LA-UR-20-25491
Approved for public release; distribution is unlimited.

Title: A Numerical Simulation of a Single Shock-Accelerated Particle

Author(s): Maxon, William Curtis

Intended for: Report

Issued: 2020-07-24

A NUMERICAL SIMULATION OF A SINGLE SHOCK-ACCELERATED PARTICLE

A Thesis presented to

the Faculty of the Graduate School

at the University of Missouri - Columbia

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

WILLIAM CURTIS MAXON

Dr. Jacob McFarland, Thesis Supervisor

July 2020

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled:

A NUMERICAL SIMULATION OF A SINGLE SHOCK-ACCELERATED PARTICLE

presented by W. Curtis Maxon,

a candidate for the degree of Master of Science and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Jacob A. McFarland

Dr. Tanner Nielsen

Dr. Carsten Ullrich

ACKNOWLEDGMENTS

This thesis is dedicated to William "Bill" Henry Maxon, who instilled the interest of thermal and mechanical systems in me from an early age.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

ABSTRACT

Particle drag models, which capture macro viscous and pressure effects, have been developed over the years for various flow regimes to enable cost effective simulations of particle-laden flows. The relatively recent derivation by Maxey and Riley has provided an exact equation of motion for spherical particles in a flow field based on the continuum assumption. Many models that have been simplified from these equations have provided reasonable approximations; however, the sensitivity of particle-laden flows to particle drag requires a very accurate model to simulate. To develop such a model, a 2D axisymmetric Navier-Stokes direct numerical simulation of a single particle in a transient, shock-driven flow field was conducted in the hydrocode FLAG. FLAG's capability to run arbitrary Lagrangian-Eulerian hydrodynamics coupled with solid mechanic models makes it an ideal code to capture the physics of the flow field around and in the particle as it is shock-accelerated – a challenging regime to study. The goal of this work is twofold: to provide a validation for FLAG's Navier-Stokes and heat diffusion solutions, and to provide a rationale for recent experimental particle drag measurements.

Chapter 1

Introduction

The motivation for this work comes from a series of experiments from the Extreme Fluids Team at Los Alamos National Laboratory. It has been observed, with previously unmatched experimental sophistication, that the drag experienced by small particles, when accelerated by a shock wave, is much higher than previous models predict. This result gave motivation to run a direct numerical simulation of the shock-accelerated particle while resolving the material of both the particle and the gas. The experiments were run using the horizontal shock tube facility running a Mach 1.3 shock into ambient conditions of the lab. The laboratory being located in Los Alamos, New Mexico requires less of the driver, and the piston-based shock tube they run is preferable to one that uses diaphragms. The experiments used 4μm diameter nylon particles and tracked them with a syncronized camera-laser system with a time resolution of 500ns. The reader is encouraged to view the work performed to gain the best understanding of the experiment to be replicated via simulations here [1].

From here, the governing equations are presented, followed by a description of the hydrodynamic instability that a better knowledge of particle drag attempts to shed light on. Then applications and models are described.

1.1 Governing Equations

The behavior of fluids often has the potential to be chaotic and difficult to describe. The mathimatical description of fluid mechanics comes from two conservation laws, the second law of thermodynamics, a constitutive relationship, and an equation of state. Conservation laws, as a result of Noether's theorem, are a result of symmetries, or invariants, of the action in the system. All equations are derived from applying the conservation laws to finite control volumes. Firstly, the principle of momentum conservation, which is a result of the symmetry of linear and rotational translations in space, yields Cauchy's first law of motion.

$$\rho \frac{d\mathbf{v}}{dt} = div(\overline{\overline{\sigma}}) + \rho \mathbf{f} \tag{1.1}$$

While very descriptive, this equation is far from well-posed as the stress tensor σ and velocity vector $\mathbf{v}$ add nine unknown quantities to the problem while only providing three equations. Losing generality of the equations, but progressing towards a well-posed problem, the following constitutive relationship is introduced.

$$\begin{aligned} \overline{\overline{\sigma}} &= -p\overline{\overline{\mathbf{I}}} + \overline{\overline{\tau}} \\ \overline{\overline{\tau}} &= \mu(-\frac{2}{3}\overline{\overline{\mathbf{I}}} + grad(\mathbf{v}) + grad(\mathbf{v})^T) \end{aligned} \tag{1.2}$$

Implicitly in this relationship, isotropy is assumed in the fluid. This assumption is appropriate for almost all fluids with few exceptions [2]. Furthermore, it is imposed in this relation that the bulk viscosity is zero, which is commonly known as the Stokes assumption, and equates the mechanical pressure to the thermodynamic pressure. This assumption is appropriate for flows that are free from extremely large rates of change in specific volume. Substituting the relation into Cauchy's first law gives the familiar Navier-Stokes momentum equations for a compressible fluid.

$$\rho\frac{d\mathbf{v}}{dt} = -grad(p) + div(\mu grad(\mathbf{v})) + \frac{1}{3}grad(\mu div(\mathbf{v})) \tag{1.3}$$

Conservation of energy is the next law to consider, and is a result of the symmetry of translations in time. This law provides two equations instead of one because of Einstein's equivalence of mass and energy, combined with the fact that the two will not change forms from one to another in this application. Conservation of mass is given as follows.

$$\frac{d\rho}{dt} + div(\rho\mathbf{b}) = 0 \tag{1.4}$$

Conservation of energy coupled with the fact that heat is a form of energy is the first law of thermodynamics. This principle applied to a continuum may be represented as follows.

$$\begin{gathered} \frac{dE}{dt} = q_{SUR=>SYS} - \overline{\overline{\sigma}} : \overline{\overline{\mathbf{D}}} \\ \rho\mathbf{f}\cdot\mathbf{v} + \rho\mathbf{v}\cdot\frac{d\mathbf{v}}{dt} + \rho\frac{d(Tc_v)}{dt} = div(\kappa grad(T)) + pdiv(\mathbf{v}) + \mu\Phi_{DIS} \\ \Phi_{DIS} = grad(\mathbf{v}) : grad(\mathbf{v}) + grad(\mathbf{v}) : grad(\mathbf{v})^T - \frac{2}{3}div(\mathbf{v})^2 \end{gathered} \tag{1.5}$$

Here, the work from a pressurized fluid expanding is generalized as the double inner product of the Cauchy stress tensor and the velocity gradient tensor. This includes the volume expansion work, but also includes work that is done due to shearing forces. In compressible fluids this component is usually negligibly small, however in ductile solids it has the potential to be the dominating component. Oftentimes when considering compressible flow, viscous diffusion is negligible in comparison to the other forces in the momentum equation, and the Navier-Stokes equations are transformed into the Euler fluid momentum equations. Even though the energy equation is less sensitive to inclusion of the viscous terms, it is important when solving the Navier-Stokes equations that viscosity is still considered in both the energy and momentum

equations. Otherwise the solution potentially fails to follow normal shock relations.

The second law of thermodynamics is used as a condition to weed out solutions to these equations that do not have physical meaning to them. This is refered to as the entropy condition and is mostly utilized in the numerical solution of the equations. In reality, shocks steepen as time goes on, becoming a spacial discontinuity of properties. One of the potential solutions that satisfy the equations will diffuse the length in which properties change over, and reports an incorrect wave velocity. The second law of thermodynamics dismisses this solution as invalid. It is worth noting that all variables with diffusion terms, i.e. those with μ and $\varkappa$ attached to them, must be positive and equated to a positive time rate of change of the correspondant variable for the second law to count it as a valid solution. This guarantees that heat does not travel from hot to cold spontaneously, and that heat will not convert entirely to kinetic energy. These are the Kelvin and Planck descriptions of the second law of thermodynamics, respectively. This is clearly the case in the momentum equation, but is less clear with the viscosity term in the energy equation because of the negative divergence squared term. The entire expression that is multiplied by the viscosity in the energy equation may be reduced to a sum of squares, thus guaranteeing that it is always positive. Because this law is not built on the conservation of a variable, it cannot be used to make the problem well-posed. However, it can be used to give insights to solutions that may not be physical, and why they exist.

It is important to note that in all differential equations presented here, the derivatives are taken constant with the material. From this, the chain rule must be applied, and in doing this it becomes explicit that the equations are nonlinear. This is unsurprising as the nature of fluids is often chaotic, which is consistent with the behavior of nonlinear dynamic systems. Unfortunately, nonlinear equations are not easily solved, and the fact there are six coupled equations constrains analytical solutions to very simple problems that do not provide many insights to the relevant physics. In order

to increase one's understanding of problems that have such complicated governing equations, oftentimes numerical solution is a powerful method.

It is worth noting that the validity of the Navier-Stokes equations, while in theory should describe the behavior of fluids perfectly within the continuum assumption, should not always be held as sacrosanct, and is still debated in the mathematical community. A proof that the Navier-Stokes equations either yield or do not yield smooth and unique solutions is one of seven millennium problems proposed by the Clay mathematical institute.

1.2 Shock-Driven Multiphase Instability

The shock-driven multiphase instability (SDMI), like all hydrodynamic instabilities (HIs), is a mechanism in which flow transitions from laminar to turbulent. The SDMI is an instability that, while not describable by a linear stability analysis like the Rayleigh-Taylor instability or the Kelvin-Helmholtz instability, is extremely widespread in nature. In addition to this fact, it is also very early in its stages of research and not much is known about its behavior. The SDMI appears in many human-created designs and is often the driving mechanism for the successful operation of the design. For example, in scramjet engines the liquid fuel particles are accelerated and mixed extremely quickly by incoming supersonic air. The SDMI is one of the ways that has the potential to describe the mixing of the fuel with the air. Additionally, in chemical weapons, it is always important to see where the hazardous material lands. These weapons typically work by detonating an explosive which propels a liquid, hazardous material.

The SDMI is a HI that necessarily has at least two phases present: One that is compressible, and the other that is either in a solid or liquid state, in the form of particles. The shock passes through the compressible phase, which holds the particles

in it as well. The region where particles are present may be treated as the region of larger density in the RMI [3]. This is due to the fact the density of the particles is always larger than that of the compressible phase. The RMI analogy becomes perfect as the diameters of the particles becomes infinitesimal. This fact is useful because continuum models that describe the particle kinematics break down as the diameters approach the mean-free path of the gas. The non-dimensional parameter that describes the jump in density at the interface for all of these HIs is known as the Atwood number. Unlike most non-dimensional parameters, the Atwood number does not come from the Buckingham-Π theorem but naturally appears in a first-order analysis of the RTI and RMI.

$$At = \frac{\rho_1 - \rho_2}{\rho_1 + \rho_2} \tag{1.6}$$

Here, ρ_1 is the density of the fluid within the interface and ρ_2 is the density of the fluid outside the interface. Notice that, unlike most non-dimensional numbers, that it is possible to have a negative Atwood number. The physical meaning of this is a morphology development in the opposite direction. The equivalent parameter that is used in the SDMI is the effective Atwood number, which is defined in a similar way, except ρ_1 is the ratio of the sum of the mass inside a representative volume element inside the interface to the volume it occupies and ρ_2 is the same for the potentially multiphase mixture outside the interface. These densities are called the effective densities, because they attempt to reduce the complexity of the SDMI to the RMI by deeming what phase the mass is in as negligible. While this analogy exists, the mechanism by which the instability develops is entirely different. The RMI deposits vorticity via the baroclinic term in the vorticity equation, which is a result of taking the curl of the Navier-Stokes equations. The baroclinic term exists when there is a misalignment between the gradients of pressure and density, and will only exist as the shock passes over the interface. This effectively deposits all vorticity instantaneously.

The SDMI deposits vorticity because of the finite amount of time for the particles to accelerate into the post-shock flow. The areas that have more particles for the fluid to travel through will lose more of its momentum to the particles than the areas with fewer particles. This creates shearing in the flow, which drives a fluid system with a large enough Reynold's number to turbulence. A lot of progress towards understanding these HIs has been made in the past century, but much work is still left.

1.3 Eulerian and Lagrangian Descriptions of Flow

In the field of continuum mechanics, it is beneficial to be able to describe kinematics in several ways. This is simply because it is often easier to describe certain motions in one way compared to another. The two ways that one may describe the kinematics of a continuum are a spacial, or Eulerian discription and a material, or Lagrangian description. The names of each of these descriptions come from the great scientists and mathematicians Leonhard Euler and his student Joseph Lagrange for their contribution to the subject. When following an Eulerian description of a continuum, the changes of properties at specified locations in space are observed while in a Lagrangian description the material is tracked. The mathematical representation of the Eulerian and Lagrangian descriptions are given respectively as follows.

$$\begin{aligned} \mathbf{v} &= \mathbf{v}(\mathbf{x}, t) \\ T &= T(\mathbf{x}, t) \end{aligned} \tag{1.7}$$

$$\begin{aligned} \mathbf{v} &= \mathbf{v}(\mathbf{X}, t) \\ T &= T(\mathbf{X}, t) \end{aligned} \tag{1.8}$$

Here, $\mathbf{x}$ is the position of the material in the current configuration and $\mathbf{X}$ is the

location of the material in the reference configuration. Note, that the governing equations that were previously presented were done so in an Eulerian description, as special treatment was taken to consider the material derivative. Governing equations in Lagrangian form are less common to see, especially for fluid mechanics. This is because in fluid mechanics, a short time after tracking the material, the current configuration potentially will not even slightly resemble the reference configuration. The Lagrangian conservation equations describe the motion of a continuum in terms of reference properties, compared to the changing properties observed at fixed points in the laboratory reference frame. Everything considered, being able to track the material of the fluid is often advantageous. The numerical methods used in this work uses a combination of the two kinematic descriptions, and has additional benefits and setbacks specific to it. These will be discussed later in greater detail.

1.4 Applications and Models

Particle drag is widespread in physical phenomena with parameter scales that span several orders of magnitude. For example, in the Crab Nebula particles can be seen in motion as electromagnetic radiation and gravity interact with them. Particles in chemical weapons are deformed and broken up by a blast wave, such that it is difficult to detect where they will land. Rotating detonation engines typically introduce liquid fuel particles ahead of a detonation wave to break them up, evaporate them, and burn them to perpetuate their thermodynamic cycle. All situations described here benefit from a better understanding of the shock-driven multiphase instability (SDMI). The SDMI is a hydrodynamic instability that is studied by several groups [3, 4], and exists due to the non-zero equilibration time of the solid or liquid particles. The equilibration time is a function of the particle and post-shock fluid's material properties, and will give a distance the particle will lag by, and the momentum it will take from the flow.

Fully understanding the drag experienced by shock-accelerated solid particles is the next step in understanding the fundamental physics and characteristic parameters of the SDMI, and in closing the larger problem of deformation and breakup in these situations.

The governing equations for this problem are a coupled set of nonlinear partial differential equations for both the solid and fluid regions. In SDMI simulations, typically thousands or millions of particle parcels 0 are present [4], and these equations cannot be used to resolve a cloud of particles in flow. This is due to the computational cost associated with resolving the huge range of relevant length scales. A simplification of these equations may be attributed to the scientists Maxey and Riley, for their rederived form of the Basset-Boussinesq-Oseen equations [5]. This reduces the complexity of the equations to be solved to a single nonlinear integro-differential equation, by assuming the particle to be rigid and the fluid to be Newtonian.

Still, applying this equation to many particles becomes unfeasible. Models are typically implemented as a solution to this. These models are ad hoc methods that do not necessarily rely on physics, but instead try to predict the outcome based on empirical data. Models that are pertenent in this application are for heat transfer and drag on a sphere. Many have been proposed over the past two centuries [6, 7, 8, 9], all while the complexity and parameter space continue to grow. The model of Clift and Gauvin [7], given by equation 1.9, has been a standard for decades.

$$C_D = \frac{24}{Re_D}\left(1 + 0.15Re_D^{0.687}\right) + 0.42\left(1 + \frac{42500}{Re_D^{1.16}}\right)^{-1} \tag{1.9}$$

This model is accurate for applications where compressibility is not a strong factor, but for particles in high speed gas flow, the incompressible models will under-predict the drag as stagnation pressure increases with the Mach number, along with many other nonlinear effects. One model that incorporates the Mach number into its parameter space is that of Parmar et al. [6]. This model uses both the Reynolds and

Mach numbers as inputs to yield the drag coefficient of the particle via a method of interpolating polynomials. Both of these models are considered in this work as a comparison to simulation results.

Models have been proposed for evaluating the heat transfer from spheres exposed to exernal flow. Because particles are often spherical in geometry, these models are able to determine heat transfer to and from particles with good accuracy. The model considered for this work is the well-validated model of Whitaker [9]. This model is given by equation 1.10, and takes into consideration the differences of temperature-dependent viscosity in between the surface of the particle and the free stream. All properties except for the viscosity at the surface of the particle are evaluated at the freestream temperature. This is atypical, as many models evaluate properties at the film temperature, which is defined as a mean of the surface and freestream temperatures.

$$\overline{Nu_D} = 2 + (0.4Re_D^{1/2} + 0.06Re_D^{2/3})Pr^{0.4}\left(\frac{\mu}{\mu_S}\right)^{1/4} \tag{1.10}$$

Nu_D is the particle Nusselt number, Re_D is the particle Reynold's number,and $\mu$and μ_S are the air viscosities evaluated at the freestream and particle surface, respectively. Experiments showed that, for a given set of flow and particle parameters, drag correlations severely under-predict the actual drag. This was measured by tracking particle positions with a high-speed camera as the particle traversed down a shock tube. The simulations conducted in this work were done so with the intent to pose an explanation to anomalous particle kinematic measurements from the Extreme Fluids team at Los Alamos National Laboratory (LANL) [10].

Chapter 2

Methods and Environment

2.1 FLAG

FLAG was the environment selected to simulate the shock-accelerated particle in this study, and is an arbitrary Lagrangian Eulerian (ALE) code. ALE codes provide the material-capturing abilities of Lagrangian codes, while also giving the robustness associated with Eulerian codes. They are executed by first solving the conservation laws while tracking the material, such that the mesh becomes deformed, then remapping the mesh using an optimization method. This step is necessary to prevent the mesh from tangling in regions of high shear strain rates, and to improve the quality of the solution. The mesh optimizer that was found to perform best, considering efficiency and effectiveness, was a condition number-based method. This method attempts to remap the mesh such that it returns cells that have been sheared back to rectangular. It is one of the more robust mesh optimizers available for this application with minimal iterations necessary. Material is advected through volume boundaries in this remapping step, as the mesh has no effect on the physics being solved. This process is then repeated in a time-marching manner.

FLAG solves hydrodynamics explicitly, such that it is limited by a CFL condition. FLAG uses a predictor-corrector method for time integration to yield a second-order solution. The cell advection is calculated using the Flux-Corrected Transport method of Boris and Book [11], to give second-order accuracy in space. FLAG is currently developed at Los Alamos National Laboratory, and is capable of running on a fully unstructured mesh in both two and three dimensions. It additionally has the capability to capture shocks via artificial viscosity. Several models for artificial viscosity exist, but the one used for this work was the modified Barton viscosity. Optional treatments are available in FLAG to smooth the velocity gradient tensors. This work utilizes the t0he hourglass treatment, which acts to smooth the differences between zone and side velocity gradient tensors. FLAG is also a highly scalable code, and is easily capable of running on tens of thousands of processers at a time. The conservation equations that can be solved with FLAG now include viscous and thermal diffusivity effects. The viscosity is capable of being evaluated using Sutherland's law (equation 2.1) [12]

$$\frac{\mu}{\mu_0} = \left(\frac{T}{T_0}\right)^{1.5} \frac{T_0 + 110.4}{T + 110.4} \tag{2.1}$$

The viscosity of the air, μ is calculated as a function of the temperature, T given a reference viscosity μ_0 measured at a given temperature T_0. All capabilities described here make FLAG ideal for tracking the motion of a shock-accelerated particle, while resolving the heat transfer and mechanical response of the particle due to the shock.

2.2 Ingen

While FLAG has a built in mesh generator, Ingen is capable of generating meshes that are more intricate and often better suited for simulations. Ingen is a Python library that has the ability to construct rectangular-structured, radial-structured, fully

unstructured, and combination meshes from a command line interface. Ingen, like FLAG, is also a code that is under development at LANL. Since its start, its operation has been overseen by the Setup team. The domain can be optionally subdivided into regions where different meshing rules may be imposed. Voronoi tesselations and Delaunay triangulation are supported as unstructured meshing rules. The Delaunay triangulation works on a set of discrete points when no point lies inside the circumscribed circle of any triangle. The Voronoi tesselation is the mathematical dual of the Delaunay triangulation of a point set. Furthermore, functions for dendritic derefinement are suppored within Ingen. This work makes use of rectangular-structured, radial-structured, and Voronoi tesselation meshes to resolve the domain. Dendritic derefinement is implemented in both the rectangular and radial structured meshes. The finalized mesh is exported as an x3d file so that it can be imported into FLAG.

2.3 Simulation Domain

Before simulating the shock-accelerated particle, the computational domain had to be built. It was decided that the three dimensional problem was to be reduced to a two dimensional axisymmetric one to reduce the computational burden associated with solving the equations in three dimensions. In this decision, the assumption of perfect symmetry was imposed, however the Reynolds number for the problem was low enough (Re = 40.8) that turbulence is not expected to onset, and therefore three dimensional effects are not expected to be present. This assumption will be checked in future work.

Kinematic measurements of the 4 μm particles were taken by the Extreme Fluids team [13] with given ambient conditions of the lab, the shock strength, and the particle properties also provided. Post shock properties are computed with the shock tube relations, given by Anderson [14], which derive from conservation of mass, momentum,

and energy for an ideal, calorically perfect gas. The particle is modeled as an elastic solid, with the yield strength given as an input to see the deformation, and possibly failure, of the particle.

This information, coupled with the post shock properties of the flow given in table 2.1, and the advertised particle diameters of four μm provides all necessary information to set up the simulation. The domain was built as rectangular with three wall boundary conditions and one inflow boundary condition. Because Lagrangian codes struggle when dealing with outflow boundary conditions, the domain had to be extended in the streamwise direction such that the reflected shock would not interfere with the accelerating particle. The distance was calculated using the initial and reflected wave speeds so that the particle could reach a reasonable percentage of the post-shock flow velocity. The downstream domain was derefined by a factor of sixteen to reduce the computational cost without losing important physics of the problem. The width of the domain was constrained to be ten times the radius of the particle in an attempt to avoid reflecting boundary effects. The fluid domain was meshed to be mostly square-structured with a transition region from the particle that is resolved with a Voronoi tessellation and a radial-structured mesh. These descriptions are illustrated in figure 2.1.

A Delaunay triangulation pattern was attempted to resolve the transitional region, but when the triangular cells deform as the simulation runs, the volume of each cell cannot be held constant by altering another degree of freedom, and the pressure in the cell artificially changes as a result. The mesh constructed with Ingen used both structured and unstructured components. This was done with the intent of building a mesh that would be robust as resolution increased while the meshing rules stayed the same. The sound speed in the solid region is much larger relative to the fluid region, so the CFL limitation typically occurs within the particle. Small cells zones in the interior of the particle will drive the time step to be extremely small

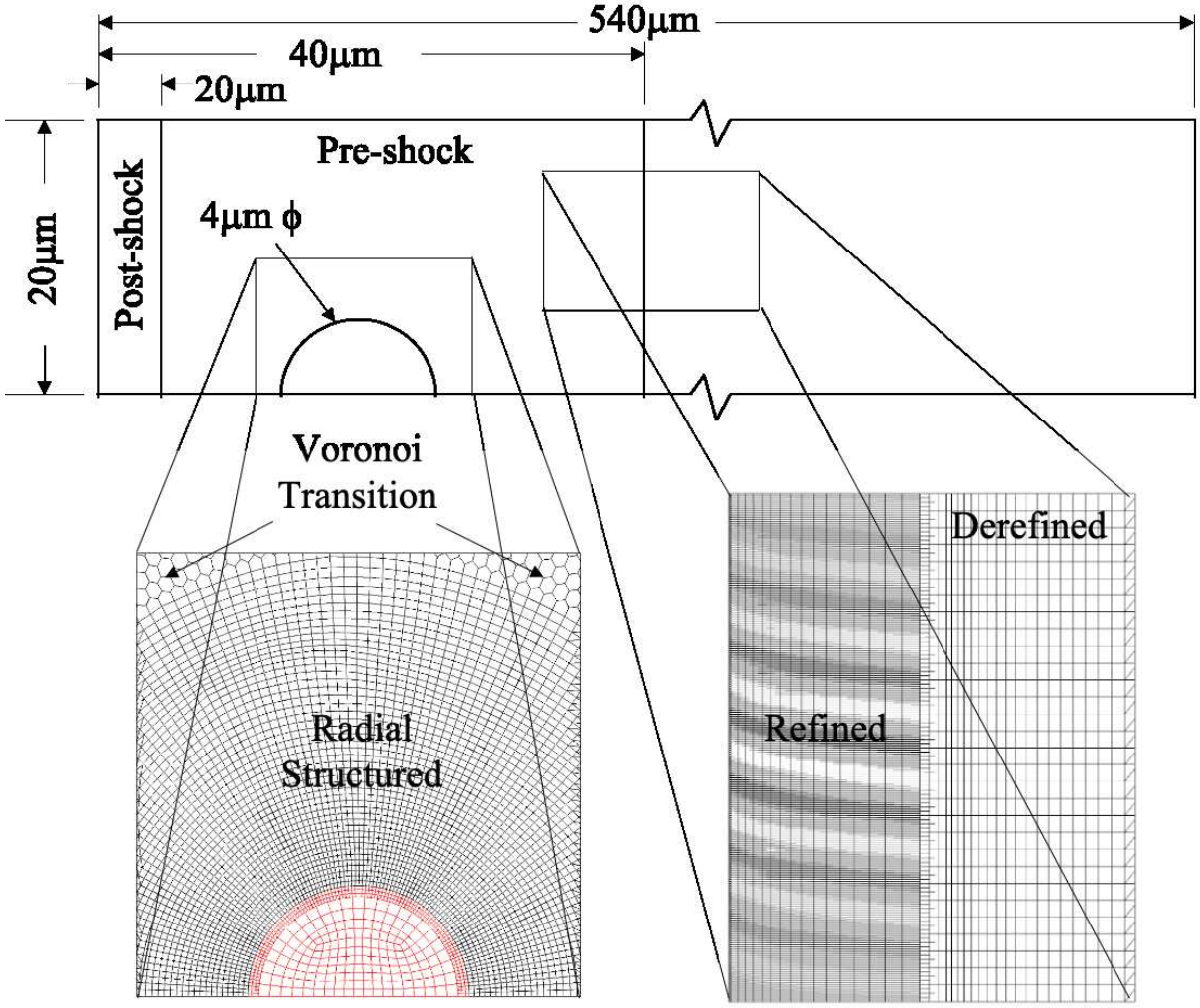

Figure 2.1: Simulation domain and meshing strategies

Table 2.1: Conditions for a Mach 1.3 Shock

Gas Variables	Pre-shock	Post-shock
T (K)	298.4	351.31
p (kPa)	78.05	140.92
ρ_{air} (kg/m^3)	0.911	1.397
V(m/s)	0	152.2
μ_{air}(μPa*s)	18.21	20.80
$d_{particle}$ (μm)	4.0	–
ρ_p (kg/m^3)	1140	–
k_p (W/(m*K))	0.25	–
$c_{v,\,p}$ (J/(kg*K))	3650	–

values unless intentional action is taken to prevent this. The mesh in the particle is dendritically derefined by a factor of four to prevent cells from shrinking and resulting in an unnecessarily small time step. A small region of high resolution is present along the rim of the particle to have mesh continuity at the boundary for calculation of the fluid stresses at the boundary. The particle is resolved with a rectangular-structured mesh so mesh stiffeners may be used in conjunction with the hydrodynamics to treat stress deviators.

Chapter 3

Results and Discussion

3.1 Knudsen Number Effects

The Knudsen number is the figure of merit when determining whether the continuum assumption is appropriate to apply to problems in fluid mechanics. It is defined as the ratio of the mean free path of the molecules in the flow field to the relevant length scale of the problem. When dealing with an ideal gas, the Knudsen number reduces to the expression given by equation 3.1.

$$Kn = \frac{Ma}{Re_D}\sqrt{\frac{\gamma\pi}{2}} \tag{3.1}$$

A general rule is that when the Knudsen number is much less than 1, the continuum assumption is a good approximation, and when the Knudsen number becomes large, free-molecular flow is a good approximation. When the Knudsen number approaches 1 from either side, the flow will behave in a less predictable manner, and adjustment models are oftentimes imposed to reduce the complexity to free molecular or continuum flow. The Knudsen number for this case is at 0.19, and is in the region traditionally called transitional flow. It is still a strongly debated topic on which value

the continuum assumption fails at; however it is observed that the error accumulated by using the continuum assumption for a Knudsen number under 0.01 is at roughly 3%, and is argued here to be an effective model.

3.2 Resolution Study

To guarantee a mesh-independent solution, a resolution study was carried out. Convergence of the solution was verified for both kinematic and thermal measurements. Ingen was utilized to develop four different mesh resolutions for the simulation to run on. The refinement of the mesh was measured by the number of faces present on the simulated particle's half-circumference. All meshing rules were kept constant otherwise to keep a fair comparison between cases. The computational cost of each of the four cases are given in table 3.1, which shows that for accuracy and computational affordability, the 645 case is the strongest. The simulations were run out to a stopping time of five hundred nanoseconds and the results were compared, as transient effects have become a negligible effect at this simulation time.

3.2.1 Particle Kinematics

First the results of the particle's kinematics were examined. The locations of the centroids of the mesh elements were tracked through simulation time along with the area of each mesh element. To determine kinematics of the particle, the position of the particle had to be defined. This was done with the geometric centroid of the particle. This variable was plotted against time and is seen in figure 3.1.

$$\mathbf{x}_{cm}(t) = \frac{\sum_{j=1}^{N_{elements}} \mathbf{x_j}(t) A_j(t)}{\sum_{j=1}^{N_{elements}} A_j(t)} \tag{3.2}$$

With the centroid of the particle given in simulation time, the other kinematic

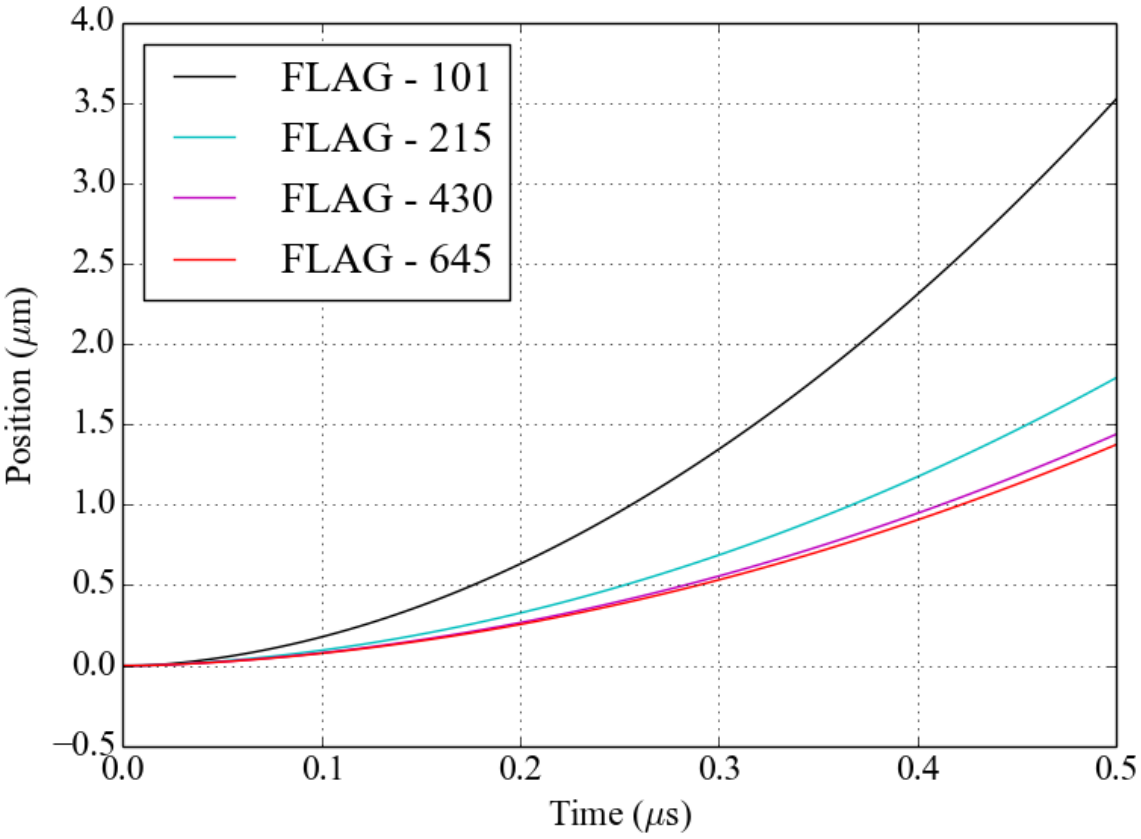

Figure 3.1: Resolution study of particle position in time

Table 3.1: Resolution study statistics

Case	Zones	Computer Time (Node Hours)	Position Relative Error (%)
101	24627	38.07	67.2
216	98463	312.42	15.1
432	220896	1335.4	3.1
645	391796	2130	–

variables could be computed. The velocity was calculated using a second-order finite difference approximation of the position and the acceleration was calculated in the same fashion with the discrete velocity data. With these kinematic variables, the drag coefficient could be computed. The method of computing this is shown in equation 3.3.

$$C_D(t) = \frac{4\rho_p d_p \|\mathbf{a_p}\|}{3\rho_a(\|\mathbf{v_p}\| - v_{ps})^2} \tag{3.3}$$

The relative error of the drag coefficient is roughly twice that of the position, because the acceleration decreases only slightly in the duration of the resolution study simulations.

Clear convergence can be seen in the position results as mesh resolution moves from the 100 faces per half-diameter case to the 645 one, as illustrated in figure 3.1. The relative error in the particle position between the coarsest two cases is over 50%, however the relative error between the most refined cases is less than 5% for position. This demonstrates the law of diminishing returns, and suggests that numerical error associated with the length scale of the mesh will not make up the last 10% required to agree with the models, but is sufficiently close that they are not to be questioned.

3.2.2 Particle Temperature

After the kinematic measurements of the particle were shown to converge to the same answer, the values for the particle temperature were checked as well. Temperature inside the particle is a continuous scalar field in reality, however it is a discontinuous field in the simulated particle. To avoid the complexities of comparing these discontinuous fields at different resolutions, the volume-averaged particle temperature is instead used as the figure of merit. To obtain this volume-averaged particle temperature, the finite volume cells of the particle are treated as infinitesimal area elements. A numerical volume integral is taken of the particle by using the method of integration by washers from elementary calculus. This method reduces to the following formula.

$$T_{FLAG}(t) = \frac{\int_0^t 2\pi r A T(\mathbf{x}, t) dr}{V_P} = \frac{\sum_{i=1}^{\#ParticleCells} r_i A_i T_i(t)}{\frac{2}{3} r^3} \tag{3.4}$$

This is repeated for every time step in each of the four resolution study simulations, and the average particle temperature is plotted against simulation time. The results are shown in figure 3.2. Interestingly, from the coarsest mesh resolution, the average particle temperature is relatively converged to the same answer. The limiting factor clearly is set by the particle kinematics agreement, and not by the average particle temperature.

The 645 faces case was chosen to be the one that was run out in time, due to the minimal relative error between it and the next case and its reasonably affordable computational cost.

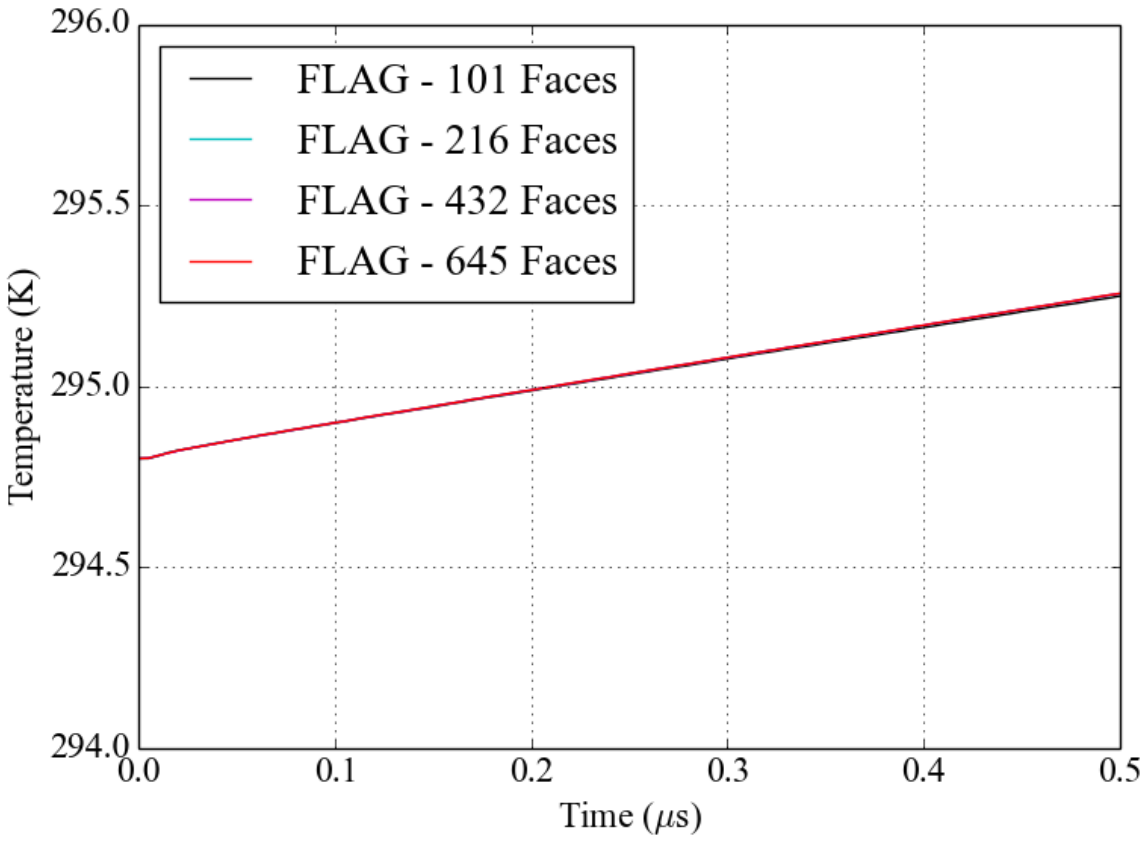

Figure 3.2: Resolution study of particle temperature in time

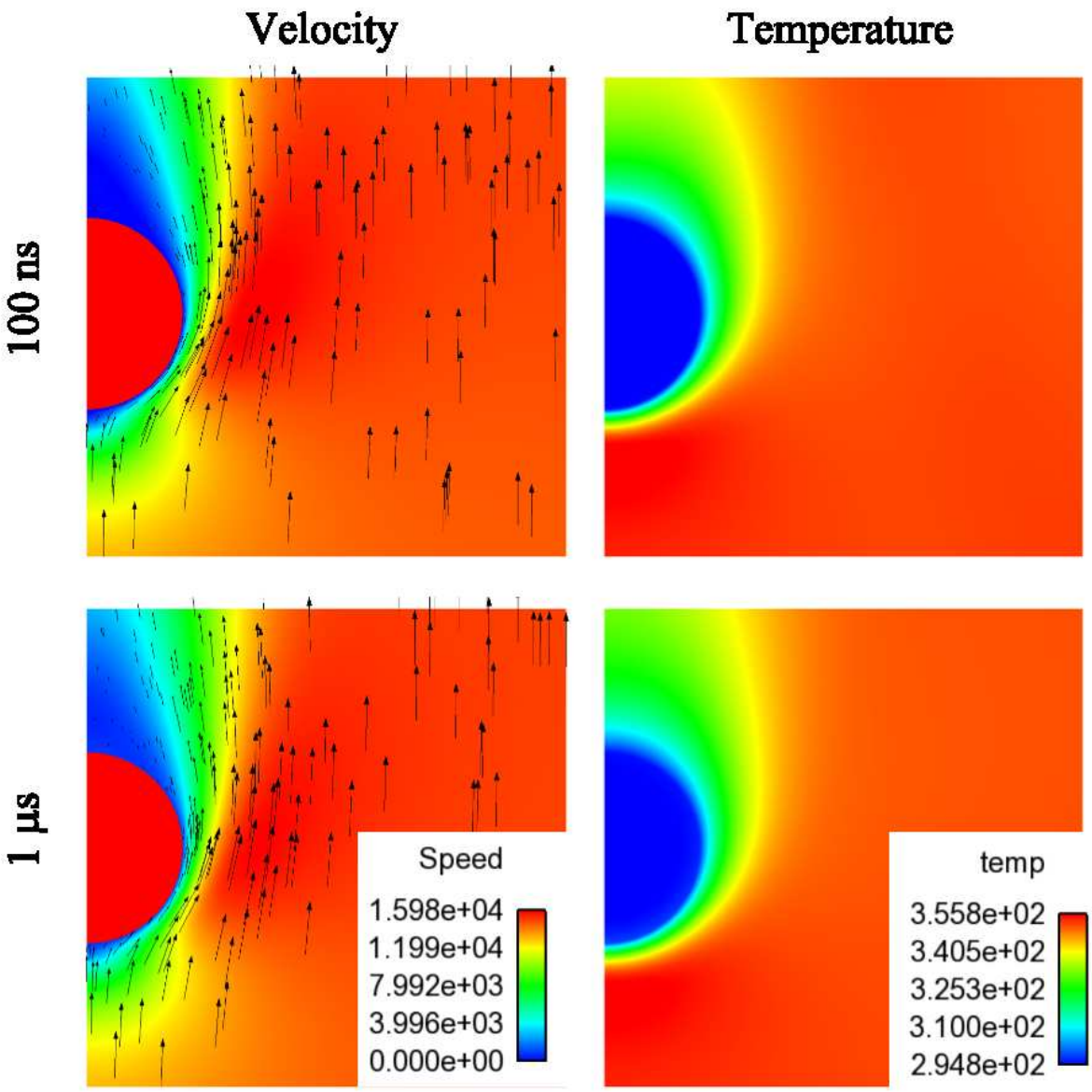

Figure 3.3: Pseudo-color plots of temperature and velocity

3.3 Model Comparison

3.3.1 Drag Model

To give a validation for the solution the kinematics converged to, the position of the simulated particle is compared to the position of a particle that is only subjected to fluid drag that obeys the model of Parmar et al. This is done by numerically integrating equation 3.5 to yield the velocity of the position in time.

$$\frac{d\mathbf{v_p}}{dt} = \frac{3C_{D,Parmar}(M, Re_d)\rho_a \|\mathbf{v_p} - \mathbf{v_{ps}}\|(\mathbf{v_p} - \mathbf{v_{ps}})}{4\rho_p d_p} + \frac{F_{i,u}}{\rho_p V_p} \tag{3.5}$$

Note that equation 3.5 is a second order ordinary differential equation because velocity is the total derivative of position with respect to time. The inviscid unsteady force, $F_{i,\,u}$, is the inviscid unsteady force and is only effective when the shock is passing over the particle. This was implemented after it was found that ignoring this parameter caused a disagreement in position larger than 5%. The model used for this force comes from Parmar et al. [15]. While this force does bring the model in closer alignment with the simulations, the viscous unsteady force is clearly one that has a large effect as well. Because this force will not fit into a one-dimensional model, it could not be implemented unfortunately.

This numerical integration is carried out by decomposing the equation into two first order ODE's and integrating with the Runge-Kutta based ODE solver built into Python. The comparison of the model to the simulated particle is shown in figure 3.4. The simulation is seen to over-predict the drag coefficient given by both the models of Clift-Gauvin and Parmar by a small amount. Both models are seen to agree with one another within 5%. The agreement of the models suggests that compressibility for the given Mach number of roughly 0.4 is not a strong factor, which is important in the validation of the heat transfer model, as compressibility is not considered in its

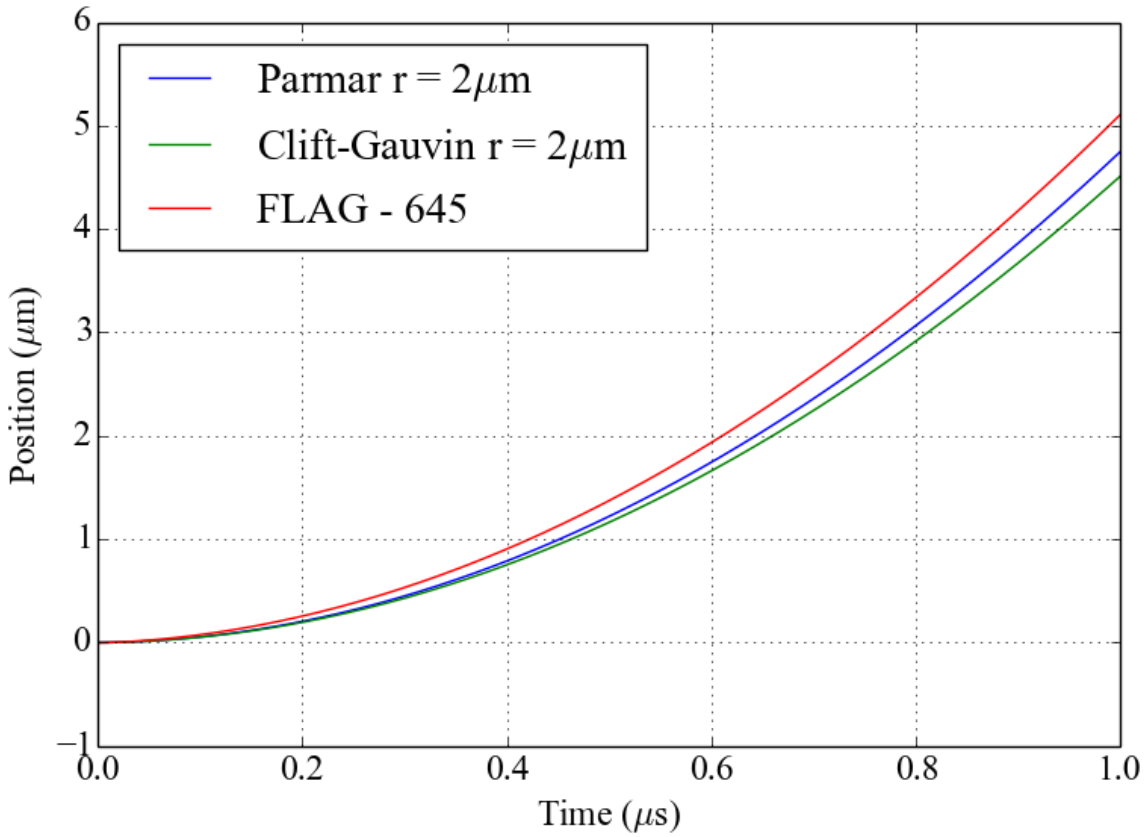

Figure 3.4: Particle position: Parmar and Clift-Gauvin model comparison to simulation

parameter space. It has previously been hypothesized that the motion of the particle has an effect on the drag experienced, as models have been built with data taken from stationary particles.

3.3.2 Heat Transfer Model

To validate the thermal conduction heat transfer solver in FLAG, the temperature values returned from the simulation were compared to those given by a heat transfer model. Compressible heat transfer models of a sphere exposed to external flow are not as well-validated as incompressible models. The model selected for comparison was the one proposed by Whitaker [9]. This model is valid when the viscosity of

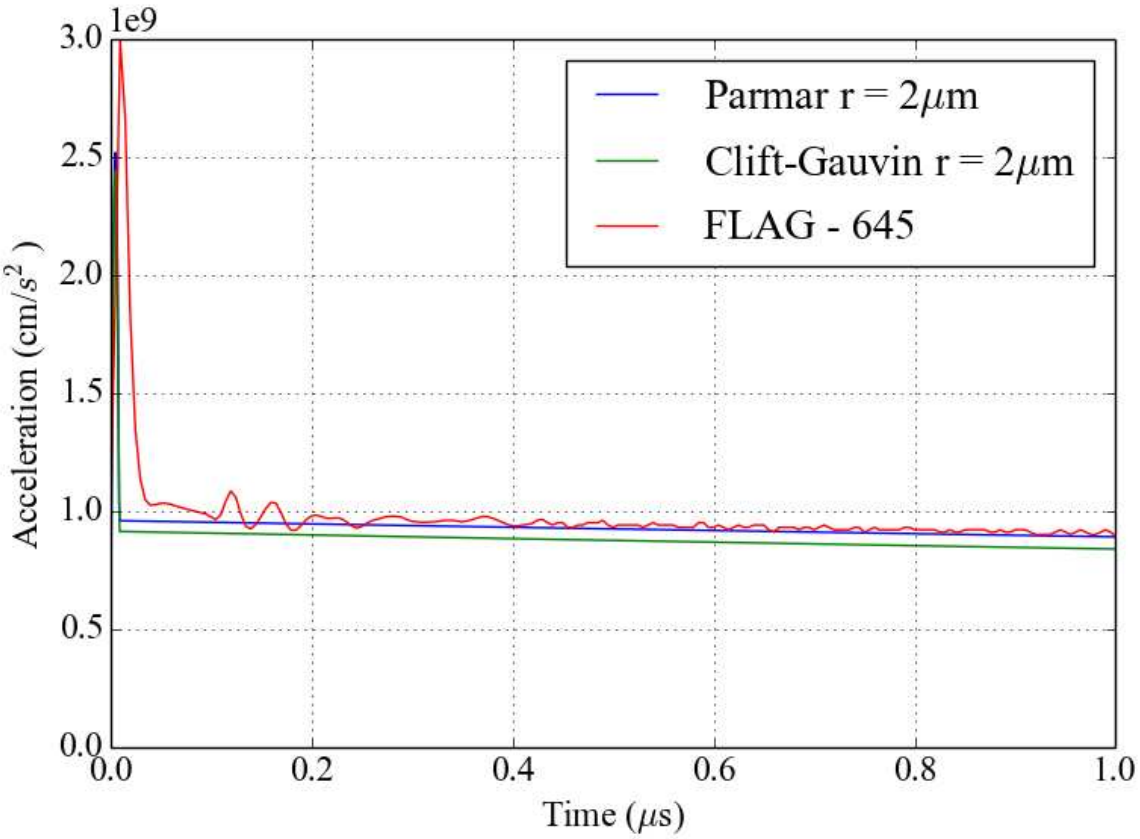

Figure 3.5: Particle acceleration: Parmar and Clift-Gauvin model comparison to simulation

the fluid at the surface of the particle is lesser than the viscosity at the free-stream. This suggests that the model is only appropriate when the free-stream is at a higher temperature with respect to the particle, as Sutherland's model for the viscosity of air increases monotonically with temperature. To test FLAG against this model, the first law of thermodynamics is applied with the particle as the control volume. It is assumed that work done to deform the particle, while non-zero in the simulation and reality, has a negligible effect. It is also assumed that the thermal properties of the particle are not temperature dependent, which is assumed by both the model and FLAG. Thermal properties for the convection correlation are computed with a quadratic interpolating polynomial with tabulated values for air. The Biot number is computed to test the validity of lumped capacitance, but is shown to be much too high at 0.6 to give an acceptable answer. To combat this issue, the particle is divided into concentric shells, such that each shell is isothermal with itself, i.e. a lumped capacitance, but not with neighboring shells. This method gives a set of coupled first-order ordinary differential equations, given by equations 3.6, 3.7, and 3.8, that must be solved simultaneously. This is done numerically in a fashion similar to the numerical integration of the particle's rigid-body dynamics equation [16].

$$\frac{dT_1}{dt} = \frac{3\alpha}{r_1^3}\left(\frac{T_2 - T_1}{\frac{1}{r_1} - \frac{1}{r_2}}\right) \tag{3.6}$$

$$\begin{aligned}\frac{dT_i}{dt} = &\frac{3\alpha}{r_i^3 - r_{i-1}^3} * \left(\frac{T_{i+1} - T_i}{\frac{1}{r_i} - \frac{1}{r_{i+1}}}\right) \\ &+ \frac{3\alpha}{r_i^3 - r_{i-1}^3}\left(\frac{T_{i-1} - T_i}{\frac{1}{r_{i-1}} - \frac{1}{r_i}}\right)\end{aligned} \tag{3.7}$$

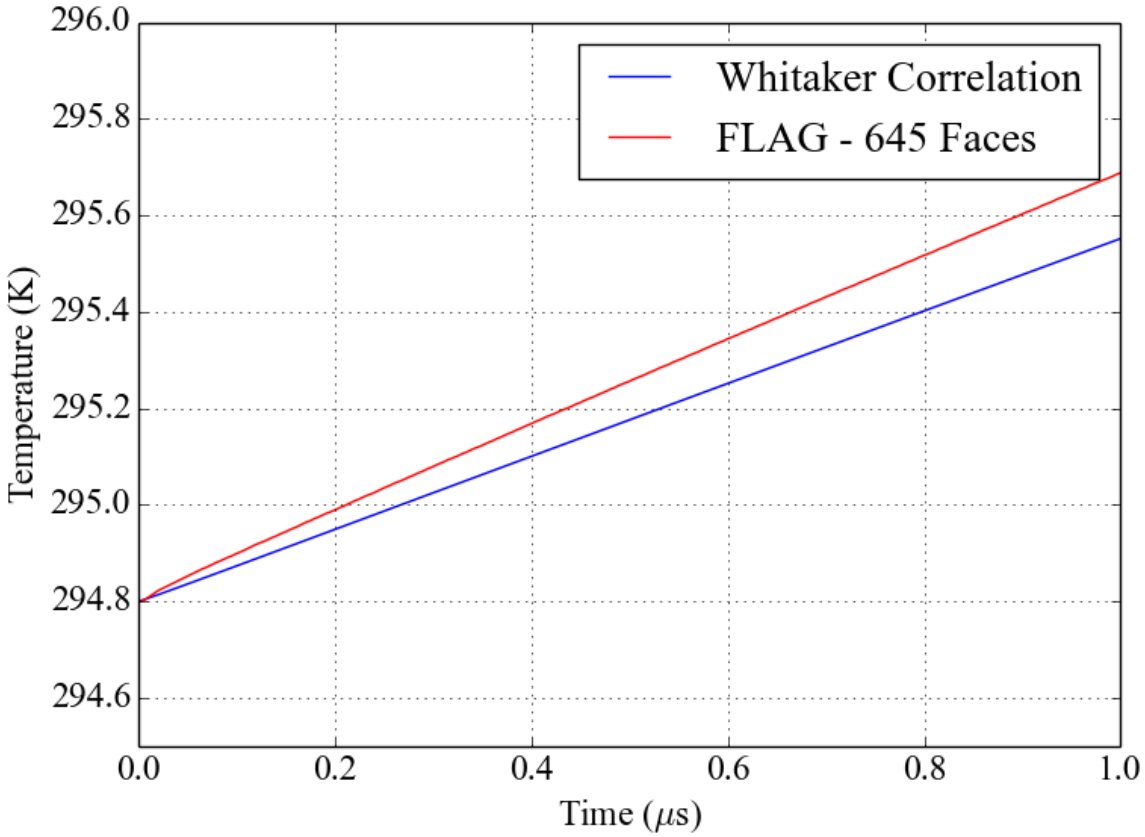

Figure 3.6: Whitaker Heat Transfer model comparison to simulation

$$\frac{dT_N}{dt} = \frac{3\alpha}{r_N^3 - r_{N-1}^3}\left(\frac{T_{N-1} - T_N}{\frac{1}{r_{N-1}} + \frac{1}{r_N}}\right) + \frac{3\overline{h}r_N^2}{\rho c\left(r_N^3 - r_{N-1}^3\right)}\left(T_{inf} - T_N\right) \tag{3.8}$$

Convergence to the final answer occurs at about 200 subdividing shells. As parameters are matched, and convergence of the lumped capacitance method of shells is realized, the convection correlation is isolated as the model to test FLAG against. This correlation is compared to the values of the FLAG particle in simulation time. Large temperature gradients appear in the particle through time, as seen in Figure 3.3. This fact is not surprising, given the computed Biot number of 0.6. After eval-

uating the average temperature of the particle through simulation time, the value is compared to that of the one-dimensional model. It is seen that the model is in good agreement with the data provided in FLAG. Given that compressibility is not taken into consideration with the model, and that heat transfer correlation of Whitaker has experimental scatter of 30% [9], the conduction heat transfer physics in FLAG is well within the range of validity for the model.

Chapter 4

"What Went Wrong"

4.1 Sphericity

In the pursuit of finding the best comparison of simulations to experiments, a few methods were attempted. First, particle sizes were searched for using a Hough transform algorithm applied to scanning electron microscopy (SEM) images, provided by the experimental team. This method searched over dozens of SEM images of different length scales to build a statistical distribution for the particle diameters. It was found after searching over these images that the distribution appeared to be log-normal with an expected value of roughly 4 μm for the diameter, as can be seen in figure 4.1. This is not surprising, as the particles were advertised as having 4 μm for their diameters. However, the observed kinematics strongly disagreed with this fact.

When using the drag models previously mentioned, it is implied that the particles are perfectly spherical, which is usually an effective enough assumption at this length scale. This is due to surface tension for liquid droplets and the frequent enough collisions to wear down anisotropic imperfections for solid particles. However, because particle sphericity is known to be an effect on the drag, sphericity was explored as a

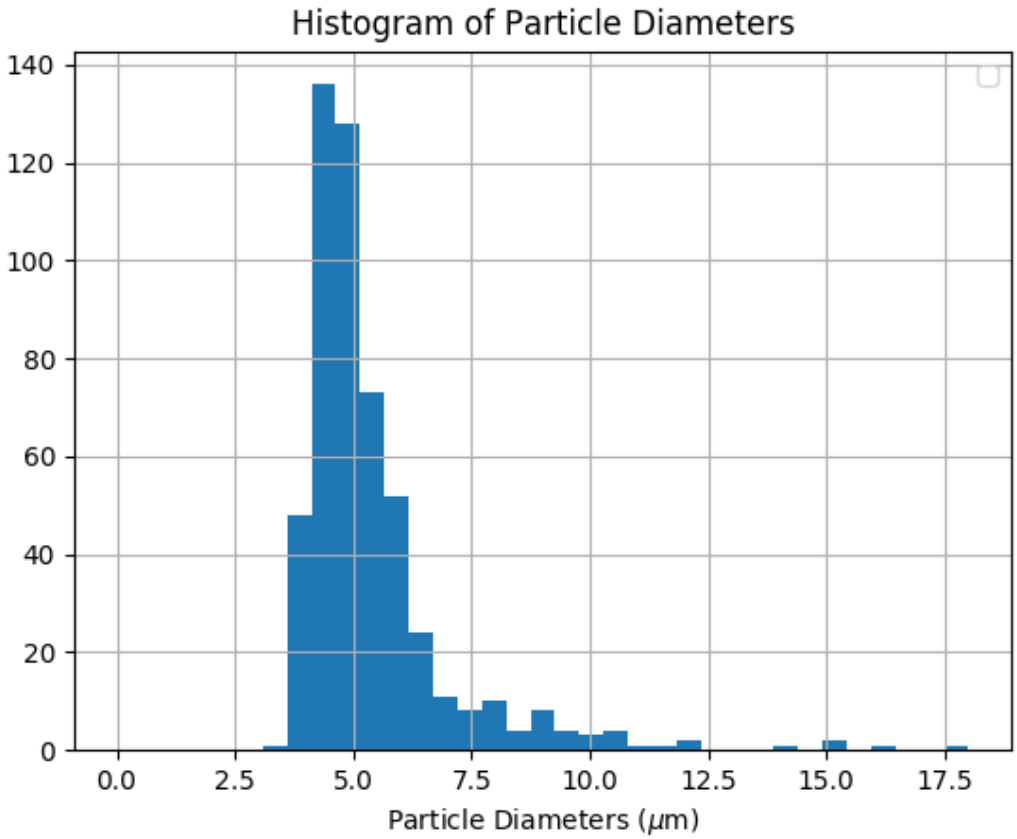

Figure 4.1: Particle size distribution

possible parameter that would explain the disagreement. To do this, the sphericity for each of the particles was computed from the SEM images. The images were two dimensional, so information about each of the particle's third dimension was unknown. With this fact, the conservative assumption that the third dimension's radius matched the major radius of the particle in the image was made. This was done so that if it was determined that the particles were not spherical, it would be because of data that was able to be proved with information from the images. The sphericity of an object is generally implicitly understood as Wadell's definition of the parameter, which is the ratio of the surface area of the object to the surface area of a sphere with the same volume as the object.

$$\Psi = \frac{\pi^{1/3}(6V_p)^{2/3}}{A_p} \tag{4.1}$$

Assuming that the manner in which the particles deviated from a sphere was in the shape of an ellipsoid, the SEM images were used again. The circle-finding algorithm used previously finds the location of the circles in an image effectively by minimizing a residual of the intensity gradient and the contour of the circle by adjusting the center and radius of the circle. This residual was plotted, and a fast Fourier transform was taken of the signal. A sample signal may be seen in figure 4.2. The amplitude of the first mode of the signal may be regarded as the difference in between the major and the minor axes of the ellipsoidal-assumed particle. With the major and minor radii of the particle known, the Wadell sphericity could be computed. The surface area of an ellipsoid is given in terms of an ellipsoidal integral, which are not analytically solvable but do have known numerical solutions. The expressions for the volume and surface area of a representative ellipsoidal particle are given as follows.

$$V_p = \frac{4}{3}\pi a^2 b \tag{4.2}$$

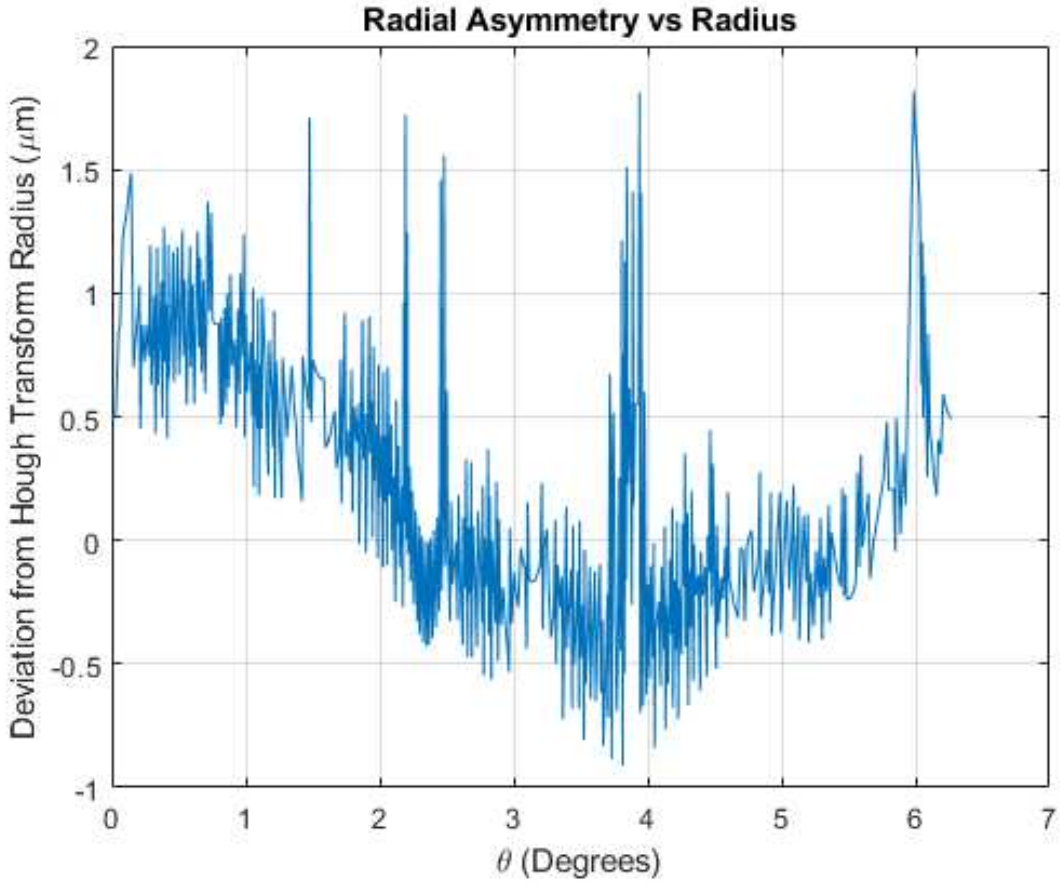

Figure 4.2: Residual of computed circle and particle

$$
\begin{aligned}
A_p &= 2\pi b^2 + \frac{2\pi a^2}{sin(\phi)}(E(\phi,1)sin^2(\phi) + F(\phi,1)cos^2(\phi)) \\
\phi &= cos^{-1}(\frac{b}{a}) \\
E(\phi,k) &= \int_0^{\phi} \sqrt{1-k^2sin^2(\theta)}d\theta \\
F(\phi,k) &= \int_0^{\phi} \frac{d\theta}{\sqrt{1-k^2sin^2(\theta)}}
\end{aligned}
\tag{4.3}
$$

Here, the major and minor radii are given by a and b respectively. This process was repeated for as many particles that had enough resolution to reliably return a signal, and it was found with the previously described process that the average sphericity of the particles was roughly 99.96%. It has been observed that effects of spherical asymmetry on particle drag do not show up until rougly 95% [17], and to

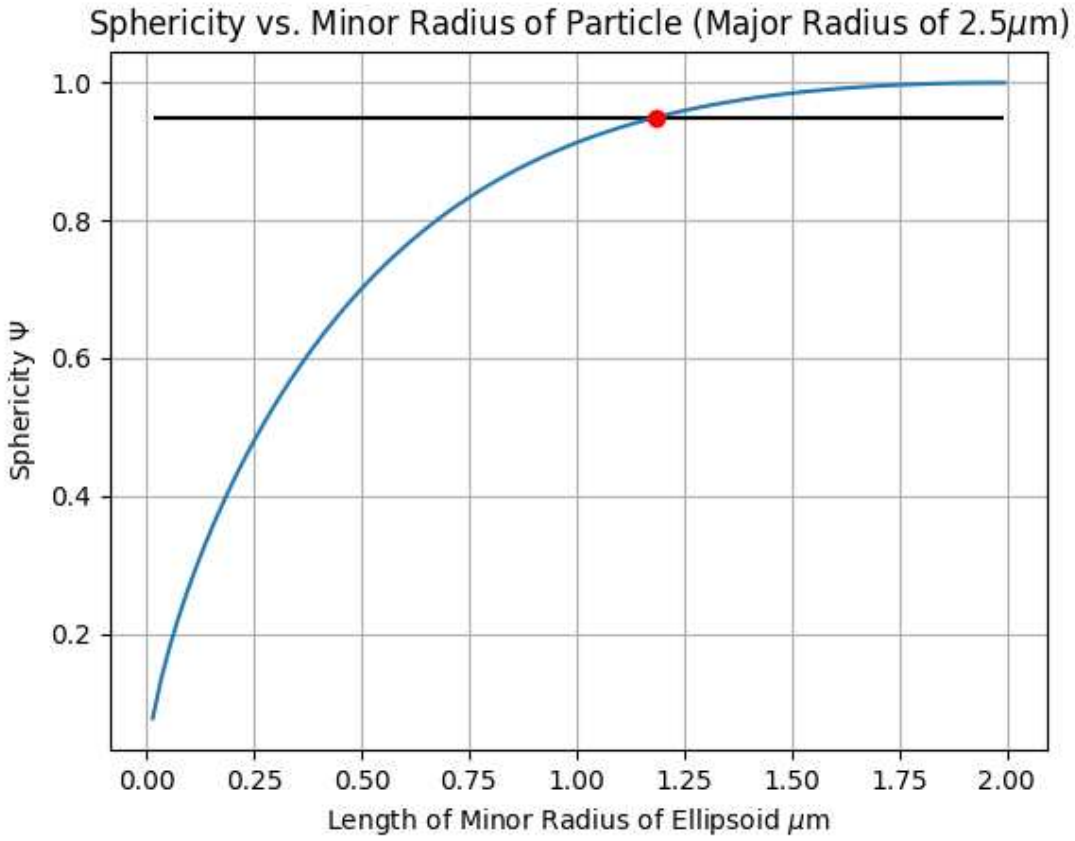

Figure 4.3: Particle Sphericity

achieve this value with an ellipsoid, the minor radius would have to be just over 1 μm, which is clearly not the case for the particles. This principle may be seen in figure 4.3. After this process of computing the sphericity of the particles, it was determined that it would not be a parameter that would even approach an explanation for the drag models' disagreement.

4.2 Equivalent Particle Size

After it was determined that the sphericity of the particle would not be capable of explaining the deviation of the measured particle kinematics from the models' predition, a solution for the equivalent particle diameter that would match experiments and the models was made. To do this, the drag model of Parmar was utilized to compute the drag coefficient of the particle as a function of the Reynold's and Mach numbers. Using the Maxey-Riley equation, and assuming only the forces of quasi-steady and inviscid-unsteady drags acting on the particle, a nonlinear first order ordinary differential equation appears, and is given by equation 3.5. This equation relates the particle velocity, $\mathbf{v_P}$, to the post-shock velocity $\mathbf{v_{PS}}$, the post-shock air density ρ_a, the post-shock Mach number M, the particle density ρ_P, the particle diameter d_P, the particle Reynold's number Re_d, and the inviscid unsteady force $F_{i,u}$. This equation can be integrated in time numerically to yield kinematics of the particle, given flow conditions and properties of the particle. Rather than solving for the velocity of the particle as a function of time, the particle diameter may be treated as the variable if given a velocity at a post-shock time. The diameter can be solved for with a Newton-Raphson style method. In this work, the numerical time integration wass done with a variable time-step Runge-Kutta method, and the numerical solution for particle diameter is done with a bisection method [16]. The particles have been determined to be nylon [10], fixing their density at 1140 kg per cubic meter. This information,

coupled with the post shock properties of the flow given in table 2.1, led to the conclusion that the particles must have had a diameter of 0.62 micrometers to agree with the model proposed by Parmar et al. As mentioned earlier, the Knudsen number is the non-dimensional parameter that is used to determine whether or not the continuum model with be effective in describing flow. Given the laboratory conditions at Los Alamos, New Mexico and the particle size calculated, the Knudsen number is computed as 0.1. This is well outside of the regime of continuum flow, and because Parmar's model is based on continuum flow, a correction factor must be implemented to adjust for this. The Cunningham correction factor, which is based on a reconcilliation of continuum flow and the kinetic theory of gases [18], was implemented into the particle size calculation procedure.

$$C = 1 + \frac{2\lambda}{d}(1.257 + 0.4e^{-0.55d/\lambda}) \tag{4.4}$$

It is used by first computing the continuum drag coefficient and then dividing by the correction factor. When progressing from continuum flow to slip flow, the drag is observed to decrease. This is due to the fact that the no slip condition is eliminated, and the velocity gradients at the particle surface decrease, and with it, the shear stress. After accounting for this effect, the equivalent particle diameter was calculated as 0.52 μm. This information was provided to the experimentalists, but a simulation based on continuum equations would be inappropriate in this flow regime. Chronologically, from here, the particle was decided to be simulated as a 4μm diameter nylon particle, as the supplier advertized, and compare to models and the experimental kinematics.

Chapter 5

Conclusions

The viscous and heat transfer physics solvers of the hydrocode FLAG have been shown to agree with validated models in their respective fields. Because more relevant physics have been considered in this work, such as the material response and dynamics of the particle from the shock, and because particle forces have been shown to agree with drag models for a fixed particle, it may be concluded that the motion of the particle may account for up to 10% error in location of the particle. Drag and thermal models continue to agree reasonably well with particle kinematics, even as the particle accelerates in the flow field. With this agreement seen, a possible explanation is that the particles being used in the experiments were smaller than previously predicted. We look forward to future experiments from the Extreme Fluids team.

Appendix A

Empirical Model Computation

A.1 Kinematic Validation

```
from scipy.integrate import odeint
    from scipy.integrate import cumtrapz
    import numpy as np
    import matplotlib.pyplot as plt
    plt.rcParams["font.family"] = "Times New Roman"
    plt.rcParams["font.size"] = "18"
    plt.rcParams["figure.autolayout"] = True
    def CdParmar(Re, M):
    if (M ¡= 1):
    xi = 0
    C = np.zeros([3, 1])
    f = np.zeros([3, 1])
    C[0] = 6.48
    C[1] = 9.28
```

C[2] = 12.21

f[0] = -1.884 + 8.422*M - 13.70*M**2 + 8.162*M**3

f[1] = -2.228 + 10.35*M - 16.96*M**2 + 9.840*M**3

f[2] = 4.362 - 16.91*M + 19.84*M**2 - 6.296*M**3

for ii in range(np.size(f)):

P = 1

for jj in range(np.size(C)):

if (ii != jj):

P *= (np.log(Re) - C[jj])/(C[ii] - C[jj])

xi += f[ii]*P

CDMcr = 24/Re*(1 + 0.15*Re**0.684) + 0.513*(1+483/Re**0.669)**(-1)

CDM1 = 24/Re*(1 + 0.118*(Re**0.813)) + 0.69*(1 + 3550/(Re**0.793))**(-1)

CD = CDMcr + (CDM1-CDMcr)*xi

return CD

if (M ¿ 1):

xi = 0

C = np.zeros([3, 1])

f = np.zeros([3, 1])

C[0] = 6.48

C[1] = 8.93

C[2] = 12.21

f[0] = -2.963 + 4.392*M - 1.169*M**2 - 0.027*M**3 - 0.233*np.exp((1-M)/0.011)

f[1] = -6.617 + 12.11*M - 6.501*M**2 + 1.182*M**3 - 0.174*np.exp(100*(1-M))

f[2] = -5.866 + 11.57*M - 6.665*M**2 + 1.312*M**3 - 0.350*np.exp((1-M)/0.012)

for ii in range(np.size(f)):

P = 1

for jj in range(np.size(C)):

```
if (ii != jj):
P *= (np.log(Re) - C[jj])/(C[ii] - C[jj])
xi += f[ii]*P
CDM1 = 24/Re*(1 + 0.118*(Re**0.813)) + 0.69*(1 + 3550/(Re**0.793))**(-1)
CDM175 = 24/Re*(1 + 0.107*(Re**0.867)) + 0.646*(1 + 861/(Re**0.634))**(-1)
CD = CDM1 + (CDM175-CDM1)*xi
return CD
def CdCG(Re):
return 24/Re*(1 + 0.15*Re**0.687) + 0.42*(1+42500/Re**1.16)**(-1)
def dvdtParmar(v, t):
rhoamb = 1.397406280968033E-3
rhop = 1.140
r = 2.0E-4
mu = 2.086E-4
ups = 15222.09907425806
Re = 2*rhoamb*r*np.sqrt((ups-v)**2)/mu
M = np.sqrt(((ups-v)/100)**2)/np.sqrt(1.401*287.06*351.3132002748754)
return 3*CdParmar(Re, M)*rhoamb*(v-ups)**2/(8*rhop*r)
def dvdtCG(v, t):
rhoamb = 1.397406280968033E-3
rhop = 1.140
r = 2.0E-4
mu = 2.086E-4
ups = 15222.09907425806
Re = 2*rhoamb*r*np.sqrt((ups-v)**2)/mu
M = np.sqrt(((ups-v)/100)**2)/np.sqrt(1.401*287.06*351.3132002748754)
return 3*CdCG(Re)*rhoamb*(v-ups)**2/(8*rhop*r)
```

```
    t = np.linspace(0, 0.000002, 100)
    v0 = 0
    vParmar = odeint(dvdtParmar, v0, t)
    vCG = odeint(dvdtCG, v0, t)
    cdParmar = np.zeros(np.size(t))
    cdCG = np.zeros(np.size(t))
    for ii in range(np.size(t)):
    rhoamb = 1.397406280968033E-3
    rhop = 1.140
    r = 2.0E-4
    mu = 2.086E-4
    ups = 15222.09907425806
    Re = 2*rhoamb*r*np.sqrt((ups-vParmar[ii])**2)/mu
    M = np.sqrt(((ups-vParmar[ii])/100)**2)/np.sqrt(1.401*287.06*351.3132002748754)
    cdParmar[ii] = CdParmar(Re, M)
    Re = 2*rhoamb*r*np.sqrt((ups-vCG[ii])**2)/mu
    cdCG[ii] = CdCG(Re)
    xParmar = np.array([0])
    xCG = np.array([0])
    xParmar = np.append(xParmar, cumtrapz(vParmar[:,0], x = t))
    xCG = np.append(xCG, cumtrapz(vCG[:,0], x = t))
    aParmar = np.gradient(vParmar[:,0], t, edgeorder = 2)
    aCG = np.gradient(vCG[:,0], t, edgeorder = 2)
    xFlag216 = np.loadtxt('Pos216Faces.csv', usecols=[1], skiprows=1, delimiter=',
')
    tFlag216 = np.loadtxt('Pos216Faces.csv', usecols=[0], skiprows=1, delimiter=', ')
    xFlag432 = np.loadtxt('Pos432Faces.csv', usecols=[1], skiprows=1, delimiter=',
```

')

tFlag432 = np.loadtxt('Pos432Faces.csv', usecols=[0], skiprows=1, delimiter=', ')

vFlag216 = np.gradient(xFlag216, tFlag216, edgeorder = 2)

vFlag432 = np.gradient(xFlag432, tFlag432, edgeorder = 2)

aFlag216 = np.gradient(vFlag216, tFlag216, edgeorder = 2)

aFlag432 = np.gradient(vFlag432, tFlag432, edgeorder = 2)

cdFlag216 = 8*2.6149*0.5E-4*aFlag216/(3*1.397406280968033E-3*(vFlag216 - 15222.099074258

cdFlag432 = 8*2.6149*0.5E-4*aFlag432/(3*1.397406280968033E-3*(vFlag432 - 15222.099074258

plt.plot(tFlag216*1E6, aFlag216, label = 'FLAG - 216 Faces', color='g')

plt.plot(tFlag432*1E6, aFlag432, label = 'FLAG - 432 Faces', color='c')

plt.xlabel(r'Time (μs)')

plt.ylabel(r'Acceleration ($\frac{cm}{s^2}$)')

plt.grid()

plt.legend(loc = 0)

plt.savefig('resolutionStudyAcc.png')

plt.close()

plt.plot(tFlag216*1E6, vFlag216, label = 'FLAG - 216 Faces', color='g')

plt.plot(tFlag432*1E6, vFlag432, label = 'FLAG - 432 Faces', color='c')

plt.xlabel(r'Time (μs)')

plt.ylabel(r'Velocity ($\frac{cm}{s}$)')

plt.grid()

plt.legend(loc = 0)

plt.savefig('resolutionStudyVel.png')

plt.close()

plt.plot(tFlag216*1E6, xFlag216, label = 'FLAG - 216 Faces', color='g')

plt.plot(tFlag432*1E6, xFlag432, label = 'FLAG - 432 Faces', color='c')

plt.xlabel(r'Time (μs)')

```
    plt.ylabel(r'Position ($\mu$m)')
    plt.grid()
    plt.legend(loc = 0)
    plt.savefig('resolutionStudyPos.png')
    plt.close()
    plt.plot(tFlag216*1E6, cdFlag216, label = 'FLAG - 216 Faces', color='g')
    plt.plot(tFlag432*1E6, cdFlag432, label = 'FLAG - 432 Faces', color='c')
    plt.xlabel(r'Time ($\mu$s)')
    plt.ylabel(r'C$D$')
    plt.grid()
    plt.legend(loc='best')
    plt.savefig('resolutionStudyCd.png')
    plt.close()
```

A.2 Heat Transfer Validation

```
import numpy as np
    from scipy.integrate import odeint
    from scipy.interpolate import interp1d
    import matplotlib.pyplot as plt
    import time
    plt.rcParams["font.family"] = "Times New Roman"
    plt.rcParams["font.size"] = "17"
    plt.rcParams["figure.autolayout"] = True
    def dTdt(tempPart, t, N, r, k, dPart, Re, Pr, mu, M, rhoPart, cpPart, kPart,
alphaPart, tempInf):
    dTdt = np.zeros(N)
```

hBar = k(tempInf)/dPart*(2 + (0.4*Re**(1/2) + 0.06*Re**(2/3))*0.71**0.4*(mu(tempInf)/mu
- 1]))**(1/4))

dTdt[0] = 3*alphaPart*((tempPart[1] - tempPart[0])/(1/r[0] - 1/r[1]))/(r[1]**3 -
r[0]**3)

for ii in range(1, N-1):

dTdt[ii] = 3*alphaPart*((tempPart[ii + 1] - tempPart[ii])/(1/r[ii] - 1/r[ii + 1]) +
(tempPart[ii - 1] - tempPart[ii])/(1/r[ii - 1] - 1/r[ii]))/(r[ii]**3 - r[ii - 1]**3)

dTdt[N - 1] = 3*alphaPart*((tempPart[N - 2] - tempPart[N - 1])/(1/r[N - 2] -
1/r[N - 1]))/(r[N - 1]**3 - r[N - 2]**3) + 3*hBar*r[N - 1]**2*(tempInf - tempPart[N
- 1])/(rhoPart*cpPart*(r[N - 1]**3 - r[N - 2]**3))

return dTdt

tempAirProp = np.loadtxt("airProperties.csv", usecols=(0,), skiprows=1, delim-
iter=",")

muAirProp = np.loadtxt("airProperties.csv", usecols=(4,), skiprows=1, delim-
iter=",")

PrAirProp = np.loadtxt("airProperties.csv", usecols=(6,), skiprows=1, delimiter=",")

kAirProp = np.loadtxt("airProperties.csv", usecols=(5,), skiprows=1, delimiter=",")

mu = interp1d(tempAirProp, muAirProp, kind='quadratic')

Pr = interp1d(tempAirProp, PrAirProp, kind='quadratic')

k = interp1d(tempAirProp, kAirProp, kind='quadratic')

t216 = np.loadtxt('temp216Faces.csv', usecols=(0,), skiprows=1, delimiter = ',')

T216 = np.loadtxt('temp216Faces.csv', usecols=(1,), skiprows=1, delimiter = ',')

t432 = np.loadtxt('temp432Faces.csv', usecols=(0,), skiprows=1, delimiter = ',')

T432 = np.loadtxt('temp432Faces.csv', usecols=(1,), skiprows=1, delimiter = ',')

N = 100

nTime = 100

dPart = 4.0E-6

```
r = np.linspace(1.0E-15, dPart/2, N)
rhoPart = 1140.0
cpPart = 3517.0
kPart = 0.25
alphaPart = kPart/(rhoPart*cpPart)
uInf = 152.2
y = 1.402
R0 = 8314.46261815324
R = R0/28.97
rhoInf = 1.397406280968033
tempInf = 351.3132002748754
M = uInf/np.sqrt(y*R*tempInf)
Re = (uInf*dPart*rhoInf/mu(tempInf))
t = np.linspace(0.0, 5.0E-7, nTime)
T0 = 294.8*np.ones(N)
tock = time.time()
T = odeint(dTdt, T0, t, args=(N, r, k, dPart, Re, Pr, mu, M, rhoPart, cpPart,
kPart, alphaPart, tempInf))
tick = time.time()
print(tick-tock)
tempAvg = np.zeros(nTime)
for ii in range(nTime):
sum = 0
for jj in range(1, N):
sum += (T[ii][jj] + T[ii][jj - 1])*(r[jj]**3 - r[jj - 1]**3)/2.0
tempAvg[ii] = sum/(dPart/2)**3
plt.plot(t*1.0E6, tempAvg, label='Whitaker Correlation', color='b')
```

```
plt.plot(t216*1.0E6, T216, label='FLAG - 216 Faces')
plt.plot(t432*1.0E6, T432, label='FLAG - 432 Faces')
plt.grid()
plt.legend(loc='best')
plt.xlabel(r'Time ($\mu$s)')
plt.ylabel('Temperature (K)')
plt.title('Average Temperature of Particle against Time')
plt.savefig('modelTemp.png')
plt.close()
plt.plot(t*1.0E6, tempAvg, label='Whitaker Correlation')
plt.plot(t216*1.0E6, T216, label='FLAG - 216 Faces', color='g')
plt.plot(t432*1.0E6, T432, label='FLAG - 432 Faces', color='g')
plt.grid()
plt.legend(loc='best')
plt.xlabel(r'Time ($\mu$s)')
plt.ylabel('Temperature (K)')
plt.savefig('resolutionStudyTemp.png')
plt.close()
```

A.3 Particle Size calculation

```
from scipy.optimize import bisect
from scipy.integrate import odeint
from scipy.integrate import trapz
import numpy as np
import matplotlib.pyplot as plt
def Cd(Re, M):
```

if (M ¡= 1):

xi = 0

C = np.zeros([3, 1])

f = np.zeros([3, 1])

C[0] = 6.48

C[1] = 9.28

C[2] = 12.21

f[0] = -1.884 + 8.422*M - 13.70*M**2 + 8.162*M**3

f[1] = -2.228 + 10.35*M - 16.96*M**2 + 9.840*M**3

f[2] = 4.362 - 16.91*M + 19.84*M**2 - 6.296*M**3

for jj in range(np.size(f)):

P = 1

for kk in range(np.size(C)):

if (jj != kk):

P *= (np.log(Re) - C[kk])/(C[jj] - C[kk])

xi += f[jj]*P

CDMcr = 24/Re*(1 + 0.15*Re**0.684) + 0.513*(1+483/Re**0.669)**(-1)

CDM1 = 24/Re*(1 + 0.118*(Re**0.813)) + 0.69*(1 + 3550/(Re**0.793))**(-1)

CD = CDMcr + (CDM1-CDMcr)*xi

return CD

if (M ¿ 1):

xi = 0

C = np.zeros([3, 1])

f = np.zeros([3, 1])

C[0] = 6.48

C[1] = 8.93

C[2] = 12.21

```
f[0] = -2.963 + 4.392*M - 1.169*M**2 - 0.027*M**3 - 0.233*np.exp((1-M)/0.011)
f[1] = -6.617 + 12.11*M - 6.501*M**2 + 1.182*M**3 - 0.174*np.exp(100*(1-M))
f[2] = -5.866 + 11.57*M - 6.665*M**2 + 1.312*M**3 - 0.350*np.exp((1-M)/0.012)
for jj in range(np.size(f)):
P = 1
for kk in range(np.size(C)):
if (jj != kk):
P *= (np.log(Re) - C[kk])/(C[jj] - C[kk])
xi += f[jj]*P
CDM1 = 24/Re*(1 + 0.118*(Re**0.813)) + 0.69*(1 + 3550/(Re**0.793))**(-1)
CDM175 = 24/Re*(1 + 0.107*(Re**0.867)) + 0.646*(1 + 861/(Re**0.634))**(-1)
CD = CDM1 + (CDM175-CDM1)*xi
return CD
def dvdt(v, t, r, l):
rhoamb = 1.397406280968033E-3
rhop = 1.14
mu = 2.086E-4
ups = 15222.09907425806
Re = 2*rhoamb*r*np.sqrt((ups-v)**2)/mu
M = np.sqrt(((ups-v)/100)**2)/np.sqrt(1.401*287.06*351.3132002748754)
C = 1 + l/r*(1.257 + 0.4*np.exp(-1.1*r/l))
return 3*Cd(Re, M)*rhoamb*(v - ups)**2/(8*C*rhop*r)
def fun(x, finalPosMeas, l):
t = np.linspace(0.0, 0.000002, 500)
v0 = 0
xdum = 0
v = odeint(dvdt, v0, t, args=(x/10000, l))
```

finalPosCalc = trapz(v[:, 0], x = t)

return (finalPosCalc - finalPosMeas/10000)

Ma = 152.2209907425806/376.00646119629

mu = 20.8E-6

rho = 1.397406280968033

m = 28.97/(1000*6.0221409E23)

kBoltz = 1.38064900E-23

T = 351.3132002748754

l = mu/rho*np.sqrt(np.pi*m/(2*kBoltz*T))

sol = bisect(fun, 0.1, 0.7, args = (176.2, l*100))

print('d = '+str(sol*2)+'um')

Bibliography

[1] Ankur D Bordoloi, Adam A Martinez, and Katherine Prestridge. Relaxation drag history of shock accelerated microparticles. *Journal of Fluid Mechanics*, 823(LA-UR-17-22125), 2017.

[2] Claudio Scherer and Antonio Martins Figueiredo Neto. Ferrofluids: properties and applications. *Brazilian Journal of Physics*, 35(3A):718–727, 2005.

[3] P Vorobieff, M Anderson, J Conroy, R White, CR Truman, and S Kumar. Analogues of rayleigh-taylor and richtmyer-meshkov instabilities in flows with nonuniform particle and droplet seeding. *WIT Transactions on Engineering Sciences*, 70:17–28, 2011.

[4] WJ Black, N Denissen, and JA McFarland. Particle force model effects in a shock-driven multiphase instability. *Shock Waves*, 28(3):463–472, 2018.

[5] Martin R Maxey and James J Riley. Equation of motion for a small rigid sphere in a nonuniform flow. *The Physics of Fluids*, 26(4):883–889, 1983.

[6] M Parmar, A Haselbacher, and S Balachandar. Improved drag correlation for spheres and application to shock-tube experiments. *Aiaa Journal*, 48(6):1273–1276, 2010.

[7] R Clift and WH Gauvin. Motion of entrained particles in gas streams. *The Canadian Journal of Chemical Engineering*, 49(4):439–448, 1971.

[8] GG Stokes. On the effect of internal friction of fluids on the motion of pendulums. trans. cambr. phil. 1851.

[9] Stephen Whitaker. Forced convection heat transfer correlations for flow in pipes, past flat plates, single cylinders, single spheres, and for flow in packed beds and tube bundles. *AIChE Journal*, 18(2):361–371, 1972.

[10] Kyle Hughes, Adam Martinez, and John Charonko. Mach number and particle size effects on the unsteady drag of shocked micro-droplets. *Bulletin of the American Physical Society*, 64, 2019.

[11] Jay P Boris and David L Book. Flux-corrected transport. i. shasta, a fluid transport algorithm that works. *Journal of computational physics*, 11(1):38–69, 1973.

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

[13] Kyle Thomas Hughes, Adam Andrew Martinez, Ankur Deep Bordoloi, and Katherine Philomena Prestridge. Compressible particle drag experiments at los alamos national laboratory. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2019.

[14] J.D. Anderson. *Modern Compressible Flow: With Historical Perspective*. Aeronautical and Aerospace Engineering Series. McGraw-Hill Education, 2003.

[15] M Parmar, A Haselbacher, and S Balachandar. Modeling of the unsteady force for shock–particle interaction. *Shock Waves*, 19(4):317–329, 2009.

[16] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser,

Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[17] A Haider and O Levenspiel. Drag coefficient and terminal velocity of spherical and nonspherical particles. *Powder technology*, 58(1):63–70, 1989.

[18] Emma Cunningham. On the velocity of steady fall of spherical particles through fluid medium. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 83(563):357–365, 1910.

VITA

I was born in Kansas City, Missouri in 1996. I pursued my bachelor of science degree in mechanical engineering at the University of Missouri - Columbia from 2014 to 2018. I joined the engineering honor societies ΤΒΠ and ΠΤΣ. It was in the latter that I met my advisor Dr. Jacob McFarland. After learning about the research that he did, I asked to join on with his laboratory as an undergraduate research assistant. I worked with the graduate research assistant Wolfgang Black, who has since earned his Ph.D in mechanical engineering, in his research with Los Alamos National Laboratory. I have had the privilege to attend Los Alamos National Laboratory in person for two summers as a student intern, and complete my masters degree through the organization. I hope to continue work with LANL in one way or another, as an organization with better resources for academic individuals is impossible to imagine.