it would have benefits for both deflagration and detonation based

engines in all applications, as noted earlier the similarities to combustion instabilities within deflagration-based rocket engines, specifically high-frequency combustion instabilities [3]. If a sufficient mathematical model were found it would accelerate the design process and safety of all combustor applications. Little work has been done on the nozzle design process for RDEs, especially in the field of optimising nozzles [45], though as noted before they are crucial for the operation of hollow RDEs only a single piece of literature could be found by the author on this topic, specifically a Master's thesis by Mark C. Schnabel [39]. In their thesis, the Angelino nozzle design [5] method was automated within MATLAB and simulated within Ansys CFX. This approach assumed purely axial flow and unsteady throat conditions. These assumptions while not incorrect, especially in the case of annular RDEs, it doesn't account for the plume effects on optimisation, which can drastically affect performance [43] or modelling an RDE itself including the nozzle. Additionally, little attention has been paid to relating RDE design to the timings between the injection, combustion, and ejection processes which Koch et al. suggested were core to the engine operating [20]. A significant majority of studies have explored the operating spaces of RDEs, varying each parameter, from contraction ratio of the nozzle [2], equivalence ratio [58] and mass flow rate [49]. This if enough resources and time, result in an optimised and technically verified design [47], but the overall methodology is slow, time-consuming, and resource-intensive. In addition, this method doesn't produce findings that are transferable to other designs, as if the fuel is changed or injector scheme altered the entire process needs to be started again. The method doesn't often produce universal findings about detonation or RDEs. This is an area of concern within the field of RDEs where more research is needed.

1.3.2 Injectors

To allow RDEs to continuously operate, fresh reactants need to be fed into the chamber to create an unburnt reactants wedge. This wedge is created by the injection and mixing of reactants. This is achieved by the use of an injector, of which the performance and design is key to the functionality of the engine [8]. RDE injectors have to operate in a highly volatile and dynamic environment, being required to inject and mix the reactants in incredibly short periods (>1.5e-4s), while also providing a uniformly adequately mixed fuel and oxidiser, with minimal combustion products left from the prior cycle. Within the field of RDE research, three different injection methodologies have been developed to solve this problem, premixed injection [4], pintle injectors [16] and the semi-impinging injector method [13].

The intention behind the premixed injector method is to avoid the issue of mixing within the detonation chamber, by mixing the reactants before injection. This has been found to work, only with lower sensitivity reactants and in very specific operating conditions [4], with the risk of flashback with H2/Air reactants too great for feasible use, and the range between blowoff and flashback with C2H4/Air remaining stubbornly small. Though ethylene and air are significantly

less sensitive to detonation than hydrogen and air this leaves premixed injection largely technically unviable. Furthermore, the geometry of the plenum dividing the mixing region from the detonation chamber has been found to increase the risks of auto-ignitions and counter-rotating detonation waves, which negatively impact the operation of the engine [1].

As noted earlier the combustion instabilities found in rocket engines have similar characteristics to detonation waves. As these instabilities have not been found in engines that make use of pintle injectors it was conjectured that they might have a stabilising effect on hollow RDEs [16], this empirically explored by Huang et al. [16, 57]. This research has been promising within one case producing a %C-J velocity of 101.25% [16]. Though a promising injection methodology the current research is largely focused on relating the findings with the pintle injector to the combustion instabilities than with using the injector method to bring about the technical viability of RDEs and maximising the efficiency and stability of RDEs.

The most studied and the primary injector method is the impinging injector method [8]. Building on the simplicity of the manufacturing of impinging injectors, often combined with an annular slot injector [3, 45] and choked to reduce the coupling between the detonation wave and injected reactants [1], this method provides a basis of RDE research. Further efforts have been made to develop the impinging injector approach such as the semi-impinging injector (SII) method [13, 14]. The semi-impinging injector method seeks to improve on the original impinging injector with the addition of an additional degree of freedom, by rotating the injectors around each other in addition to the jet impingement angling. This additional degree of freedom to the injector allows the off-centring of the two colliding flows combining both the jet and sheer mixing [13]. It was found that this combination of jet and sheer mixing performed better in mixing H2/O2 reactants in comparison to purely jet or shear mixing, with an approximately 5% and a 17.5% improvement in mixing efficiency respectively with a periodic arrangement of injector elements [13].

1.4 Future Research and This Thesis

Briefly summed up, the future direction of RDE research is on the technical and commercial viability of the technology as a platform for more efficient rocket engines, and turbine engines. For this technology to become a viable product, concerns over the controllability and control systems would need to be addressed though progress has been made with the RDE analogue [29], to the author's knowledge no study has yet come out implementing the suggested control system on an actual RDE. Additionally, regulatory concerns over noise and emissions may further hinder the commercial implementation of RDEs, especially within turbine usage, as little research has been done on NOx production [1] as detonation produces the extremely high temperatures required for NOx to be produced when air is used as the oxidiser, this factor needs exploration as NOx pro-

duction in engines is heavily regulated. The area of optimisation of RDEs and RDE components is severely lacking, as the optimisation process would be a key part of any engine development cycle, with the full benefits and gains of an optimised engine still, not understood [45]. Within the optimisation of components, optimisation of RDE injectors is of the highest importance, given the most important area is the mixing of reactants [8].

While researching and implementing the SII method during the author's Bachelor's degree thesis, there was an issue of the injector legs intersecting and the spacing for the fuel and oxidiser plenums being too tight, these issues were also found by Gaillard [13]. Within Gaillard et al. these issues were solved by arbitrarily rotating the injector legs around the centre of the injector cell at the cost of a more inefficient mixing process. This solution to the author seemed to be an unsatisfactory solution to a pressing issue in the method. This inspired the method developed and described in this thesis called the Modified Semi-Impinging Injector (MSII) method. Iterating on the original SII method by adding an additional degree of freedom to angle the injector perpendicular to the SII's two degrees of freedom. This built off the original SII method development process, where a degree of freedom was applied to the impinging injector approach, yielding a more efficient injector and optimum design, repeating this step could have the same effect of improving the mixing efficiency while at the same time angling the injector legs away from each other solving the issue of the legs intersecting. Additionally, no SII or MSII method has yet been applied to hollow RDEs, this being another gap in the literature. Firstly, the new method would need to be quantified, and validated as better than the original SII approach, but if successful, designing and optimising an injector with this method and then testing it on an RDE could provide partly a solution to both the lack of research on the optimisation process and design stages for RDEs, solving a gap in the literature. Therefore, in this thesis, the Modified Semi-Impinging Injector method will be explored and optimised in comparison to the Semi-Impinging Injector method. Then the MSII method was applied to a hollow RDE to find its effect on the performance.

Part I

Modified Semi-Impinging Injector Methodology, Design and Optimisation

Chapter 2

Methodology

2.1 Modified Semi-Impinging Injector Methodology

The MSII method continues the assumptions of the SII method, where the area taken up by the injector cell is comparatively minor to the overall injection area of the engine. The geometry of the mixing volume is that of a cuboid to simplify the geometry as seen in Figure 2.1. The injector legs feed into the centre of the mixing region. The key parameters that define the geometry are:

- a the width of the mixing region
- b the breadth of the mixing region
- L the height of the mixing region
- l the length of the injector legs
- d_1 the diameter of the fuel inlet
- d_2 the diameter of the oxidiser inlet
- α the angle of the injector leg with respect to the y-axis in the x-y plane
- β the angle of the injector leg with respect to the x-axis in the x-z plane
- ϵ the angle of the injector leg with respect to the y-axis in the y-z plane

With a, b, d_1, d_2, α, β, and ϵ required to calculate the centring of the bounding box that surrounds the intersection point between the injector legs and the mixing region (see Figure 2.2a) the cross-section of the injector legs often is elliptical with the greater the angle α the greater the eccentricity. With the additional degree of freedom acting on the cross-section (i.e.,

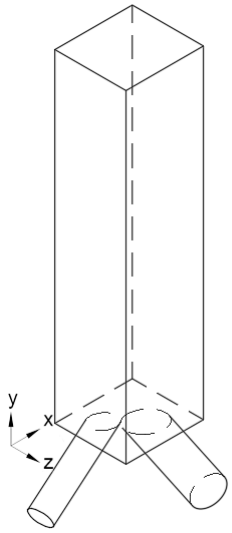
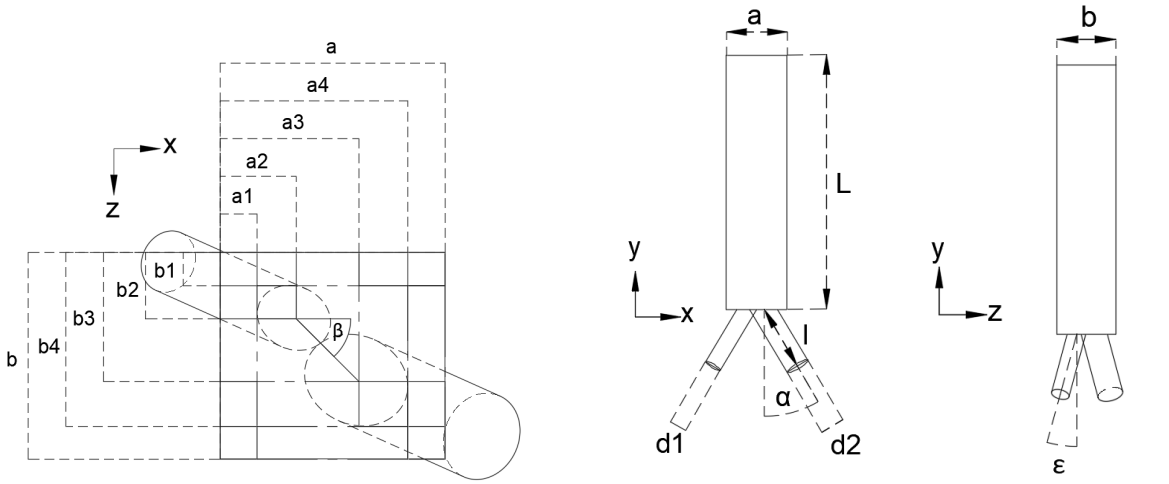


Figure 2.1: Isometric view of an semi-impinging injector



(a) Top side view (b) Right side view (c) Right side view

Figure 2.2: Injector geometry

ϵ) increasing this angle further works to increase the eccentricity but also rotate the major and minor axis of the ellipses. This is due to the angling of both degrees of freedom act to create two theoretical separate and perpendicular ellipses with the resulting cross-section being the sum of these two ellipses resulting in the increased eccentricity and rotated axes. This all has an effect on the centring of the injectors and requires the following method to calculate: For each inlet diameter, given the calculation of the ellipses requires the radii, therefore the radius of the minor axis for each inlet is calculated.

$$r_{1,2} = \frac{d_{1,2}}{2} \tag{2.1}$$

With then the diameter of the major axis, required to take into effect the increase in length due to the angling in both DOFs applied to each inlet and then the radii calculated for the major axis.

$$D_{1,2} = \frac{\frac{d_{1,2}}{\cos(\alpha)}}{\cos(-\epsilon)} \tag{2.2}$$

$$R_{1,2} = \frac{D_{1,2}}{2} \tag{2.3}$$

To find the rotation angle of both the major and minor axes μ is calculated. As the angling applied to both inlets is the same, the rotation of the axes is the same.

$$\mu = \tan^{-1}\left(\frac{\sin(-\varepsilon)\sin\left(\frac{\pi}{4}-\alpha\right)}{\cos\left(\frac{\pi}{4}-\alpha\right)}\right) \tag{2.4}$$

In finding the effect of the off-centring of the injectors, with angle β, while still continuing the one point of contact between the two injector cross-sections, by modifying the equation of the polar radius of an ellipse, the distance between the two injector cross-sections can be found while considering the rotating of the ellipses and the increased eccentricities. This is applied to both injectors with the full distance between found by summing both resulting radii.

$$r_{f,ox} = \frac{R_{1,2}r_{1,2}}{\sqrt{\left(r_{1,2}\cos\left(-\beta-\varepsilon\right)\right)^2 + \left(R_{1,2}\sin\left(-\beta-\varepsilon\right)\right)^2}} \tag{2.5}$$

$$r_3 = r_f + r_{ox} \tag{2.6}$$

The distance r_3 calculates the distance between the centres of each ellipse with this distance broken down into its resulting x and y components to allow the ellipses to be offset from one another while still remaining tangent at a single point.

$$r_x = r_3\cos\left(-\beta\right) \tag{2.7}$$

$$r_y = r_3\sin\left(-\beta\right) \tag{2.8}$$

The equation of an ellipse is modified to incorporate the rotation and then the derivative calculated. The derivative and equation of the cross-section ellipse are equated to zero, and then

the intersection points are found to find the maximum x and y positions to then finally create the bounding box around the injector cross-section which is finally centred from the origin point in the top left corner as viewed in Figure 2.2a. From there the centring of the injector can be calculated.

$$f(x,y)_{1,2} = \left(\frac{xcos(\mu)+ysin(\mu)}{R_{1,2}}\right)^2 + \left(\frac{-xsin(\mu)+ycos(\mu)}{r_{1,2}}\right)^2 - 1 \tag{2.9}$$

$$\frac{d}{dy}\left(f(x,y)_{1,2}\right) = \left(\frac{2\cos(\mu)(xcos(\mu)+ysin(\mu))}{R_{1,2}^2} \pm \frac{2\sin(\mu)(-xsin(\mu)+ycos(\mu))}{r_{1,2}^2}\right) \tag{2.10}$$

- x_{fmax} the absolute maximum x value of the fuel injector
- y_{fmax} the absolute maximum y value of the fuel injector
- x_{oxmax} the absolute maximum x value of the oxidiser injector
- y_{oxmax} the absolute maximum y value of the oxidiser injector

Equations 2.11 to 2.15 calculate the positioning of each of the injector locations within the injector cell as seen in Figure 2.2a.

$$l_a = r_x + x_{fmax} + x_{oxmax}, \;\; l_b = r_y + y_{fmax} + y_{oxmax} \tag{2.11}$$

$$a_1 = \frac{a-l_a}{2}, \;\; b_1 = \frac{b-l_b}{2} \tag{2.12}$$

$$a_2 = a_1 + x_{fmax}, \;\; b_2 = b_1 + y_{fmax} \tag{2.13}$$

$$a_3 = a_2 + r_x, \;\; b_3 = b_2 + r_y \tag{2.14}$$

$$a_4 = a_3 + x_{oxmax}, \;\; b_4 = b_3 + y_{oxmax} \tag{2.15}$$

2.2 Simulation Methodology

In modifying the SII method, a similar meshing, simulation, and post-processing approach was taken to the original methodology. Following reference [13] the mixing region is meshed with a cubic mesh using a 5×10^{-5}m element size, while from a height of 12.5 mm from the base of the mixing region the mesh is coarsened to 1×10^{-4}m. Though the method diverges in that a single 5×10^{-5}m high layer of tetrahedral cells occurs at the base of the mixing region, given the large range of often awkward angles the injector legs come into the mixing region, so to keep the quality of the cubic mixing region mesh, this layer is placed. The legs are meshed with an inflation mesh by the injector leg walls meshed with tetrahedral elements using the same 5×10^{-5}m element size, using the default growth rate of 1.2, a total thickness of 1×10^{-4}m and 5 layers applied. Again, in divergence with the original method, and to further develop the flow inside the injector legs, "o" type meshing is applied, with the inner square region, having sides, half the diameter that of the injector leg it is applied to. The total element count comes to between 8×10^5 to 1.1×10^6 and the average y^+ value is 5.146.

Given the finding that the periodic layout of SII method injector elements gives a significant improvement in mixing efficiency over symmetric layouts [13], the decision to assume a purely periodic layout was chosen. As a different CFD software was used in the simulation process, the naming of the boundaries differs, but the same type and processes are applied. The boundary types are as shown in Figure 2.3, with the red indicating (total) pressure inlets, blue adiabatic, no-slip walls, green the periodic boundaries to replicate the periodic element layout and orange the pressure outlet. The inlet pressures were calculated by subtracting the operating pressure from the total inlet pressure.

The boundary conditions were as follows matching Gaillard et al. [13]:

- An operating pressure 100kPa, to match the pressure outlet's pressure.
- An inlet pressure for each injector (due to similar specific heat ratios) of 37.8 kPa
- An inlet and outlet temperature of 300K, to allow flow in each injector leg to reach Mach 0.7.
- An outlet fuel mass fraction of 0.1111, and oxidiser mass fraction of 0.8889, with the simulation initialised with the outlet conditions.

a	b	d_1	d_2	α	β	ϵ	l	L
3.58 mm	3.29 mm	1 mm	1.41 mm	30°	45°	0°	4 mm	15 mm

Table 2.1: Table of design values for reference from Gaillard et al. [13]

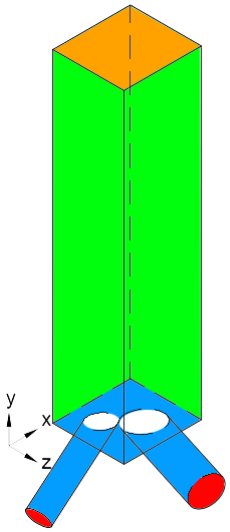


Figure 2.3: Coloured isometric view of injector geometry displaying the boundary types as different colours.

To provide a reference and point of validation, the results and simulation methodology was based on Gaillard et al. [13], notably the periodic semi-impinging case, the geometry design values for such found in Table 2.1. The boundary conditions referred to in Figure 2.3 are as listed, red being the pressure inlet, blue the adiabatic no-slip wall, green the periodic boundaries and orange the pressure outlet. In the simulation methodology, the viscous model used was the Smagorinsky-Lilly Large-Eddy Simulation (LES) method, with the constant Cs = 0.1. This is as the LES method provide the greatest detail turbulent simulations which are key to fully modelling the turbulent mixing inside the injector. The calculation methods for each of the components of the species were as follows, density-ideal gas, specific heat - mixing law, thermal conductivity, and viscosity - mass-weighted mixing law, mass diffusivity and thermal diffusion coefficient - kinetic theory. The selection of the inlet diffusion, diffusion energy source and thermal diffusion was made to enable diffusion at the inlet, to fully model the large Mach numbers involved and therefore likely large changes in temperatures over the injector allowing the effect on enthalpy due to the diffusion and mixing of the reactants and temperature on diffusion would better match the realities of mixing the reactants. Additionally, the reasoning behind the use of the mixing law, mass-weighted mixing law, ideal gas and kinetic theory was due to the compressible, multi-component flow, with large temperature variations that would affect the diffusion rate and use of hydrogen an extremely light molecule. The solution method was as follows, the pressure-velocity coupling scheme was the coupled method, due to the author's prior experience with the meshing of similar injector geometries, and the creation of bad elements, making the use of the coupled method justified as it is more stable. Furthermore, given the use of LES methods, the transient formulation was the bounded second-order implicit scheme. The time steps were broken down into two sequences, the first sequence of a time step of 1×10^{-6}s for a total of 1ms, to fully

develop the flow in the mixing region and a second time-step of 1×10^{-7}s for a total of 0.1ms to fully refine the flow, this second step differs from the original study in the time step of 1×10^{-7}s vs 1×10^{-8}s, given the number of simulations required for the parameter study, to simplify and speed up the simulation process.

As Ansys Fluent was used through this study, a number of drawbacks were found, one of them being the inability of Ansys to calculate the mass fraction fluctuations, without the use of user-defined functions which were beyond the abilities and time scales available to the author at the time. Therefore, to solve this issue to collect the data required to calculate the injector performance, discrete datasets were taken at intervals during the refinement stage of the simulation. The data was sampled at the beginning of the stage, and during 1×10^{-5}s periods, leading to 11 sets of data captured. Within each dataset, the data within a plane perpendicular to the y-axis were used to select the data at set positions within the geometry. To calculate the mass fraction fluctuation equation 2.16 was applied where Y' is the mass faction fluctuation Y the mass fraction, and the overbar denoting a time-averaged value. Equation 2.17 was used to calculate the equivalence ratio where the subscript denotes the reactant W the molar mass of the reactant and φ the equivalence ratio.

$$Y' = \left(\overline{Y} - Y\right) \tag{2.16}$$

$$\overline{\varphi} = \frac{M_{O_2}\overline{Y_{H_2}}}{2M_{H_2}\overline{Y_{O_2}}} \tag{2.17}$$

In using discrete data, the post-processing equations were obtained from Gaillard et al. [13]

$$\eta_{mix} = \frac{\iint_{S_y}\overline{Y_{O_2}}min(\overline{\varphi},1)d\overline{\dot{m}_y}}{\iint_{S_y}\overline{Y_{O_2}}max(\overline{\varphi},1)d\overline{\dot{m}_y}} \approx \frac{\sum\overline{Y_{O_2}}\min\left(\overline{\varphi},1\right)\overline{\dot{m}_y}}{\sum\overline{Y_{O_2}}\max\left(\overline{\varphi},1\right)\overline{\dot{m}_y}} \tag{2.18}$$

$$\sqrt{\overline{Y'_{H_2}{}^2}} = \frac{\iint_{S_y}\sqrt{\overline{Y'_{H_2}{}^2}}d\overline{\dot{m}_y}}{\iint_{S_y}d\overline{\dot{m}_y}} \approx \frac{\sum\sqrt{\overline{Y'_{H_2}{}^2}}\,\overline{\dot{m}_y}}{\sum\overline{\dot{m}_y}} \tag{2.19}$$

$$\overline{P_{t,y}} = \frac{\iint_{S_y}\overline{P_t}d\overline{\dot{m}_y}}{\iint_{S_y}d\overline{\dot{m}_y}} \approx \frac{\sum\overline{P_t}\overline{\dot{m}_y}}{\sum\overline{\dot{m}_y}} \tag{2.20}$$

$$\overline{P_{t,inj}} = \frac{\dot{m}_{O_2}\overline{P_{t,O_2}} + \dot{m}_{H_2}\overline{P_{t,H_2}}}{\dot{m}_{O_2} + \dot{m}_{H_2}} \tag{2.21}$$

Degrees of Freedom		
α	β	ϵ
15°	0°	0°
30°	45°	15°
45°	90°	30°

Table 2.2: Table of explored parameter values in the optimisation study

$$\eta_{rec} = \frac{\overline{P_{t,y}}}{\overline{P_{t,y}}} \tag{2.22}$$

Where η_{mix} is the mixing efficiency, η_{rec} the pressure recovery efficiency, and $\sqrt{\overline{Y'^{2}_{H_2}}}$ the mass averaging of the RMS of the time fluctuations of the fuel mass fraction (referred to from here on out as the mixing consistency). The mixing efficiency equation was developed by T. Gillard as a means of quantifying the quality of the injected flow over the whole of the mixing region while normalising the data between 0 and 1. The pressure recovery efficiency is a ratio of the mass averaged total pressure in a given y section to the mass-averaged total pressure at the inlets. This allows the analysis of the total pressure losses to either wall friction in the injector legs and to mixing, with the less total pressure loss, the more desirable the injector. Finally, the mixing consistency was used as it provides a quantified measure of the flow within the mixing region matching the requirements for efficiency and stable RDE operation with there being a reliable, constant, and consistently well mixed injected flow. The lower the fluctuations, in combination with the mixing efficiency can show how an injector meets these criteria.

To optimise the injector geometry parameter study was conducted. This was obtained by varying each of the three degrees of freedom as described in Table 2.2, while keeping the mixing region geometry and injector leg length and diameters constant. To simplify the method, the same values for the geometry, boundary conditions and mesh, were used as described before. Another condition on the degrees of freedom was made, that the angling of the DOF's was not such that they would result in the bounding box exceeding the mixing region. This led to 27 simulations being conducted with each combination of the angles simulated and the results found.

2.3 Automating Calculations

As any injector design methodology would be used in an iterative manner in any practical design cases and be used in a wide range of different reactants, and initial conditions, automating the calculation process would drastically improve the efficiency and reliability of the design process. Additionally integrating the calculation of simulation initial conditions would further improve

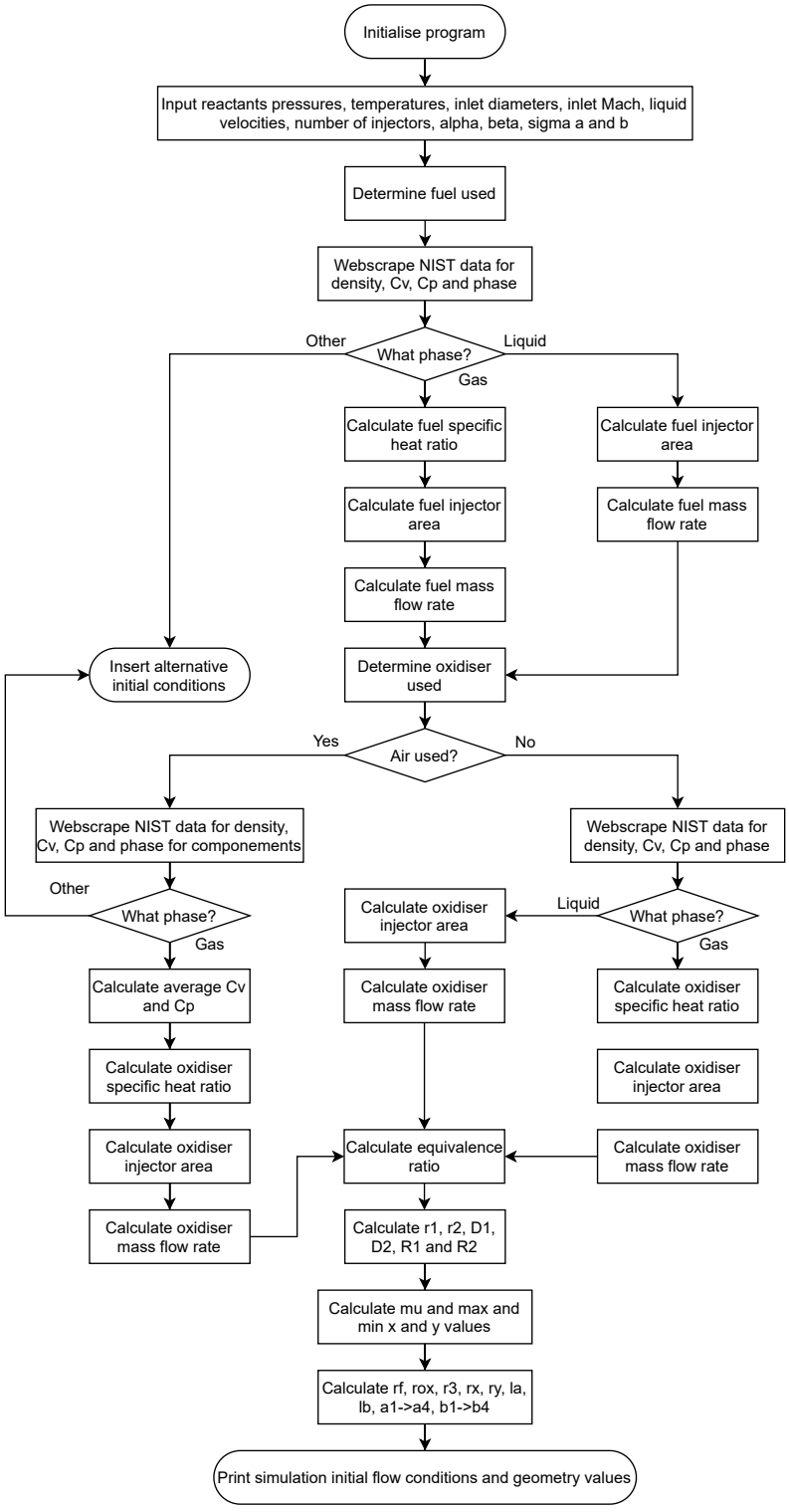


Figure 2.4: Flowchart of the design and initial conditions code.

the efficiency of the process. Therefore the design and initial conditions calculations were programmed and automated follwing the flowchart in Figure 2.4 following the design methodology described in section 2.1 and initial conditions described further on in this section. The calcula-

Reactant	Formula (1 Mole)	Molar Mass	Specific Gas Constant
Hydrogen	H2	0.00201588	4124.48291
Oxygen	O2	0.0319988	259.83670
Methane	CH4	0.016425	506.20777
Ethylene	C2H4	0.0280532	296.38197
Air (dry)	0.79N2 + 0.21O2	0.0296058	280.83898

Table 2.3: Table of reactants available within the program

tions were coded in Python on Google Colab's cloud jupyter notebooks as it is an open and simple platform to program within a browser while making use of Google Cloud Platform's spare computing power. Additionally, as the post-processing calculations would also be automated, its installation of Pandas would allow the easy manipulation and modification of the instantaneous data produced by the simulations.

To allow the code to design injectors for most RDEs it would have to work with a variety of reactants. Within Anand et al. [1] the reactants where successful rotating detonation has been achieved are listed, the most common reactants where, oxygen, air, hydrogen, methane, and ethylene. The molar masses and specific heats were obtained from the National Institute of Standards and Technology (NIST) chemistry workbook [36] because of NIST's history of reliability and accuracy as a source of chemistry data. Using the molar masses, sourced from NIST and equation 2.23, equation 2.24 is used given the specific gas constant = $8.31446261815324\left(\frac{J}{Kmol}\right)$ to calculate the specific gas constants shown in Table 2.3. It should be noted that M_i referrers to the component molar mass, x_i the component mass fraction, R the gas constant, $C_{v,i}$ and $C_{p,i}$ the component specific heats, and st_i the global stoichiometric ratio.

$$M_{mixture} = \sum x_i M_i \tag{2.23}$$

$$R_{spesific} = \frac{R}{M_{mixture}} \tag{2.24}$$

As the specific heats of any reactant vary with temperature, to allow the code to work with as wide a range of inlet conditions as possible and still have accurate outputs the specific heats were web-scraped off of the NIST chemistry workbook website, along with the phase of the reactant and reactant density. If the reactant was a mixture, equations 2.25 and 2.26 were applied to the components of the mixture to calculate the specific heats. The specific heat ratio if the reactant is a gas, then is calculated using equation 2.27. The stoichiometric ratios were calculated using equation 2.28.

$$C_p = \sum x_i C_{p,i} \tag{2.25}$$

Reactant	Stoichiometric Equations
Hydrogen + Oxygen	$2H_2 + O_2 \rightarrow 2H_2O$
Methane + Oxygen	$CH_4 + 2O_2 \rightarrow CO_2 + 2H_2O$
Ethylene + Oxygen	$C_2H_4 + 3O_2 \rightarrow 2CO_2 + 2H_2O$
Hydrogen + Air	$2H_2 + 3.76N_2 + O_2 \rightarrow 2H_2O + 3.76N_2$
Methane + Air	$CH_4 + 7.52N_2 + 2O_2 \rightarrow CO_2 + 2H_2O + 7.52N_2$
Ethylene + Air	$C_2H_4 + 11.28N_2 + 3O_2 \rightarrow 2CO_2 + 2H_2O + 11.28N_2$

Table 2.4: Table of reactant stoichiometry

$$C_v = \sum x_i C_{v,i} \tag{2.26}$$

$$\gamma = \frac{C_p}{C_v} \tag{2.27}$$

$$st_i = \frac{\left(n_{oxidiser}\right)_{reactants} \left(M_{mixture}\right)_{fuel}}{\left(n_{fuel}\right)_{reactants} \left(M_{mixture}\right)_{oxidiser}} \tag{2.28}$$

The injection area is calculated using equation 2.29, allowing the injector design geometry to be related to the initial flow conditions. The initial flow conditions specified are the total pressures and temperatures, and in the case of liquid reactants flow velocity. The mass flow rate was then calculated with equation 2.30 used for gases reactants and 2.31 for liquid reactants. The gaseous flow was assumed to be isentropic and compressible while the liquid flow was incompressible. Once this process has been applied to both the fuel and oxidiser, the equivalence ratio is calculated using equation 2.32. The geometry is then calculated as shown in section 2.1.

$$S_i = n_{inj} \frac{\pi}{4} d_i^2 \tag{2.29}$$

$$\dot{m} = \sqrt{\frac{\gamma}{R}} \frac{p_t}{\sqrt{T_t}} S M \left(1 + \frac{\gamma - 1}{2} M^2\right)^{-\frac{\gamma+1}{2(\gamma-1)}} \tag{2.30}$$

$$\dot{m} = \rho u S \tag{2.31}$$

$$\phi = st_i \frac{\dot{m}_{fuel}}{\dot{m}_{oxidiser}} \tag{2.32}$$

The post-processing code can be broken down into two sections, the first calculating the non-location specific $\overline{P_{t,H_2}}$ as this is a constant applied to each of the 7 planes taken throughout the

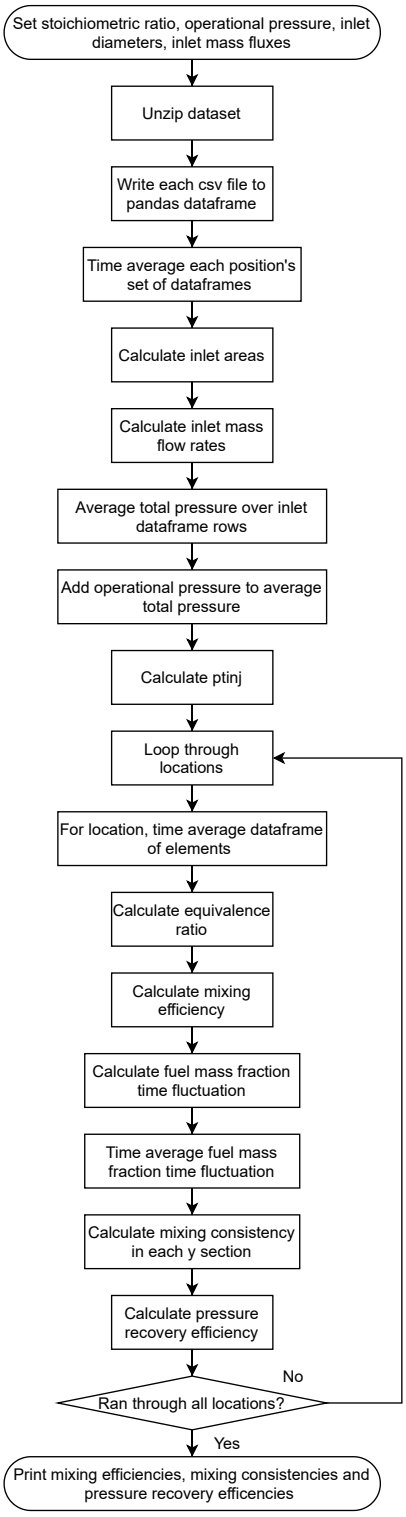


Figure 2.5: Flowchart of the post-processing code.

mixing region. The data time-averaged within this section was the total pressure readings taken at each injector leg inlet these combined with the calculated inlet mass flow rates are used in equation 2.21.

Source	Fuel mass flux $\left(\frac{kg}{m^2 s}\right)$	Oxidiser mass flux $\left(\frac{kg}{m^2 s}\right)$	Equivalence ratio	Fuel mass flow rate $\left(\frac{kg}{s}\right)$	Oxidiser mass flow rate $\left(\frac{kg}{s}\right)$
Gaillard et al. [13]	85.4	340.2	1	6.7073003e-5	5.3120532e-4
Code output	78.1256436	310.586733	1.00417985	6.1359737e-5	4.8496568e-4
Error	9.31109951%	9.53462080%	0.41624519%	9.31109951%	9.53462081%

Table 2.5: Table of comparison of code and reference study values [13]

The second section goes through each mixing region plane, time averages the data, calculating the equivalence ratio using equation 2.33 applied to the fuel and oxidiser resulting in an equation 2.17 and for hydrogen and oxygen reactants. The mixing efficiency is then calculated using equation 2.18. Then going through each time step the fuel mass fraction is calculated using equation 2.16 and then time-averaged throughout the entire flowtime recorded. The root-mean-square of this value is found and averaged over the entire plane as shown in equation 2.19. The pressure recovery efficiency is then calculated by using equations 2.20 and 2.22.

$$\varphi_i = \frac{\left(n_{oxidiser}\right)_{reactants} \left(M_{mixture}\right)_{oxidiser} Y_{fuel}}{\left(n_{fuel}\right)_{reactants} \left(M_{mixture}\right)_{fuel} Y_{oxidiser}} \tag{2.33}$$

To validate the design and initial conditions code, the design inputs for the initial conditions used in T. Gaillard et al. [13] were input and the outputs compared. The error rates were all under 10% validating the code. Due to the scale of the areas used to calculate the mass flow rates from the mass fluxes listed in T. Gaillard et al. [13] are likely the main reason for the % error as any slight change in the number of decimal places used to calculate the inlet areas, would have a significant effect on the mass flow rates given the minuscule scale of the values. The post-processing code would require validating as part of the methodology.

2.4 Methodology Validation

Given the slight deviations from the meshing technique in the reference study, but also the lack of a mesh study in the original study, a meshing study was conducted. Two separate variables were varied, firstly the average y^+ value at the injector leg walls, by varying the number of divisions in the inflation mesh. Secondly the number of elements by varying the element size, while keeping the ratio of the coarsened mesh size to the mixing region mesh size constant. The y^+ values were obtained with the inflation layer numbers of 3, 5 and 10, to obtain y^+ values of 9.93, 5.146 and 1.354, respectively. The element counts were obtained with element sizes of (coarsened region element sizes: mixing region element sizes): 5×10^{-4}m : 1×10^{-4}m, 2.5×10^{-4}m : 7.5×10^{-5}m

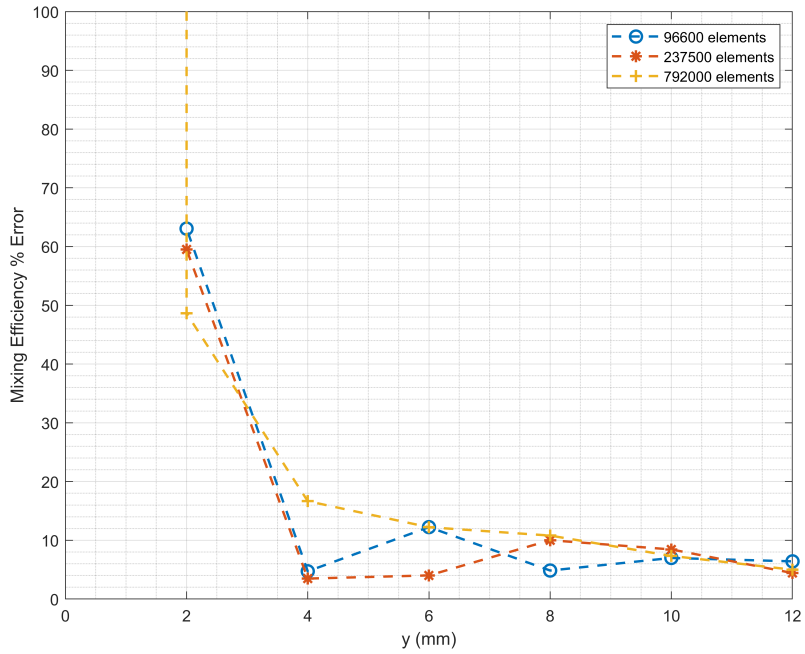


Figure 2.6: Mixing efficiency error against injector height for different element counts.

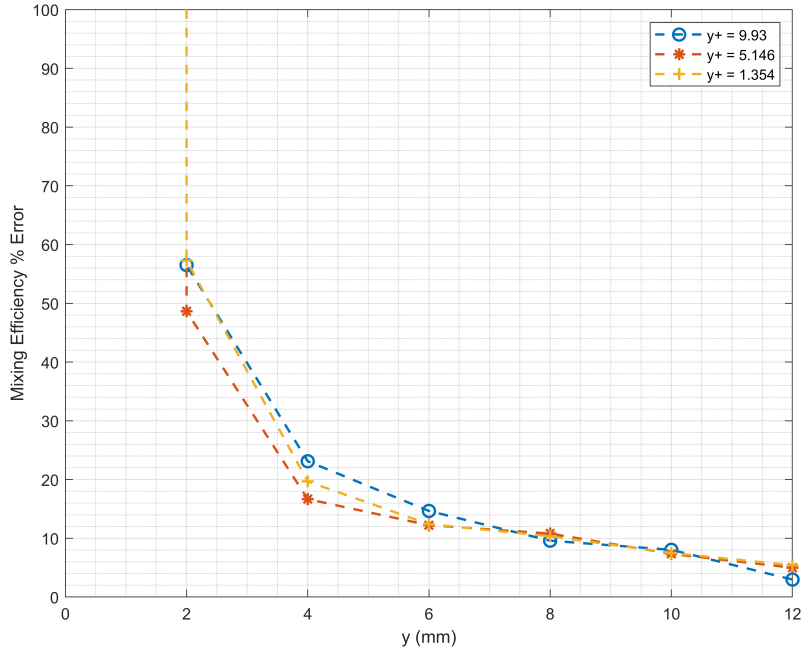


Figure 2.7: Mixing efficiency error over injector height for different average y^+ values.

and 1×10^{-4}m : 5×10^{-5}m, leading to total element counts of 9.66×10^4, 2.375×10^5 and 7.92×10^5 elements, respectively. In varying the y^+ value, it was found that there was little effect on the error over the mixing region, with the y^+ value of 5.146 equating to the given methodology described in the meshing method producing the best results. In varying the element count as the resolution of the mesh decrease as did the stability of the results, partially seen in the error rate, outside the initial mixing region the error rate jumps around, though, given the low resolution, the error rate at points is lower than that of the initial meshing methodology. Given the instability

of the lower resolution mesh results, it was decided to go forward with the original element sizes

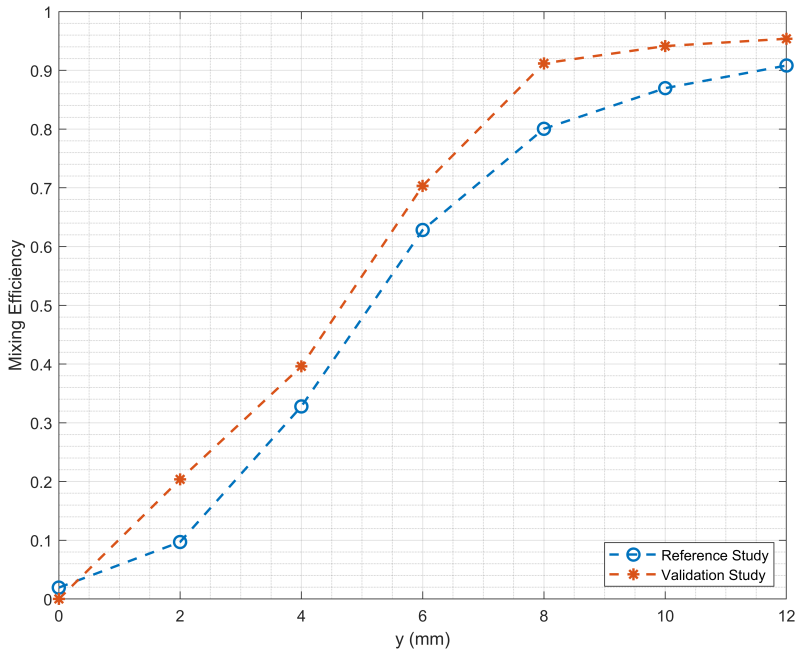


Figure 2.8: Methodology validation study mixing efficiency.

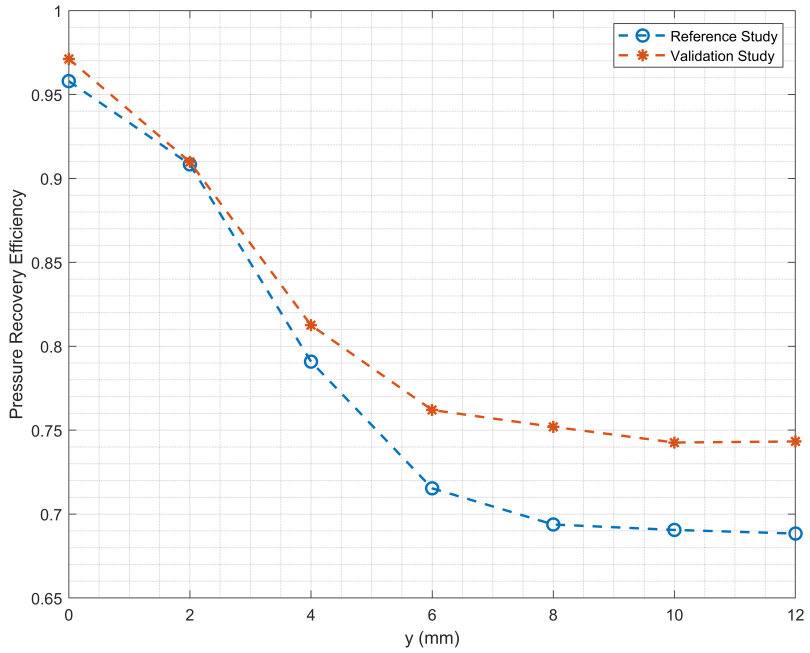


Figure 2.9: Methodology validation study pressure recovery efficiency.

As it was found that the data sampling approach and meshing technique may lead to some instability in the results, a reliability study was conducted. Using the original meshing, simulation and post-processing approach, the same validation study simulation was repeated 5 times and the results were collated. It was found that the mixing efficiency standard deviation error, excluding the error rate at 0 mm height was on average 2.51% with a maximum of 4.57%. The

error rates taken at 0 mm were discounted as any deviation given the extremely small values that occur at that height give unusable and inaccurate error rates for the mixing efficiency and mixing consistency. The standard deviation for mixing consistency error was on average 13.84% with a maximum of 32.48%. The pressure recovery efficiency average standard deviation error rate was 0.27% with a maximum of 0.4%.

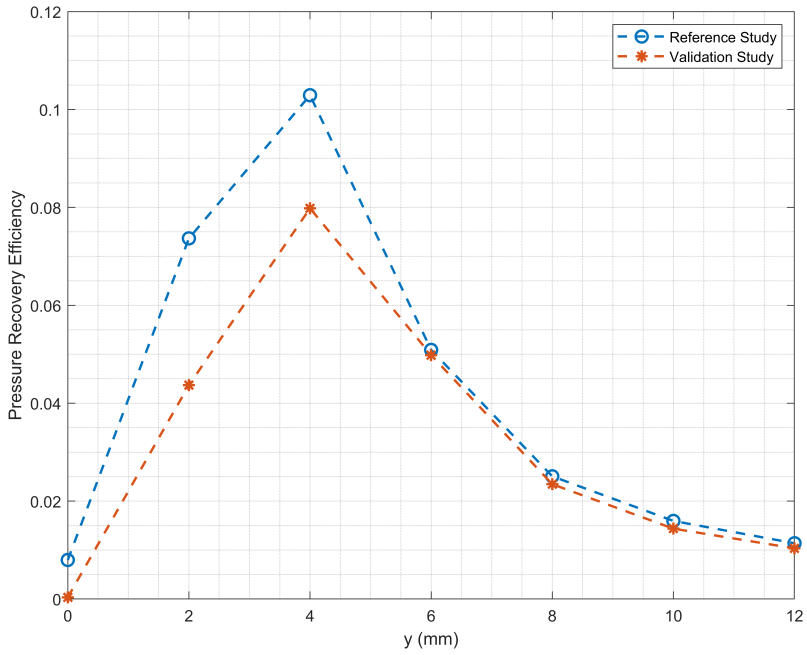


Figure 2.10: Methodology validation study mixing consistency.

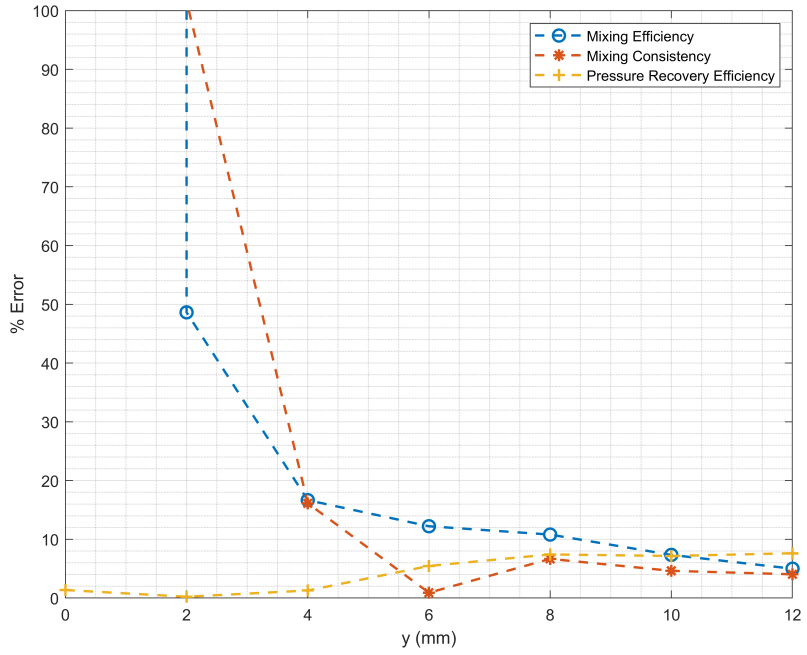


Figure 2.11: Methodology validation study error.

To validate the methodology, a reference point was required. After searching a suitable reference study could not be found, therefore, given the circumstances, the use of a numerical ref-

erence study was considered suitable. The reference study was case 3A in Gaillard et al. [13] where the design details and inital conditions are listed in Tables 2.1 and 2.5 respectively. This was chosen as it was used as the basis of the methodology used in this study and used more refined results in the use of smaller simulation step times in the refinement stage. Furthermore, though Gaillard et al. does not refer to what exact data sampling technique was used, it is likely that it is higher than the approach used in this study, leading to more detailed and refined results. Following the beforehand described simulation set-up, running and post-processing, the results were found and compared to the reference study. The methodology used seems to overestimate the performance of the injector against the reference study, as seen in Figures 2.8 and 2.9, there is a consistently higher mixing efficiency and pressure recovery efficiency though showing a similar loss in total pressure from the injector leg walls, and in the initial mixing phase, from a hight of 4 mm and on, the error rate of 7%. Mixing consistency shows the inverse of this with the initial mixing divergence remaining relatively, likely due to the sampling rate used, in conjunction with the initial mixing region being highly turbulent, therefore the time averaging not fully capturing the mass fraction fluctuations. The turbulence decreases as the high increases. This leads to the mass fraction fluctuation reducing and the data sampling capturing the averaged flow better. Looking at the error rate, it should be noted that for the initial mixing region (up to 4 mm), given the minuscule values any slight deviation leads to significant error rates, even when the divergence between the validation study and reference study results is relatively consistent, as seen in the mixing efficiency case. The error rates once the flow stabilises past the initial mixing region are within the range of $>12\%$. That all being said, given the range of measures taken that would likely reduce the accuracy of the results, that being the reduce refinement flow time set and significantly reduced data sampling the results broadly match those of the reference study, showing the method to be valid.

Chapter 3

Results and Discussion

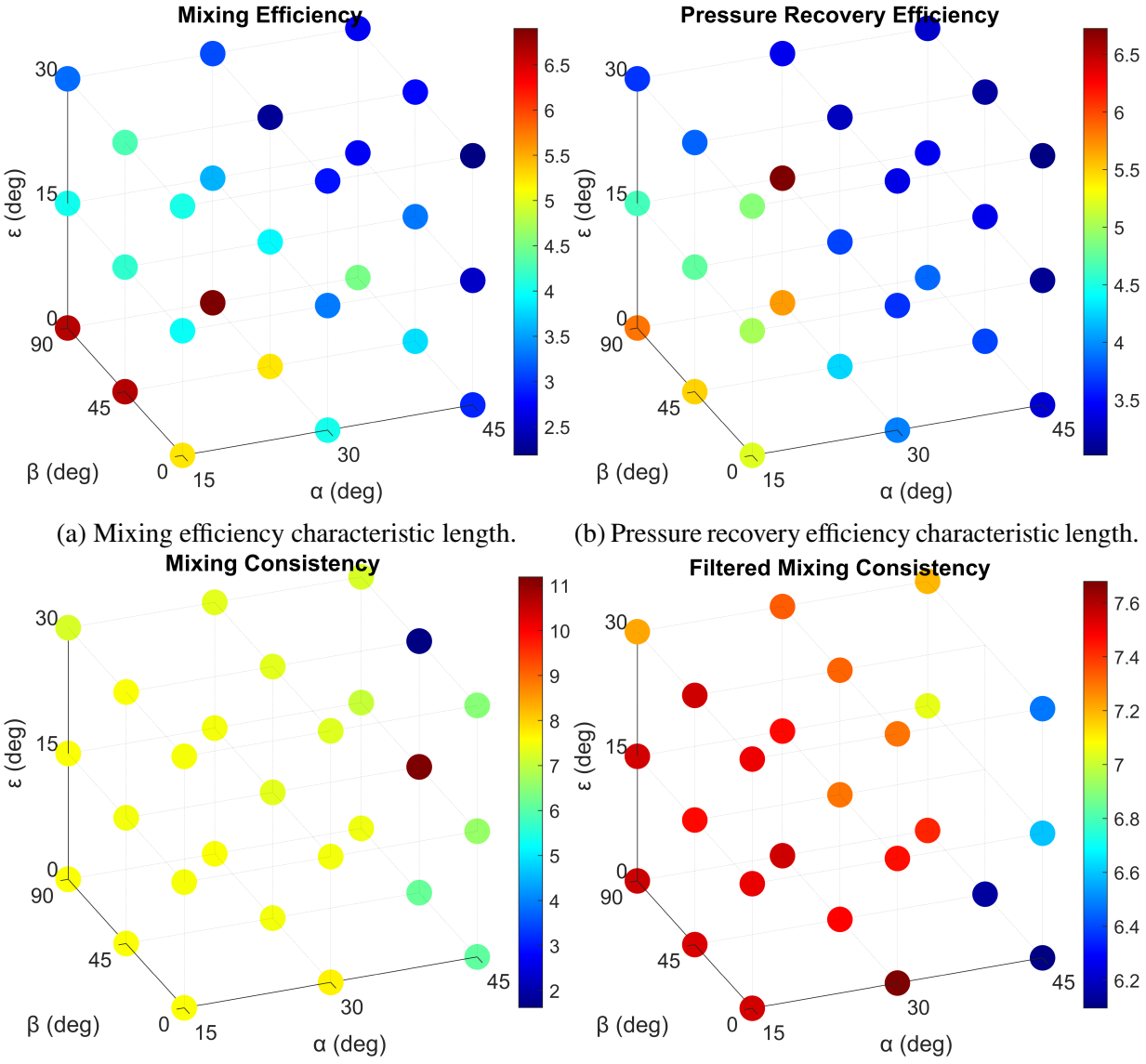
3.1 Post-Processed Results

The post-processed results are plotted to visualise the distribution of resulting flows from the ranges set in the parameter study. The plots can be divided into two sections, the turbulent mixing phase, and the dampening phase. This typically occurs with the initial conditions and set-up used in this parameter study at 6 mm from the base of the injector. As seen in Figures 3.1-3.5 the plots start from a singular point and diverge. The plots then start converging around a similar end value. These two stages are called the mixing phase and dampening phase respectively. It was found that mixing efficiency and pressure recovery efficiency had a more consistent set of plots than mixing consistency, this was likely due to mixing consistency peaks displaying the collision point between the two injected flows. As this was dependent on the angles of the injected flows, it was dependent on the parameter values used and therefore will be less consistent given the large ranges of the parameter study. Within the range of plots, similar patterns of performance and flows are found, with the mixing efficiency tending to converge around 0.9, with the average value found at 12 mm being 0.91762 with a standard deviation of 0.02120. The same result was found in the pressure recovery efficiency, with approximately 3-4% of the total pressure lost to friction within the injector legs, and 21-23% loss of the initial total pressure lost in the mixing process. The main variation found within these two parameters was an analogue of the mixing speed, found in the gradient of the mixing efficiency in the mixing phase. The larger the gradient the faster the mixing, and less distance and therefore time gave consistent mass flow rates. The inverse of this was found for the pressure recovery efficiency as the faster the mixing occurs, the faster the drop in total pressure, as the energy described by the total pressure was used up in the process of mixing the reactants. The mixing consistency average value found at y = 12 mm was 0.01020. It was found that in the pressure recovery and mixing consistency data in some cases diverged from the typical plots. These issues were often small and are possible due to a number of causes, either the simulated flows in these certain cases just significantly diverged from the

norms or possibly user error.

3.2 Characteristic Lengths

Given the difficulty in assessing the effect of changing the angle of the injector's degrees of freedom, there was a need to summarise the performance of each injector to a more manageable form. As the majority of the performance parameter plots converge consistently on a small range of final values and inspired by the use of characteristic time τ in systems analysis a characteristic length was used to define each performance parameter with a single value. The characteristic length was the distance taken to reach 63.2% of the output at a height of 12 mm found by linear interpolation, as the characteristic time was the time taken to reach 63.2% of the final output and use of discrete data. In addition, given the need for RDEs to mix the reactants in as short a time as possible, and the consistent mass flow rates, and therefore flow velocity, the distance is a reliable measure of the speed of mixing. In the case of the mixing efficiency and mixing consistency the shorter the characteristic length, the better the injector. The opposite is found in the pressure recovery efficiency, as the further the total pressure is retained the lower the loss of total pressure. Plotting the characteristic lengths as a 4D scatter plot shows the effectiveness of changing each parameter. In Figure 3.5 the mixing efficiency characteristic lengths are shown. It can be seen that a general trend emerges with the larger the angle α and ϵ, the smaller the characteristic length. This was likely because α and ϵ affect the total magnitude of the angling of the injector leg, and this affecting the collision angle of the two semi-impinging flows. β seemingly has a less consistent effect on the mixing performance, as seen when $\epsilon = 0°$, increasing β consistently increases the characteristic length, whereas when $\epsilon = 15°$ and $30°$, β has a less consistent effect on τ, as in these cases when $\beta = 45°$, the performance drops compared to when $\beta = 0°$ and $90°$, but except for α=30°, $\beta = 45°$, ϵ=30° providing one of the lowest characteristic lengths out of the injectors studied. This inconsistent and varied effect by β is due to beta's use in rotating the two injected flows in, in-between and out of collision with each other, and likely shows the limits of the approach, but also the usefulness of making use of all degrees of freedom within the injector design. The approach taken with the pressure recovery efficiency deviated the method as described above slightly in one aspect, it took the length to reach a pressure recovery efficiency of 0.8, as when using the original method, the output would vary by large amounts due to the use of linear interpolation and the almost flat and sometimes negative gradients at the end of the pressure recovery efficiency plots providing unreliable results. Comparing Figures 3.1a and 3.1b there are a large number of similarities with the larger the angles for α and ϵ the larger the characteristic length. This though is in the context of the pressure recovery efficiency, where the larger the value the better the performance showing that the two parameters are inversely proportionate coinciding with prior research [13] and Figure 3.3.



(a) Mixing efficiency characteristic length.

(b) Pressure recovery efficiency characteristic length.

(c) Mixing consistency characteristic length

(d) Filtered mixing consistency characteristic length

Figure 3.1: Injector geometry

For the mixing consistency the smaller the characteristic length, the shorter the distance for the flow to stabilise and homogenize. Two figures were produced as a result of the $\alpha = 30°$, $\beta = 45°$, $\epsilon = 0°$ and $\alpha = 30°$, $\beta = 45°$, $\epsilon = 15°$, as seen in Figure 3.1c have a characteristic length below 12 mm, this was due to the mixing consistency increasing before the final measurement taken this was possibly due to either irregular flow in the injector, human error and/or statistical error given the data sampling. Removing these two values allows the better visualisation of the results as shown in Figure3.1d. Figure 15 shows only a few trends with generally the best performance resulting from when $\epsilon = 0°$, as increasing ϵ, increases the unsteadiness of the flow, over purely using α to angle the injector, with β having little influence on the steadiness.

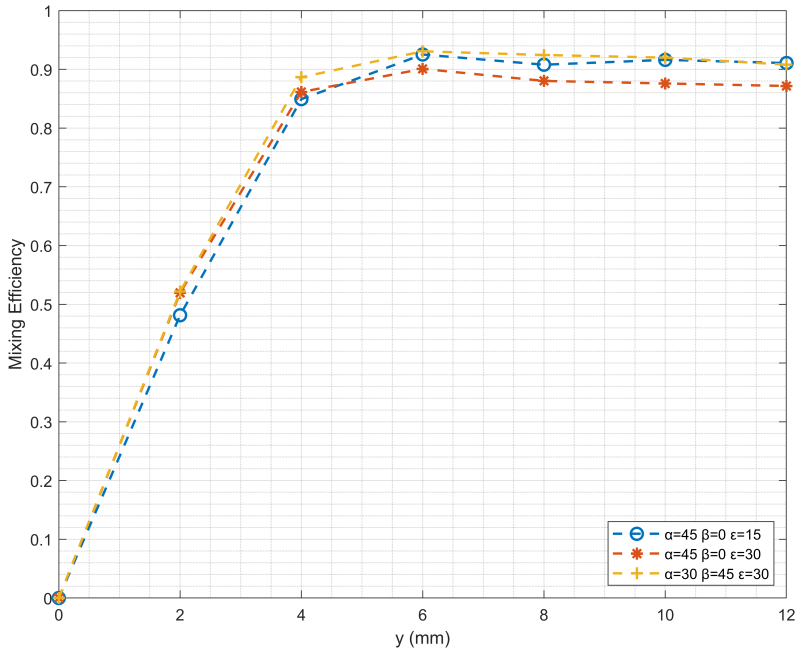


Figure 3.2: Top performing injectors mixing efficiency plots.

With all the results considered mixing efficiency seems to be the most relevant optimisation reward factor as it provides the most insights into optimising the performance. This is shown as the pressure recovery efficiency can be directly related to the mixing efficiency as seen in reference [13] and Figure 3.3 negating the need to measure it, additionally the total pressure drop over the injector remained relatively consistent allowing greater gains to be made in the mixing efficiency at a low cost in pressure recovery efficiency. Additionally, the mixing consistency has few general trends to optimise the injector though can give some insights into the flow in the injector, doesn't provide as much relative performance data compared to the mixing efficiency, which is required for optimisation. Therefore, to optimise the injector, the mixing efficiency parameter seems to be the most adequate. Using mixing efficiency to find the top-performing injector geometries gives us the characteristic lengths of 2.512mm, 2.286mm and 2.184mm for ($\alpha = 45°$, $\beta = 0°$, $\epsilon = 15°$), ($\alpha = 30°$, $\beta = 45°$, $\epsilon = 30°$) and ($\alpha = 45°$, $\beta = 0°$, $\epsilon = 30°$). As seen in Figures Figures 3.2-3.5 there are relatively few differences between the plots, with the largest

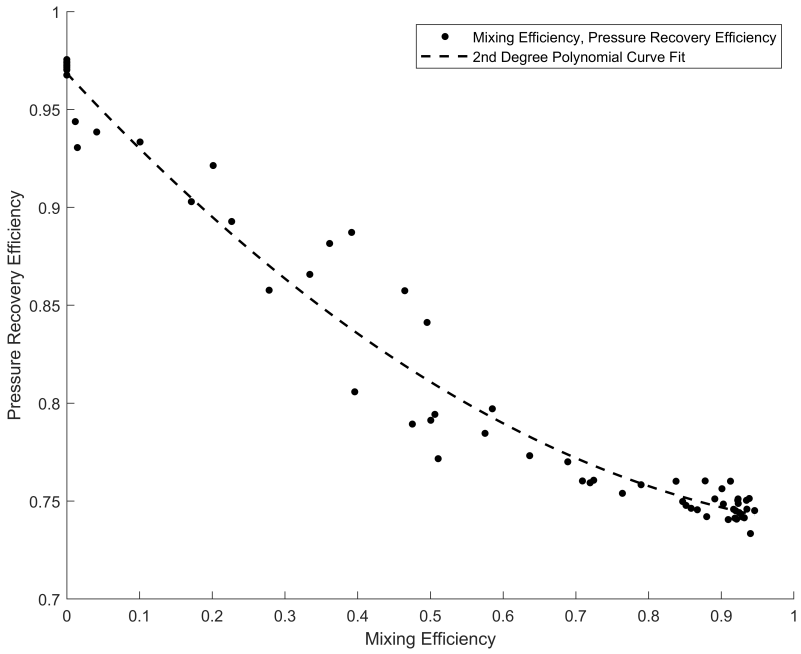


Figure 3.3: Plot showing the correlation between mixing efficiency and pressure recovery efficiency.

divergence seen in the mixing consistency with ($\alpha = 45°$, $\beta = 0°$, $\epsilon = 30°$) having a lower peak value but a higher-end value while ($\alpha = 30°$, $\beta = 45°$, $\epsilon = 30°$) has a plot which was the opposite. Given the similarities, it can be reasonably said that any of the geometries shown in Figures 3.2-3.5 can be classified as the most optimum injector design meeting the goal of the parameter study. In addition, given that the top-performing injectors all make use of the additional degree of freedom it shows the positive effect that adding ϵ has done on the effectiveness of the injector, validating the theory that expanding the degrees of freedom of the SII method would improve performance and opportunities for optimisation. To summarise the key findings in this phase:

- The best parameter to characterise the flow is the mixing efficiency, especially when reduced to calculate the mixing efficiencies characteristic length.

- By using the mixing efficiency characteristic length, the top-performing injector geometries were identified by having the shortest length. The top geometries and their corresponding characteristic lengths were ($\alpha = 45°$, $\beta = 0°$, $\epsilon = 15°$) with 2.512mm , ($\alpha = 45°$, $\beta = 0°$, $\epsilon = 30°$) with 2.184mm and ($\alpha = 30°$, $\beta = 45°$, $\epsilon = 30°$) with 2.286mm. It should be noted that all made use of all the degrees of freedom, validating the Modified SII method as an optimisation approach for cold flow analysis of RDE injectors.

- Within the 4-dimensional space of the degrees of freedom, and characteristic length, For the mixing efficiency, as α and ϵ increased, the speed of the mixing efficiency is increased. β has an inconsistent effect on the mixing efficiency characteristic length, and likely requires a better resolution within the range of angles studied to better find the effectiveness

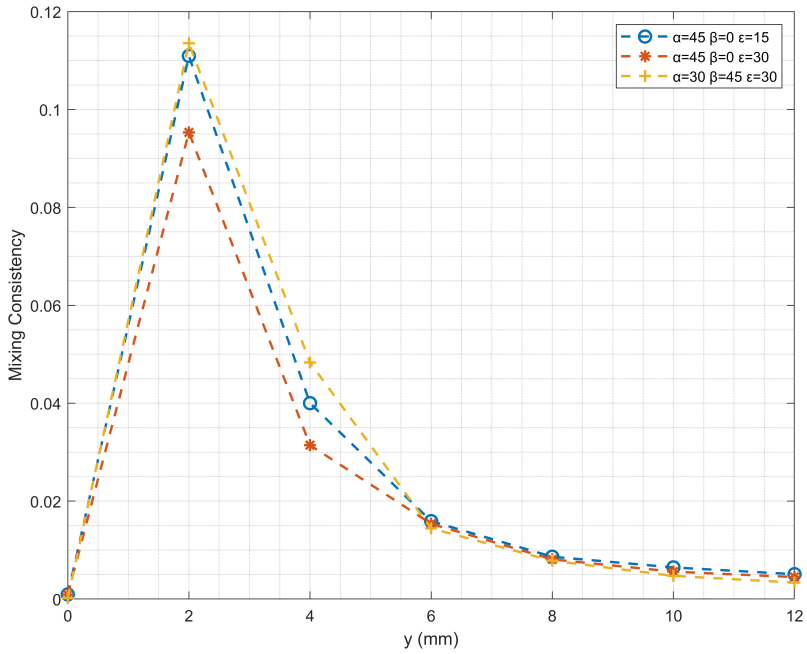


Figure 3.4: Top performing injectors mixing consistency plots.

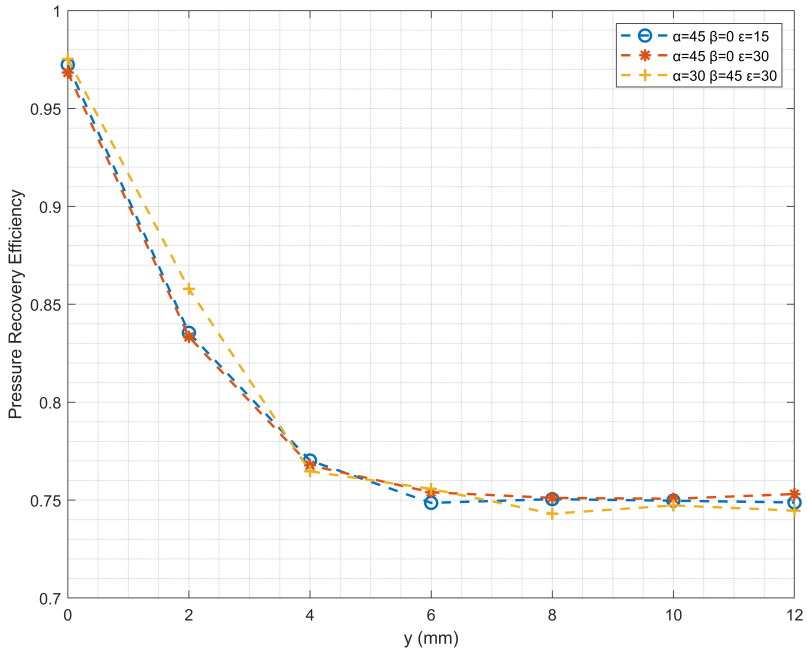


Figure 3.5: Top performing injectors pressure recovery efficiency plots.

of varying. This would be beneficial to all degrees of freedom and is the natural next step within RDE injector analysis.

Part II

An Optimised Semi-Impinging Injector's Effect on an RDE

Chapter 4

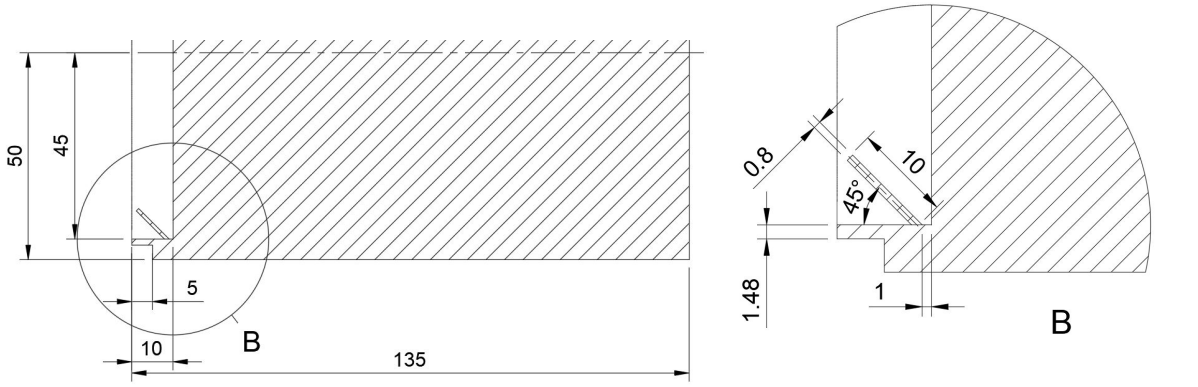
Methodology

4.1 Reference RDE

Given the success of the MSII method in cold flow conditions, the method was tested in a simulated hollow RDE, continuing the aims of the project. To achieve this, a design case and validation reference case must be set. Whilst within the prior exploration of the MSII method no empirical validation was used, the justification being that no empirical data could be found that would be applicable to the method, studies, and their data for hollow RDEs were available. The most appropriate and pressing study to fill this need was obtained from Huang et al. [16]. This is because the study explores the pintle injector method, a competitor to the MSII method, this would allow the comparison of an optimised MSII method injector against an unoptimised but operational pintle injector showing the merits of either method. Additionally, it would fill a gap in the literature as Huang et al. [16], notes the lack of numerical simulations of a pintle injector RDE, solving this would allow further analysis of the recirculation phenomenon in this injector methodology. The use of hydrogen and air as the reactants would also be most useful, as it allows the demonstration of the functionality of the MSII method with different reactants and given the widespread use of these reactants within the literature, the ease of finding an appropriate reaction mechanism would assist in the timeliness of the project. Within Huang et al. [16] the best performing injector geometry where all major parameters were listed was test no. 1, therefore this was chosen as the reference RDE. As not all key geometry dimensions were listed in the original study, using the dimensions available in diagrams from the original study, and assuming the diagrams were to scale the dimensions listed in figures 4.1a and 4.1b were found.

Pintle Injector Diameter (mm)	Pintle Injector Height (mm)	Fuel mass flow rate $\left(\frac{kg}{s}\right)$	Oxidiser mass flow rate $\left(\frac{kg}{s}\right)$	Equivalence Ratio	Detonation Velocity $\left(\frac{m}{s}\right)$
90	5	0.30044	0.00852	0.97	2057.74

Table 4.1: Reference Pintle RDE design and performance parameters



(a) Cross-section view

(b) Injector leg detailed view with 60 injector legs applied to the engine

Figure 4.1: Pintle RDE geometry

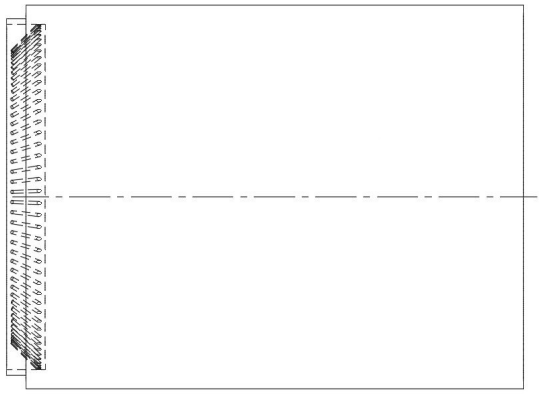


Figure 4.2: Side view of the Pintle Hollow RDE

4.2 Implementation of the MSII Method On a Hollow RDE

As it was found that the key factors of RDE injector performance are the mixing efficiency to simplify the method and increase efficiency, the total pressure loss and mixing consistency measurements would be discarded. Additionally, the focus on the mixing efficiency allows the time averaging and post-processing calculations to be automated within Ansys Fluent itself. This was not initially done as the fuel mass fluctuation, key to calculate the mixing consistency cannot be calculated within the software without the use of a user-defined function (UDF) which when loaded into Fluent allows the modification of the simulation process and must be coded in C. As

the author is not experienced with C this would have further increased the scope and timeline of the project beyond acceptable means, threatening the viability of the project, so it was decided to automate the mixing efficiency post-processing. The automated calculation was conducted by the use of user-defined field functions, and the time averaging of the mass flow rate and mass fractions. On top of the simplicity of this implementation of the post-processing, it would also allow the comparison of different data sampling rates, with the user-defined field function averaging over the entire refinement time, in contrast to the coarser sampling of using the exported instantaneous data.

$$\min(x, y) = 0.5\,(x + y - |x - y|) \tag{4.1}$$

$$\max(x, y) = 0.5\,(x + y + |x - y|) \tag{4.2}$$

$$\min\left(\overline{\varphi}, 1\right) = 0.5\left(\overline{\varphi} + 1 - \left|\overline{\varphi} - 1\right|\right) \tag{4.3}$$

$$\max\left(\overline{\varphi}, 1\right) = 0.5(\overline{\varphi} + 1 + \left|\overline{\varphi} - 1\right|) \tag{4.4}$$

$$\eta_{mix} = \frac{\iint_{S_y} \overline{Y_{O2}} 0.5\left(\overline{\varphi} + 1 - \left|\overline{\varphi} - 1\right|\right) d\overline{\dot{m}_y}}{\iint_{S_y} \overline{Y_{O2}} 0.5\left(\overline{\varphi} + 1 + \left|\overline{\varphi} - 1\right|\right) d\overline{\dot{m}_y}} \tag{4.5}$$

Within Ansys Fluent's user-defined field functions a limited set of mathematical operators can be used. This did not include min and max functions, by expanding the functions to their core operations the functions could be applied within the user-defined field function as seen with equations 4.1 to 4.4 and this then applied to equation 2.18 the mixing efficiency, resulting in the user-defined field function equation 4.5. To validate the new method the methodology validation study simulation from section 2.4 was repeated but using the updated mixing efficiency calculation approach.

The resulting mixing efficiency is shown in Figure 4.3 comparing the updated method, to the method used in section 2.4 and to the reference study from T. Gaillard et al. Whilst both methods recreate the results relatively well, the additional time sampling increased the overestimation of the mixing efficiency, where the reduced data sampling in the original method helps correct the overestimation, but with a higher uncertainty due to the under-sampling of the data. In all, the results do reasonably recreate the results of the reference study, and that with the refinement time-step reduced by a factor of 10, explaining the high error rates (above 10%) in the most turbulent regions, below 8 mm as seen in Figure 4.4. Though the justification of reducing the refinement

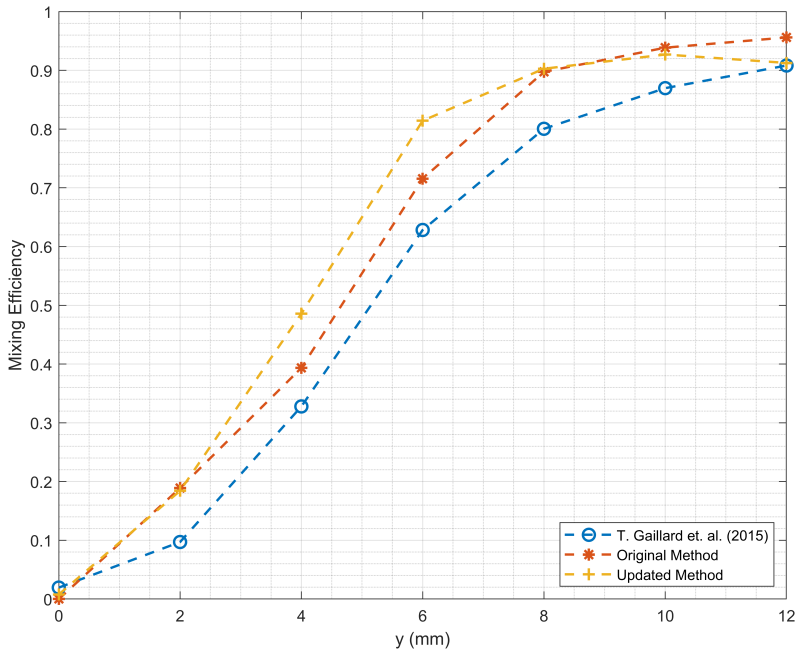


Figure 4.3: Updated Post-Processing Mixing Efficiency

time-step is reasonable considering the computing resources available to conduct this study. In all, the updated method is considered valid, with the gains in the efficiency of the calculation process worth the increase in error with a decrease in uncertainty.

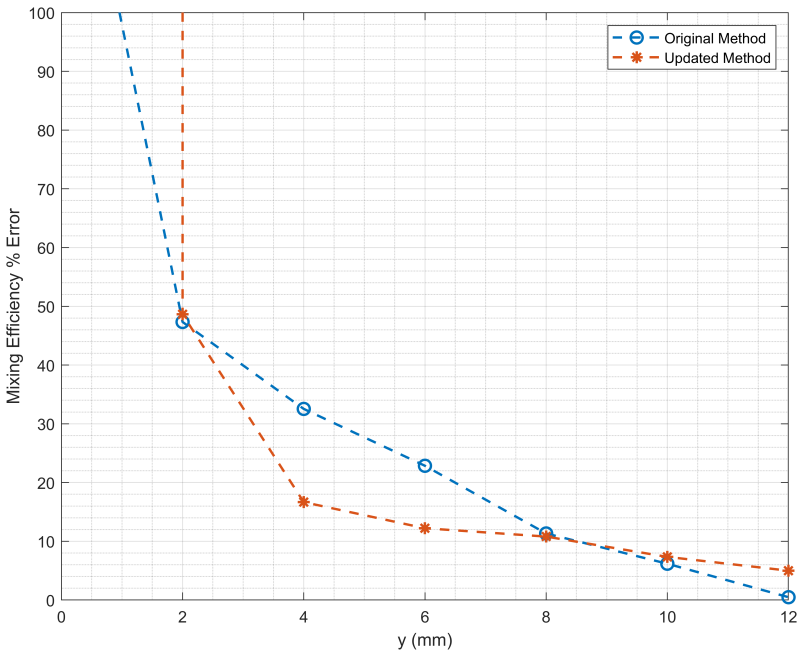


Figure 4.4: Updated Post-Processing Mixing Efficiency Error

Using the updated post-processing method and the key design requirements from the reference RDE an MSII method injector was designed. It was decided to use 60 injectors with a length of 5 mm to match the same injector area as the reference RDE. As the diameter of the combustion chamber was 100 mm, equally spacing each injector along the circumference gave

an injector cell width of 0.5233 mm. With the injector cell area set, the diameters and inlet parameters were found. Through a process of trial and error, keeping the Mach number in each inlet subsonic as had been explored in past SII and MSII injector designs, and with a preference for each inlet diameter to be a whole number the following inlet parameters were found:

- Fuel and oxidiser inlet diameters of 1 mm and 3 mm respectively
- Inlet pressures of 104000 Pa and 114000 Pa for the fuel and oxidiser inlets respectively and each with a total temperature of 300 K (note that the total pressure is the sum of the operating pressure and inlet pressure)
- An operating pressure of 200000 Pa
- A Mach number of 0.8 in each inlet leg

The length of the injector legs was extended over the original 4 mm as given the larger diameter of the legs, specifically the air injector leg, in any practical implementation, the legs would have required a larger distance to sufficiently diverge to space the feeding plenums apart. In following the MSII method the same ranges of the degrees of freedom were initially explored, throughout the trial and error exploration of setting the injector design values it was found that when $\alpha = 45°$ the injectors would exceed the injector cell area, therefore it was decided to reduce the range to just between 15° and 30° for α as seen in Table 4.2. Additionally, as it was found in the MSII study that generally the more angled the injector the higher the performance, on top of reducing the number of injector simulations within the optimisation study it was decided to keep $\beta = 45°$ again seen in Table 4.2. The simulation and meshing method followed in section 2.2 was followed with the initial conditions and design parameters changed to match those within this section.

Though a comparatively small sample of injector simulations was conducted, in comparison to the MSII optimisation study notable insights into the MSII method have been found. Unlike that of the hydrogen and oxygen reactants, hydrogen and air mixing performance results in a larger range of mixing efficiency plots and a noticeably slower rate of mixing. As seen in Figure 4.5 the results did not all converge within the height of the mixing region, this is likely due to the fact that the nitrogen within the air dilutes the oxygen, increasing the difficulty for the hydrogen to

Degrees of Freedom		
α	β	ϵ
15°	45°	0°
30°		15°
		30°

Table 4.2: Table of explored parameter values in the MSII RDE injector optimisation study

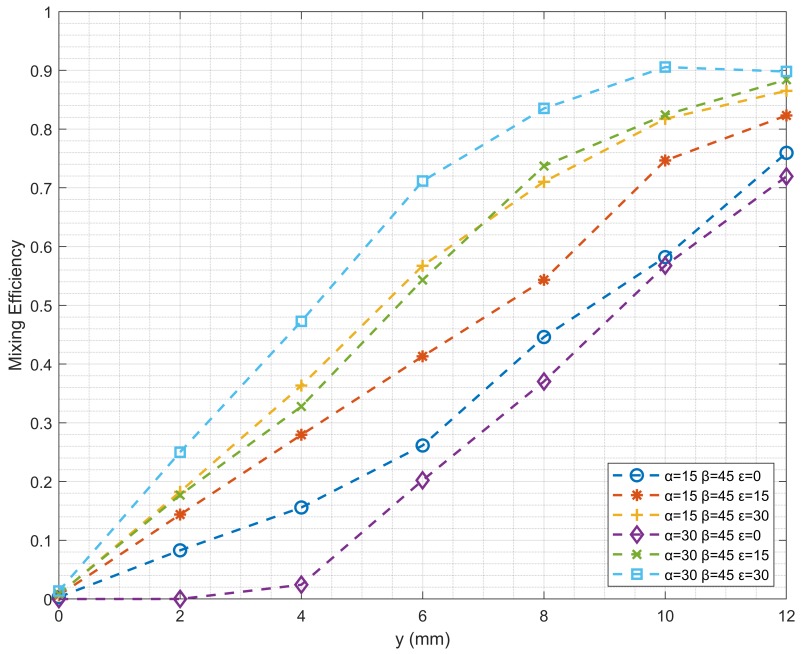


Figure 4.5: RDE Injector Study Results

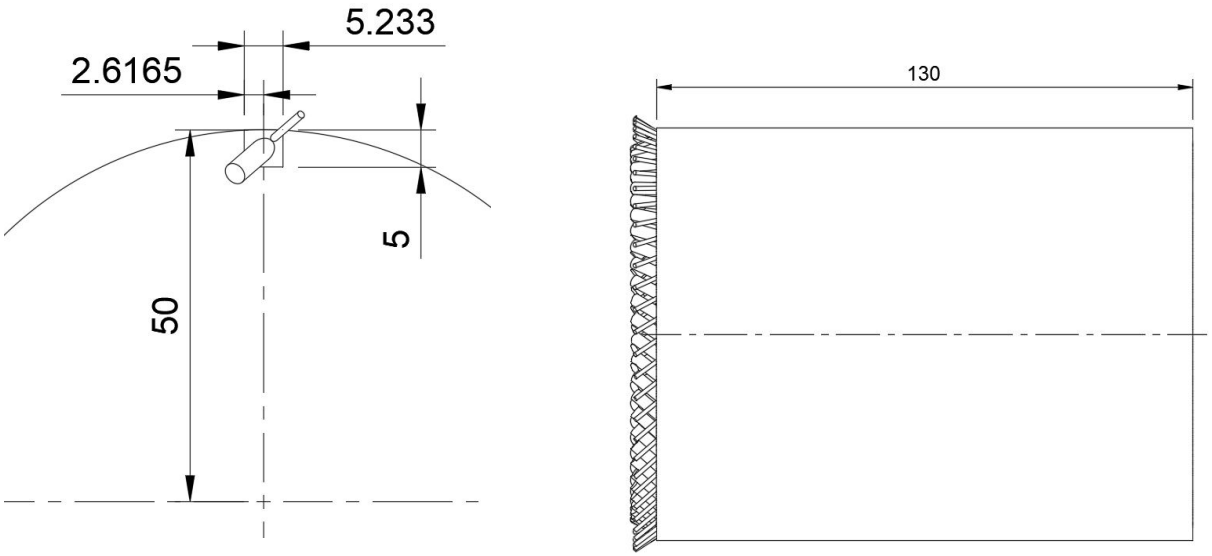
a	b	d_1	d_2	α	β	ϵ	l	L
$5.23mm$	$5mm$	$1mm$	$3mm$	$30°$	$45°$	$30°$	$7mm$	$15mm$

Table 4.3: Table of design values for the optimised MSII RDE injector

disperse equally within the mixture. This slows down the mixing reducing the mixing efficiency and speed. The characteristic mixing efficiency length was not calculated for these results as the results did not converge, making the comparison between different characteristic lengths inconsistent. From Figure 4.5 it can be seen that ($\alpha = 30°$, $\beta = 45°$, $\epsilon = 30°$) was the best performing injector in both overall and final mixing efficiency and mixing speed. This matches one of the most optimum injectors explored in the MSII optimisation study, and validates the assertion that generally the greater the angling of the injector the better the injector. The top-performing injector also matches the same DOF angles as that of the MSII optimisation study, this is most likely due to the limited range of injector values explored but does provide additional validation to the prior assertion. The optimised MSII RDE injector is applied to the combustion chamber as seen in Figure 4.6a.

4.3 RDE Simulation Methodology

Within the current literature, the general trend within numerical simulation meshes is towards using structured meshes. In the Ansys software package, the main meshing program is Ansys Meshing, this allows both structured and unstructured meshes to be generated for imported geometries. The Pintle RDE or the MSII RDE were imported into the software and the main



(a) View of injector placement on the MSII RDE

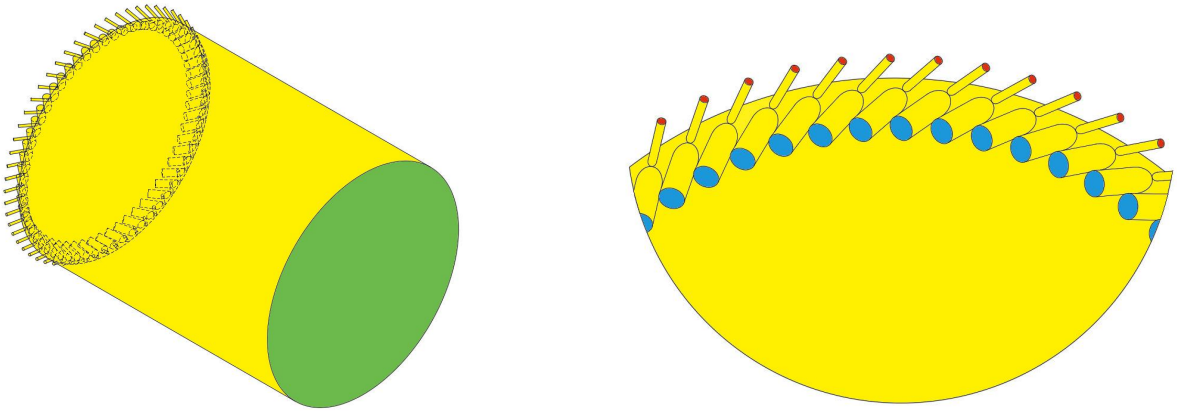
(b) Side view of the MSII RDE

Figure 4.6: Detailed views of the MSII RDE

structured meshing approach for dealing with geometry with discontinuous cross-sections; the multi-body approach, was applied both. With either injector geometries with fuel and oxidiser plenums or only the injector legs, the program resulted in a failure to generate the mesh. Therefore an alternative approach was required, this was achieved by applying a structured mesh in the continuous cross-section region of the geometry that being the cylindrical combustion chamber, and an unstructured mesh to the injector legs/region. To improve the mesh transition from the furthest position in the mixing region a plane was placed 5e-4m along the axial direction parallel to the outlet plane separating the geometry into two sections, the mixing body and combustion body. On the combustion body, the multi-body meshing method was applied, this is because the continuous cross-section allowed for a more structured mesh to be created. To produce a hexahedral mesh to reduce the mesh element count, the mapped mash type was set to Hexa, the surface mesh method set to pave (as this was found to produce the best mesh through trial and error), and free mesh type set to Hexa dominant. On the mixing body, Ansys's default unstructured meshing approach was applied, with an element and max element size set to 5e-4m, to minimise the total element count whilst still being adequate to simulate the detonation wave. No mesh study was conducted as the effects of mesh size has been extensively studied within the literature [45], with meshes ranging from 2e-4 to 5e-5m adequate to simulate the key detonation mechanisms. To allow the mesh to adapt to the varied geometry of the injectors the capture proximity setting was turned on with default settings and mesh defecting and capture curvature settings turned off. The other settings within Ansys Meshing were kept at their default values. As seen in table 4.4 the element counts differ by about 1 million elements, this is due to the mixing body being larger in the Pintle RDE than the MSII RDE, thereby having a larger unstructured mesh and due to the

RDE	Pintle	MSII
Min	0.16453	0.18846
Max	1	1
Average	0.95989	0.96971
Standard Deviation	7.1386e-2	5.934e-2
Elements	9569491	7633809

Table 4.4: RDE mesh quality statistics



(a) MSII RDE outlet and wall boundaries

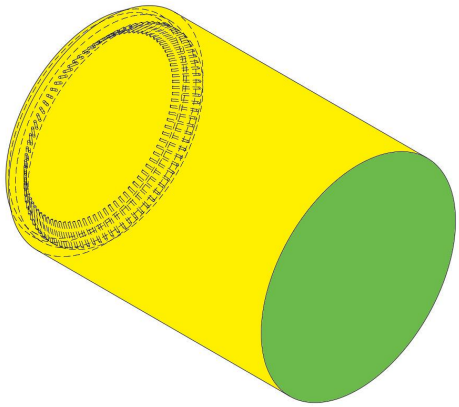
(b) Side view of the MSII RDE

Figure 4.7: MSII RDE boundary conditions

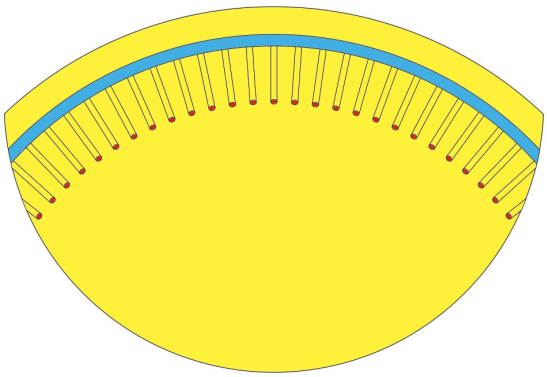
unstructured mesh higher element density, a larger element count.

It was noted in section 1.2, that the simulation methodologies generally fell into two categories, inviscid simulations largely being the Euler method and viscous simulations most commonly being the SST K-ω method which take account for viscous and diffusive effects. As the meshing approach taken within this project differed from most of the literature, both simulation methods were tested being called the simplified method and detailed method respectively. Both methods made use of the same meshes, Ansys Fluent's default boundary condition settings and wall treatments for their respective models and boundary locations (that being no-slip conditions for both and the two layer model wall treatment for the detailed method), with the boundary locations shown in Figures 4.7a to 4.8b, where red denotes the fuel inlet, blue the oxidiser inlet, green the pressure outlet, and yellow the no-slip adiabatic wall.

The simplified method was largely adapted from Yuhui Wang (2016) [48], with the vicious method changed to an inviscid approach. The method made use of Ansys's Transient, Explicit formulation Density-Based type solver as the result wasn't a steady-state solution, and the solver selected has an advantage in its accuracy in simulating high-velocity compressible flows, that being the type expected within the RDE combustion chamber [7, 48]. The inviscid Euler solver was used in conjunction with an ideal gas density solver as these solvers made use of the com-



(a) Pintle RDE outlet and wall boundaries



(b) Pintle RDE boundary conditions

Figure 4.8: Detailed views of the MSII RDE

pressible Euler Equations to solve the flow [6], this being appropriate as noted in section 1.2 that the Euler equations can model the key dynamics of RDE accurately enough, the specific heat calculation method was the mixing law. The reaction mechanism used was the same as Yuhui Wang (2016) [48] a single-step mechanism requiring only H_2, N_2 and O_2 species with a laminar finite-rate flame method. Inlet and outlet conditions were set by their respective mass flow rates, with the inlets specified by the fuel and oxidiser mass flow rates and outlet by the total mass flow rate, this was done to simulate the flow through the section, and as the nozzle was removed to reduce the mesh size and time, to make sure that it would match the actual flow from the empirical reference study. This poses a potential issue through a pressure outlet was not used as the outlet in this method, the geometry did not model the full geometry from the original study wherein past studies, not simulating the full geometry, especially in the case of a hollow RDE without a nozzle, especially with a pressure outlet can affect the RDEs stability and performance [45]. The air inlet mole fractions for nitrogen and oxygen were 0.79 and 0.21 respectively and total temperature values for all boundary conditions was 300K. Following the method described within Yuhui Wang (2016) [48] the Roe-FDS flux solver was used along with the least-squares based gradient solver given the use of an unstructured mesh. The Flow solver and transient formulation were set to second-upwind and second-order implicit to improve the accuracy of the results. The solution was initialised with stoichiometric conditions and operational pressure of 0.1 MPa, this was done to further simplify the simulation method, reducing the required simulation time. In the simulation 4 pressure measurement point were set at P2=(0.03,0,0.049), P1=(0.03,0.049,0), P4=(0.03,0,-0.049) and P3=(0.03,-0.049,0) as seen in Figure 4.9, with the origin in each of the geometries set at the centre of the combustion chamber, 130 mm from the outlet. A spherical region was selected with a centre at (0.03,0.05,0) and a radius of 0.01 m and patched with a pressure of 1 MPa and a z velocity of 2000 m/s to initialise detonation in the RDE. The simulation was then ran with a time step of 1e-7s.

The detailed method diverged from the simplified method by a wide margin. This was done

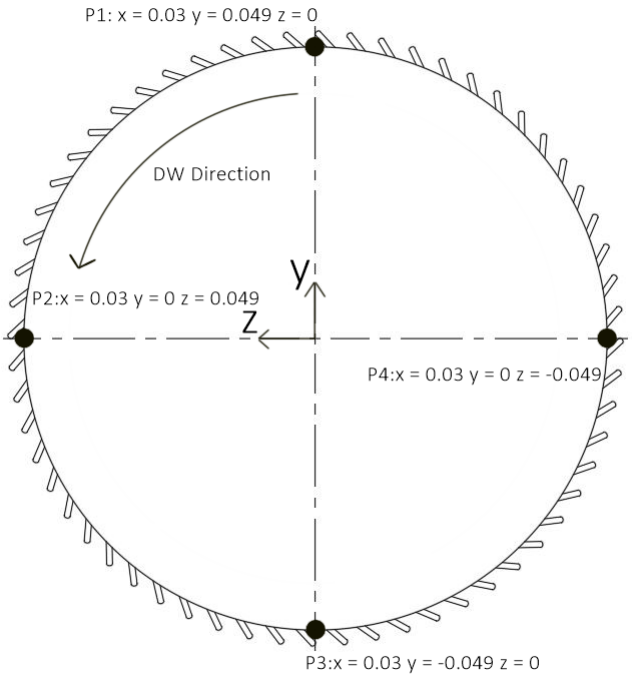


Figure 4.9: RDE pressure measurement points

through a trial and error approach as just replacing the simplified method's approach with a viscous solver and a more detailed reaction mechanism. The addition of a mixing period to fill the chamber instead of initialising with a stoichiometric mixture proved unobtainable using the density-based type solver and mass flow inlets and outlets. Keeping the operational pressure and inlet species constant over the two methods, the stability issues were solved by substituting the transient pressure-based type solver and pressure inlets and outlets. The pressure-based type solver though noted in the Ansys Fluent manual as being less appropriate for the flow expected within an RDE [7], can still accurately model high-velocity compressible flow, and in this case, proved the more stable solver. The pressure inlets and outlets also provided another issue as the expected mass flow rate through the RDE had to be provided by the pressure values rather than a specific mass flow rate, when monitoring the mass flow rate through the engine with the pressure inlet and outlets an 8.82% drop in mass flow rate was found. The pressures at each inlet were the same as in the RDE injector study, to provide the same mass flow rate. The viscous solver for the detailed method was the SST K-ω method with a large amount of the detailed method adapted from J. Sun et. al. [44]. In keeping with the simplified method the density was calculated by the ideal gas equations and specific heat by the mixing law, given the use of a solver that takes account for the viscosity and diffusion, the thermal conductivity and viscosity were calculated by the mass-weighted mixing law, and Mass diffusivity by Kinetic theory [48]. The inlet pressure-velocity coupling method used was the PISO scheme as it was found to provide the best simulation stability and speed. The pressure, momentum, turbulent kinetic energy, specific dissipation rate, species and energy equations were all second-order upwind and the transient

formulation second-order implicit, in keeping with the simplified method. To allow the engine to fill up the simulation was initialised with ISA sea level condition air, and then ran until the engine filled with the mixed fuel and oxidiser. The reaction mechanism was then implemented using the same laminar flame finite rate combustion model as the simplified method, but the reaction mechanism was changed to the 8-species 7-step reaction mechanism used in J. Sun et. al. [44]. The same detonation initialisation procedure was used as in the simplified method. The simulation was then ran with a time step of 1e-8s with a pressure measurement point at (0.03,0,0.049).

Chapter 5

Results and Discussion

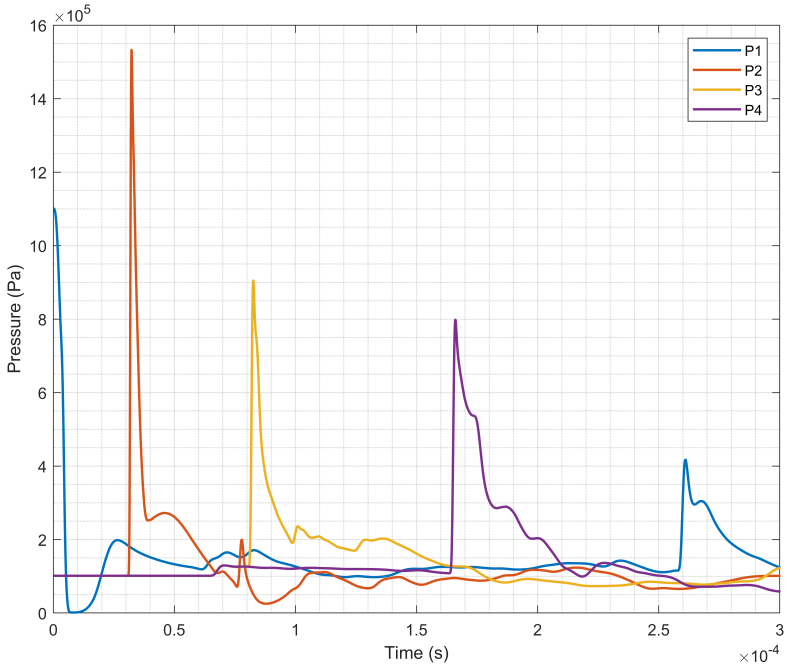


Figure 5.1: Pintle simplified method pressure measurements where P1-4 refer to the points described in Figure 4.9

It should be noted that all methods had issues with running for more than 1 cycle whilst keeping the Courant number below 1, this is likely due to the meshing method used. Though the meshing the meshing method did produce usable meshes, with a minimum element quality above 0.1, the best minimum mesh quality obtained using Ansys Meshing for the given geometries was around 0.17. It should be noted that this was the highest mesh quality obtained after experimenting with the full range of meshing approaches available within Ansys Meshing. Given the large gradients, and similarly large in the context of detonation, time steps, this resulted in unstable simulations. Steps to reduce the time step could possibly have been taken to further reduce the Courant number and improve the stability of the simulation but this would have required

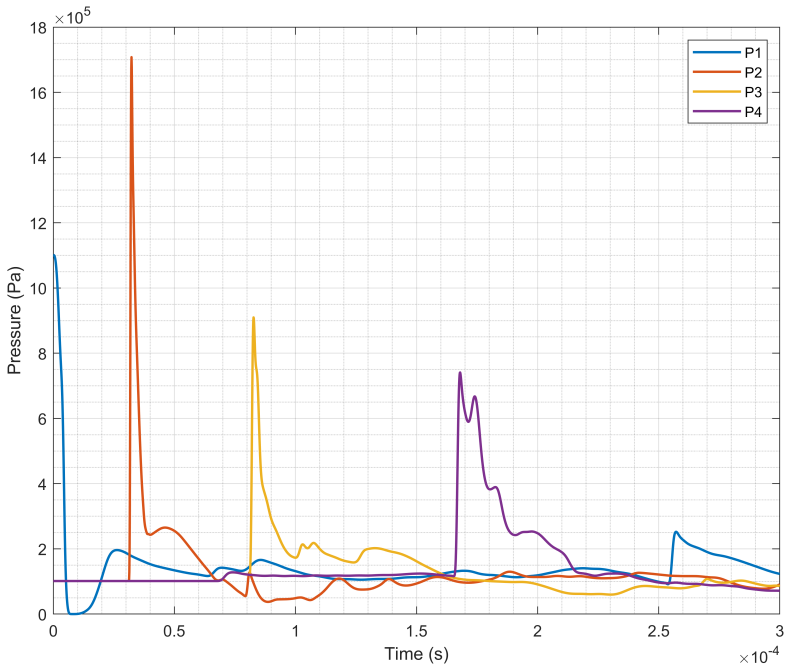


Figure 5.2: MSII simplified method pressure measurements where P1-4 refer to the points described in Figure 4.9

more computing resources than were available at the time. Additionally, as each simulation took approximately a month to complete, this shows the limitations of the computational resources available to this project. This could have been solved by using a different meshing software that produced a fully structured hexahedral mesh as opposed to the unstructured one produced by Ansys Meshing and possibly reduce the element mesh size and improve the mesh quality. This helping to solve some of the issues with both the run time and stability. Unfortunately sticking within Ansys's software and time limitations restricted exploring other meshing software. Though these limitations restricted the results and the ability to test the effect on performance by the MSII injector, nonetheless results were obtained for both the detailed and simplified methods allowing within the limited run times, notable if not limited comparisons to be made. Though these comparisons require further research to be made fully reliable, especially by emperical results.

The simplified method resulted in similar detonation velocities for both RDE geometries as measured at each of the pressure measurement points (note: the pressure measurement points were taken at 90° increments around the circumference of the chamber) as seen in Figure 5.3. Over the entire cycle, the detonation wave velocity for the MSII RDE and Pintle RDE was 1221.459 m/s and 1202.754 m/s respectively. Using the initial conditions from the MSII injector study, NASA's CEA Code calculated the C-J velocity to be 1973.8 m/s. This resulted in a %C-J velocity of 61.884% and 60.936% for the MSII and Pintle RDEs respectively. The empirical Pintle RDE detonation velocity was 2057.743 m/s and %C-J velocity was 104.253%. Given an only 1% difference between the simplified methods detonation velocities and an around 40%

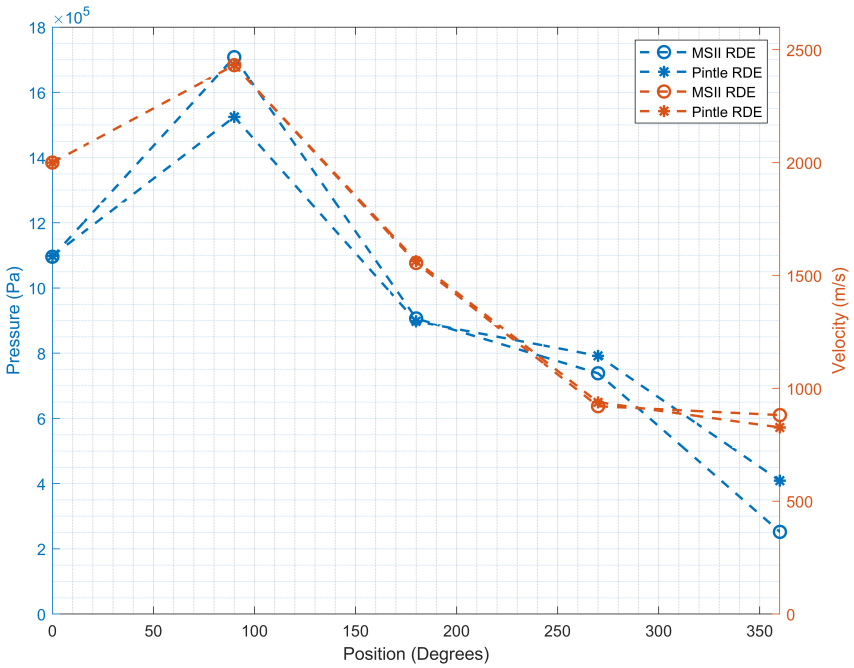


Figure 5.3: Simplified method detonation values for both RDE geometries at the P1-4 measurement points, where the orange plots describe instantaneous detonation velocities and blue describe peak pressures

difference to the empirical results, this shows the limitations of the single-step reaction mechanism used within the simplified method. The key difference between the geometries being the detonation pressures as seen in Figure 5.3 with the MSII RDEs detonation pressures initially higher, then dropping off below the Pintle RDE. The detonation pressures at the measurement point P2 are conducive to stable detonation but from P3 and onwards the peak pressures decrease, marking the change from detonation to deflagration, and then to a weak deflagration wave. Typically, within RDEs, this is seen in engines with inadequately mixed reactants, where the global equivalence ratio is below the detonation limit. As it can be seen in Figures 5.1 and 5.2 the final pressure measurement varies significantly, as this position has only been filled by the injector, and not the stochiometric initialised mixture it allows the performance of the injector designs to be compared. In Figure 5.1 the Pintle detonation pressure is almost double that of the MSII RDE whilst obtaining a similar RDE velocity, this shows that within the simplified method, the better performing injector is the Pintle. With the caveat that both RDEs seemingly failed to initialise and begin continuous rotating detonation. It should also be noted that in the author's experience while attempting to recreate the results of [48] in a prior project, using the reaction mechanism listed within the study resulted in a similar issue, where the detonation wave pressure values produced were lower than the ones listed within the study and the detonation wave failing to continuously rotate. At the time it was initially assumed to be an issue with the other parts of the methodology. Given that the same issue occurred with a different meshing and solver method, the reaction mechanism is the most likely origin of the issue. The reaction mechanism seems

to have a too low pre-exponential factor or too high activation energy, resulting in a lower than expected reaction rate, releasing too little energy to sustain detonation, even in stoichiometric conditions.

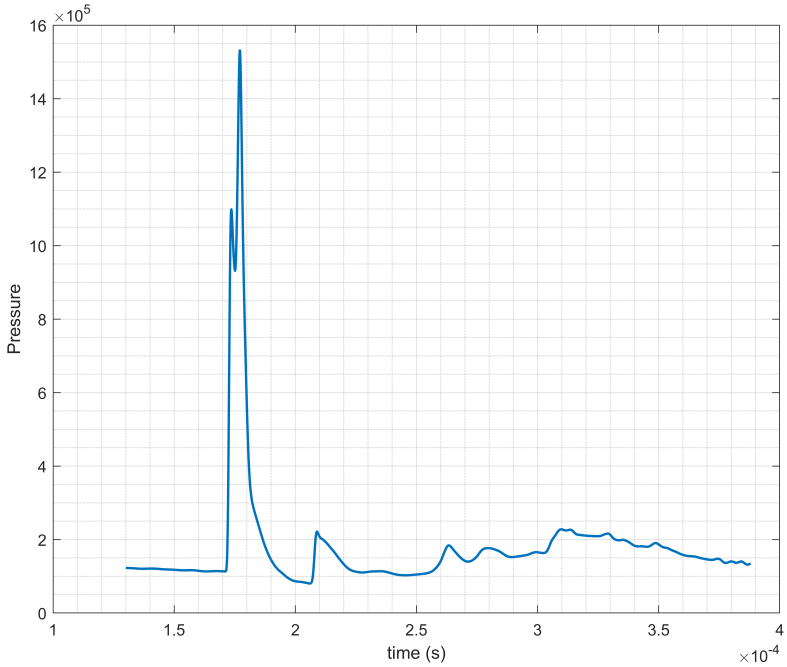


Figure 5.4: Pintle detailed method pressure measurements

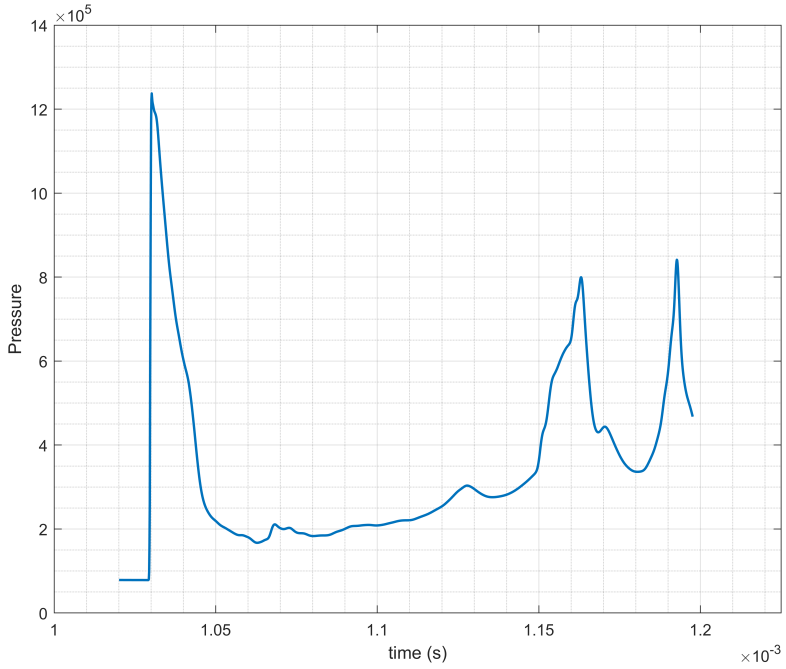


Figure 5.5: MSII detailed method pressure measurements

The detailed method results differed drastically from the simplified method, with only one of the two RDEs completing a full cycle. The effect of the injectors on the performance of the engines is drastic with a reduction in the time required to fill the chamber by a factor of 10 for

the Pintle RDE in comparison to the MSII RDE. But as the Pintle RDE failed to cycle the quality of the mixing was likely insufficient to initialise the detonation wave whereas though the MSII RDE took longer to fill, it must have had a significantly higher quality mixture. It should be noted that the simulation was run until the fuel mass fraction at the mass flow outlet was greater than 0 showing the injector had filled the chamber. This possibly resulted in an inadequate mixing time for the pintle RDE and show a higher axial velocity for the pintle RDE than the MSII RDE. The detonation pressures for the detailed method both were lower than the peak pressure at P2 in the simplified method, with the performance of each engine reversed, where the MSII RDE detonation pressure was lower than the Pintle RDE. The second set of detonation pressures in the MSII RDE are lower likely due to the detonation wave splitting into two separate waves, as these two waves both consume the same fuel the pressure gradient of each wave will decrease but will have a similar detonation wave speed. As only one pressure measurement point was recorded, it is impossible to know whether the two waves are counter-rotating or transitioning into the two-wave mode though both have peak pressure values associated with detonation. The detonation wave velocity for the 2nd wave is 2362.010 m/s and the 3rd 1932.099 m/s with a %C-J velocity of 119.673% and 97.887% respectively. This validates the reaction mechanism used especially in comparison to the single-step mechanism, as they coincide with the empirical Pintle RDE's %C-J velocity. Though the mode is different, a two-wave mode in comparison to the TDPO mode of the empirical result, as this is just an initial cycle of the RDE the mode could transition down to the TDPO mode as the cycles progressed. Additionally, the RDE was not in a stable two-wave mode as the waves are too close together and have a 20% gap in % C-J velocity and are required to be closer together in detonation velocity and have an approximately 180-degree phase shift between each other to operate within the two-wave mode. Regarding recirculation within the simulated Pintle RDE given that the simulations ran for a very short simulated time, it's unlikely that the recirculation regions developed also the lack of a GPU on the VMs used makes it unable to produce post-processed images and data of the flow.

Comparing the two methods the results seem to contradict each other, with the MSII RDE performing better in the detailed method, but the Pintle RDE narrowly outperforming the MSII RDE in the simplified method. Each method had its advantages, and possibly combining the simplified method with the multi-step reaction mechanism, a compromise between the compute time and quality of the results could have been found. But as the study was time-limited this approach would need to be tested in a future study. For future study, the MSII and SII method must be empirically validated to match empirical results as the lack of non-numerical evidence calls into question the results produced so far. As seen with the simplified method, the limitations of the single-step reaction mechanism are clear and future studies should focus on the use of multi-step reaction mechanisms. Though even within the current field of multi-step reaction mechanisms, the lack of a more standardised mechanism or approach limits the ability to compare and recreate results. A large amount of time within this study was spent assessing the various reaction mech-

anisms to find a valid mechanism, a frequent issue found was for the same reaction mechanism's reaction equations and constants, the units would change from study [42] to study [44], this taking time to test each version with unit conversions. Both methods, given the long compute times and short runs of the RDEs, difficulty in producing adequate meshes for the complex injector geometries, and issues finding an adequate and reliable reaction mechanism are examples of why the field of RDEs was sparsely populated until the 2010s. Given the field is relatively young and adaptable, setting a more standardised method of simulating RDEs is of key importance to the field. The problem of a lack of a standardised method applies also to injector design and integration. The standard method of validating injector design results by using empirical data has yet to be applied to the MSII and SII methods, for the MSII method developed within this study this was done to fit within the budget, covid restrictions and time limitations imposed on the project. The same cannot be said of the SII method where all results have been numerical, casting some doubt on the validity of the method at the present for use within RDEs. This is not to neglect the lack of a standardised design methodology for pintle injectors for RDEs, though empirical testing has validated the use of this type of injector within RDEs. At the current state of the field and the limited data gathered from this study, the pintle injector is a better approach, as it is simpler to manufacture, as opposed to using hundreds of small and long injector legs that would be difficult and expensive to manufacture, additionally, the lack of a flow choke point on either of the MSII or SII methods introduces issues of pressure coupling and back-flow to the inlet plenums and the detonation wave, this poses both safety issues and stability issues to the designs. The pintle method is widely used in other rocket engines and has been successful in operating with various reactants, additionally it provides another control aspect and the ability to physically shut off flow to the engine by fully retracting the pintle and adjusting its height. More research is needed into possible ways to accelerate RDE CFD simulations, as to this day they are still computationally costly to run limiting the research that can be carried out on the field. These issues could be solved by using a different meshing software and then importing the mesh into Ansys with meshing software such as GMSH, ICEM, Pointwise or StarCCM+ all widely used. Secondly to accelerate and reduce the computational load of CFD simulations [21] for RDEs using machine learning [33] and reduced-order models [32] by modelling the core dynamics and applying the patterns found to numerical simulations to reduce the need to solve the complex CFD PDEs for each cell of a mesh. Finally, possibly using other solvers, especially open-source CFD solvers such as OpenFOAM can increase the accessibility of running RDE simulations rather than using commercial software and have been successful at running RDE numerical simulations [50]. To summarise the key findings in this phase:

- That the Pintle RDE appears to be the best injector design method found within the literature and this study and the way forward as it is both empirically validated to operate well with RDEs unlike either the SII or MSII methods, and a proven design being used on other types of rocket engines, but within the RDE field lacks a standardised design methodology.

- Both methods showed more about the issues of simulating RDEs that the comparison between the injectors with a more than 40% difference in %C-J velocity between the single-step and the multi-step reaction mechanisms, with only the multi-step reaction mechanism reaching a similar %C-J velocity to the empirical results.

- Solutions are needed to be able to simulate RDEs in a more computationally efficient way, this could possibly be achieved by integrating machine learning and reduced-order models into the solver to accelerate the process.

Chapter 6

Conclusion

Within this thesis the objectives were to; first quantify, explore and optimise the Modified Semi-Impinging Injector method and secondly apply the MSII method to an RDE and find the effect on the performance. The first objective was met in Part 1 by developing the equations to calculate the MSII geometry, and then these equations were coded in python to allow the design process to be automated and applicable to a variety of initial conditions, reactants and phases. The MSII method was explored in both parts 1 and 2, with a variety of reactants used between the two sections, and the effect of modifying the geometry explored in an optimisation study in each allowing the goal of optimising the MSII method to be met. The second objective was addressed in Part 2 by implementing an optimised MSII injector designed to match the mass flow rate, equivalence ratio and reactants of an empirically tested pintle injector RDE. With the MSII RDE and Pintle RDE numerically simulated and compared to the empirical results, to validate the results, the simulations were unable to discern the direct effect of the MSII injector. This is due to methodological issues causing instability in the simulation calculation, all simulations ran for less than 2 cycles and failed to reach stable continuous detonation.

The findings of this thesis are of importance to the RDE field and future research. Such as the development of the MSII method which though not empirically validated does show promise in cold flow numerical simulations and does outperform the SII method which has been shown to work in simulated RDEs [14]. The SII or MSII methods were could still provide an alternative to the pintle and impinging injector methods currently in use if empirically tested in RDEs and as the design approach and equations have been derived and validated within this study is the next logical step in this injector methodology. Beyond this, the exploration of the direct effect of various reactants on the mixing speed should be explored given the large differentiation between the hydrogen-oxygen results and hydrogen-air results, as the mixing speed impacts the ability of an RDE to operated with shorter cycle times, and therefore smaller diameters directly affect the geometry of the engine, which then goes to affect the injector design, the interdependence of these two factors needs to be untangled. The development of the characteristic length as a means

of streamlining the analysis of the optimisation of RDE injectors is another important finding. As RDE injectors must readily mix reactants in as short a time and space as possible, when applied to the mixing efficiency allows the mixing speed to be found. It also provides a simple measure for automating the injector design process, as a single value to define the performance of an injector allows easy comparison between different injector geometries. Which given that the mixing efficiency calculation was automated and integrated within Ansys, could allow a more full exploration of the injector design space and solve the issue of the wide jumps taken between the design points within the injector optimisation study.

The assessment that the most important factor for measuring RDE injector performance is that of the mixing efficiency. As the relationship between the mixing efficiency and pressure recovery efficiency was found to be inversely correlated, this matching the results of T. Gaillard [13], the need to calculate both measurements was redundant. Additionally, as the mixing consistency was found to coalesce around a similar plot and values, with no discernible patterns, in addition to the difficulty calculating this measure within Ansys Fluent it was shown that the most important RDE injector performance measurement was the mixing efficiency. It should be noted that it is also shown by its use within the literature unlike that of the pressure recovery efficiency or mixing consistency [44, 60]. In addition, the comparison of two separate RDE simulation methods showed the direct effect of different reaction mechanisms and limitations within the current literature as the move from a single-step to multi-step resulted in a 40% increase in %C-J velocity. It also resulted in the change from a failure to initiate in both RDEs to the RDE detonation velocity aligning with the empirical result. This difference is indicative of the limitations of the single-step reaction mechanism inaccurate simulation of RDEs. To this day the single-step mechanism is still used [58] and though it can produce adequate detonation velocities, with around 60% %C-J velocities seen in this study, the pressure waves it produces were found to match deflagration waves. This finding is indicative of the need to both move to only using multi-step methods and the use of a standardised RDE reaction mechanism, depending on the reactants used, as consistent difficulties were found in both validating and implementing various reaction mechanisms. A move to standardise the nomenclature, measurements, design processes and experimental practises in the field of RDE research would be highly beneficial, with the key contribution this study makes being a template for the design and implementation of an RDE injector. Though with the possible improvements on the resources, time, computational power and automation of the optimisation process would be a core step in the design process of a commercially viable RDE.

The broader societal impacts of this work are in the development of the MSII method. As a new injector design approach, the use cases of this method and its particular benefits range widely outside the RDE field. These are not limited to industrial applications of rapidly and effectively mixing or dispersing gasses and potentially liquids in as short a distance as 6mm, with mixing efficiencies greater than 90%. Additionally, the MSII method could also have potential use

cases in traditional rocket engines, with their high efficiency and simple design, though empirical validation and testing would be needed. This study also stands as a testament to the increasing democratisation of researching RDEs and the expansion of the field. This comes with the benefits of a greater number of perspectives and data on which the field can use to reach greater notoriety and eventually commercial viability and begin to play a part in reducing emissions. Additionally, this thesis is to the author's knowledge the first formal research study conducted into RDEs and notably hollow RDEs within Scotland and the United Kingdom, building a foundation for further research to be conducted putting these nations on the map for rotating detonation research, and being one of a few that have so far.

Besides the limitations previously discussed, the key limitations of this thesis are the simplifications made to each of the methods, be that using the largest accepted RDE element sizes, larger time-steps or the enlargement of by a factor of 10 made to the injector refinement stage time-step. These simplifications were made to reduce the compute time and power required but resulted in less accurate results and reduced quality meshes. With more computing power available to the project, more time to run the simulations and refine the meshes and meshing technique these issues could have been solved. Regarding the future direction of the field and questions for further research, the top injector methodology within the RDE field is the Pintle injector method, given the successful and high performing empirical results, simplicity of the design and ease of manufacturing, these benefits outperform those of the MSII and impinging injector methods at the current time. The core question around the Pintle method is the lack of a standardised design method allowing the standardisation of the design and recreation and validation of prior findings. Though the MSII method requires empirical testing and validation and given the design method listed within this study, is the logical next step regarding this design approach.

Appendix A

Injector Design Python3 Code

```
#setup
import numpy as np
import requests
import pandas as pd
import urllib.request
import re
from bs4 import BeautifulSoup
import sympy as sym

def equivilence_ratio(fuel,oxidiser,ptfuel,ptoxidiser,
Ttfuel,Ttoxidiser,dfuel,doxidiser,Machfuel,Machoxidiser,
velocityfuel,velocityoxidiser,ninjector):
  #determine fuel used and scrape data from NIST Chemistry
  #WebBook for density, Cp, Cv, and Phase of material
  # density is in kg/m^3
  # pressure (total) is in Pa
  # temperature (total) is in K
  # diameter is in mm
  # veocity is in m/s
  # area is m^2
  if fuel == "Water": # C7732185

    Rspesificfuel = 461.52230 # spesific gas constant

    P = ptfuel / 1000000 # MPa
    T = Ttfuel # K
```

```
p_url = str(P)
t_url = str(T)
t_maxval = T + 10
t_max_url = str(t_maxval)

url = ( "https://webbook.nist.gov/cgi/fluid.cgi?Action"+
"=Load&ID=C7732185&Type=IsoBar&Digits=9&P=" + p_url +
"&THigh=" + t_max_url + "&TLow=" + t_url +
"&TInc=10&RefState=DEF&TUnit=K&PUnit=MPa&DUnit="+
"kg%2Fm3&HUnit=kJ%2Fmol&WUnit=m%2Fs&VisUnit=Pa*s&STUnit=N%2Fm" )
response = requests.get(url)

soup = BeautifulSoup(response.text, "html.parser")

found = soup.findAll('td')
foundDensity = re.findall('\d*\.?\d+',str(found[2])) # Density
foundCv = re.findall('\d*\.?\d+',str(found[7])) # Cv
foundCp = re.findall('\d*\.?\d+',str(found[8])) # Cp

found2 = re.findall('vapor',str(found[13])) # Phase
if found2 == ['vapor']:
  phasefuel = found2[0]
else:
  found3 = re.findall('liquid',str(found[13]))
  if found3 == ['liquid']:
    phasefuel = found3[0]
  else: print("Non liquid or gas phase, please enter a different '
  "total pressure and/or temperature for the fuel")

foundDensityprocessing = foundDensity[0].split('.') # Density
pt1 = int(foundDensityprocessing[0])
pt2 = int(foundDensityprocessing[1])
length = (int(len(foundDensityprocessing[1])))
pt2_decimal = pt2 / (10**length)
densityfuel = pt1 + pt2_decimal

foundCvprocessing = foundCv[0].split('.') # Cv
pt1 = int(foundCvprocessing[0])
```

```
    pt2 = int(foundCvprocessing[1])
    length = (int(len(foundCvprocessing[1])))
    pt2_decimal = pt2 / (10**length)
    Cvfuel = pt1 + pt2_decimal

    foundCpprocessing = foundCp[0].split('.') # Cv
    pt1 = int(foundCpprocessing[0])
    pt2 = int(foundCpprocessing[1])
    length = (int(len(foundCpprocessing[1])))
    pt2_decimal = pt2 / (10**length)
    Cpfuel = pt1 + pt2_decimal


  elif fuel == "H2": # C1333740

    Rspesificfuel = 4124.48291 # spesific gas constant

    P = ptfuel / 1000000 # MPa
    T = Ttfuel # K
    p_url = str(P)
    t_url = str(T)
    t_maxval = T + 10
    t_max_url = str(t_maxval)

    url =  "https://webbook.nist.gov/cgi/fluid.cgi?Action="+
    "Load&ID=C1333740&Type=IsoBar&Digits=9&P=" + p_url +
    "&THigh=" + t_max_url + "&TLow=" + t_url +
    "&TInc=10&RefState=DEF&TUnit=K&PUnit=MPa&DUnit="+
    "kg%2Fm3&HUnit=kJ%2Fmol&WUnit=m%2Fs&VisUnit=Pa*s&STUnit=N%2Fm"
    response = requests.get(url)

    soup = BeautifulSoup(response.text, "html.parser")

    found = soup.findAll('td')
    foundDensity = re.findall('\d*\.?\d+',str(found[2])) # Density
    foundCv = re.findall('\d*\.?\d+',str(found[7])) # Cv
    foundCp = re.findall('\d*\.?\d+',str(found[8])) # Cp
```

```
    found2 = re.findall('vapor',str(found[13])) # Phase
    if found2 == ['vapor']:
      phasefuel = found2[0]
    else:
      found3 = re.findall('liquid',str(found[13]))
      if found3 == ['liquid']:
        phasefuel = found3[0]
      else: print("Non liquid or gas phase, please enter a "+
      "different total pressure and/or temperature for the fuel")

    foundDensityprocessing = foundDensity[0].split('.') # Density
    pt1 = int(foundDensityprocessing[0])
    pt2 = int(foundDensityprocessing[1])
    length = (int(len(foundDensityprocessing[1])))
    pt2_decimal = pt2 / (10**length)
    densityfuel = pt1 + pt2_decimal

    foundCvprocessing = foundCv[0].split('.') # Cv
    pt1 = int(foundCvprocessing[0])
    pt2 = int(foundCvprocessing[1])
    length = (int(len(foundCvprocessing[1])))
    pt2_decimal = pt2 / (10**length)
    Cvfuel = pt1 + pt2_decimal

    foundCpprocessing = foundCp[0].split('.') # Cv
    pt1 = int(foundCpprocessing[0])
    pt2 = int(foundCpprocessing[1])
    length = (int(len(foundCpprocessing[1])))
    pt2_decimal = pt2 / (10**length)
    Cpfuel = pt1 + pt2_decimal

  elif fuel == "CH4": # C74828

    Rspesificfuel = 506.20777 # spesific gas constant

    P = ptfuel / 1000000 # MPa
    T = Ttfuel # K
    p_url = str(P)
```

```
t_url = str(T)
t_maxval = T + 10
t_max_url = str(t_maxval)

url =  "https://webbook.nist.gov/cgi/fluid.cgi?Action"+
"=Load&ID=C74828&Type=IsoBar&Digits=9&P=" + p_url +
"&THigh=" + t_max_url + "&TLow=" + t_url +
"&TInc=10&RefState=DEF&TUnit=K&PUnit=MPa&DUnit="+
"kg%2Fm3&HUnit=kJ%2Fmol&WUnit=m%2Fs&VisUnit=Pa*s&STUnit=N%2Fm"
response = requests.get(url)

soup = BeautifulSoup(response.text, "html.parser")

found = soup.findAll('td')
foundDensity = re.findall('\d*\.?\d+',str(found[2])) # Density
foundCv = re.findall('\d*\.?\d+',str(found[7])) # Cv
foundCp = re.findall('\d*\.?\d+',str(found[8])) # Cp

found2 = re.findall('vapor',str(found[13])) # Phase
if found2 == ['vapor']:
  phasefuel = found2[0]
else:
  found3 = re.findall('liquid',str(found[13]))
  if found3 == ['liquid']:
    phasefuel = found3[0]
else: print("Non liquid or gas phase, please enter a "+
  "different total pressure and/or temperature for the fuel")

foundDensityprocessing = foundDensity[0].split('.') # Density
pt1 = int(foundDensityprocessing[0])
pt2 = int(foundDensityprocessing[1])
length = (int(len(foundDensityprocessing[1])))
pt2_decimal = pt2 / (10**length)
densityfuel = pt1 + pt2_decimal

foundCvprocessing = foundCv[0].split('.') # Cv
pt1 = int(foundCvprocessing[0])
pt2 = int(foundCvprocessing[1])
```

```
    length = (int(len(foundCvprocessing[1])))
    pt2_decimal = pt2 / (10**length)
    Cvfuel = pt1 + pt2_decimal

    foundCpprocessing = foundCp[0].split('.') # Cv
    pt1 = int(foundCpprocessing[0])
    pt2 = int(foundCpprocessing[1])
    length = (int(len(foundCpprocessing[1])))
    pt2_decimal = pt2 / (10**length)
    Cpfuel = pt1 + pt2_decimal

  elif fuel == "C2H4": # C74851

    Rspesificfuel = 296.38197 # spesific gas constant

    P = ptfuel / 1000000 # MPa
    T = Ttfuel # K
    p_url = str(P)
    t_url = str(T)
    t_maxval = T + 10
    t_max_url = str(t_maxval)

    url =  "https://webbook.nist.gov/cgi/fluid.cgi?Action="+
    "Load&ID=C74851&Type=IsoBar&Digits=9&P=" + p_url +
    "&THigh=" + t_max_url + "&TLow=" + t_url +
    "&TInc=10&RefState=DEF&TUnit=K&PUnit=MPa&DUnit="+
    "kg%2Fm3&HUnit=kJ%2Fmol&WUnit=m%2Fs&VisUnit=Pa*s&STUnit=N%2Fm"
    response = requests.get(url)

    soup = BeautifulSoup(response.text, "html.parser")

    found = soup.findAll('td')
    foundDensity = re.findall('\d*\.?\d+',str(found[2])) # Density
    foundCv = re.findall('\d*\.?\d+',str(found[7])) # Cv
    foundCp = re.findall('\d*\.?\d+',str(found[8])) # Cp

    found2 = re.findall('vapor',str(found[13])) # Phase
    if found2 == ['vapor']:
```

```
      phasefuel = found2[0]
    else:
      found3 = re.findall('liquid',str(found[13]))
      if found3 == ['liquid']:
        phasefuel = found3[0]
      else: print("Non liquid or gas phase, please enter a "+
      "different total pressure and/or temperature for the fuel")

    foundDensityprocessing = foundDensity[0].split('.') # Density
    pt1 = int(foundDensityprocessing[0])
    pt2 = int(foundDensityprocessing[1])
    length = (int(len(foundDensityprocessing[1])))
    pt2_decimal = pt2 / (10**length)
    densityfuel = pt1 + pt2_decimal

    foundCvprocessing = foundCv[0].split('.') # Cv
    pt1 = int(foundCvprocessing[0])
    pt2 = int(foundCvprocessing[1])
    length = (int(len(foundCvprocessing[1])))
    pt2_decimal = pt2 / (10**length)
    Cvfuel = pt1 + pt2_decimal

    foundCpprocessing = foundCp[0].split('.') # Cv
    pt1 = int(foundCpprocessing[0])
    pt2 = int(foundCpprocessing[1])
    length = (int(len(foundCpprocessing[1])))
    pt2_decimal = pt2 / (10**length)
    Cpfuel = pt1 + pt2_decimal

  else:
    print("please enter a different fuel")

  # if a vapor (i.e. a gas, use ideal gas approach to
  #calculate mass flow rate, if liquid use
  #incompressible approach)

  Sfuel = (ninjector * (np.pi / 4) *
  ((dfuel/1000)**2)) # fuel injection area
```

```
if phasefuel == "vapor":

  gammafuel = Cpfuel / Cvfuel # specific heat ratio

  psfuel = ptfuel*((1 + (((gammafuel - 1) / 2) *
  (Machfuel**2)))**(-1*(gammafuel/(gammafuel-1))))
  Tsfuel = (Ttfuel*((1 + (((gammafuel - 1) / 2) *
  (Machfuel**2))))**(-1))

  pt1fuel = ptfuel
  Tt1fuel = Ttfuel

  massflowratefuel = (np.sqrt(gammafuel / Rspesificfuel)
  * ( pt1fuel / np.sqrt(Tt1fuel) ) * Sfuel * Machfuel *
  ((1 + (((gammafuel - 1) / 2) * (Machfuel**2)))**
  (-1*((gammafuel + 1)/(2 * (gammafuel - 1))))))

else:

  massflowratefuel = Sfuel * densityfuel * velocityfuel
  Tt1fuel=0
  pt1fuel=0

if oxidiser == "Water": # C7732185

  Rspesificoxidiser = 461.52230 # spesific gas constant

  P = ptoxidiser / 1000000 # MPa
  T = Ttoxidiser # K
  p_url = str(P)
  t_url = str(T)
  t_maxval = T + 10
  t_max_url = str(t_maxval)

  url =  "https://webbook.nist.gov/cgi/fluid.cgi?Action="+
  "Load&ID=C7732185&Type=IsoBar&Digits=9&P=" + p_url +
  "&THigh=" + t_max_url + "&TLow=" + t_url +
```

```
"&TInc=10&RefState=DEF&TUnit=K&PUnit=MPa&DUnit="+
"kg%2Fm3&HUnit=kJ%2Fmol&WUnit=m%2Fs&VisUnit=Pa*s&STUnit=N%2Fm"
response = requests.get(url)

soup = BeautifulSoup(response.text, "html.parser")

found = soup.findAll('td')
foundDensity = re.findall('\d*\.?\d+',str(found[2])) # Density
foundCv = re.findall('\d*\.?\d+',str(found[7])) # Cv
foundCp = re.findall('\d*\.?\d+',str(found[8])) # Cp

found2 = re.findall('vapor',str(found[13])) # Phase
if found2 == ['vapor']:
  phaseoxidiser = found2[0]
else:
  found3 = re.findall('liquid',str(found[13]))
  if found3 == ['liquid']:
    phaseoxidiser = found3[0]
  else: print("Not gas phase, please enter a different"+
" total pressure and/or temperature for the oxidiser")

foundDensityprocessing = foundDensity[0].split('.') # Density
pt1 = int(foundDensityprocessing[0])
pt2 = int(foundDensityprocessing[1])
length = (int(len(foundDensityprocessing[1])))
pt2_decimal = pt2 / (10**length)
densityoxidiser = pt1 + pt2_decimal

foundCvprocessing = foundCv[0].split('.') # Cv
pt1 = int(foundCvprocessing[0])
pt2 = int(foundCvprocessing[1])
length = (int(len(foundCvprocessing[1])))
pt2_decimal = pt2 / (10**length)
Cvoxidiser = pt1 + pt2_decimal

foundCpprocessing = foundCp[0].split('.') # Cv
pt1 = int(foundCpprocessing[0])
pt2 = int(foundCpprocessing[1])
```

```
    length = (int(len(foundCpprocessing[1])))
    pt2_decimal = pt2 / (10**length)
    Cpoxidiser = pt1 + pt2_decimal

  elif oxidiser == "O2": # C7782447

    Rspesificoxidiser = 259.83670 # spesific gas constant

    P = ptoxidiser / 1000000 # MPa
    T = Ttoxidiser # K
    p_url = str(P)
    t_url = str(T)
    t_maxval = T + 10
    t_max_url = str(t_maxval)

    url =  "https://webbook.nist.gov/cgi/fluid.cgi?Action="+
    "Load&ID=C7782447&Type=IsoBar&Digits=9&P=" + p_url +
    "&THigh=" + t_max_url + "&TLow=" + t_url +
    "&TInc=10&RefState=DEF&TUnit=K&PUnit=MPa&DUnit="+
    "kg%2Fm3&HUnit=kJ%2Fmol&WUnit=m%2Fs&VisUnit=Pa*s&STUnit=N%2Fm"
    response = requests.get(url)

    soup = BeautifulSoup(response.text, "html.parser")

    found = soup.findAll('td')
    foundDensity = re.findall('\d*\.?\d+',str(found[2])) # Density
    foundCv = re.findall('\d*\.?\d+',str(found[7])) # Cv
    foundCp = re.findall('\d*\.?\d+',str(found[8])) # Cp

    found2 = re.findall('vapor',str(found[13])) # Phase
    if found2 == ['vapor']:
      phaseoxidiser = found2[0]
    else:
      found3 = re.findall('liquid',str(found[13]))
      if found3 == ['liquid']:
        phaseoxidiser = found3[0]
      else: print("Not gas phase, please enter a different"+
    " total pressure and/or temperature for the oxidiser")
```

```
    foundDensityprocessing = foundDensity[0].split('.') # Density
    pt1 = int(foundDensityprocessing[0])
    pt2 = int(foundDensityprocessing[1])
    length = (int(len(foundDensityprocessing[1])))
    pt2_decimal = pt2 / (10**length)
    densityoxidiser = pt1 + pt2_decimal

    foundCvprocessing = foundCv[0].split('.') # Cv
    pt1 = int(foundCvprocessing[0])
    pt2 = int(foundCvprocessing[1])
    length = (int(len(foundCvprocessing[1])))
    pt2_decimal = pt2 / (10**length)
    Cvoxidiser = pt1 + pt2_decimal

    foundCpprocessing = foundCp[0].split('.') # Cv
    pt1 = int(foundCpprocessing[0])
    pt2 = int(foundCpprocessing[1])
    length = (int(len(foundCpprocessing[1])))
    pt2_decimal = pt2 / (10**length)
    Cpoxidiser = pt1 + pt2_decimal

  elif oxidiser == "Air":
    # scrapes individual components Cp and Cv, and checks
    #if the component is a vapor (gas) to keep air
    #only in a gaseous state

    Rspesificoxidiser = 287.00568 # spesific gas constant

    P = ptoxidiser / 1000000 # MPa
    T = Ttoxidiser # K
    p_url = str(P)
    t_url = str(T)
    t_maxval = T + 10
    t_max_url = str(t_maxval)

    #O2 C7782447
```

```
url =  "https://webbook.nist.gov/cgi/fluid.cgi?Action="+
"Load&ID=C7782447&Type=IsoBar&Digits=9&P=" + p_url +
"&THigh=" + t_max_url + "&TLow=" + t_url +
"&TInc=10&RefState=DEF&TUnit=K&PUnit=MPa&DUnit="+
"kg%2Fm3&HUnit=kJ%2Fmol&WUnit=m%2Fs&VisUnit=Pa*s&STUnit=N%2Fm"
response = requests.get(url)

soup = BeautifulSoup(response.text, "html.parser")

found = soup.findAll('td')
foundCv = re.findall('\d*\.?\d+',str(found[7])) # Cv
foundCp = re.findall('\d*\.?\d+',str(found[8])) # Cp

found2 = re.findall('vapor',str(found[13])) # Phase
if found2 == ['vapor']:
  phaseoxidiser = found2[0]
else: print("Not gas phase, please enter a different"+
" total pressure and/or temperature for the oxidiser")

foundCvprocessing = foundCv[0].split('.') # Cv
pt1 = int(foundCvprocessing[0])
pt2 = int(foundCvprocessing[1])
length = (int(len(foundCvprocessing[1])))
pt2_decimal = pt2 / (10**length)
CvO2 = pt1 + pt2_decimal

foundCpprocessing = foundCp[0].split('.') # Cv
pt1 = int(foundCpprocessing[0])
pt2 = int(foundCpprocessing[1])
length = (int(len(foundCpprocessing[1])))
pt2_decimal = pt2 / (10**length)
CpO2 = pt1 + pt2_decimal

#N2 C7727379

url =  "https://webbook.nist.gov/cgi/fluid.cgi?Action="+
"Load&ID=C7727379&Type=IsoBar&Digits=9&P=" + p_url +
"&THigh=" + t_max_url + "&TLow=" + t_url +
```

```
"&TInc=10&RefState=DEF&TUnit=K&PUnit=MPa&DUnit="+
"kg%2Fm3&HUnit=kJ%2Fmol&WUnit=m%2Fs&VisUnit=Pa*s&STUnit=N%2Fm"
response = requests.get(url)

soup = BeautifulSoup(response.text, "html.parser")

found = soup.findAll('td')
foundCv = re.findall('\d*\.?\d+',str(found[7])) # Cv
foundCp = re.findall('\d*\.?\d+',str(found[8])) # Cp

found2 = re.findall('vapor',str(found[13])) # Phase
if found2 == ['vapor']:
  phaseoxidiser = found2[0]
else: print("Not gas phase, please enter a different"+
" total pressure and/or temperature for the oxidiser")

foundCvprocessing = foundCv[0].split('.') # Cv
pt1 = int(foundCvprocessing[0])
pt2 = int(foundCvprocessing[1])
length = (int(len(foundCvprocessing[1])))
pt2_decimal = pt2 / (10**length)
CvN2 = pt1 + pt2_decimal

foundCpprocessing = foundCp[0].split('.') # Cv
pt1 = int(foundCpprocessing[0])
pt2 = int(foundCpprocessing[1])
length = (int(len(foundCpprocessing[1])))
pt2_decimal = pt2 / (10**length)
CpN2 = pt1 + pt2_decimal

#Ar C7440371

url =  "https://webbook.nist.gov/cgi/fluid.cgi?Action="+
"Load&ID=C7440371&Type=IsoBar&Digits=9&P=" + p_url +
"&THigh=" + t_max_url + "&TLow=" + t_url +
"&TInc=10&RefState=DEF&TUnit=K&PUnit=MPa&DUnit="+
"kg%2Fm3&HUnit=kJ%2Fmol&WUnit=m%2Fs&VisUnit=Pa*s&STUnit=N%2Fm"
response = requests.get(url)
```

```
    soup = BeautifulSoup(response.text, "html.parser")

    found = soup.findAll('td')
    foundCv = re.findall('\d*\.?\d+',str(found[7])) # Cv
    foundCp = re.findall('\d*\.?\d+',str(found[8])) # Cp

    found2 = re.findall('vapor',str(found[13])) # Phase
    if found2 == ['vapor']:
      phaseoxidiser = found2[0]
    else: print("Not gas phase, please enter a different"+
    " total pressure and/or temperature for the oxidiser")

    foundCvprocessing = foundCv[0].split('.') # Cv
    pt1 = int(foundCvprocessing[0])
    pt2 = int(foundCvprocessing[1])
    length = (int(len(foundCvprocessing[1])))
    pt2_decimal = pt2 / (10**length)
    CvAr = pt1 + pt2_decimal

    foundCpprocessing = foundCp[0].split('.') # Cv
    pt1 = int(foundCpprocessing[0])
    pt2 = int(foundCpprocessing[1])
    length = (int(len(foundCpprocessing[1])))
    pt2_decimal = pt2 / (10**length)
    CpAr = pt1 + pt2_decimal

    # Air = 0.78N2 + 0.21O2 + 0.01Ar, Rule of mixtures

    Cvoxidiser = 0.78*CvN2 + 0.21*CvO2 + 0.01*CvAr
    Cpoxidiser = 0.78*CpN2 + 0.21*CpO2 + 0.01*CpAr

  else: print("please enter a different oxidiser")

  # if a vapor (i.e. a gas, use ideal gas approach to
  #calculate mass flow rate, if liquid use incompressible approach)

  Soxidiser = (ninjector * (np.pi / 4) *
```

```
((doxidiser/1000)**2)0 # oxidiser injection area

if phaseoxidiser == "vapor":

  gammaoxidiser = Cpoxidiser / Cvoxidiser # specific heat ratio

  psoxidiser = (ptoxidiser*((1 + (((gammaoxidiser - 1) / 2) *
  (Machoxidiser**2)))**(-1*(gammaoxidiser/(gammaoxidiser-1))))
  Tsoxidiser = Ttoxidiser*((1 + (((gammaoxidiser - 1) / 2) *
  (Machoxidiser**2))))**(-1))

  pt1oxidiser = ptoxidiser #/ ((1 + (((gammaoxidiser - 1) / 2) *
  (Machoxidiser**2)))**(-1*(gammaoxidiser/(gammaoxidiser-1))))
  Tt1oxidiser = Ttoxidiser #/ ((1 + (((gammaoxidiser - 1) / 2) *
  (Machoxidiser**2)))**(-1*(gammaoxidiser/(gammaoxidiser-1))))

  massflowrateoxidiser =( np.sqrt(
  gammaoxidiser / Rspesificoxidiser) *
  ( pt1oxidiser /
  np.sqrt(Tt1oxidiser) ) * Soxidiser * Machoxidiser *
  ((1 + (((gammaoxidiser - 1) / 2) *
  (Machoxidiser**2)))**
  (-1*((gammaoxidiser + 1)/(2 * (gammaoxidiser - 1))))))

else:

  massflowrateoxidiser =( Soxidiser * densityoxidiser
  * velocityoxidiser
  pt1oxidiser=0)
  Tt1oxidiser=0
if fuel == "H2" and oxidiser == "O2":
  st = 7.936682739
  # stoichiometric ratio (oxidiser to fuel ratio (oxidiser/fuel))
elif fuel == "CH4" and oxidiser == "O2":
  st = 3.89635312
  # stoichiometric ratio (oxidiser to fuel ratio (oxidiser/fuel))
elif fuel == "C2H4" and oxidiser == "O2":
  st = 3.421941169
```

```
  # stoichiometric ratio (oxidiser to fuel ratio (oxidiser/fuel))
elif fuel == "Water" and oxidiser == "O2":
  st = 0
  # stoichiometric ratio (oxidiser to fuel ratio (oxidiser/fuel))
elif fuel == "H2" and oxidiser == "Air":
  st = 34.21607657
  # stoichiometric ratio (oxidiser to fuel ratio (oxidiser/fuel))
elif fuel == "CH4" and oxidiser == "Air":
  st = 16.79768754
  # stoichiometric ratio (oxidiser to fuel ratio (oxidiser/fuel))
elif fuel == "C2H4" and oxidiser == "Air":
  st = 14.75243561
  # stoichiometric ratio (oxidiser to fuel ratio (oxidiser/fuel))
elif fuel == "Water" and oxidiser == "Air":
  st = 0
  # stoichiometric ratio (oxidiser to fuel ratio (oxidiser/fuel))
elif fuel == "H2" and oxidiser == "Water":
  st = 0
  # stoichiometric ratio (oxidiser to fuel ratio (oxidiser/fuel))
elif fuel == "CH4" and oxidiser == "Water":
  st = 0
  # stoichiometric ratio (oxidiser to fuel ratio (oxidiser/fuel))
elif fuel == "C2H4" and oxidiser == "Water":
  st = 0
  # stoichiometric ratio (oxidiser to fuel ratio (oxidiser/fuel))
elif fuel == "Water" and oxidiser == "Water":
  st = 0
  # stoichiometric ratio (oxidiser to fuel ratio (oxidiser/fuel))

globalequivilenceratio = (st *
(massflowratefuel/massflowrateoxidiser))
totalmassflowrate = massflowratefuel + massflowrateoxidiser
peroxidisernjector = massflowrateoxidiser / ninjector
perfuelinjector = massflowratefuel / ninjector
massfluxfuel = massflowratefuel / Sfuel
massfluxoxidiser = massflowrateoxidiser / Soxidiser

return (globalequivilenceratio,totalmassflowrate,
```

```
  massflowratefuel,massflowrateoxidiser,perfuelinjector,
  peroxidisernjector,massfluxfuel,massfluxoxidiser,psfuel
  ,Tsfuel,psoxidiser,Tsoxidiser)

def geometry(dfuel,doxidiser,alpha,beta,sigma,a,b):

  d1 = dfuel
  d2 = doxidiser

  r1 = d1 / 2
  r2 = d2 / 2

  D1 = (d1 / (np.cos(np.radians(alpha)))) / np.cos
  (np.radians(-1 * sigma))
  D2 = (d2 / (np.cos(np.radians(alpha)))) / np.cos
  (np.radians(-1 * sigma))

  R1 = D1 / 2
  R2 = D2 / 2

  mu = (np.degrees(np.arctan((np.sin(np.radians
  (-1 * sigma))*(np.sin(np.radians(90 -
  alpha)))/(np.cos(np.radians(90 - alpha)))))))
  c = np.radians(mu)

  rfuel = (R1 * r1) / np.sqrt((r1*np.cos(
  np.radians(-1* beta - mu)))**2 + (R1*np.sin(
  np.radians(-1* beta - mu)))**2)
  roxidiser = (R2 * r2) / np.sqrt((r2*np.cos(
  np.radians(-1* beta - mu)))**2 + (R2*np.sin(
  np.radians(-1* beta - mu)))**2)
  r3 = rfuel + roxidiser

  rx = r3 * np.cos(np.radians(-1 * beta))
  ry = r3 * np.sin(np.radians(beta))
  #equates equaition of rotated ellipse and its
  #derivative (ie the sum of the two degrees of freedom
  #on the cross section of the inlet cylinder))
```

```
#to find the maximum distances from the centre

x,y = sym.symbols('x,y')
eq1 = sym.Eq(((x*sym.cos(c)+y*sym.sin(c))/(R1))**2 +
((-1*x*sym.sin(c)+y*sym.cos(c))/(r1))**2,1)
eq2 = sym.Eq(( 2*sym.cos(c) * (
x*sym.cos(c) + y*sym.sin(c) ) ) / R1**2) - ((
2*sym.sin(c) * (-1*x*sym.sin(c) + y*sym.cos(
c) ) ) / r1**2) ,0)
result = sym.solve([eq1,eq2],(x,y))

ymaxfuel = result[0][1]


x,y = sym.symbols('x,y')
eq1 = sym.Eq(((x*sym.cos(c)+y*sym.sin(c))/(R1))**2 +
((-1*x*sym.sin(c)+y*sym.cos(c))/(r1))**2,1)
eq2 = sym.Eq(( 2*sym.sin(c) * (
x*sym.cos(c) + y*sym.sin(c) ) ) / R1**2) + (
( 2*sym.cos(c) * (-1*x*sym.sin(c) + y*sym.cos(
c) ) ) / r1**2) ,0)
result = sym.solve([eq1,eq2],(x,y))


xmaxfuel = result[1][0]


x,y = sym.symbols('x,y')
eq1 = sym.Eq(((x*sym.cos(c)+y*sym.sin(c))/(R2))**2 +
((-1*x*sym.sin(c)+y*sym.cos(c))/(r2))**2,1)
eq2 = sym.Eq(( 2*sym.cos(c) * (x*sym.cos(c) + y*sym.sin(
c) ) ) / R2**2) - (( 2*sym.sin(c) * (
-1*x*sym.sin(c) + y*sym.cos(c) ) ) / r2**2) ,0)
result = sym.solve([eq1,eq2],(x,y))


ymaxoxidiser = result[0][1]
```

```
x,y = sym.symbols('x,y')
eq1 = sym.Eq(((x*sym.cos(c)+y*sym.sin(c))/(R2))**2 +
((-1*x*sym.sin(c)+y*sym.cos(c))/(r2))**2,1)
eq2 = sym.Eq(( 2*sym.sin(c) * (x*sym.cos(
c) + y*sym.sin(c) ) ) / R2**2) + (( 2*sym.cos(
c) * (-1*x*sym.sin(c) + y*sym.cos(c) ) ) / r2**2) ,0)
result = sym.solve([eq1,eq2],(x,y))


xmaxoxidiser = result[1][0]

if ymaxfuel < 0:
  ymaxfuel = ymaxfuel* -1
if xmaxfuel < 0:
  xmaxfuel = xmaxfuel* -1
if ymaxoxidiser < 0:
  ymaxoxidiser = ymaxoxidiser* -1
if xmaxoxidiser < 0:
  xmaxoxidiser = xmaxoxidiser* -1

if  ymaxfuel < (ymaxoxidiser - ry):
  ymaxfuel = ymaxoxidiser

if  xmaxfuel < (xmaxoxidiser - rx):
  xmaxfuel = xmaxoxidiser

la = rx + xmaxfuel + xmaxoxidiser
lb = ry + ymaxfuel + ymaxoxidiser

a1 = (a - la) / 2
b1 = (b - lb) / 2


a2 = a1 + xmaxfuel
b2 = b1 + ymaxfuel

a3 = a2 + rx
```

```
  b3 = b2 + ry

  a4 = a3 + xmaxoxidiser
  b4 = b3 + ymaxoxidiser

  return a1,b1,a2,b2,a3,b3,a4,b4

#test1 = equivilence_ratio("H2","Air",
304000,314000,300,300,1,3,0.8,0.8,1,1,60)
test1 = equivilence_ratio("H2","Air",
304000,314000,300,300,1,3,0.8,0.8,1,1,60)
test2 = geometry(1,3,15,45,15,5.233,5)
print(test1)
print(test2)
```

Appendix B

Post-Processing Python3 Code

```
from zipfile import ZipFile

# Create a ZipFile Object and load sample.zip in it
with ZipFile('5e-5m1e-5mmesh.zip', 'r') as zipObj:
#eg. /content/GaseousSIIMethodValidationStudy_Data.zip
   # Extract all the contents of zip file in current directory
   zipObj.extractall()

import pandas as pd
import numpy as np
import glob
#H2 + O2 version

st = 7.936682739 #stochiometric ratio
opper = 100000 #Pa operational pressure
dh2 = 1
do2 = 1.41
h2mflux = 85.4
o2mflux =  340.2



i = []
j = []
k = []

#presinj
```

```
presdirlist = glob.glob("/content/*p8_*.csv")
df1t0 = pd.read_csv(presdirlist[0],skiprows=5)
df1t1 = pd.read_csv(presdirlist[1],skiprows=5)
df1t2 = pd.read_csv(presdirlist[2],skiprows=5)
df1t3 = pd.read_csv(presdirlist[3],skiprows=5)
df1t4 = pd.read_csv(presdirlist[4],skiprows=5)
df1t5 = pd.read_csv(presdirlist[5],skiprows=5)
df1t6 = pd.read_csv(presdirlist[6],skiprows=5)
df1t7 = pd.read_csv(presdirlist[7],skiprows=5)
df1t8 = pd.read_csv(presdirlist[8],skiprows=5)
df1t9 = pd.read_csv(presdirlist[9],skiprows=5)
df1t10 = pd.read_csv(presdirlist[10],skiprows=5)
#time averaging
df1 = (df1t0 + df1t1 + df1t2 + df1t3 + df1t4 + df1t5
+ df1t6 + df1t7 + df1t8 + df1t9 + df1t10)
dfp8 = df1.div(11)
presdirlist = glob.glob("/content/*p81_*.csv")
df2t0 = pd.read_csv(presdirlist[0],skiprows=5)
df2t1 = pd.read_csv(presdirlist[1],skiprows=5)
df2t2 = pd.read_csv(presdirlist[2],skiprows=5)
df2t3 = pd.read_csv(presdirlist[3],skiprows=5)
df2t4 = pd.read_csv(presdirlist[4],skiprows=5)
df2t5 = pd.read_csv(presdirlist[5],skiprows=5)
df2t6 = pd.read_csv(presdirlist[6],skiprows=5)
df2t7 = pd.read_csv(presdirlist[7],skiprows=5)
df2t8 = pd.read_csv(presdirlist[8],skiprows=5)
df2t9 = pd.read_csv(presdirlist[9],skiprows=5)
df2t10 = pd.read_csv(presdirlist[10],skiprows=5)
#time averaging
df1 = (df2t0 + df2t1 + df2t2 + df2t3 + df2t4 + df2t5
+ df2t6 + df2t7 + df2t8 + df2t9 + df2t10)
dfp81 = df1.div(11)
presdirlist = glob.glob("/content/*p811_*.csv")
df3t0 = pd.read_csv(presdirlist[0],skiprows=5)
df3t1 = pd.read_csv(presdirlist[1],skiprows=5)
df3t2 = pd.read_csv(presdirlist[2],skiprows=5)
df3t3 = pd.read_csv(presdirlist[3],skiprows=5)
df3t4 = pd.read_csv(presdirlist[4],skiprows=5)
```

```
df3t5 = pd.read_csv(presdirlist[5],skiprows=5)
df3t6 = pd.read_csv(presdirlist[6],skiprows=5)
df3t7 = pd.read_csv(presdirlist[7],skiprows=5)
df3t8 = pd.read_csv(presdirlist[8],skiprows=5)
df3t9 = pd.read_csv(presdirlist[9],skiprows=5)
df3t10 = pd.read_csv(presdirlist[10],skiprows=5)
#time averaging
df1 =( df3t0 + df3t1 + df3t2 + df3t3 + df3t4 + df3t5
+ df3t6 + df3t7 + df3t8 + df3t9 + df3t10)
dfp811 = df1.div(11)
presdirlist = glob.glob("/content/*p9_*.csv")
df4t0 = pd.read_csv(presdirlist[0],skiprows=5)
df4t1 = pd.read_csv(presdirlist[1],skiprows=5)
df4t2 = pd.read_csv(presdirlist[2],skiprows=5)
df4t3 = pd.read_csv(presdirlist[3],skiprows=5)
df4t4 = pd.read_csv(presdirlist[4],skiprows=5)
df4t5 = pd.read_csv(presdirlist[5],skiprows=5)
df4t6 = pd.read_csv(presdirlist[6],skiprows=5)
df4t7 = pd.read_csv(presdirlist[7],skiprows=5)
df4t8 = pd.read_csv(presdirlist[8],skiprows=5)
df4t9 = pd.read_csv(presdirlist[9],skiprows=5)
df4t10 = pd.read_csv(presdirlist[10],skiprows=5)
#time averaging
df1 = (df4t0 + df4t1 + df4t2 + df4t3 + df4t4 + df4t5
+ df4t6 + df4t7 + df4t8 + df4t9 + df4t10)
dfp9 = df1.div(11)
presdirlist = glob.glob("/content/*p91_*.csv")
df5t0 = pd.read_csv(presdirlist[0],skiprows=5)
df5t1 = pd.read_csv(presdirlist[1],skiprows=5)
df5t2 = pd.read_csv(presdirlist[2],skiprows=5)
df5t3 = pd.read_csv(presdirlist[3],skiprows=5)
df5t4 = pd.read_csv(presdirlist[4],skiprows=5)
df5t5 = pd.read_csv(presdirlist[5],skiprows=5)
df5t6 = pd.read_csv(presdirlist[6],skiprows=5)
df5t7 = pd.read_csv(presdirlist[7],skiprows=5)
df5t8 = pd.read_csv(presdirlist[8],skiprows=5)
df5t9 = pd.read_csv(presdirlist[9],skiprows=5)
df5t10 = pd.read_csv(presdirlist[10],skiprows=5)
```

```
#time averaging
df1 = (df5t0 + df5t1 + df5t2 + df5t3 + df5t4 + df5t5
+ df5t6 + df5t7 + df5t8 + df5t9 + df5t10)
dfp91 = df1.div(11)
presdirlist = glob.glob("/content/*p911_*.csv")
df6t0 = pd.read_csv(presdirlist[0],skiprows=5)
df6t1 = pd.read_csv(presdirlist[1],skiprows=5)
df6t2 = pd.read_csv(presdirlist[2],skiprows=5)
df6t3 = pd.read_csv(presdirlist[3],skiprows=5)
df6t4 = pd.read_csv(presdirlist[4],skiprows=5)
df6t5 = pd.read_csv(presdirlist[5],skiprows=5)
df6t6 = pd.read_csv(presdirlist[6],skiprows=5)
df6t7 = pd.read_csv(presdirlist[7],skiprows=5)
df6t8 = pd.read_csv(presdirlist[8],skiprows=5)
df6t9 = pd.read_csv(presdirlist[9],skiprows=5)
df6t10 = pd.read_csv(presdirlist[10],skiprows=5)
#time averaging
df1 = (df6t0 + df6t1 + df6t2 + df6t3 + df6t4 + df6t5
+ df6t6 + df6t7 + df6t8 + df6t9 + df6t10)
dfp911 = df1.div(11)

areah2 = (np.pi/4)*((dh2/1000)**2)
areao2 = (np.pi/4)*((do2/1000)**2)

mh2 = h2mflux * areah2
mo2 = o2mflux * areao2

h2i1 = dfp8.append(dfp81,ignore_index=True)
h2i = h2i1.append(dfp811,ignore_index=True)

h2pres = h2i[" Total Pressure [ Pa ]"].mean()
h2prestotal = h2pres + opper

o2i1 = dfp9.append(dfp91,ignore_index=True)
o2i = o2i1.append(dfp911,ignore_index=True)

o2pres = o2i[" Total Pressure [ Pa ]"].mean()
o2prestotal = o2pres + opper
```

```
ptinj = ((h2prestotal*mh2) + (o2prestotal*mo2)) / (mh2 + mo2)


h2m = h2i[" Mass Flow [ kg s^-1 ]"].sum()
o2m = o2i[" Mass Flow [ kg s^-1 ]"].sum()



for x in range(1,8):
  y = str(x)

  dirlist = glob.glob("/content/*p" + y + "*.csv")
  dirlist.sort()
  dft0 = pd.read_csv(dirlist[0],skiprows=5)
  dft1 = pd.read_csv(dirlist[1],skiprows=5)
  dft2 = pd.read_csv(dirlist[2],skiprows=5)
  dft3 = pd.read_csv(dirlist[3],skiprows=5)
  dft4 = pd.read_csv(dirlist[4],skiprows=5)
  dft5 = pd.read_csv(dirlist[5],skiprows=5)
  dft6 = pd.read_csv(dirlist[6],skiprows=5)
  dft7 = pd.read_csv(dirlist[7],skiprows=5)
  dft8 = pd.read_csv(dirlist[8],skiprows=5)
  dft9 = pd.read_csv(dirlist[9],skiprows=5)
  dft10 = pd.read_csv(dirlist[10],skiprows=5)

  #time averaging
  df1 = (dft0 + dft1 + dft2 + dft3 + dft4 + dft5
  + dft6 + dft7 + dft8 + dft9 + dft10)
  df2 = df1.div(11)

  #equivilence ratio
  dfer1 = df2[" H2.Mass Fraction"] / df2[" O2.Mass Fraction"]
  dfer = dfer1.mul(st)


  # max/min functions
  dfmax = pd.DataFrame(np.where(dfer < 1, 1, dfer))
```

```
dfmin = pd.DataFrame(np.where(dfer > 1, 1, dfer))

#Yo2 * (max and min) * mass flow rate
df3 = dfmax * df2[" O2.Mass Fraction"]*df2[" Mass Flow [ kg s^-1 ]"]
df3 = pd.DataFrame(df3[0])
df4 = dfmin * df2[" O2.Mass Fraction"]*df2[" Mass Flow [ kg s^-1 ]"]
df4 = pd.DataFrame(df4[0])

#mix efficiency approximation
bottom = df3.sum()
top = df4.sum()
mixeff = top/bottom
i.append(mixeff)

#Y2 h2 mass fraction fluctuation
y2fluct0 = (df2[" H2.Mass Fraction"] -
dft0[" H2.Mass Fraction"])**2
y2fluct1 = (df2[" H2.Mass Fraction"] -
dft1[" H2.Mass Fraction"])**2
y2fluct2 = (df2[" H2.Mass Fraction"] -
dft2[" H2.Mass Fraction"])**2
y2fluct3 = (df2[" H2.Mass Fraction"] -
dft3[" H2.Mass Fraction"])**2
y2fluct4 = (df2[" H2.Mass Fraction"] -
dft4[" H2.Mass Fraction"])**2
y2fluct5 = (df2[" H2.Mass Fraction"] -
dft5[" H2.Mass Fraction"])**2
y2fluct6 = (df2[" H2.Mass Fraction"] -
dft6[" H2.Mass Fraction"])**2
y2fluct7 = (df2[" H2.Mass Fraction"] -
dft7[" H2.Mass Fraction"])**2
y2fluct8 = (df2[" H2.Mass Fraction"] -
dft8[" H2.Mass Fraction"])**2
y2fluct9 = (df2[" H2.Mass Fraction"] -
dft9[" H2.Mass Fraction"])**2
y2fluct10 = (df2[" H2.Mass Fraction"] -
dft10[" H2.Mass Fraction"])**2
```

```
  #time average
  y2flucsum = (y2fluct0 + y2fluct1 + y2fluct2 +
  y2fluct3 + y2fluct4 + y2fluct5 + y2fluct6 +
  y2fluct7 + y2fluct8 + y2fluct9 + y2fluct10 )
  y2flucavg = y2flucsum.div(11)
  y2flucsqr = y2flucavg**(0.5)
  y2flucmass = y2flucsqr*df2[" Mass Flow [ kg s^-1 ]"]
  y2mass = df2[" Mass Flow [ kg s^-1 ]"]
  bottom = y2mass.sum()
  top = y2flucmass.sum()
  y2eff = top/bottom
  j.append(y2eff)

  #Pressure Recovery Efficiency
  df11 = (df2[" Total Pressure [ Pa ]"]
  + opper) * df2[" Mass Flow [ kg s^-1 ]"]
  top = df11.sum()
  y2mass = df2[" Mass Flow [ kg s^-1 ]"]
  bottom = y2mass.sum()
  presef1 = top/bottom
  presef = presef1 / ptinj
  k.append(presef)



display(i,j,k)
```

Bibliography

[1] Vijay Anand and Ephraim Gutmark. Rotating detonation combustors and their similarities to rocket instabilities. *Progress in Energy and Combustion Science*, 73:182–234, 2019.

[2] Vijay Anand, Andrew St. George, Robert Driscoll, and Ephraim Gutmark. Investigation of rotating detonation combustor operation with h2-air mixtures. *International Journal of Hydrogen Energy*, 41(2):1281–1292, 2016.

[3] Vijay Anand, Andrew St. George, Charles Farbos de Luzan, and Ephraim Gutmark. Rotating detonation wave mechanics through ethylene-air mixtures in hollow combustors, and implications to high frequency combustion instabilities. *Experimental Thermal and Fluid Science*, 92:314–325, 2018.

[4] Ionio Q. Andrus, Paul I. King, Marc D. Polanka, Fred R. Schauer, and John L. Hoke. Design of a premixed fuelâ€“oxidizer system to arrest flashback in a rotating detonation engine. *Journal of Propulsion and Power*, 33(5):1063–1073, 2017.

[5] GlANFRANCO ANGELINO. Approximate method for plug nozzle design. *AIAA Journal*, 2(10):1834–1835, 1964.

[6] Ansys. 1.7.1 euler equations, Jan 2009.

[7] Ansys. 26.1 overview of using the solver, Jan 2009.

[8] Fedor A. Bykovskii, Sergey A. Zhdan, and Evgenii F. Vedernikov. Continuous spin detonations. *Journal of Propulsion and Power*, 22(6):1204–1216, 2006.

[9] Ralf Deiterding. Detonation structure simulation with amroc. In *Lecture Notes in Computer Science*, pages 916–927, 09 2005.

[10] Li Deng, Hu Ma, Can Xu, Xiao Liu, and Changsheng Zhou. The feasibility of mode control in rotating detonation engine. *Applied Thermal Engineering*, 129:1538–1550, 2018.

[11] Brian Dunbar. Launch america - a partnership between nasa and private space companies – will help open the space above earth to people besides government astronauts.

[12] Marjorie W. Evans and C. M. Ablow. Theories of detonation. *Chemical Reviews*, 61(2):129–178, 1961.

[13] T. Gaillard, D. Davidenko, and F. Dupoirieux. Numerical optimisation in non reacting conditions of the injector geometry for a continuous detonation wave rocket engine. *Acta Astronautica*, 111:334–344, 2015.

[14] Thomas Gaillard. *Étude numérique du fonctionnement d'un moteur à détonation rotative*. Theses, Université Paris Saclay (COmUE), March 2017.

[15] Kurt R. Glaesemann and Laurence E. Fried. Improved wood–kirkwood detonation chemical kinetics. *Theoretical Chemistry Accounts*, 120(1):37–43, May 2008.

[16] Siyuan Huang, Yangpeng Li, Jin Zhou, Shijie Liu, and Haoyang Peng. Effects of the pintle injector on h2/air continuous rotating detonation wave in a hollow chamber. *International Journal of Hydrogen Energy*, 44(26):14044–14054, 2019.

[17] Harry W. Jones. 48th international conference on environmental systems. In *The Recent Large Reduction in Space Launch Cost*. NASA Ames Research Center, 2018.

[18] Jan Kindracki, Stanislaw Siatkowski, and Borys Lukasik. Influence of inlet flow parameters on rotating detonation. *AIAA Journal*, 58(12):5046–5051, 2020.

[19] James Koch, Mitsuru Kurosaka, Carl Knowlen, and J. Nathan Kutz. Mode-locked rotating detonation waves: Experiments and a model equation. *Physical Review E*, 101(1), Jan 2020.

[20] James Koch, Mitsuru Kurosaka, Carl Knowlen, and J. Nathan Kutz. Multi-scale physics of rotating detonation engines: Autosolitons and modulational instabilities, 2020.

[21] Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21), 2021.

[22] Baoxing Li, Yuwen Wu, Chunsheng Weng, Quan Zheng, and Wanli Wei. Influence of equivalence ratio on the propagation characteristics of rotating detonation wave. *Experimental Thermal and Fluid Science*, 93:366–378, 2018.

[23] Qin Li, Pengxin Liu, and Hanxin Zhang. Further investigations on the interface instability between fresh injections and burnt products in 2-d rotating detonation. *Computers and Fluids*, 170:261–272, 2018.

[24] Jianhan Liang, Xiaodong Cai, Zhiyong Lin, and Ralf Deiterding. Effects of a hot jet on detonation initiation and propagation in supersonic combustible mixtures. *Acta Astronautica*, 105(1):265–277, 2014.

[25] Michael Liberman. *Introduction to Physics and Chemistry of Combustion*. Springer International Publishing, 01 2008.

[26] Wei Lin, Jin Zhou, Shijie Liu, and Zhiyong Lin. An experimental study on ch4/o2 continuously rotating detonation wave in a hollow combustion chamber. *Experimental Thermal and Fluid Science*, 62:122–130, 2015.

[27] Pengxin Liu, Qilong Guo, Dong Sun, Chen Li, and Hanxin Zhang. Wall effect on the flow structures of three-dimensional rotating detonation wave. *International Journal of Hydrogen Energy*, 45(53):29546–29559, 2020.

[28] Xiang-Yang Liu, Yan-Liang Chen, Zhi-Jie Xia, and Jian-Ping Wang. Numerical study of the reverse-rotating waves in rotating detonation engine with a hollow combustor. *Acta Astronautica*, 170:421–430, 2020.

[29] John Z. Ma, Ming-Yi Luan, Zhi-Jie Xia, Jian-Ping Wang, Shu-jie Zhang, Song-bai Yao, and Bing Wang. Recent progress, development trends, and consideration of continuous detonation engines. *AIAA Journal*, 58(12):4976–5035, 2020.

[30] Konrad Malik, Mateusz Å»bikowski, and Andrzej Teodorczyk. Detonation cell size model based on deep neural network for hydrogen, methane and propane mixtures with air and oxygen. *Nuclear Engineering and Technology*, 51(2):424–431, 2019.

[31] John Matson. Phased out: Obama's nasa budget would cancel constellation moon program, privatize manned launches, 2010.

[32] Ariana Mendible, James Koch, Henning Lange, Steven L. Brunton, and J. Nathan Kutz. Data-driven modeling of rotating detonation waves. *Physical Review Fluids*, 6(5), May 2021.

[33] JO NAPOLITANO. New argonne computational model to accelerate engine development for next-generation hypersonic flight: Argonne national laboratory, May 2019.

[34] NASA. Commercial crew program - essentials, 2021.

[35] J. A. NICHOLLS, R. E. CULLEN, and K. W. RAGLAND. Feasibility studies of a rotating detonation wave rocket motor. *Journal of Spacecraft and Rockets*, 3(6):893–898, 1966.

[36] NIST. Nist chemistry webbook, srd 69.

[37] Haoyang Peng, Weidong Liu, Shijie Liu, and Hailong Zhang. Experimental investigations on ethylene-air continuous rotating detonation wave in the hollow chamber with laval nozzle. *Acta Astronautica*, 151:137–145, 2018.

[38] K.W. Ragland and K.M. Bryden. *Combustion Engineering*. CRC Press, 2011.

[39] Mark C. Schnabel. *Pressure Distribution and Performance Impacts of Aerospike Nozzles on Rotating Detonation Engines*. PhD thesis, Naval Postgraduate School, 2017.

[40] Jonathan Sosa, Robert Burke, Kareem A. Ahmed, Daniel J. Micka, John W. Bennewitz, Stephen A. Danczyk, Eric J. Paulson, and William A. Hargus. Experimental evidence of h2/o2 propellants powered rotating detonation waves. *Combustion and Flame*, 214:136–138, 2020.

[41] Panagiotis Stathopoulos. Comprehensive thermodynamic analysis of the humphrey cycle for gas turbines with pressure gain combustion. *Energies*, 11(12), 2018.

[42] Jian Sun, Jin Zhou, Shijie Liu, and Zhiyong Lin. Numerical investigation of a rotating detonation engine under premixed/non-premixed conditions. *Acta Astronautica*, 152:630–638, 2018.

[43] Jian Sun, Jin Zhou, Shijie Liu, Zhiyong Lin, and Wei Lin. Plume flowfield and propulsive performance analysis of a rotating detonation engine. *Aerospace Science and Technology*, 81:383–393, 2018.

[44] Jian Sun, Jin Zhou, Shijie Liu, Zhiyong Lin, and Wei Lin. Effects of air injection throat width on a non-premixed rotating detonation engine. *Acta Astronautica*, 159:189–198, 2019.

[45] Jian Sun, Jin Zhou, Shijie Liu, Zhiyong Lin, and Wei Lin. Numerical investigation of a non-premixed hollow rotating detonation engine. *International Journal of Hydrogen Energy*, 44(31):17084–17094, 2019.

[46] William B. Todd (ed.). *The Glasgow Edition of the Works and Correspondence of Adam Smith, Vol. 2: An Inquiry into the Nature and Causes of the Wealth of Nations, Vol. 2*. Oxford University Press, 1976.

[47] Ucf. Ucf researchers develop groundbreaking new rocket-propulsion system, Apr 2020.

[48] Yuhui Wang. Rotating detonation in a combustor of trapezoidal cross section for the hydrogenâ€“air mixture. *International Journal of Hydrogen Energy*, 41(12):5605–5616, 2016.

[49] Yuhui Wang and Jialing Le. A hollow combustor that intensifies rotating detonation. *Aerospace Science and Technology*, 85:113–124, 2019.

[50] Zhi-Jie Xia, Ming-Yi Luan, Xiang-Yang Liu, and Jian-Ping Wang. Numerical simulation of wave mode transition in rotating detonation engine with openfoam. *International Journal of Hydrogen Energy*, 45(38):19989–19995, 2020.

[51] Zhijie Xia, Xinmeng Tang, Mingyi Luan, Shujie Zhang, Zhuang Ma, and Jianping Wang. Numerical investigation of two-wave collision and wave structure evolution of rotating detonation engine with hollow combustor. *International Journal of Hydrogen Energy*, 43(46):21582–21591, 2018.

[52] Qiaofeng Xie, Haocheng Wen, Weihong Li, Zifei Ji, Bing Wang, and Piotr Wolanski. Analysis of operating diagram for h2/air rotating detonation combustors under lean fuel condition. *Energy*, 151:408–419, 2018.

[53] Chian Yan, Honghui Teng, and Hoi Dick Ng. Effects of slot injection on detonation wavelet characteristics in a rotating detonation engine. *Acta Astronautica*, 182:274–285, 2021.

[54] Songbai Yao, Xinmeng Tang, Mingyi Luan, and Jianping Wang. Numerical study of hollow rotating detonation engine with different fuel injection area ratios. *Proceedings of the Combustion Institute*, 36(2):2649–2655, 2017.

[55] Songbai Yao and Jianping Wang. Multiple ignitions and the stability of rotating detonation waves. *Applied Thermal Engineering*, 108:927–936, 2016.

[56] Tae-Hyeong Yi, Jing Lou, Cary Kenny Turangan, and Piotr Wolanski. Numerical study of detonation processes in rotating detonation engine and its propulsive performance. *Transactions on Aerospace Research*, 2020(3):30–48, 2020.

[57] Si yuan Huang, Jin Zhou, Shi jie Liu, and Hao yang Peng. Effects of pintle injector on ethylene-air rocket-based continuous rotating detonation. *Acta Astronautica*, 164:311–320, 2019.

[58] Hailong Zhang, Luxin Jiang, Weidong Liu, and Shijie Liu. Characteristic of rotating detonation wave in the h2/air hollow chamber with laval nozzle. *International Journal of Hydrogen Energy*, 46(24):13389–13401, 2021.

[59] Hailong Zhang, Weidong Liu, and Shijie Liu. Experimental investigations on h2/air rotating detonation wave in the hollow chamber with laval nozzle. *International Journal of Hydrogen Energy*, 42(5):3363–3370, 2017.

[60] Majie Zhao and Huangwei Zhang. Large eddy simulation of non-reacting flow and mixing fields in a rotating detonation engine. *Fuel*, 280:118534, 2020.