7-12-2014

A Simple Computational Model for Particle Resuspension Behind a Normal Moving Shock

Nyssa Gilkey

Nyssa Gilkey

Candidate

Mechanical Engineering

Department

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Thesis Committee:

C. Randall Truman, Chairperson

Arsalan Razani

Svetlana Poroseva

A SIMPLE COMPUTATIONAL MODEL FOR PARTICLE RESUSPENSION BEHIND A MOVING NORMAL SHOCK

BY

NYSSA GILKEY

BACHELOR OF SCIENCE, MECHANICAL ENGINEERING

THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Mechanical Engineering

The University of New Mexico

Albuquerque, New Mexico

May 2014

ACKNOWLEDGEMENTS

I would first like to thank my committee chair and advisor, Dr. Truman, without whom this wouldn't have been possible. He has been a large influence in my academic career, both in undergraduate and graduate school, and has always pushed me to try harder, while simultaneously guiding me towards the best methods and results.

I would also like to thank my committee members, Dr. Poroseva and Dr. Razani, both of whom were last minute replacements for my committee. They both took up the task, each challenging me to examine my own way of conducting research and building simulations.

This research was supported by grant "Experimental and Numerical Studies of Respirable Particle Transport from Surfaces by Acoustic/Shock Waves," PIs: Profs. C. Randall Truman and Peter Vorobieff, from Agent Characterization, Threat Agent Science (TAS), RD-CB Basic and Supporting Science, DTRA Project #CB10-CBSFTE2-2-0071.

Finally, I would like to extend special thanks to my family and friends. Without them I could not have done this, nor would I have been able to make it this far.

A SIMPLE COMPUTATIONAL MODEL FOR PARTICLE RESUSPENSION BEHIND A MOVING NORMAL SHOCK

by

Nyssa Gilkey

B.S., Mechanical Engineering, University of New Mexico, 2010
M.S., Mechanical Engineering, University of New Mexico, 2014

ABSTRACT

The simulation of particle resuspension from a surface due to shock passage and subsequent piston flow presents a means to analyze the post-shock conditions of an environment, such as after a "dirty" bomb is detonated. This computational model is based on the "Rock'n Roll" models of particle detachment by Reeks, Reed, and Hall. The attractive forces used in the model are based on measurements by Truman et al. (2011). The simplifying assumptions of this model are: the simulation is two-dimensional, the particles are perfect spheres of identical size and are arranged in a hexagonal pattern in a bed of specified length and height. Each particle is categorized as being in one of five situations with respect to surrounding particles. These situations are used to model the forces and moments acting on the particles for resuspension. A random particle arrangement was generated within MATLAB, as well as a visual display of the particle layout as the shock wave passes over the particles. The model employs a turbulent velocity profile acquired from a STAR-CCM+ simulation with randomly-varying attractive forces between particles. Particle rolling and the dynamics of resuspending particles are computed during the passage of the shock and its following piston flow. A variety of multi-particle interactions was observed. Particles "zippered off" along the

direction of the flow. Mountains and canyons were eroded away due to either strongly-attracted or weakly-attracted particles. After the shock passes over the particle bed, predictions reveal that all particles are detached above a certain height due to high velocity piston flow. The simulation also predicts the percentage of particles resuspended when exposed to the shock.

Contents

List of Figures

List of Tables

Nomenclature

a = speed of sound

C = Cunningham Factor, eqn (8)

C_D = drag coefficient

C_L = lift coefficient

d = diameter of the particle

d^+ = dimensionless diameter, eqn (12)

F_A = attractive force

F_D = drag force

F_L = lift force

H_o = position of equilibrium separation for which adhesion force balances with the elastic rebound force, eqn (7)

Kn = Knudson number, eqn (9)

L = length scale

Ma = Mach number

M_p = moment about the pivot point

r = radius of the particle

R = gas constant

u* = shear velocity

V_p = piston velocity

V_s = shock velocity

y^+ = dimensionless wall distance

α = relative approach between particle and surface, eqn (7)

λ = mean free path of air, eqn (9)

μ = dynamic viscosity of air

ν = kinematic viscosity of air

ρ = density of air

σ = standard deviation of height distribution for asperity, eqn (7)

Chapter 1 – Introduction

Particle resuspension is both a natural and a manmade occurrence. It occurs when wind blows across a sand dune or when an explosive device detonates. All a particle needs to be resuspended in the flow is to be exposed to a high enough flow velocity for drag to overcome the attractive forces keeping it down. The topic is of interest in a variety of fields, from clean room development to HVAC system development.

This study examines particle resuspension due to the passage of a shock wave, such as when respirable particles are lifted up due to a bomb detonation. The particles are suddenly subjected to the piston velocity in the flow trailing the shock. The particles are subjected to turbulent velocity fluctuations in flows up to Mach 1.7, as well as a wide variety of attractive forces. Particle interactions are of great interest as they dictate how many particles will resuspend out of a given sample, and in what manner they resuspend.

Chapter 2 – Literature Review

A wide variety of particle models was reviewed before choosing the Reeks, Reed, and Hall "Rock'n Roll" Model. Then this model was combined with research done in particle resuspension to create a new particle simple computation resuspension model.

2.1 – General Resuspension Models

In order to understand particle resuspension, a survey of models had to be conducted. The first models that were examined were summarized by Ziskind et al. (2000). The models were focused on particle behavior on surfaces that were subjected to external excitations. They focused on fluid flow over a particle-laden surface, which was determined to be the most feasible method for particle removal through hydrodynamic moments derived from the drag force acting parallel to the surface. Prior to this, the usual method was the hydrodynamic lift force exceeding the adhesion force. With the newer model, if the drag force is not strong enough, it can cause the particles to oscillate, which led to Reeks et al. (1988) using energy balance instead of force balance to determine resuspension. Vainsthein et al. (1997) introduced a model for particle oscillations parallel to the surface. Resuspension was caused by turbulent drag force and the rate of resuspension caused by that drag force was larger than that of the laminar drag force.

The modeling of particle oscillations was based on the following concepts: 1) when a particle is placed on a surface, it forms a contact with the surface; 2) the particle formation is complete when the adhesion forces acting on the particle are fully balanced by the elastic force due to particle deformation; 3) equilibrium exists when there are no external forces, making this the reference state; and 4) particles are considered perfect

spheres and a particle may be removed immediately if an appropriate external force of moment exceeds the adhesion-based counterpart.

Linear models are the simplest 1-degree of freedom oscillation model, which have a particle of mass *m* and a spring with a constant *k*. The linear model has no damping. Reeks et al. (1988) and Lazaridis et al. (1998) came to this model, the differences in their models being how they calculated the spring stiffness. Ziskind et al. (2000) split the spring in two, with the spring constant of each spring half of the stiffness of the original spring. When the particle was turned, one spring was extended while the other was compressed. The resulting stiffness was the same, and the distances between the springs was equal to *r*.

For nonlinear models, the same basic setup is used: a particle on a smooth surface with an external force acting on it parallel to the wall. The two main factors to consider for linear models are the surface roughness and the mean external forces (the hydrodynamic lift and drag forces). These external forces cannot reach the natural frequency of the submicron particles by any existing method of particle removal from a surface.

Nonlinear models have a particle on a smooth surface with external forces acting parallel to the wall, but the adhesion models do not consider the particle response to an applied load that are not normal to the wall. This leads to peeling, where the tangential force is capable of causing normal separation of forces. If the tangential loading does not approach its critical value, the system will remain in a regime of peeling. The models predict different behaviors for hard and soft particles. Nonlinear models are best used for a perfectly smooth surface, if the oscillations are parallel to the surface. The resonant

oscillations are unstable. Once again, there is no possibility of removal for the particles by the application of a force with a frequency equal to the harmonic frequency of the particle. However an application of a force with a frequency lower than the natural frequency may still cause resonance within the system.

Ziskind et al. (1998) also studied the effects of shear on particle motion near a surface, and its application to resuspension. They split the process into two stages, where Stage One was when a particle is detached (leaves its initial site on the surface) and Stage Two occurred when a particle may either move away from the surface or return to it depending on flow conditions. They considered the mean shear rate to be on the order of $10^5 s^{-1}$ (fully developed turbulent flow) where the viscous sublayer was equal to five wall units. The viscous sublayer thickness was on the order of 10 microns so that particles with diameters between 0.1-10 microns were fully submerged in the viscous sublayer.

Ziskind et al. (1998) determined that a particle at fixed flow conditions may be either stable or unstable. When motion is unstable, the particle moves very rapidly away from the surface and this happens when the initial velocity of the particle is smaller than the local fluid velocity and the wall induced lift is larger than gravity. When wall induced lift is smaller than gravity, the particle that at first moves away from the surface may change its direction of motion and come closer to the wall. The particle will slow down and the motion eventually becomes stable. They also showed how the fluid shear rate, the particle size, initial location and velocity determine the character of motion.

Vainshtien et al. (1997) focused on semiconductor manufacturing, clean room technologies, indoor air contamination, and particle behavior in respiratory tracks. The

particles were held in place by very strong forces (physical attractions, chemical bonds and mechanical stresses) which they called the adhesion forces. The group placed emphasis on the prevention of particle deposition on surfaces rather than on the subsequent removal. They then studied particle removal by various means such as air jets, and high frequency sonic waves. The calculations were completed with an energy balance with a fluctuating lift force where the drag force moment was in equilibrium with the adhesion force moment. They found that the resuspension rate caused by the drag force is considerably larger than that of the lift force and is determined to be consistent with turbulent flow.

From this survey of resuspension models, a few important notes became apparent. A particle must peel before it can move, the lift and drag forces are important to determining the eventual resuspension of a particle and that the adhesion forces are the forces that must be overcome before resuspension can be achieved.

2.2 – Ibrahim et al.

Ibrahim et al. (2003) studied microparticle detachment from surfaces that were exposed to turbulent air flow in controlled experiments to develop a model. The mode of detachment was the process of separation by rolling, sliding, or direct lift-off of a microparticle adhering in static equilibrium to a surface. Entrainment is defined as the capture of the microparticle by the flow after being detached. Re-entrainment (or resuspension) of a microparticle is the removal of a microparticle from a surface, where the microparticle was previously airborne and subsequently deposited on a surface. A microparticle can be considered in a state of equilibrium when it is attached to a fixed

surface by adhesion. A microsphere is held on the surface by its gravitational (mg) and adhesion (F_A) forces. If it is in the viscous sublayer, it experiences a linear mean shear flow that produces drag (F_D) and lift (F_L) forces, as well as moments of these forces.

Ibrahim et al. started by doing wind tunnel experiments. In Phase One, flow is accelerated for a period of time and some microparticles are removed either in groups or individually. In Phase Two, mean flow reaches steady state. The detachment rate is coupled to the flow acceleration and detachment rates during Phase One can be as many as two orders of magnitude more than the rates during Phase Two.

They found that resuspension rate was reduced in high relative humidity, due to the absorption of water vapor at the particle-surface interface and its effects on adhesion. Ibrahim et al. also looked at surface roughness, as all surfaces are rough in some capacity.

In the study, Ibrahim et al. also examined Particle-Particle collisions. Once microparticles were detached, they moved along the surface and impacted other microparticles. This supplied enough momentum to the stationary microparticles to overcome their adhesion with the surface. Ibrahim et al. looked at detachment of stainless steel and glass microspheres and Lycopodium spore microparticles from a glass substrate.

The major factors controlled in the experiments were air-flow acceleration, final freestream velocity, relative humidity, initial number density of deposited microparticles, microparticle counting technique and the microparticle material and size. Microparticle detachment on the surface occurred in discrete, intermittent events, either in groups or individually. When two or more microparticles detached simultaneously, particles between them did not detach. Microparticle detachment occurred as rolling and/or sliding

rather than as direct lift-off. The modeling supports that this motion is rolling rather than sliding. Detachment does not always result in entrainment. Microparticles that remain adherent to the surface do not detach when subjected again to the same flow velocity history in a subsequent experiment. All microparticles do not detach at a single value of the freestream velocity, but over a range of velocities.

Ibrahim et al. (2004a) returned to their study, this time focusing on the effects of substrate cleanliness, deposition technique, storage duration up to 48 days, and moisture concentration on removal of microparticles from the surface. Typically the flow velocity is increased with time over an initial transient period until a steady mean velocity is reached. They investigated the effects of several flow and particle deposition characteristics on the detachment behavior. They looked at effects of mean flow acceleration, where there were two distinct phases of detachment: short term phase characterized by a high detachment rate and long term phase characterized by a much lower detachment rate. Dependence on flow acceleration in the considered range (transition) is small and within the uncertainty measurements.

Ibrahim et al. further examined the effects of density deposition and collisions. Depending on relative magnitude of the aerodynamic drag, lift, pull-off, gravity forces, and moments, the particle may detach from the surface in direct lift-off, pure rolling, and sliding modes. Relatively heavy particles move along the surface and collide with other particles on the surface and that impact supplies force and moments to other particles that could overcome adhesion, possibly causing detachment and resuspension. Collisions were more effective than flow alone in causing detachment. The density of particles causes particles to detach at a velocity large enough that the detaching moment supplied

is much larger than that of a similar volume of air. The density was high enough that there were enough particles to start many chains of collisions. High density particles colliding mostly resulted in detachment instead of sticking, unless freestream velocity was less than 1m/s, which resulted in sticking.

Ibrahim et al. (2004b) then focused on microparticle detachment from surfaces exposed to turbulent air flow, specifically the microparticle motion after detachment. They studied elongated counter-rotating streamwise vortices that occur randomly in space and time which cause near wall fluid burst and sweep events. The first pattern they looked at (ejection sweep pattern) was similar to classical description of burst sweep events. The second pattern (macro sweep pattern) was observed less frequently than the former and characterized by sustained high velocity periods with large sweeps. In general larger microspheres detach at lower freestream velocities. Since they studied humidity before, it was noted that with relative humidity 52%, microspheres detach in the freestream velocity range of 5 to 22 m/s. The results indicated that the normal direction is governed practically by the balance of the Hertzian and adhesion forces. In the entrainment case, the mass center displacement and the contact radius are approximately at their equilibrium position after detachment. In the tangential direction, the microspheres undergo pure rolling on the surfaces with rolling detachment and a high acceleration. High acceleration was caused mainly by the sweep part of the burst sweep event. Shortly after detachment, the adhesions dissipation moment can be neglected compared to the drag moment.

From their work, it was apparent that surface roughness and relative humidity were very important, and that particle-particle collisions also contributed to particle resuspension.

The particles once again were found to roll rather than lift off the surface in discrete intermittent events in large groups or individually. However just because two particles resuspended at the same time did not mean a particle that rested between them would as well. Particle collisions accounted for some of the particle resuspension downstream from where the particle resuspended.

2.3 – Particle Resuspension in Severe Accident Conditions

Severe Accident Conditions arise when an accident has occurred at a nuclear facility. There is rarely any data immediately following such an accident. For example, the Chernobyl release lasted two weeks, so initial deposition was poorly defined, and resuspension data was not collected until years later (Loosemore, 2003) However in studying this behavior, it can be determined when buildings are safer to enter in terms of respirable radioactive particle.

Loosmore (2003) examined particle resuspension in regards to aerosols. They relied on simple models based on a resuspension factor (the air concentration at breathing height) and resuspension rate (resuspension flux from the surface, divided by the original surface concentration). However the models showed great variability and uncertainty. Most existing data on resuspension was obtained months to years after initial deposition. These long term data sets are inadequate substitutes for a short-term emergency response scenario. For the selected data sets, the deposited material was unaltered before exposure to wind, so the resuspension could be calculated easily from measured quantities. The surfaces used were bare soil, concrete, and grass, with a friction velocity ranging from 0.1 to 1.4 m/s. Two types of particles were used with different densities, ranging in size from

submicron to silt. Roughness heights for the surface were estimated to be $1/10^{th}$ obstacle height.

The model was based on five parameters: friction velocity, time since wind began, the particle diameter, the particle density, and the roughness. The models were physically realistic: resuspension rates increased with friction velocity, and the particle diameter decreased with time, surface roughness and particle density. Larger particles protrude up higher in boundary layer, and thus experience larger removal forces. Larger surface roughness provides more shielding, acting against resuspension. The gravitation attraction is not expected to be the primary adhesive force, with Van der Waals and capillary forces providing the higher attraction, so therefore resuspension was not based directly on density. The model predicted resuspension to fall sharply with time, so that $2/3^{rd}$ the total removal in the first year occurs in the first day. This emphasizes the need for a robust model for short-time resuspension.

Hontanñón et al. (2000) used the CAESAR code for aerosol resuspension in turbulent pipe flows due to aerosol resuspension possibly being an important source of radioactivity to the environment in the late states of a severe accident in a nuclear power plant, when highly turbulent flows pass over the aerosol deposits as a consequence of the disruptive phenomena occurring in the reactor coolant system and containment building. The experimental data is limited and does not correspond to conditions found in several accidents, so the correlations are based on non-representative conditions. An older model used a force balance that only used one of aerodynamic forces (lift or drag). Their new model used both and balances with frictional and adhesive forces where previously surfaces were perfectly smooth and aerodynamic drag was only with turbulent mean flow

and under predicted particle resuspension. This model accounted for surface roughness and turbulent fluctuations.

The CÆSAR Code is a 2D Lagrangian particle tracking code that calculates trajectory within the viscous sublayer. It can solve particle equation of motion in both axial and radial directions. Within the code, gravitational force is neglected and is considered to be proportional to the cube of particle diameter. The adhesive force is proportional to particle diameter as well. Trajectory is modeled as a succession of interactions with turbulent eddies where an eddy is characterized by random axial and radial velocities. At the beginning, instantaneous components are evaluated and assumed to be constant during particle-eddy interaction. There are new positions and velocity at the end of the interaction, where it then interacts with a new eddy. For the adhesive force, particles are assumed to be hard smooth spheres where the substrate is a rigid rough shell. A particle translating in shear flow undergoes transversal force (lift force) which causes particles to travel perpendicular to flow direction. The Reynolds number associated with shear rate and particle slip is important and fluctuations of lift force are consequences of fluctuations of the axial flow velocity, assumed to follow Gaussian probability distribution. The drag force includes corrections due to both inertia and wall effects. The friction force is the resistance of the substrate to particle slide. It acts in a direction parallel to surface of contact between particle and substrate and helps maintain the particle at a position of static equilibrium.

The STORM Experiments were an experimental program on aerosol resuspension in turbulent pipe flows under conditions representative of nuclear reactors during severe accidents. The horizontal pipe was 6.3cm diameter and 5m long with aerosols of SnO_2

with steam and nitrogen as gas carriers. In resuspension phase, gas velocity was increased stepwise where Reynolds number varied from 50000 to 150000. Roughness had a drastic effect on adhesion, where 0.5μm was enough to reduce adhesive force by three orders of magnitude compared to a smooth surface. Particle surface adhesive force was proportional to particle diameter and inversely proportional to surface roughness. According to Hontannon et al., lift and drag forces only apply to particle Reynolds numbers less than unity. If particle is in contact with wall, shear Reynolds number and slip Reynolds number are related to particle Reynolds number.

They found that the steady flow conditions results in two periods of particle resuspension rate: short and long term. In the short term, resuspension rate was high initially and responsible for significant fraction of total resuspended mass. In the long term, resuspension rate decreased sharply and continued to decrease with time exposure to flow. A large portion of resuspension takes places over a short period (less than 10 ms) with the remaining material resuspending at rate inversely proportional to time exposure to the flow. An inverse relationship is maintained over wide range of times (up to several hours) and long term behavior is resistant to variations in the flow, particle size, and surface roughness.

This code captured the existence of initial phase with large particle rentrainment (where most resuspension occurs) followed by an abrupt drop in resuspension. There was a period of strong resuspension predicted by the code that is shorter than experiments indicate. The code, however, did overestimate the amount of material resuspended, which can be explained by a single-layer model, which assumes all particles are exposed to flow and the STORM tests were multilayer. It also neglected cohesion between particles.

Biasi et al. (2001) used a simple model for the interpretation of experimental data on particle resuspension in turbulent flows. They chose to focus on this as particle resuspension plays an important role in the release of radioactive material from a nuclear reactor following a severe accident (where the possible resuspension of deposited material in the primary circuit either during or after the accident will increase the eventual release to the environment. They also used the STORM experiments, as well as the "Rock'n Roll" resuspension model to develop their model. They used the same ideas of mean lift and drag force, the RMS lift and drag, the geometric factor, forcing frequencies and the distribution of adhesive forces. The other three experiments they used focused on single and multilayer resuspension: Hall's experiments, Braaten's experiments and the Oak Ridge National Laboratory (ORNL) experiments. They eventually compared to the CAESAR code results. They charted results for comparisons between experimental data and determined that several features play an important role. The model was simplistic and depended on a small number of variables. The model was mechanistic, taking into account the correct physical processes that determine particle resuspension. The model had been properly bench-marked in an experiment in which the resuspension and the distribution of adhesive forces were both measured. They looked at two particle regimes, where particles were (10-30) μm and roughly spherical and monodisperse particles with less than a mono-layer coverage. Multi-layer deposits of particles were generally sub-micron particles typical of those involved in the STORM and ORNL experiments and thus typical of deposits occurring in the light water reactor severe accident. Both regimes of particles in the adhesion were much reduced from that due to prefect contact, consistent with the influence of roughness and reduced contact due

to surface asperities. Influence of roughness is less for the smaller size multi-layer particles indicating a greater than linear dependence on particles size for the adhesion as has previously been assumed. Geometric standard deviation of the adhesion is less.

Lazaridis et al. (2010) studied turbulent resuspension of small non-deformable particles. They also chose to examine this phenomenon because resuspension can have a strong effect on timing and magnitude of radioactive sources released to the containment and the environment. The current models used force or energy balance models, where the force balance resuspension occurs when aerodynamic lift forces become greater than adhesive forces (rolling and sliding). They worked from a modified Reeks, Reed, and Hall (RRH) model, where effect of drag force on resuspension rate is included and improved agreement with experimental results. The hydrodynamic forces in turbulent boundary layer flow are usually decomposed into two parts: mean part (shear flow field) and fluctuating part (turbulent velocity fluctuations in boundary layer). The adhesive force was proportional to the particle radius and inversely proportional to the square of distance to the surface. For the small metallic particles, elastic flattening is not important, thereby providing an indirect justification for the assumption of non-deformable particles. Roughness was determined to cause a reduction and spread of adhesive force, and can be modeled by introducing effective particle radius. The mean part of the velocity field was determined by the point of detachment from the surface. Higher friction velocities corresponded to easier resuspension. Resuspension occurred when the particle is closer to the surface. An increase in particle size has same effect since larger particles feel a larger mean lift force and are resuspended more easily.

Severe Accidents provide a long term look at particle resuspension, where particles are subjected to a low speed for a long period of time. However, they do indicate how particles react with flow, with large particles being subjected to more forces and layered particles having a more linear dependence on the particle size for adhesion.

2.4 – Parmer and Shock-Particle Interaction

Parmer et al. (2009a) studied prediction and modeling of shock-particle interactions. They found that as a shock wave propagates into a gas-particle mixture, the gas velocity increases instantaneously across the shock. By contrast a particle velocity approaches the post shock gas velocity only slowly due to the finite inertia of the particles. The particles generally approach the equilibrium state faster than predicted by standard drag relation. The drag force on a particle, and hence drag coefficient, are substantially enhanced in the post shock flow. Using an accelerometer installed inside a sphere, they examined the stress-wave drag balance and reported time dependent force measurement on a stationary particle with an estimated error of less than 15%.

The force on a particle increases significantly as the shock wave passes over it. The peak force on particle can be more than 1 order of magnitude greater than steady state force in post shock flow. The propagation of shock consists of regular and irregular shock wave reflection, diffraction, and focusing phenomena. Simulations captured the increase of instantaneous drag force by more than an order of magnitude as the shock wave propagates over the particle, taking a non-monotonic approach to the steady state. Shock wave Mach Numbers were sufficiently low that the Mach number of the flow behind the shock wave was subcritical. The Lagranian-Eulerian point particle approach with the

proposed force model can thus be used as an efficient approach to compute compressible multiphase flows involving shock waves propagating through suspensions containing large numbers of particles.

They determined that the inviscid unsteady force is crucial to capturing the peak in the unsteady force on particles due to the interaction with a shock wave. The impact of shock is on the front of the particle, and the pressure near the front stagnation point is that behind a reflected shock wave. Assuming the sphere does not move, pressure at the rear stagnation point remains unchanged until the shock wave reaches the rear end.

Parmer et al. (2009b) modeled the unsteady forces on particles in compressible flow. The limitations were rooted in the relationship between the added mass and the instantaneous acceleration. They modeled the effect of the Mach number, where added mass force is realized instantaneously upon the application of acceleration due to the infinite acoustic propagation speed implicit in the incompressibility assumption. The force is proportional to the applied instantaneous acceleration and ceases to exist once the acceleration is removed. They also looked at Mach number expansion where the qualitative behavior had increasing added effect at a finite Mach number.

Parmer et al. (2010) studied improved drag correlations for spheres and their applications to shock tube experiments. For subcritical Mach numbers, the flow around a spherical particle is shock free. The drag coefficient is only weakly affected by compressibility effects. For supercritical but subsonic Mach numbers, a shock wave of limited radial extent exists on the sphere, and the drag coefficient becomes more strongly dependent on the Mach number. For supersonic Mach numbers, a bow shock exists that leads to a large

increase of the drag coefficient (the bow shock does not appear at precisely sonic conditions but the proceeding simple separation into regimes is sufficient for modeling purposes). The upper limit of Mach numbers examined was 1.75. The boundary layer effects were eliminated by hanging the spherical particles from a spider web thread. Interference between particles was reduced by testing no more than three particles simultaneously. A large part of particle trajectory was recorded using multiple shadowgraphs in a single run.

Fedorov et al. (2002) developed a numerical simulation of shock wave interaction with a near wall particle layer. They were hoping to prevent dust explosion in mines and other similar industries. gas-dust mixture behind a shock wave was modeled. They examined dust lifting related to compression and expansion waves following multiple reflections of the shock wave off of a contact surface. The numerical solution was carried out for a one-velocity, one-temperature mathematical model. They found two different wave pictures in a dense layer of dust by examining the shock wave reflection.

Suzuki et al. (2006) studied particle motion behind a planar shock wave. They used horizontally placed shock tubes and direct photograph technique that was synchronized to the shock wave. They found that the speed of rotation and the velocity of the particles were strongly affected by the floor conditions. They speculated that the upward forces responsible for particle resuspension were generated by the shock wave reflections between the particles and the floor.

Wayne et al. (2013) examined shock-driven particle transport off of both smooth and rough surfaces. This is an area with little research conducted, so they modified an

existing shock tube to study respirable particles resting on a horizontal surface that are then detached by a shock-driven flow. They measured the effects of surface roughness on the resulting particle cloud. They found that the growth of the particle cloud was dictated by several factors, where the most influential was the adhesive force between particles and the surfaces. The stronger the attraction, the smaller and slower the resulting cloud. They also noted that the particle clouds extended above the boundary layer, which they believe suggests that particle lag may play a role in the evolution of the flow.

Jacobs et al. (2012) studied high-order resolution Eulerian-Lagrangian simulations of dispersion of particles that were accelerated by the piston flow following a moving shock. Their work was two-dimensional and looked at bronze particles with a volume concentration of 4%. The particles were originally arranged in various shapes (rectangle, triangle and circle). They found that the flow had an impact on the cloud formation, such as reflected shocks and unstable wakes. The particles would be pulled out of the clouds in sharp corners, so the smoothest transport was with the circular cloud.

Shock-Particle interaction is of great importance to the model being developed as the particles go from a no flow state to a post shock state. They found that as the shock passes, the particles are subjected to forces up to one order of magnitude greater than what they would be using the piston flow. They are also subjected to a rapid change in flow velocity.

2.5 – Powders and Spores

Krauter and Biermann (2007) studied the resuspension of fluidized spores in ventilation systems. They used Bracillus anthracis (Anthrax) particles that were less than 5 microns

in diameter which allowed penetration into pulmonary aveoli. There is minimal data on human infective doses available. There were variations in individual susceptibility, strain virulence, spore preparation, and physical characteristics. The experimental air flow was similar to air circulation in ordinary buildings. They focused their study in postal buildings, which other studies show that an area could still be contaminated several days after original contamination. The particles were deposited and resuspended from ducts at different rates depending on size, velocity, physical configuration, duct surface, and environmental factors (humidity, dirt, and biofilm formation). The particle resuspension rate was qualified as being uncertain within 2 to 3 orders of magnitude, values ranging from 10^{-13} to 10^{-14} 1/s . Bioparticles smaller than 10 microns moved with the airflow. Spores on the surface tended to move from a source location to some other location on the duct surfaces. After being viewed for seven hours, the initial cloud of spores moved through a ventilation duct for 25 seconds, but spores continued to move though the duct for the next several hours (50% decrease took approximately 3 hours). Resuspension on plastic was no different than resuspension on steel.

Gac et al. (2008) studied the turbulent flow energy for resuspending powder particles. They looked at the interactions between particles in powder agglomerate structure and between particles and the support surface, on the microscale. They focused on the short range effects (van der Waals forces), the electrostatic forces and the capillary attraction induced by presence of condensed fluid between particles. They worked with the Eddy Fluid Particle Model which uses a damping coefficient, including two effects: damping in solids and in a fluid. It also uses kinetic, turbulent and thermal energy. The mean flow kinetic energy of a moving fluid degrades to thermal energy, but usually the first part of it

is transformed into turbulent energy. The turbulent viscosity produces drag which is independent of the molecular viscosity of the fluid. They used the Verlet algorithm for their calculations and found that efficient resuspension of particles from a powder sample to the form suitable for inhalation requires considerable energy for fluidization and breaking up of particle aggregates. Inter-particle cohesive forces have to be overcome by the stresses of the interactions between the flowing air and the aerosolized aggregates.

2.6 – Resuspension from Indoor Surfaces

Kim et al. (2008) focused on resuspension on indoor surfaces, where it may be of importance to improve indoor air quality in modern buildings. They also wanted to examine how best to protect against indoor dispersal of toxins. They modeled room flow dispersion profiles and then looked at how these profiles interacted with particles. They examined the mechanisms that caused particle resuspension on indoor surfaces, identified parameters that were relevant to resuspension as well as evaluated their model against experimental data. They developed empirical correlations that can predict particle resuspension as a function of time, particle size, friction velocity, surface roughness and the van der Waal's interactions between the particle and the surface.

Boor et al. (2011) also studied resuspension from indoor surfaces. Particles deposited on indoor surfaces can be resuspended through a variety of mechanisms, such as a passing fluid stream and can therefore increase the particle concentrations in the air. This in turn can lead to the particles being inhaled and could cause respiratory problems. Boor et al. focused on monolayer and multilayer particle deposits, which were generated by two separate seeding procedures, so the multilayer particle deposits could be thicker, with a

higher quantity of particles. They also developed a micro-scale wind tunnel. They used fluorescent particles to detect the number of particles resuspended. They found that particle resuspension was greater for multilayer deposits. For the purposes of their research, this suggested that heavy dust loads would have a greater resuspension rate than light dustings.

Kassab et al. (2012) examined micrometer particle detachment from a variety of surfaces. They examined glass, ceramic, and hardwood substrates experimentally, where the particles were deposited on the lower surface of a wind tunnel by gravitational settling. The airflows went up to 16m/s and individual particle trajectories were mapped using high-speed imaging. They found three different types of motion: immediate lift off, where the particles would completely leave the surface with no rolling or bouncing; rolling/bouncing, which were caused by the particles moving over an uneven surface; and complex motion, where particles rolled and bounced before lifting off. Surface roughness would affect what type of motion.

2.7 – Reeks, Reed, and Hall (RRH) Models

Reeks et al. (1988) developed the basis of the "Rock'n Roll" model for particle resuspension. Prior to this model, turbulent energy was not taken into account in the modeling process which left the force balance models incomplete. The turbulent energy transferred to the particle influences resuspension, as it allows the particle to move within a certain distance from the wall, but not actually detach from the wall. They classified detachment as when a particle accumulates enough vibrational energy to escape from the

surface adhesive potential well. This allowed them to develop an equation for rate constant *p*

$$p \sim \omega_o \exp[-\frac{Q}{2(PE)}] \quad (1)$$

where ω_o is the natural frequency of vibration, Q is the depth of the surface adhesive potential well and (PE) is the average potential energy of a particle within the well. In this model, potential energy depends on the fluid and mechanical damping, as well as the energy spectrum of the fluctuating aerodynamic force, especially when it is near the resonant frequency of the particle-surface vibration. The rate constant *p*, based on the van der Waals adhesive forces for a particle on a surface (with both particle and surface being elastically deformed), indicates that particles can be resuspended from a surface more easily than anticipated from a balance of adhesive and aerodynamic forces. The observed dependence of resuspension on flow and particle size was the same as that predicted by this model. The resuspension rates from surfaces where there is a wide range of adhesive forces (usually due to particle roughness) were shown to decay over time, almost inversely with the duration of exposure to the flow.

Reeks and Hall (2001) expanded on their 1988 model. They began with measurements of short-term resuspension of alumina spheres, ranging in size from 10 to 20 μm, as well as graphite particles. These were being resuspended from a polished, stainless steel flat plate in a fully developed turbulent flow. Measurements were made of the normal and tangential forces holding the particles to the surface. These forces covered a broad range of values, but were on average much lower than the values for a smooth contact. The tangential forces were typically $1/100^{th}$ of the average normal adhesive forces, indicating

that the drag forces play a more important role in resuspension than the lift forces. Forces and dimensions used in the Reeks, Reed, and Hall (RRH) "Rock'n Roll" model are shown in Figure 1. This causes their force balance to be dominated by drag forces. Their results also showed that the contribution of resonant energy to resuspension is relatively small, especially when considering the overwhelming influence of the drag forces. This simplifies the resuspension rate constant so that a moment balance of the adhesion and aerodynamic forces about points of contact can be made. These points of contact are created by surface asperities. This way of analyzing particle resuspension resulted in values of resuspension rate similar to the original RRH model, though they are much improved.

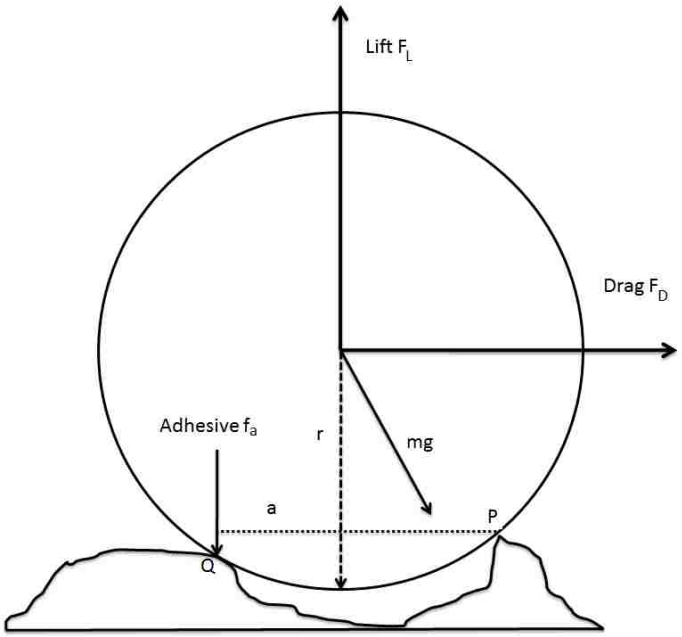


Figure 1 – Rock'n Roll Model forces and dimensions

As will be discussed below, this model will be used within the numerical method discussed herein. For each particle there is a point of contact (P) and a point of adhesion

(Q). The flow moves from left to right, as indicated by the drag force. There is a lift force and a drag force, as well as a distance between points Q and P (indicated by a). The particle will pivot about point P, where moments about point P due to drag, lift and the adhesion force will be computed. Then the particle will twist off to be resuspended into the flow after it has exited the surface adhesion well, where particles are caught due to low velocity and high attractive force.

2.8 – Particle Resuspension Force Measurement

Zhang and Ahmadi (1995) examined particle detachment from rough surfaces in turbulent flows. They assumed that the real area of contact between the particle and the surface was determined by elastic deformation of surface asperities, so they examined the surface properties in their research. They used both the sliding and rolling methods of detachment, as well as examining near wall eddies and turbulent motion, with particular focus on critical shear velocities. They found that the dominant resuspension mechanism for spherical particles on a rough surface is rolling, and that the critical shear velocity was reduced as the roughness increased. However, it increased when the radius of surface asperities decreased. They also concluded that gravitational effects on the particle were negligible for small particles. They examined particles ranging in size from 0.1 microns to 100 microns.

Zhang and Ahmadi (2000) examined aerosol particle resuspension in turbulent channel flow. They used direct numerical simulations of the Navier-Stokes equation to generate the instantaneous fluid velocity field. They examined the particle resuspension mechanisms, and they compared their results with Reeks et al. (1988), and found good

agreement between Reeks' results and their simulated resuspension. rates. They found that larger size particles (60 microns) move perpendicular to the wall when resuspended due to the lift force. Near-wall turbulent flow structures play a significant role in particle resuspension. They examined particles ranging in size from 30 microns to 60 microns.

In collaboration with Profs. Truman and Vorobieff at UNM, D. Srivastava and Prof. Hugh Smyth at the Univ. of Texas College of Pharmacy measured adhesion forces between glass particles and artificially-roughened surfaces (Truman et al., 2011). Particles were classified as respirable or small (diameter <5 microns) or as large (mean diameter ~10 microns). Surfaces with varying degrees of roughness (smooth, nano-rough, micro-rough, and nano-and-micro-rough) and varying surface chemistries (-CH_3, -COOH, and –NH_2) were produced. The CH3-modified surface is hydrophobic, while the other two are hydrophilic.

To produce nano-roughness, silver was etched onto 10 mm stainless steel discs followed by sputter coating. Micro-roughness was obtained by physically roughening stainless steel discs with a ¼" 240-grit sanding band. Nano-and-micro-rough surfaces received both treatments. An Optical Profiler was used to measure the roughness of these surfaces. To vary the surface chemistry, self-assembled monolayers (SAMs) of 1-dodecanethiol, 11-Mercaptoundecanoic acid, and 11-amino-1-undecanethiol, hydrochloride were used. To improve the adhesion of gold to the discs, a 20 nm layer of smooth silver was sputter coated onto the smooth and micro-rough discs. Then, a 15 nm layer of gold was thermally evaporated onto all the discs.

Resulting surface roughness measured with a scanning electron microscope (SEM) varied from 0.19-0.38 microns for the nano-rough surfaces, from 1.1-1.8 microns for the micro-rough surfaces, and from 1.9-2.2 microns for the nano-and-micro-rough surfaces. The resulting adhesion forces for large particles, shown in Figure 2, generally increase with increasing roughness except that the smooth glass surface has the highest adhesion force. While there are some differences between adhesion forces with different surface chemistry, there is no clear trend. The hydrophilic NH2 surfaces were significantly more adhesive on the nano-rough surfaces, while the hydrophobic CH3 surfaces were significantly more adhesive on the nano-and-micro-rough surfaces. Thus for the present study, the values of adhesion force considered in the present work were selected to span the range of values at 0.5, 5, and 20 nN.

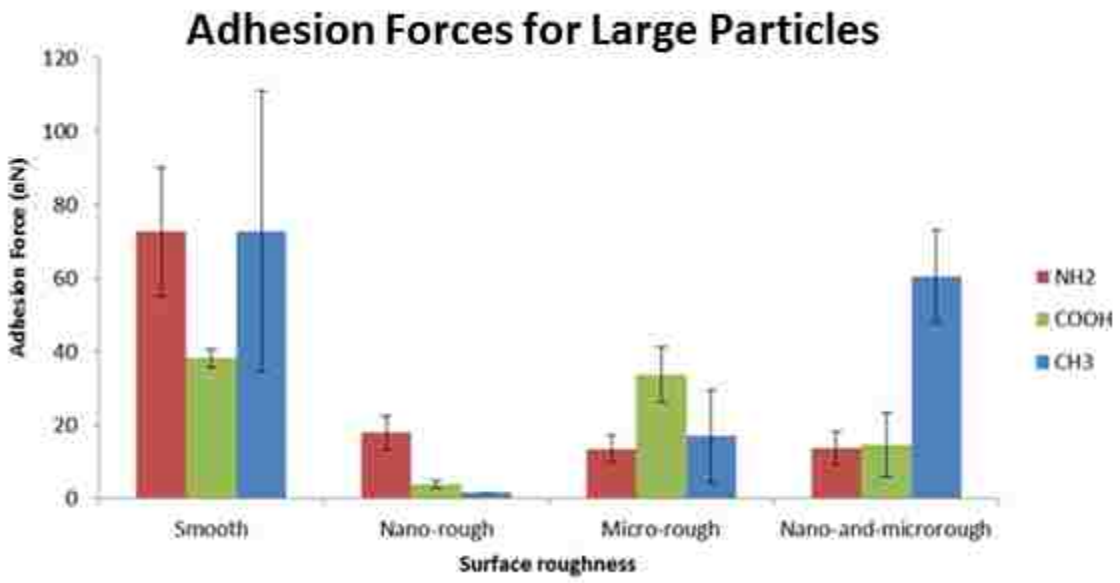


Figure 2 – Average adhesion forces for large particles with different surface chemistries (Truman et al., 2011)

2.9 – Literature Review Summary

These ideas, the neglect of resonance effects, the importance of moment balances and effect of roughness on particle adhesion, were brought together to form a new method to analyze particle resuspension that was developed for this study. Reeks, Reed, and Hall (1988) determined how particles resuspended, and which forces are the most pertinent for a model of a particle that pivots about a point before lifting off. This model is applicable to the case of a nano-rough surface, with few contact points between the surface and the particle. The nano-rough adhesion forces measured by Truman et al. (2011) had the smallest variance. When spherical particles are arranged in a hexagonal grid, so that particles touch at two contact points with the surface beneath them, they interact with each other in a way that is similar to a nano-rough surface particle interaction.

Chapter 3 – Particle Movements

For this model the particle dynamics were simplified with the assumptions described in this section. A particle's geometry relative to its neighbors falls into a limited number of situations, which will also be described and analyzed within this section.

Ten basic assumptions are made for the model.

Assumption 1 – The model is two-dimensional on a vertical plane including the streamwise and wall-normal directions.

Assumption 2 – Particles are of equal size, shape and mass. They are hard particles, meaning they do not deform. In the vertical plane, the particles are circular although their mass is computed for spheres.

Assumption 3 – The collection of particles is arranged in a hexagonal grid pattern. The grid is defined by specifying the number of rows and the number of columns, where columns are diagonal. In Figure 3, Particle A is located in one row, whereas Particles B, C and D are located in another row. The columns are oriented upwards to the right, thus Particles A and B are located in one column, and Particle C is located in the next column.

Assumption 4 – From Assumption 2, particles ABC form an equilateral triangle, with side lengths equal to the diameter of the particles, and thus angles A, B and C are each 60°, or $\pi/3$ radians. The points of contact lie on the line between centers of adjacent particles, and form equilateral triangles with the particle centers.

Assumption 5 – At the point of contact between particles, there is an attractive force or reactive force.

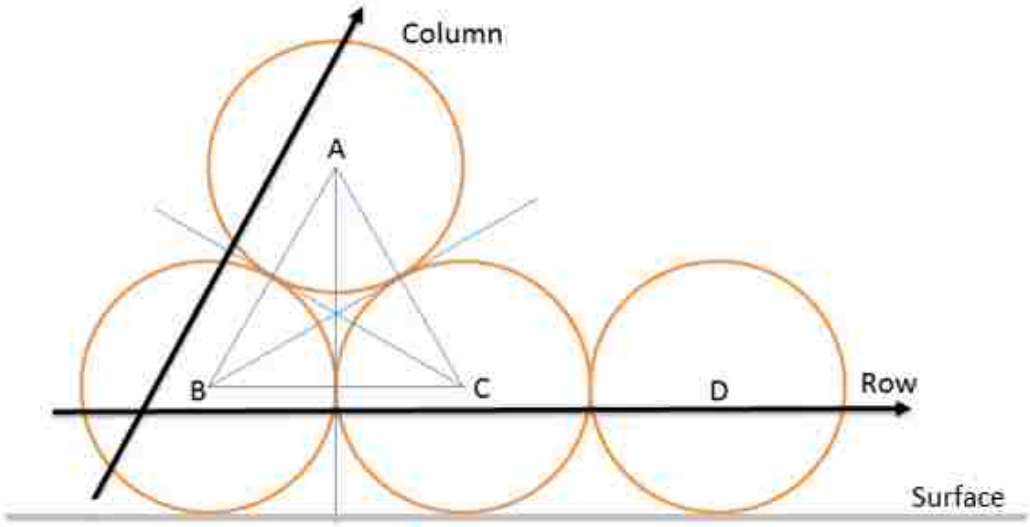


Figure 3 – The layout of four particles in a perfect hexagonal grid. Particles ABC form an equilateral triangle, with side lengths equal to particle diameter.

Assumption 6 – Particle mass is negligible with respect to the attractive forces and lift and drag forces. Assuming a glass particle has a radius of 5 microns, the weight of the particle would be 1.23×10^{-17} N, which is far smaller than the attractive forces on the order of 10^{-9} N.

Assumption 7 – The simulation takes place in a dry environment, with low relative humidity.

Assumption 8 – Once a particle resuspends, it is no longer considered a part of the simulation. It does not bounce or collide with other particles, triggering their resuspension.

Assumption 9 – The particles are not resuspended by the shock, but rather the piston velocity following the shock.

Assumption 10 – Since the particles are stacked in a grid pattern, each particle can be arranged in one of five ways, as described below.

Each of these five arrangements is defined by the particle's ability to move and its interactions with the particles that surround it. A particle is immobile when it is in contact with another particle above it. For example, in Figure 3, Particles B and C cannot move because Particle A rests on top of both of them. Particle D can move as there is no particle above it. An immobile particle is defined as Situation Zero. The particle does not enter into the resuspension equations until it is able to move.

3.1 – Situation One

Situation One occurs when a particle sits atop two other particles, with no particles to the immediate left or the immediate right. This is shown in Figure 4a. Assuming the air flow is in the direction indicated by Figure 4a, then the free body diagram (FBD) of the particle is shown in Figure 4b. There is a pivot point (P) and an attraction point (A). The particle is acted on by four forces: the lift force (F_L), the drag force (F_D), the weight (mg) and the contact force (F_A) at point A There is also an attraction force at P, but it produces no moment about point P.

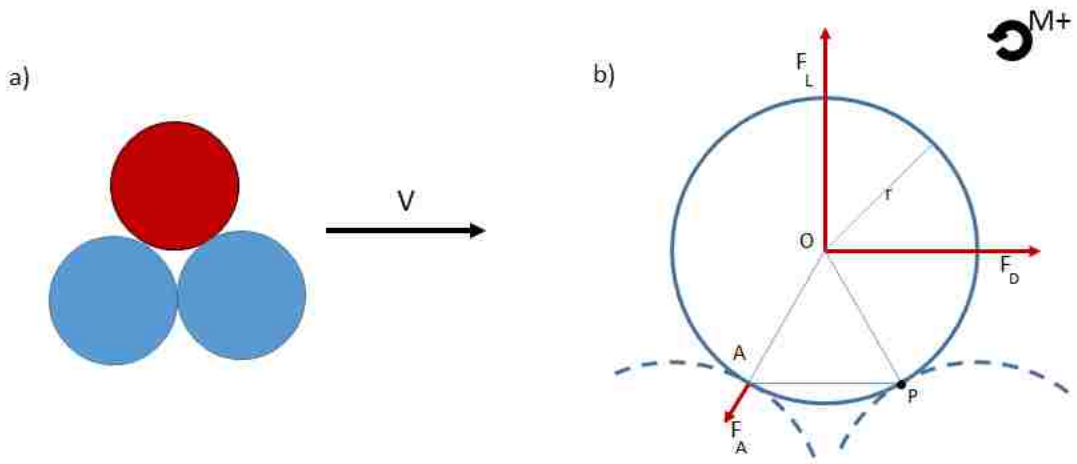


Figure 4 – a) Particle atop two particles, with no particles to either side. b) Free body diagram of a Situation One particle.

Taking the moment about pivot point P, when there is no motion, results in the following equation:

$$\sum M_{1P} = 0 = F_A r \cos\left(\frac{\pi}{6}\right) - F_D r \cos\left(\frac{\pi}{6}\right) - F_L r \cos\left(\frac{\pi}{3}\right) \quad (2)$$

where r is particle radius.

3.2 – Situation Two

Situation Two is a particle atop two other particles, with no particle to the immediate left. However, it does have a particle to the immediate right. This is shown in Figure 5a. Assuming the airflow is in the direction indicated by Figure 5a, then the free body diagram of the particle can be seen in Figure 5b. There is a pivot point (P) and two attraction points (A1 and A2). The pivot point P was chosen as it is the point of contact where the smallest lift force would be required to overcome the attraction forces in the moment balance. The particle is acted on by four forces: the lift force (F_L), the drag force

(F_D) and two contact forces (F_{A1} and F_{A2}). The drag force (F_d) creates no movement or moment, as it goes through the pivot point P. There is also an attractive force at point P, but it produces no moment about point P.

Taking the moment about point P, when there is no motion, results in the following equation:

$$\sum M_{2P} = 0 = F_{A1}\cos\left(\frac{\pi}{6}\right)r + F_{A2}\cos\left(\frac{\pi}{6}\right)r - F_L r \tag{3}$$

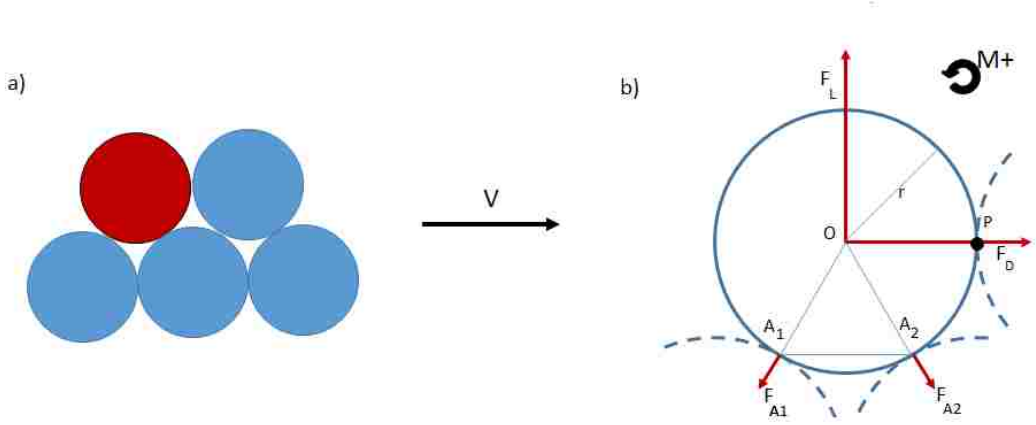


Figure 5 – a) Particle atop two particles, with a particle to the immediate right. b) Free body diagram of a Situation Two particle.

3.3 – Situation Three

Situation Three is a particle atop two other particles, with no particle to the immediate right. However, it does have a particle to the immediate left. This can be seen in Figure 6a. Assuming the airflow is in the direction indicated by Figure 6a, then the free body diagram of the particle can be seen in Figure 6b. There is a pivot point (P) and two attraction points (A1 and A2). The pivot point P was chosen as it will result in resuspension with the smallest lift force. The particle is acted on by four forces, the lift

force (F_L), the drag force (F_D) and two contact forces (F_{A1} and F_{A2}). There is also an attractive force at point P, but it produces no moment about point P.

Taking the moment about point P, when there is no motion, results in the following equation:

$$\sum M_{3P} = 0 = F_{A1} r \cos\left(\frac{\pi}{6}\right) + F_{A2} r \cos\left(\frac{\pi}{6}\right) + F_L r \cos\left(\frac{\pi}{3}\right) - F_D r \cos\left(\frac{\pi}{6}\right) \qquad (4)$$

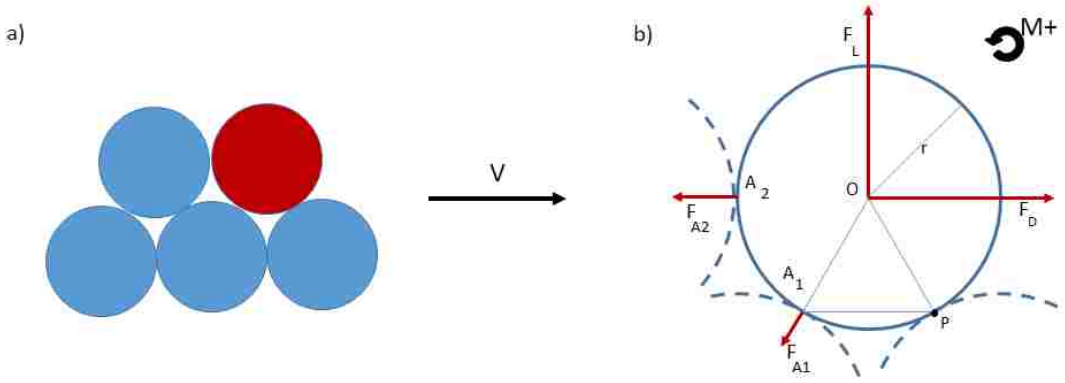


Figure 6 – a) Particle atop two particles with a particle to the immediate left. b) Free body diagram of a Situation Three particle.

3.4 – Situation Four

Situation Four is a particle atop two other particles, with particles to the immediate left and the immediate right. This can be seen in Figure 7a below. Assuming the velocity is in the direction indicated by Figure 7a, then the free body diagram of the particle can be seen in Figure 7b. There is a pivot point (P) and three attraction points (A1, A2 and A3). The pivot point P was chosen as it is the point of contact where the forces required for resuspension are the smallest. The particle is acted on by five forces, the lift force (F_L), the drag force (F_D) and three contact forces (F_{A1}, F_{A2} and F_{A3}). The drag force is acting

through the pivot point P. There is also an attractive force at point P, but it produces no moment about point P.

Taking the moment about P, when there is no motion, results in the following equation:

$$\sum M_{4P} = 0 = F_{a1}\cos\left(\frac{\pi}{6}\right) + F_{a3}\cos\left(\frac{\pi}{6}\right) - F_L r \tag{5}$$

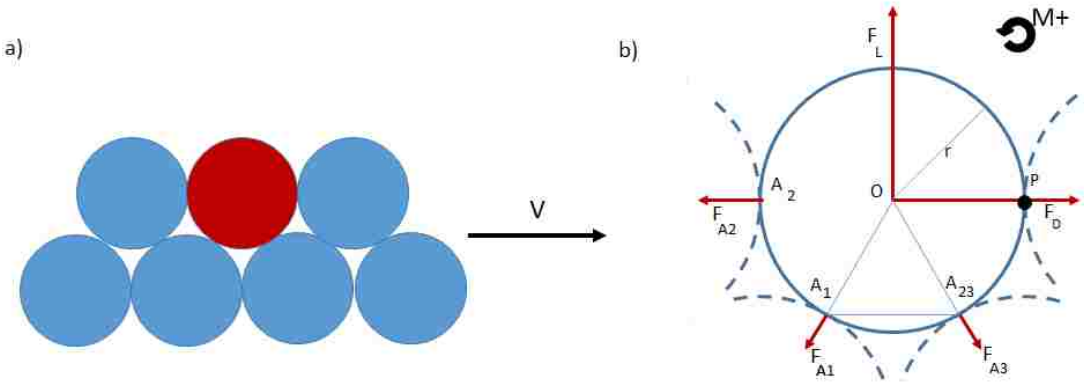


Figure 7 – a) Particle atop two particles, with particles to the immediate left and right. b) Free body diagram of a Situation Four particle.

3.5 – Simplification and Summary of Moment Equations

With these equations, it can be seen that Situations Two and Four have the same moment balance. For Situation Four, the third contact force (between the particle and the particle to its immediate left) produces no moment. The drag force does not enter into the moment balances for Situations Two and Four. Thus the moment balances for Situations One, Three and Two/Four, respectively, are:

$$\sum M_{1P} = 0 = F_A r\cos\left(\frac{\pi}{6}\right) - F_D r\cos\left(\frac{\pi}{6}\right) - F_L r\cos\left(\frac{\pi}{3}\right) \tag{2}$$

$$\sum M_{3P} = 0 = F_{A1} r \cos\left(\frac{\pi}{6}\right) + F_{A2} r \cos\left(\frac{\pi}{6}\right) + F_L r \cos\left(\frac{\pi}{3}\right) - F_D r \cos\left(\frac{\pi}{6}\right) \quad (4)$$

$$\sum M_{4P} = 0 = F_{a1} \cos\left(\frac{\pi}{6}\right) r + F_{a3} \cos\left(\frac{\pi}{6}\right) r - F_L r \quad (5)$$

F_A is the contact force and for the nano-rough surfaces, it falls into three categories. NH2 surfaces have attractive forces ranging from 15 to 25 nN, COOH surfaces range from 4 to 6 nN and the CH3 has the smallest attractive forces, ranging from 0 to 1 nN. All three of these will be examined.

For the lift and drag forces, lift and drag coefficients had to be calculated. For the drag, Soltani and Ahmadi (1995) was used, specifically the equation for Stokes Drag force.

$$F_D = \frac{5.8\pi\rho d {u^*}^2 L}{C} \quad (6)$$

The density, ρ of air at 70°F is 1.184 kg/m^3, d is the diameter of the particle and u* is the shear velocity (3.96 m/s for Mach 1.2, 13.4 m/s for Mach 1.7). The length scale L is defined as

$$L = \frac{d}{2} + 2.76\sigma + H_o - \alpha \quad (7)$$

where σ is the standard deviation of the height distribution for asperity, H_o is the position of the equilibrium separation for which the asperity adhesion force balances with the elastic rebound force, and α is the relative approach between the particle and the surface. Because α, σ and H_o are on the order of the asperity height, or approximately 200 nm according to Truman et al. (2011), they are negligible compared to the diameter of 10 microns. Thus the length scale is L = d/2.

C is the Cunningham factor, defined as

$$C = 1 + Kn\left[1.257 + 0.4\exp\left(-\frac{1.1}{Kn}\right)\right] \tag{8}$$

where Kn is the Knudson number and is defined as

$$Kn = \frac{2\lambda}{d} \tag{9}$$

The mean free path of air, λ, is also on the order of nanometers, and is much smaller than the diameter, so we can assume that C = 1.

The drag force written in terms of drag coefficient is

$$F_D = \frac{1}{2}\rho V^2 C_D \pi r^2 \tag{10}$$

where V is the freestream velocity (104m/s) and r is the particle radius (5 microns). Thus the drag coefficient based on eqn (6) is

$$C_D = \frac{23.2{u^*}^2}{V_P^2} = \begin{cases} 0.0337\ for\ Mach\ 1.2 \\ 0.0417\ for\ Mach\ 1.7 \end{cases} \tag{11}$$

Ahmadi (2014) discusses various proposals for computing lift forces on small particles due to shear. The form that provides the best agreement with experimental data is dimensionless lift force $F_L^+ = F_L/(\rho\nu^2)$ = 15.75*$d^{+1.87}$ = 88.3 for Mach 1.2 and 873.4 for Mach 1.7, where the dimensionless particle diameter is

$$d^+ = \frac{du^*}{\nu} = \begin{cases} 2.53\ for\ Mach\ 1.2 \\ 8.56\ for\ Mach\ 1.7 \end{cases} \tag{12}$$

where ν is the kinematic viscosity of the air ($\nu = 1.565 \times 10^{-5}$ m^2/s at 70°F).

Lift force written in terms of lift coefficient is

$$F_L = \frac{1}{2}\rho V^2 C_L \pi r^2 \tag{13}$$

so that the lift coefficient is

$$C_L = \frac{2F_L^+ \nu^2}{V^2 \pi r^2} = \begin{cases} 0.0509 \; for \; Mach \; 1.2 \\ 0.0546 \; for \; Mach \; 1.7 \end{cases} \tag{14}$$

These values will be used for the subsequent calculations relating to the particle resuspension.

3.5.1 – Threshold Velocity for Surface NH2, FA = 20 nN

The moment equations for the four situations were placed into MATLAB, where $F_A = 20 \times 10^{-9}$N, which is the average attractive force for the surface NH2. The lift-off velocity can be determined as the minimum flow velocity for which the moment is negative and the particle can pivot free and detach into the flow. This occurs when the total moment due to attractive forces between the particles is overcome by the moment due to the lift and drag forces. This code, listed in Appendix A.1, is called MomentCalc.

Figure 8 shows the flow velocity versus net moment on particles in each of the situations (where Situations Two and Four are identical). Lift-off velocities are indicated by the symbol at zero net moment. As expected, Situation One requires the lowest velocity for the particle to detach. It is somewhat surprising that Situation Three requires a higher

lift-off velocity than Situations Two and Four. This is due to the different pivot point location.

At a high velocity, if the moment is less than zero, the particle pivots about P and the particle will soon resuspend. For this specific value of the attraction force, a particle in Situation One would resuspend when the velocity exceeded 84m/s, Situations Two and Four (given they are based on the same equation) would resuspend at 122 m/s and Situation Three would resuspend at 129 m/s. Note that Situation One has only one attractive force to overcome, whereas Situation Three has two attractive forces to overcome and they are arranged in such a way that they are more difficult to overcome than the two attractive forces in Situations Two and Four.

However, as the freestream velocity is equal to approximately 104 m/s, only the Situation One particles will resuspend due to moments breaking the attractive bonds, and even those particles may not resuspend as they require a lift-off velocity of 84m/s.

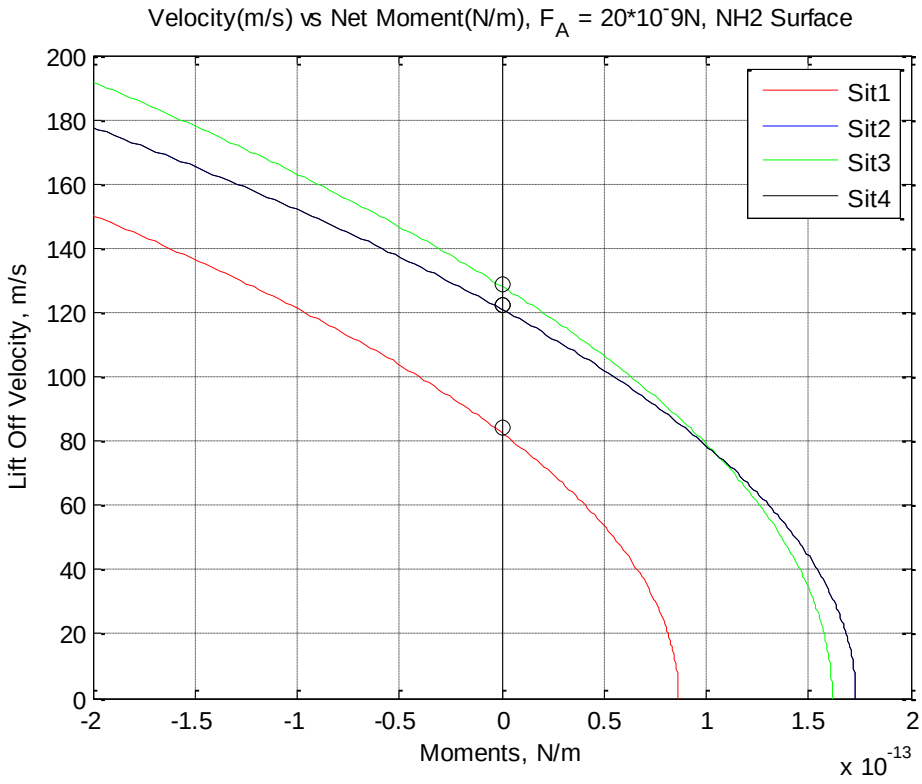


*Figure 8 – Velocity vs net moment for each situation where F_A = 20*10-9N. Symbols indicate minimum velocity for particle to detach, termed lift-off velocity. The moment balance is identical for Situations Two and Four.*

3.5.2 – Threshold Velocity for Surface COOH, F_A = 5x10nN

If the attractive force between particles is equal to $5x10^{-9}$N, the mean attractive force for the surface COOH, the results are different as presented in Figure 9.

Situation One would start to pivot at 42 m/s, Situations Two and Four start to resuspend at 61 m/s and Situation Three starts to move at 65 m/s. Note that as the attractive force is less, the lift-off velocity is significantly decreased. All of these particles has the potential to resuspend as the lift-off velocity for each is less that 104 m/s. However, due to the velocity profile, they will only resuspend if the velocity is high enough at that height.

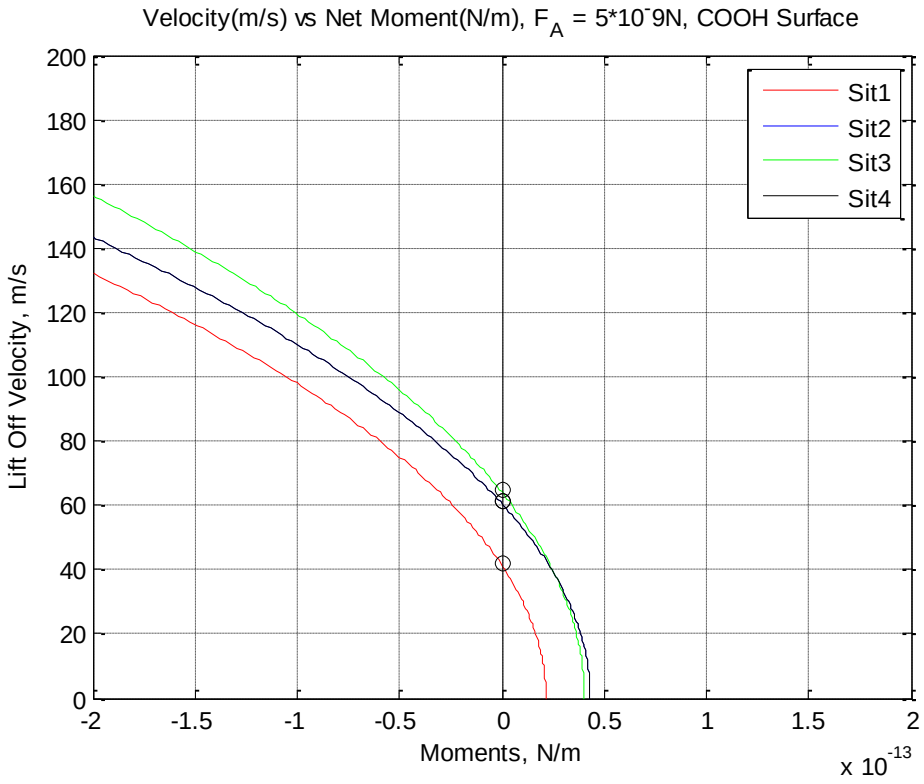


*Figure 9 – Velocity vs net moment for each situation, F_A = 5*10-9 N. Symbols indicate minimum velocity for particle to detach, termed lift-off velocity. The moment balance is identical for Situations Two and Four.*

3.5.3 – Threshold Velocity for Surface CH3, $F_A=0.5x10^{-9}N$

If the attractive force between particles is equal to $0.5x10^{-9}$N, the mean value for a nano-rough particle-surface interaction on a CH3 surface, the results are different as can be seen in Figure 10.

Situation One would start to pivot at 14 m/s, Situations Two and Four start to resuspend at 20m/s and Situation Three starts to move at 21m/s. These attractive forces are the smallest, and thus the particles will resuspend the easiest. They will also resuspend at heights smaller than those that would be found using the COOH attractive forces, as the smallest COOH lift-off velocity (40 m/s) is twice that of the largest NH3 lift-off velocity

(20m/s). A complete tabulation of the results can be found in Table 1. Note that as the attraction force decreases, so does the lift-off velocity of the particle.

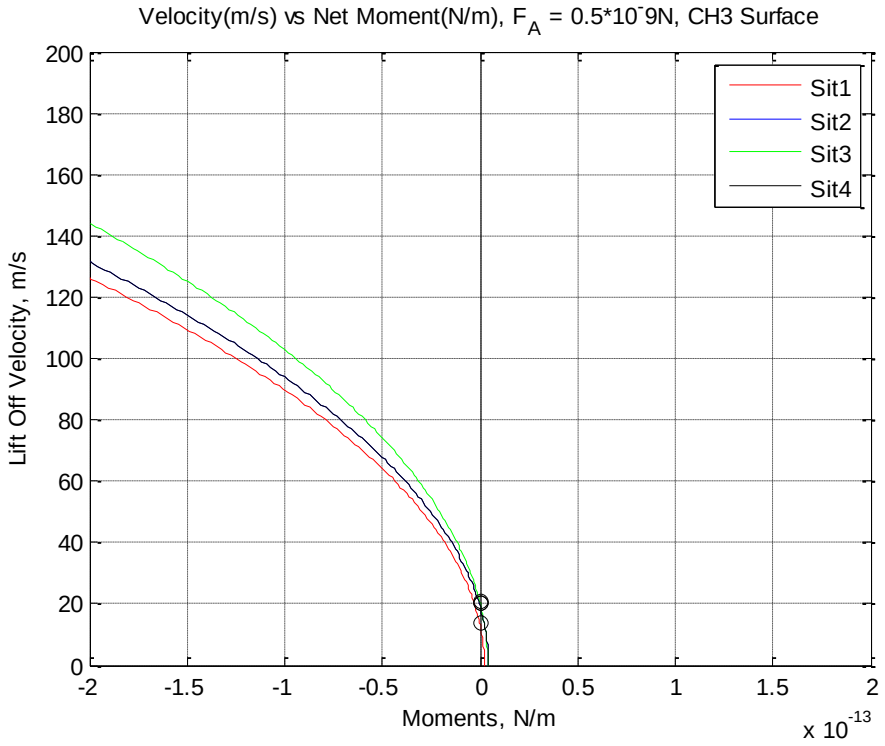


*Figure 10 – Velocity vs net moment for each situation, F_A = 0.5*10-9N. Symbols indicate minimum velocity for particle to detach, termed lift-off velocity. The moment balance is identical for Situations Two and Four.*

Table 1 – Lift-Off velocities for each Situation and each attractive force estimate

	Lift-off Velocities (m/s)		
	NH3 $20x10^{-9}N$	COOH $6x10^{-9}N$	CH3 $0.05x10^{-9}N$
Situation One	85	42	14
Situation Two	122	61	20
Situation Three	129	65	21
Situation Four	122	61	20

Chapter 4 – Grid Generation

In order to run the simulations, a grid had to be generated that would determine the random placement of the particles that were being tested. Each time the simulation would run, a new random positioning of the particles would be used, meaning that each solution would be unique and would be calculated as a whole. In the notation system, [#] indicates a value in the matrix, and {#} indicates the situation into which each particle is categorized.

In Figure 11, two random particle arrangements are shown that are based on the same two basic requirements: there are four rows and 13 columns. Each arrangement is unique and would yield a different solution. Each particle in arrangements A and B is categorized according to the particles surrounding them. Figure 11 A, for example, has two Situation One particles in the 4th row whereas Figure 11 B has only one. Figure 11 B also has a Situation Four particle in the 4th row, whereas Figure 11 A does not.

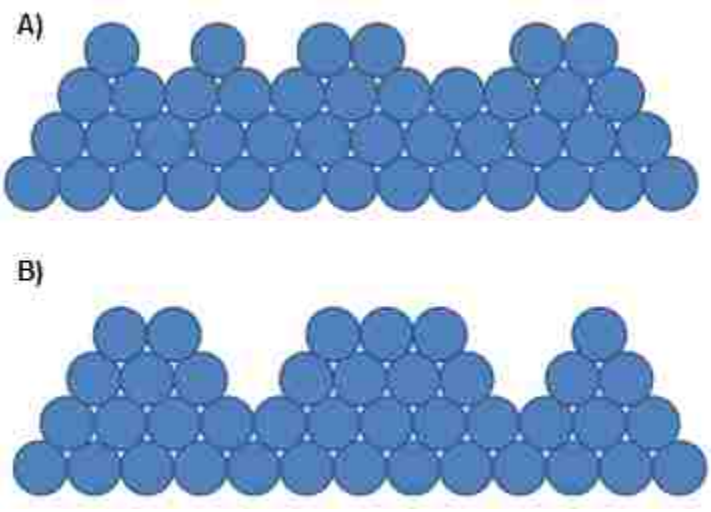


Figure 11 – Two random particle arrangements based on the same parameters

For a specified number of rows and columns, the particle layout was determined first. The code in its entirety is listed in Appendix A.2, called gridgen. An [MxN] matrix was established where M was the minimum number of rows and N was the minimum number of columns. This matrix was then filled with random values between 0 and 1 that were rounded to the nearest integer (being 0 or 1). Zero (0) indicates that no particle fills that position, while one (1) indicates that a particle is present. Since it is not physically possible for particles to lie atop empty spaces, any gap (0) with a particle (1) above it was converted to (1) to indicate the presence of a particle. For each [1] in the original matrix, every space in the column beneath that particle also had to have a [1] in it, thus filling any empty gaps. Furthermore, each particle must be supported by two below, one directly beneath and one below in the column to the right. Figure 12 shows an original 4 row by 13 column matrix where each red [0] in the original matrix indicates a gap replaced with a particle (or red [1]) in the filled-in matrix. The filled-in matrix in Figure 12 corresponds to the particle arrangement shown in Figure 13. The zeroes in the upper right of the matrix are generated by adding empty columns, and then adding particles to form the base of particles that require it. This does not happen for every random arrangement.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Figure 12 – The original 4 row by 12 column matrix (top) and the resulting filled-in matrix (bottom). Note that row 1 is the lowest row in each matrix.

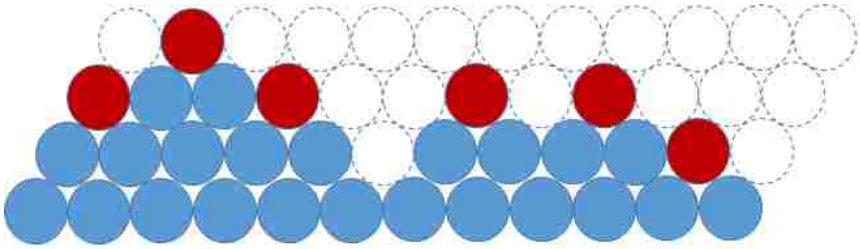
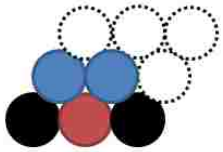
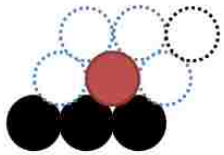
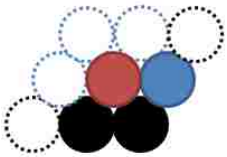
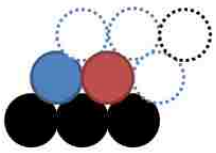
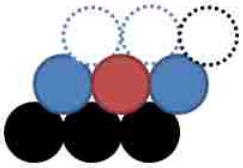


Figure 13 – Particle arrangement corresponding to filled-in matrix of Figure 12, where only red particles can move. The empty circles are the [0] values in Figure 12.

In the resulting particle arrangement, each particle was categorized into the appropriate situation. Values within the matrix were set to represent one of six possibilities: empty space, a particle that cannot move, or particles in Situations One through Four. Situations Two and Four are kept as distinct because when particles resuspend new situations will arise. As shown in Figure 13, six particles in the arrangement can move, indicated by the color red.

Here {0} is the value for an empty space. Each [0] remains as {0}. No particle may fill an empty space because particles that move about their pivot point are assumed to be entrained in the flow as they detach. The next step is more complex because the first nonzero value in each column may be a particle that cannot move. In Table 2, all six possible layouts for a specific particle (highlighted in red) are illustrated. The spaces around it that must be checked are shown in blue. Black particles provide a base but do not require a check. Situations One through Four are represented in the matrix with the numbers {1} through {4}. Situation Zero (a particle that cannot move) is represented by an {8} since {0} represents an empty space.

Table 2 – Table of particle checks for each possible arrangement. The red particle is being checked, blue particles must be checked, black particles cannot move; dash line indicates no particle present.

Matrix	Diagram	Explanations	Situation Representation
$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$		With a particle immediately above, the Red Particle cannot move.	{8}
$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$		The Red Particle cannot move because the space above and to the left is occupied.	{8}
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$		The Red Particle is in Situation One because it can move and has no particles to the immediate left or right.	{1}
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$		The Red Particle is in Situation Two because it can move and has a particle to the immediate right.	{2}
$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$		The Red Particle is in Situation Three because it can move and has a particle to the immediate left.	{3}
$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$		The Red Particle is in Situation Four because it can move and has a particles to the immediate left and immediate right.	{4}

The most efficient way to determine the situation of a particle is first to check if it can move. This is done by checking whether both the space above it and the space above it and to the left are empty. If it can move, it is categorized in one of Situations One through

Four by checking the particles to the left and to the right. Figure 14 shows the completed matrix with each particle categorized corresponding to the arrangement in Figure 13.

$$Matrix \rightarrow \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 8 & 8 & 3 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 3 & 0 & 0 \\ 0 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 0 \\ 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \end{bmatrix}$$

Figure 14 – The completed matrix describing the particle arrangement of Figure 13, with elements corresponding to the situation for each particle.

There are a few other values that must be added to the grid generation code. To simplify coding, an empty top row must be added, so that the code will not result in an error when checking for empty spaces above the top row of particles. An additional empty column to the left and to the right must also be added for the same reason. Finally, a full row of particles is allocated to the bottom, making it so each particle generated has a stable base beneath it.

Chapter 5 – The Boundary Layer Model

A boundary layer model is needed because the particles are within the boundary layer. A velocity profile is needed to compute the lift and drag forces on the particles. This section describes the setup for the problem in STAR-CCM+ and the boundary layer model that was produced. In this model, the particles will resuspend due to the piston velocity, which is the velocity of air following the shock. Initially, there will be no flow until the shock passes.

5.1 – Piston Flow Behind Moving Shock

The particle resuspension model will analyze two cases, with flows of air at Mach 1.2 and Mach 1.7. Since these cases are moving shocks, the pertinent values of pressure, temperature, density and velocity can be calculated from normal shock relations in a moving reference frame and fluid properties typical for Albuquerque NM. The speed of the shock V_s is calculated as the product of speed of sound

$$a = \sqrt{\gamma g_c R T} \tag{15}$$

where R = 287.1 J/(kg*K), γ = 1.4, and g_c = 1 kg*m/(N*sec^2), and Mach Number

$$V_s = aMa_1 \tag{16}$$

Table 3 – Shock conditions

Ma_1	**Pressure (kPa)**	**Temperature (K)**	**Density (kg/m^3)**	**Initial Velocity, V_1 (m/s)**	**Speed of the Shock, V_s (m/s)**
1.2	86.18	289	0.979	0	409
1.7	86.18	289	0.979	0	579

In a reference frame moving with the shock at speed V_S, the flow behind the shock is denoted by 2' with properties computed from normal shock relations, and is the air velocity relative to the moving shock.

Table 4 – Flow properties behind the shock

Ma_1	Ma_2'	Pressure (kPa)	Temperature (K)	Speed of sound, a (m/s)	Velocity of the air, V_2' (m/s)
1.2	0.8422	130	325.9	361.8	304.8
1.7	0.6405	276	421.2	411.5	236.3

Piston flow values (denoted by 2) are those properties behind the moving shock in a stationary reference frame including the Reynolds number

$$Re_2 = \frac{\rho V_2 L}{\mu g_c} \tag{17}$$

Table 5 – Piston flow conditions

Ma_1	Pressure (kPa)	Temp (K)	Density (kg/m^3)	Velocity, V_2 (m/s)	Viscosity, μ ($N*s/m^2$)	Reynolds Number
1.2	130	325.9	1.32	104	2.04E-05	6.84E+05
1.7	276	421.2	2.15	316	2.47E-05	2.80E+06

The high Reynolds numbers indicate that both flows will be turbulent. Once these values are calculated, an estimated thickness of the boundary layer can be calculated using (White, 2006)

$$h = \frac{0.37 * (Length)}{Re_L^{1/5}} \tag{18}$$

This gives a result for roughly how high the prism layer mesh in STAR-CCM+ must be.

Table 6 – Boundary layer parameters

Ma_1	Length (m)	Re_L	Boundary Layer Thickness (m)
1.2	0.61	4.11E+06	0.0108
1.7	0.61	1.68E+07	0.0081

A numerical simulation of piston flow in the shock tube was carried out using STAR-CCM+ before further development of the model could be carried out. The numerical simulation was a test section with dimensions 0.10x0.10x0.61 meters, with the center at 0.305 meters. However, because the model being developed is two-dimensional, the simulation domain only has to be 0.10x0.305 meters. The STAR-CCM+ simulation is symmetric around the vertical midplane of the shock tube, which means that the simulation domain can be further reduced to 0.025x0.305 meters if the top plane is a symmetry line.

The STAR-CCM+ simulation can be reduced further because the velocity profile is approximately constant for distances from the wall between 0.025 to 0.05 meters. The part that is of most interest is closest to the wall, where the velocity profile changes the most and where the velocity profile will be used by the particle resuspension model. By reducing the size of the STAR-CCM+ simulation, the mesh will be more refined close to the wall. Therefore, the simulation was 0.025 meters high and 0.305 meters long.

5.2 – The Prism Layer

STAR-CCM+ creates the computational mesh by using prism layers. Because the STAR-CCM+ model of the shock tube segment is rectangular, the prism cells will also be

rectangular for this simulation. However the prism layers allows STAR-CCM+ to create a stretching mesh, with smaller cells closer to the wall and larger cells towards the mean flow. The mesh requires a set of input values, as follows:

Table 7 – Prism layer terms

base	The value that all later calculations will be based upon
maximum %	The maximum amount of the base that the stretching prism layer will occupy
# of layers	The total number of layers within the stretching prism
stretching factor	The ratio of each prism layer thickness to the one below it (a value greater than one)
% of prism layer	The true percentage amount of the base that the stretching prism layer occupies
depth	The height of the stretching prism layer

It is, in fact, impossible to specify all of the values listed in Table 7. The Maximum % and the true percentage cannot be simultaneously specified. The true percentage, the number of layers and the stretching factor have to be chosen so that they are consistent with each other; two values must be selected and used to calculate the third in order to specify the prism layers.

The maximum % is the maximum amount of the Base that the prism layer is allowed to occupy. This does not mean that it will take up this entire space, as will be seen in the following calculations. A good value is 10%, because by that point the flow velocity will have increased significantly and is no longer requires as much resolution for future

calculations. The mesh does not need to be as refined from 10% to 100%, though there does need to be a well-defined mesh to acquire good results.

The number of layers is directly correlated to the number of data points. Each cell will provide data, so it is better to have more layers of cells. There is a trade-off between having enough layers to model the details of the flow and having superfluous layers that do not contribute to the solution. The number of layers outside the prism layer needs to be the same as the number of layers within the prism layer.

The stretching factor is defined as: Stretching factor = (height of second layer – height of first layer)/height of first layer as shown in Figure 15. The example is for a stretching factor of 1.5.

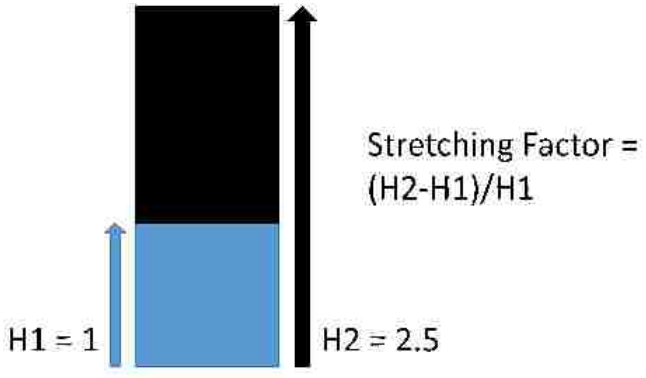


Figure 15 – Stretching factor example

This does not give a quick and effective way to determine the most useful stretching factor based solely on the top of the layers. This is especially useless when y+ is being used for the calculations. y+ is the dimensionless wall distance,

$$y^+ = \frac{u^* y}{v} \tag{19}$$

$u^* = \sqrt{\tau_{wall/\rho}}$ is the friction velocity, y is the distance to the wall and $\upsilon = \mu/\rho$ is the kinematic viscosity of the fluid. Where μ is dynamic viscosity, ρ is density and τ_w is wall sheer stress.

A calculator within STAR-CCM+ can be used for many of these values, based upon the flow. So the best way to get these values is to run a simple simulation with a uniform mesh. This will provide approximate values for the friction velocity u_*, the density ρ and the dynamic viscosity μ. The values for the piston flow following a Mach 1.2 shock are tabulated below:

Table 8 – Variables for y+

u*	3.96	m/s
ρ	1.185	Kg/m^3
μ	1.85E-05	$N*s/m^2$

An initial layer height can be calculated from these three values by stating y+ = 1. Thus the height of the first layer can be calculated using

$$h = \frac{\mu}{u^*\rho} = 1.85x10^{-5}\frac{N*s}{m^2} * \frac{1}{3.96}\frac{s}{m} * \frac{1}{1.185} * \frac{m^3}{kg} = 3.937x10^{-6}m \quad (20)$$

A stretching factor of 1.15 was chosen and the next layer height can be determined by simply multiplying the height of the first layer ($3.937x10^{-6}$ meters) by 1.15, giving a value of $4.521x10^{-6}$ meters. This is not the top of the second layer, but the thickness of it. From this, the value of y+ for this cell can be calculated using eqn (19). Subsequent values can then be calculated using the same method, resulting in Table 9.

Table 9 – Calculated y+ values for STAR-CCM+

Layer #	height (m)	y+
1	3.94E-06	1.000
2	4.52E-06	2.150
3	5.23E-06	3.473
4	5.99E-06	4.993
5	6.91E-06	6.742
6	7.95E-06	8.754
…	…	…
32	4.59E-04	577.100

Adding the heights together, a value for the total prism depth is calculated. This is used to determine if the total calculated height is greater than the maximum percentage that the prism layer is allowed to occupy, as specified in Table 7. This also gives the total number of prism layers should allow for the value of y+ to range from 1 to about 500. The completed table of values needed by STAR-CCM+ is listed in Table 10.

Table 10 – Completed table for STAR-CCM mesh generation

base (m)	0.0254
maximum %	10
# of layers	32
stretching factor	1.15
% of prism layer	6.759
depth (in)	0.00171

This means that there are 32 prism layers within the prism layer, the stretching factor is 1.15 and the depth of the prism layer is about 0.00171 meters, or 6%, which is less than the maximum of 10%. It should be noted that the value calculated here is approximately the same size as the value calculated initially for the size of the boundary layer, and most of the boundary layer is encapsulated within the prism mesh.

5.3 – STAR-CCM+ Results

With the above values entered into STAR-CCM+, a mesh can be generated to compute the velocity profile. Using STAR-CCM+, a velocity inlet/pressure outlet simulation was used for the turbulent velocity profile using a k-ω turbulence model, and was then run for 1000 iterations until the residuals became steady, indicating convergence of the output results.

5.3.1 – Velocity Profile

Figure 16 shows the velocity profile for air traveling at 104 m/s, the calculated piston velocity for the Mach 1.2 case. With a 0.0254 meter domain height and symmetry plane at the top surface, mass flow conservation requires that the velocity in the inviscid region increases to 109 m/s. Position is distance from the wall.

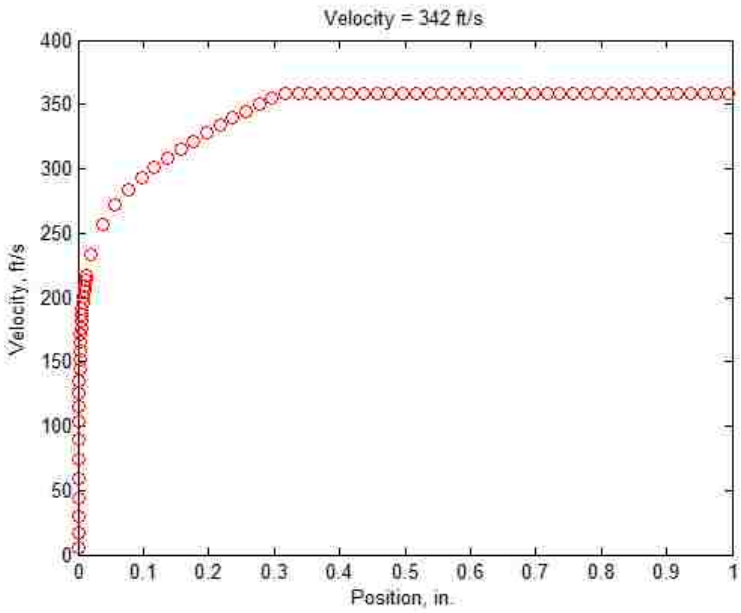


Figure 16 – Velocity profile from STAR-CCM+for Mach 1.2 case.

5.3.2 – Friction Velocity

The velocity profile in wall variables, where $u^+ = u/u^*$ is shown in Figure 17, where position is distance from the wall. The value of the friction velocity can be extracted and used to determine the suitability of the wall-normal mesh. The velocity profile satisfies the $u^+ = y^+$ behavior for $y^+ <\sim 5$ and demonstrates logarithmic behavior for $20 < y^+ < 1000$.

5.3.3 – Turbulent Kinetic Energy

The STAR-CCM+ results for turbulent kinetic energy shown in Figure 18 are suspect because the peak kinetic energy is expected to occur at y+ ~ 10 rather than y+>100 (Bernard, 2002). There was no response to inquiries made to STAR-CD, the distributor of STAR-CCM+, about this discrepancy. The computed velocity profile is strongly dependent on the tubulent kinetic energy profile, so it is believed that something within

the code for outputting kinetic energy is in error. Because the computed velocity profile appears to correct, it will be used in the particle resuspension modeling using MATLAB.

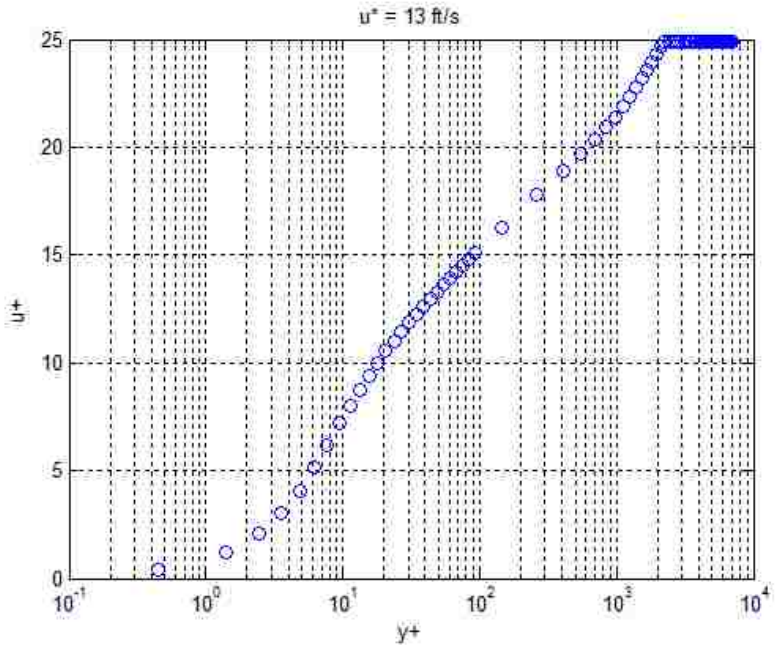


Figure 17 – Velocity profile for Mach 1.2 case from STAR-CCM+ using wall variables

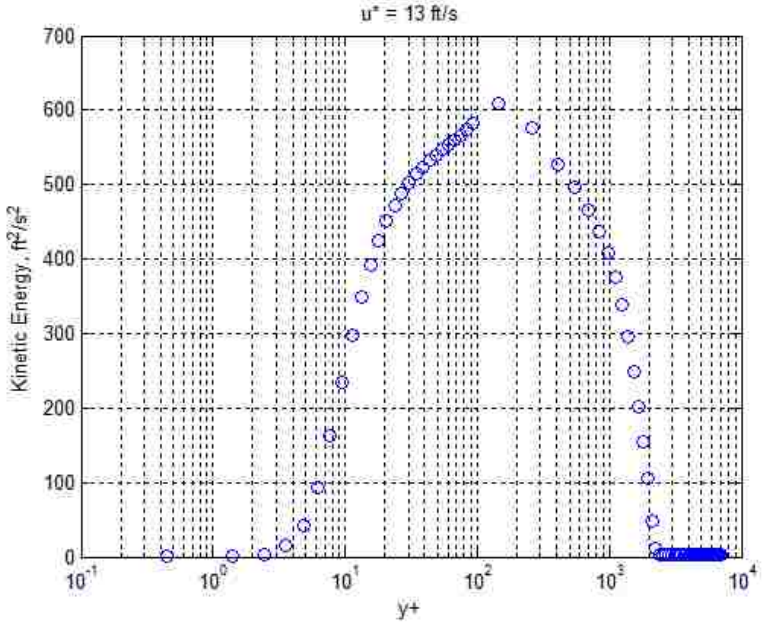


Figure 18 – Turbulent Kinetic Energy profile for Mach 1.2 case from STAR-CCM+

Chapter 6 – Visualization

A MATLAB function was created to display the particle grid, with input from the above grid generation. The code in its entirety is listed in Appendix A.3, called partmap. First, a color was assigned to each particle situation listed in Table 11.

Table 11 – Color assignments for visual representation

Situation	Matrix #	Color
No Particle	0	None
Situation One	1	Green
Situation Two	2	Red
Situation Three	3	Black
Situation Four	4	Magenta
Situation Zero	8	Blue

A radius is chosen for plotting the particles. Then each particle is assigned a coordinate for plotting. Because the particles are assumed to be in a packed hexagonal array, each row is shifted relative to the row beneath it by the radius of a particle. In the packed hexagonal array, the y-coordinate for a row is the same. The x-coordinate is determined by the matrix index of the particle. The plotting coordinates for each particle are calculated from the indices of the particle.

For example, a particle arrangement where the input is 4 rows and 10 columns could potentially look like Figure 19. Recall that a bottom row is added so there is a stable base, and particles are added to the right to support "hanging particles". Each of the particles

is represented with a color that represents what type of situation it is in. This will allow for a quick visual inspection of the results.

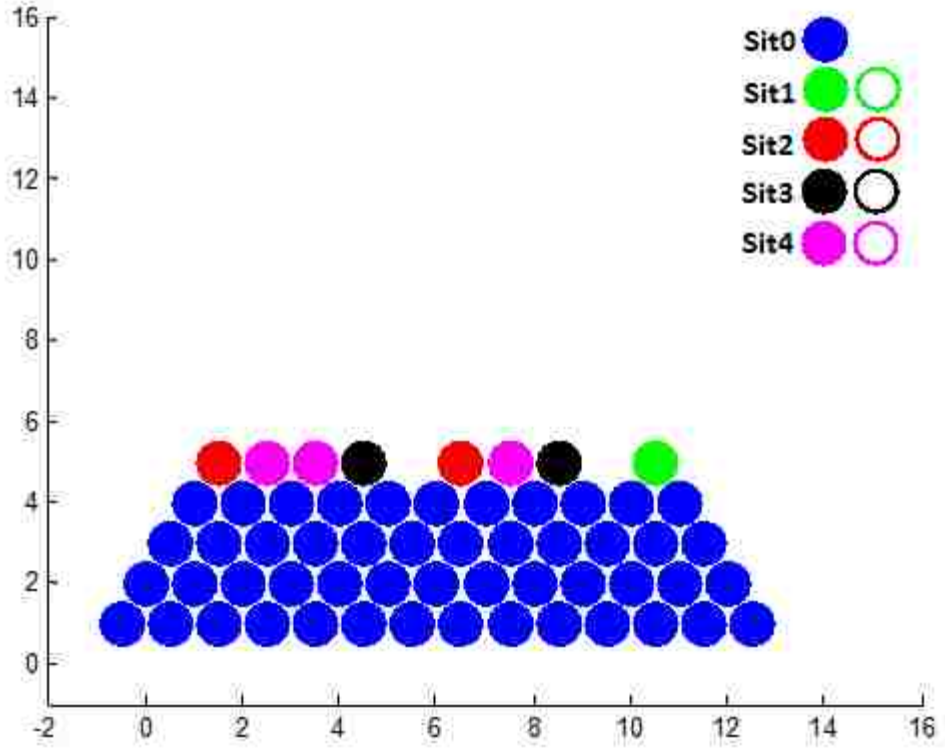


Figure 19 – Particle bed visualization, where color indicates Situation of each particle.

Chapter 7 – Test Code: Uniform Attractive Force and Velocity

The next step is building the code that will determine which particles resuspend. The first version of the code had simplifying assumptions so the code could be created in an incremental manner. The code in its entirety is listed in Appendix A.4, called PartSimSimple.

A basic grid of ten rows and fifty columns was used. The velocity profile was uniform, with no boundary layer profile. The attractive force is assigned as uniform throughout the grid. A force of $1*10^{-9}$N was chosen initially, as it was small enough to allow for debugging purposes. A time step t = 10^{-14} seconds is chosen for these calculations in order to solve the dynamic equations.

Table 12 – Equations for forces in Y for each Situation, basic simulation

Situation	Equation
One	$F_{Y1} = -F_a \cos\left(\frac{\pi}{6}\right) + F_L - mg$
Two	$F_{Y2} = -2F_a \cos\left(\frac{\pi}{6}\right) + F_L - mg$
Three	$F_{Y3} = -2F_a \cos\left(\frac{\pi}{6}\right) + F_L - mg$
Four	$F_{Y4} = -2F_a \cos\left(\frac{\pi}{6}\right) + F_L - mg$

Next, a system of equations was solved for each situation. The forces in y are found first. These equations are then used to determine the acceleration in y, as F = ma. With the acceleration, the velocity in y can then be calculated with $V = V_i + at$. Using the

velocity and the acceleration, a position in y can then be calculated, using the equation $y = y_i + Vt + at^2$. These equations require the logging of the position and velocity of the previous time step.

Next a value must be determined to decide whether or not a particle has resuspended. Once the particle breaks free at one attraction point, it has broken free of all attraction points, and has started to move. It is only a matter of time before it resuspends. However, this time does need to be standardized so it can be factored into the calculations. The easiest way is to assign a vertical distance y that the particle must travel before it is considered to have 'resuspended'. The value chosen can be altered to better visualize the results. A distance equal to half the radius of the particles was chosen as a reasonable value for these initial calculations.

After a particle is determined to have resuspended, it is allocated a new number within the grid, thus remembering its situational identity when it has suspended. Note that these resuspended particles are no longer being used within the calculations. This was done by adding (-9) to the situation number when a particle resuspends. The new values are listed in Table 13. The Grid visualization was expanded upon to display the resuspended particles as empty circles.

Table 13 – New Situation allocations after resuspension

Situation	Matrix #	Resuspend Value
No Particle	0	NA
Situation One	1	-8
Situation Two	2	-7
Situation Three	3	-6
Situation Four	4	-5
Situation Zero	8	NA

For example, Figure 20 shows the Particle Arrangement for a [10,40] initial grid. It has nine Situation One particles, five Situation Two particles, four Situation Three particle and three Situation Four particles.

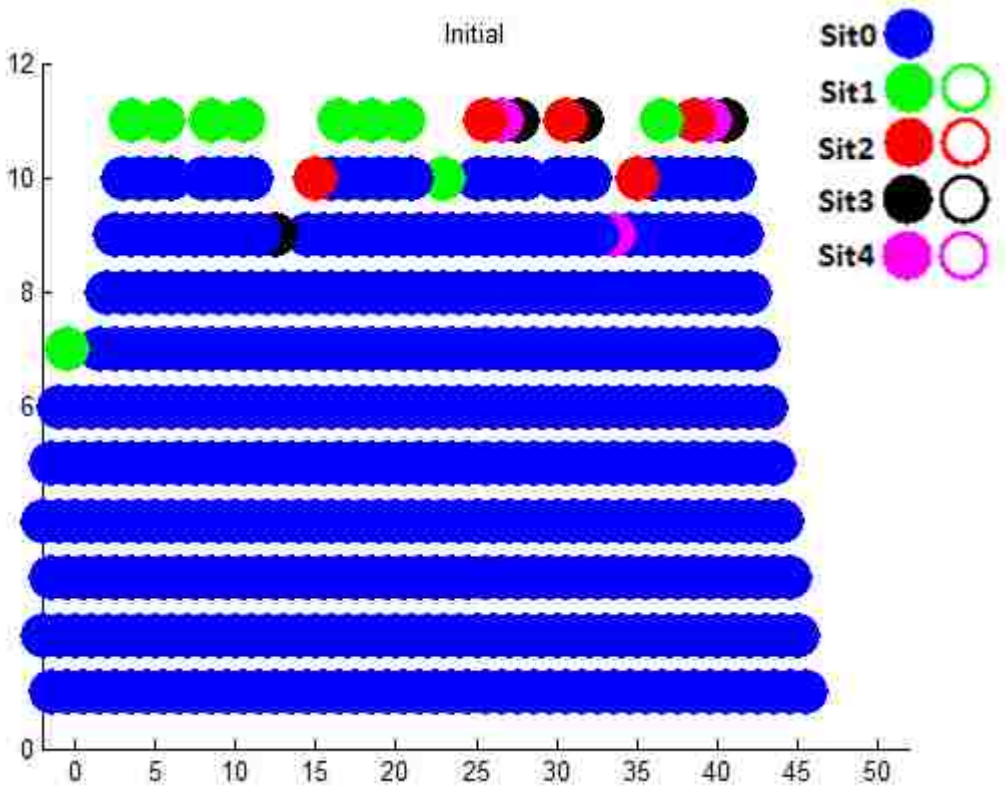


Figure 20 – Initial particle bed, with Situation indicated by color.

When this arrangement is used in the test simulation, after fifteen time steps, the grid looks like Figure 21. Notice the now empty circles whose color indicate the Situation of the particle when it resuspended. These empty circles maintain their location and their color to help determine how particles resuspended. It can be seen in Figure 21 that the particles seemed to lift off in rows, with the Situation Fours all detaching at once. If each time step is depicted it will show the particles resuspending as time passes, also in rows, though with time steps in between where nothing resuspends. There is no sudden

emergence of a Situation One particle, and the particles resuspend in a logical fashion, assuming that all the particles are the same size, weight, are experiencing the same attractive force and subject to the same airflow velocity.

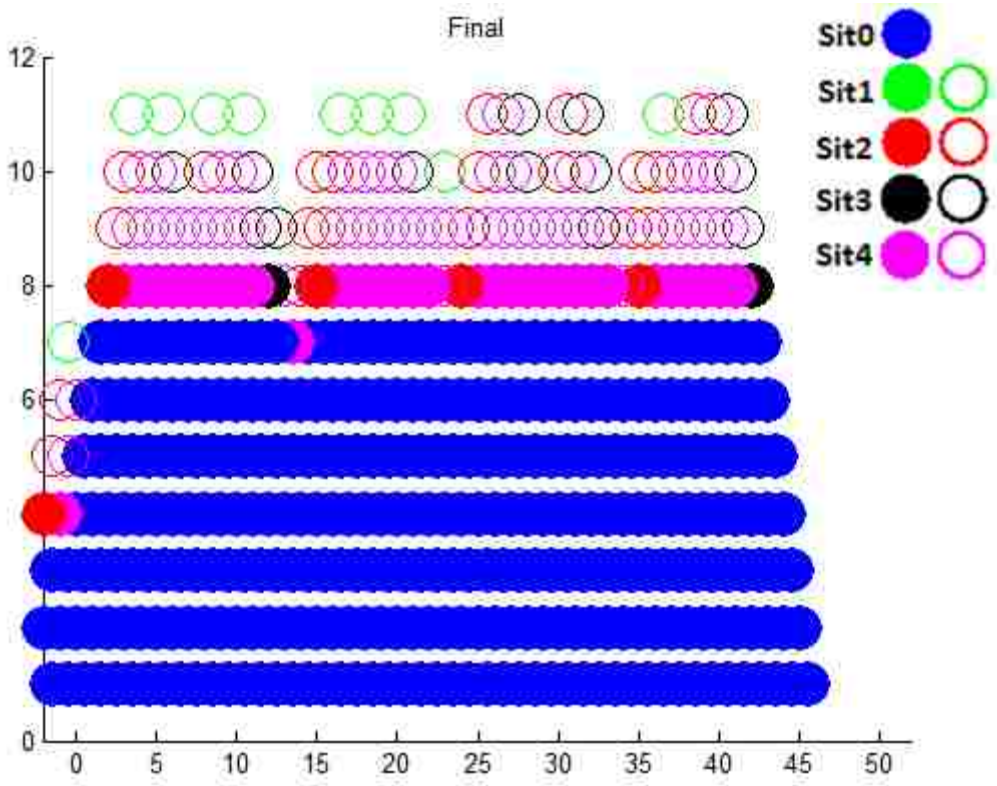


Figure 21 – Particle bed after 15 time steps have elapsed. Empty circles indicate the Situation of the resuspended particle.

Chapter 8 – Simulating the Airflow

The preliminary simulation is expanded upon by including the variable attractive forces, the velocity profile with a boundary layer and the addition of more particles. The attractive force of each particle was assigned a maximum value of $1*10^{-9}$N. Since it is a uniform random distribution, the MATLAB random function was used to assign each particle a value between 0 and 1, and would then later be scaled to the appropriate values, with a bias added so it covers the correct range of attractive forces for a surface. This assigns each particle an attractive force which will now be used in the force balance equations. As can be seen in Table 14, each attractive force is in represented in the forces in the y-direction. The code in its completed form is listed in Appendix A.5, and is called PartSimComplex.

Table 14 – Reduced force balance equations

Situation	Equation
One	$F_{Y1} = -F_a \cos\left(\frac{\pi}{6}\right) + F_L - mg$
Two	$F_{Y2} = -F_{a1} \cos\left(\frac{\pi}{6}\right) - F_{a2} \cos\left(\frac{\pi}{6}\right) + F_L - mg$
Three	$F_{Y2} = -F_{a1} \cos\left(\frac{\pi}{6}\right) - F_{a2} \cos\left(\frac{\pi}{6}\right) + F_L - mg$
Four	$F_{Y2} = -F_{a1} \cos\left(\frac{\pi}{6}\right) - F_{a2} \cos\left(\frac{\pi}{6}\right) + F_L - mg$

8.1 – Turbulent Velocity Profile

The particles are subjected to a turbulent airflow. This airflow is thus not a uniform flow and is more parabolic in nature. This means that the airflow closest to the wall is much

slower than the air away from the wall. This airflow was modeled in a STAR-CCM+, as described in Section 5. A plot of the velocity profile for the piston flow flowing a Mach 1.2 shock is shown in Figure 22.

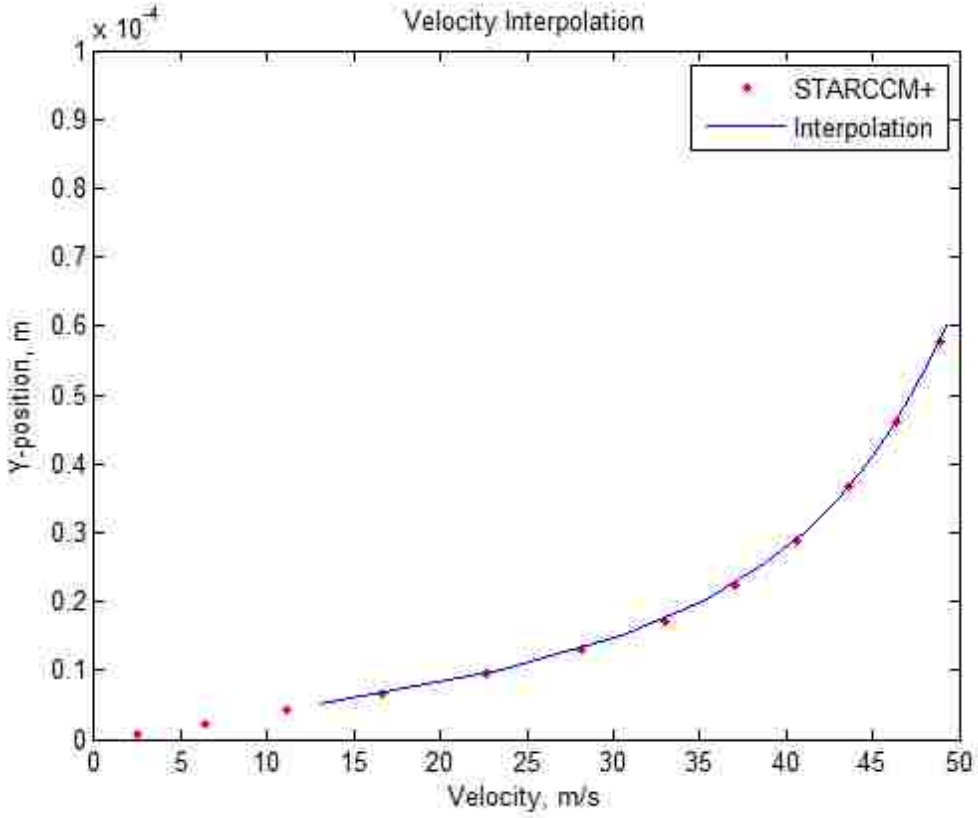


Figure 22 – Velocity interpolation from STAR-CCM+ values

At the wall, the velocity is almost zero, though with greater distance from the wall, the airflow speeds up. This needs to be represented within the simulation itself. The results from STAR-CCM+ are imported and a linear interpolation is used to solve for the velocity at a given coordinate. This is depicted with the red symbols in the figure above. The blue line is the results from STAR-CCM+. This velocity profile based on the y location of the particle is now used. Previously, when subjected to a uniform flow with

uniform attractive forces, several of the particles resuspended, and given enough time, all of them would. This is no longer the case as can be seen in Figure 23.

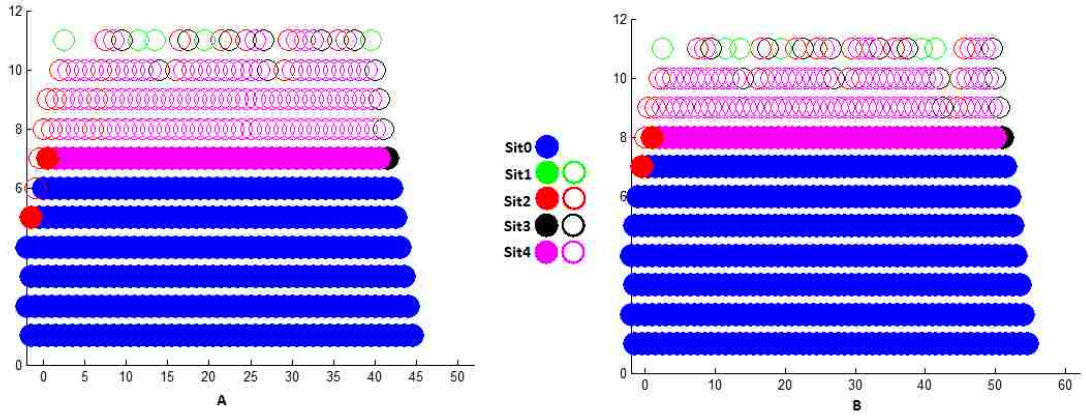


Figure 23 – Particle bed Situations after 15 time steps. A) Exposed to constant velocity, B) Exposed to velocity varying with height. Empty circles indicate the Situation of the resuspended particle. Fewer particles in B resuspended.

Far fewer particles have resuspended over the same amount of time. This is due to the boundary layer. Within the boundary layer, the airflow velocity decreases closer to the wall, to the point where the lift and drag forces are insufficient to overcome the attractive forces. The randomized attractive forces make it possible for particles to stick together in ways that are not apparent until the surrounding particles have resuspended.

8.2 – Simulation Refinement

After the initial simulations, the grid of particles was expanded to a much larger scale. A larger model requires more computing time. The code starts on the outermost layer of particles and then works downward towards the base. Logic was added to the code to determine when an entire row of particles was unable to move. As every subsequent row would also not be able to move, the checks past this point were not needed. The

visualization was changed so it only displayed the rows above the first unmovable row. This made the results easier to interpret and increased the speed of the program immensely for large [100,100] or a [500,500] particle arrangements.

Memory requirements were also reduced by not saving the time history of the net forces, accelerations, velocities and positions of every particle. Instead, only the values from the current and previous time step are saved, as they are the only two needed for the propagation of the motion of a particle.

The time step for the simulation was calculated by first determining the time to resuspend a particle. The fastest a particle would resuspend is a Situation One particle when it is exposed to the maximum flow, with a weak attractive force holding it to the other particles. This resuspension time is then divided by 10, so that the particle dynamics can be calculated and allow the simulation to capture the motion of that particle. Other particles take longer to resuspend as they may have more than one attractive force to overcome, which may also be stronger, and the aerodynamic forces are smaller with a lower velocity. This time step allows for all of the motion to be captured for all situations of particles.

8.3 – Particle Rolling

The next step is to model the particles with moments. While the particles themselves can be resuspended purely by the lift forces, some of the particles can move and rotate before lifting off. Situation One is the best example, as the particle can roll onto the adjacent particle before resuspending.

Further code was added to roll the particles for Situations One and Three (the only situations where there is no particle to the immediate right). For Situations Two and Four, it is impossible for them to roll, and they can only resuspend by breaking the attractive bonds between them and the surrounding particles through lift. The code was changed to check for resuspension due to either rolling due to a moment or a strong enough lift force.

8.4 – Shock Passage

This simulation was further expanded upon to model the passage of a shock over the particles. While shock itself is not modeled within the simulation, the passage of the shock can be, as the particles behave differently before and after the shock. In this simulation, the boundary between pre and post shock situations is approximated as being infinitesimally small. This means that the particles are either exposed to zero velocity flow, or exposed to the piston velocity flow at any given time.

The length of time it would take for the shock to pass over the entirety of the particles can be calculated from the Mach number of the shock, and the length of the particle arrangement, calculated by the number of particles in the bottom row and their diameters. This gives a value for the time the simulation must run to model the passage of the shock. The vast majority of particles are resuspended as the shock passes over the particles. The simulation can be run longer to capture the few additional particles that will resuspend with more exposure to piston flow.

The next step is to model the passage of the shock. As the speed of the shock and the time step and current time of the simulation are known, the distance the shock has traveled is also known. So the next step is to have the simulation check, particle by particle, if the

shock has passed it before doing the calculations for modeling resuspension. If the shock has passed, the particle is exposed to the piston velocity flow. If it has not, the particle is exposed to no flow and thus will not be considered for resuspension. This will generate results that are closer to the real world dynamics as the particles are exposed to flow for a very short interval of time and either exposed to the piston flow or no flow at all.

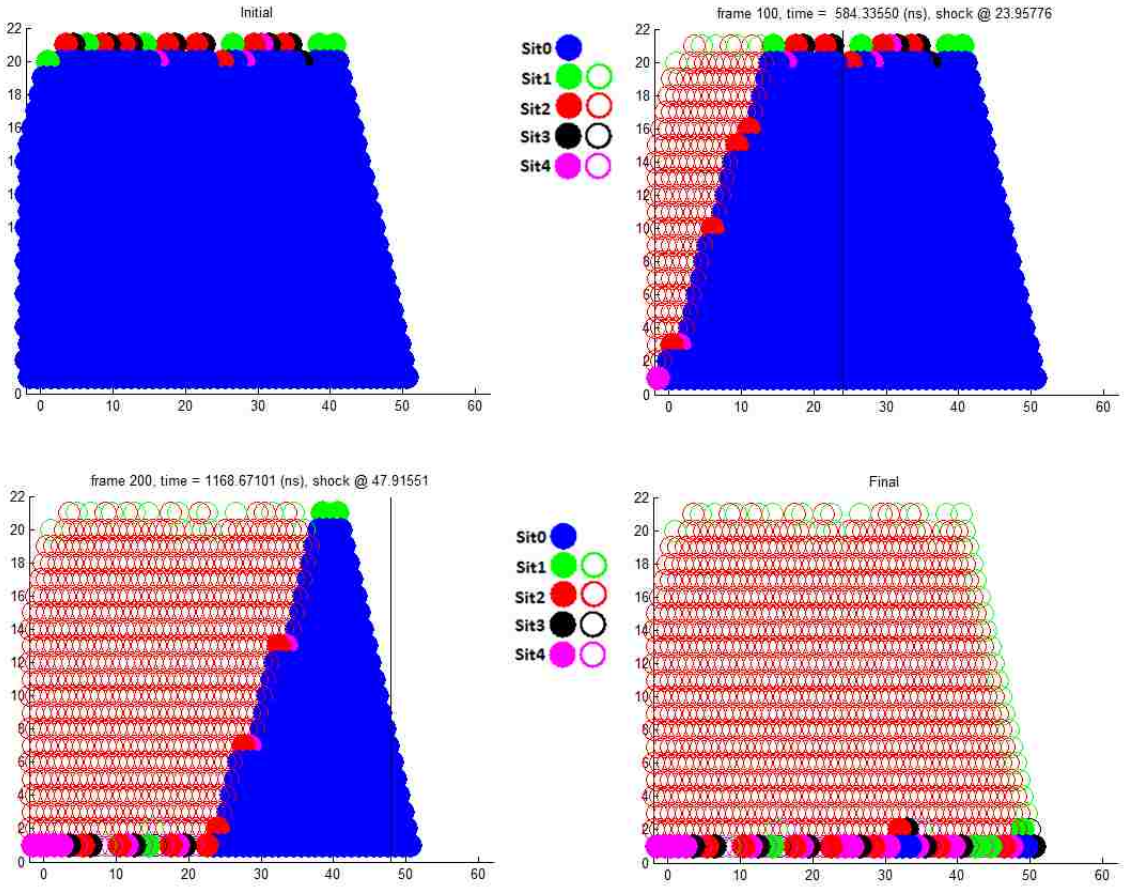


Figure 24 – Particle bed at four time intervals as the shock passes. The black line indicates the location of the shock passing over the particles. Empty circles indicate the Situation of the resuspended particle. To the left of the shock, particles are exposed to the piston velocity, whereas to the right of the shock, they experience no flow.

Chapter 9 – A Usable Code

As the air flow has been fully modeled, a new code had to be crafted that could be used to look at the results, both individually, and also for trends. This required the air flow model to be written as a subroutine for a larger code that would use PartSimComplex and run it in a variety of ways to produce different results, depending on the user's needs.

9.1 – Focusing the Results

The first step was to determine what results were appropriate to examine. The most readily apparent of these results was the number of particles that resuspended. The code previously counted how many particles of each type resuspended, and how many particles overall resuspended. Now it was time to take those results and export them. Code was added to PartSimComplex to export this data, as well as the time it took the simulation to run and the initial and final particle counts.

With this, some basic statistical analysis could be performed, notably determining the percentage of the particles that resuspended. However, one run was not enough to recognize trending, so multiple runs were looked at, which was the basis for PartBatchRun, which may be found in its completed form in Appendix A.6.

PartBatchRun allows for the simulation to be run multiple times, each time logging the data results. In order to determine an appropriate number of runs, some statistics had to be calculated from the results of these runs: average, minimum, maximum, range and standard deviation. It was determined by looking at a few samples of data, that while 25

runs was sufficient to determine trending behavior, there was not enough data for some of the smaller size particle arrangements.

For testing the statistical analysis, the attractive force range was shifted so as to increase particle attraction. The attractive force is now between 1-2 nN. This will allow more trends to become visible. The results of the 12x52 particle arrangement over various numbers of runs are displayed in Table 15.

Table 15 – Results from PartSimComplex over various runs, using a 12x52 particle arrangement grid

	Initial particles	Final particles	Resuspended particles	%Sit1	%Sit2	%Sit3	%Sit4	%Total
12x52 25 Runs								
mean	465.6	120.7	344.9	6.503	88.81	0.1513	4.534	74.07
min	447	113	327	4.735	87.16	0	2.941	72.72
max	479	128	359	7.951	91.45	0.5882	5.556	75.06
range	32	15	32	3.216	4.297	0.5882	2.614	3.332
stdev	8.4552	4.326	8.086	0.7737	1.042	0.1709	.7269	0.8846
12x52 50 Runs								
mean	458.9	120.6	338.3	6.381	89.25	0.1813	4.184	73.71
min	427	108	306	4.902	86.57	0	2.194	71.56
max	479	130	359	7.837	91.50	0.8721	6.000	76.32
range	52	22	53	2.935	4.932	0.8721	3.806	4.756
stdev	10.56	5.272	10.87	0.741	0.8819	0.2031	0.6677	1.206
12x52 100 Runs								
mean	461.4	120.0	341.4	6.367	89.15	0.1661	4.318	73.99
min	423	105	307	5.099	85.63	0	2.874	71.02
max	479	133	360	8.750	90.99	0.6173	5.625	77.07
range	56	28	53	3.651	5.363	0.6173	2.751	6.050
stdev	10.45	5.587	9.493	0.6862	1.011	0.2204	0.6593	1.106

While the results vary with the number of runs included in the statistical analysis, the changes from 50 to 100 runs are relatively small, a minimum run count of 100 was chosen for initial trending analysis.

9.2 – The Time Scale, τ

The next step was to determine the appropriate time scale. Initially, the particles within the simulation were exposed to the flow for the amount of time it took the shock to pass over the particles. However, this will not capture particle detachment in the subsequent piston flow behind the shock. The length of time it takes for the flow at the piston velocity to pass over the particle bed is selected as a time scale τ = (bed length)/V_p. For Mach 1.2, the piston velocity (342 ft/s) is roughly $1/4^{th}$ the velocity of the shock (1342 ft/s). Thus the shock passes over the particle bed in 0.25τ and 0.55τ, respectively, for M = 1.2 and 1.7. In order to make sure that all motion was captured, run times equal to 5, 10 and 15τ were used to determine a minimum run time. This was done by running the simulation using a variety of particle arrangements and plotting various results. In Figure 25, the 12x52 grid arrangement is used. About 70% of all particles resuspended, with 87% of them being Situation Twos, followed by small percentages of Situation Ones, Fours and very few Threes. However each of these results is consistent across all three time scales, with negligible variance between them. This was repeated with the 22x102 and 32x62 grid arrangements with similar results that are shown in Figure 26 and Figure 27.

For all of the simulations, Situation Twos are the overwhelming majority of resuspended particles, followed by Situation Ones, Fours and finally Threes, which are a rare event.

The larger the grid, the greater percentage of particles resuspend, though adding more rows increases the percentage resuspended compared to adding more columns. This is because all particles above a certain height (where the velocity of the flow is fast enough) will resuspend. These results indicated that 5τ was a more than adequate run time.

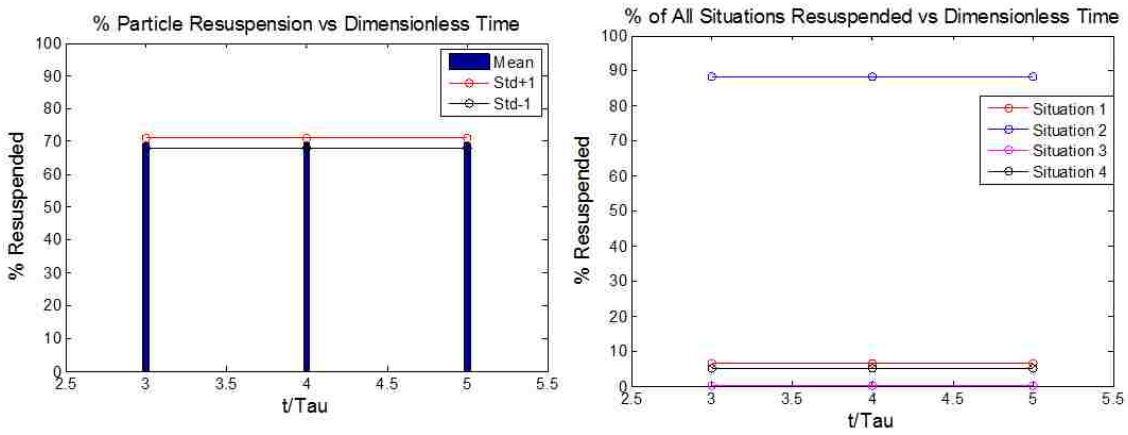


Figure 25 – Average percent resuspension vs dimensionless time for 12x52 grid size.

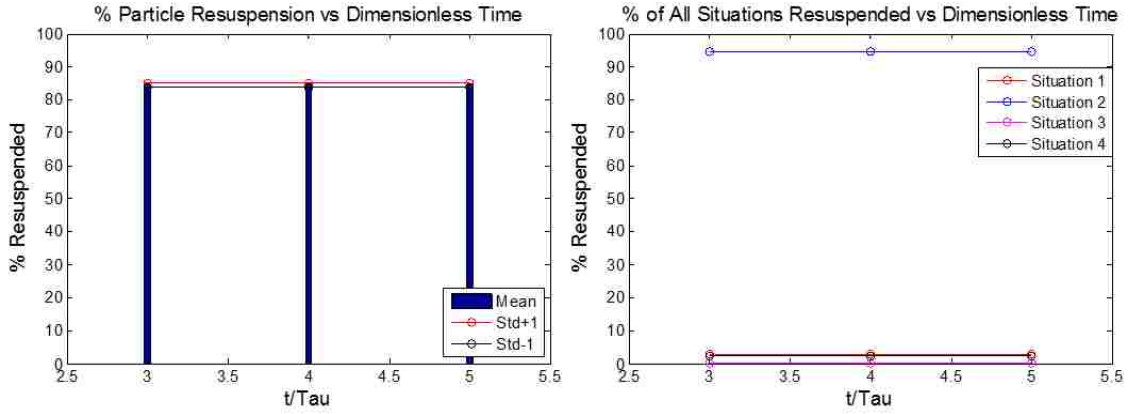


Figure 26 – Average percent resuspension vs dimensionless time for 22x102 grid size.

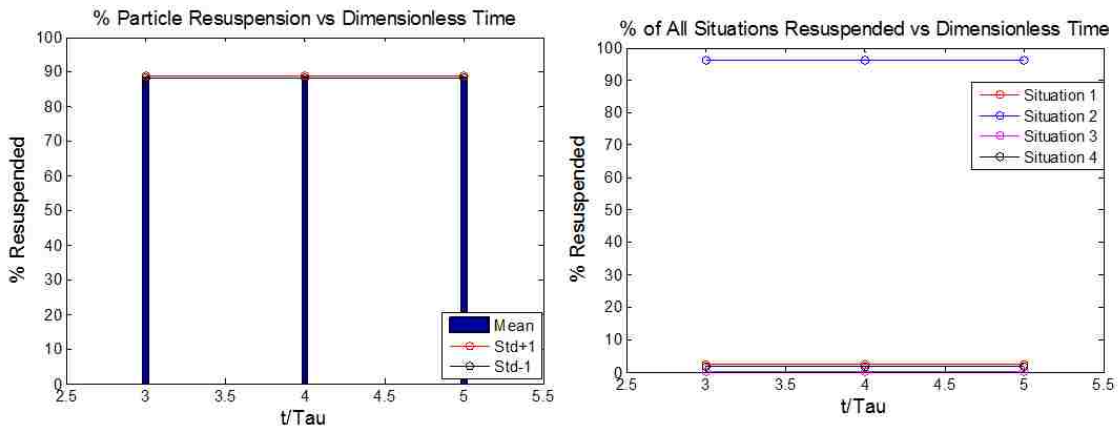


Figure 27 – Average percent resuspension vs dimensionless time for 32x92 grid size.

9.3 – Capturing Particle Resuspension

Particle Resuspension happens very early in exposure to flow, as was evidenced by the amount of particle resuspension that happens when the run time was equal to the length of time it took for the shock to cross the particles. A significant number of the particles resuspend as the shock passes over the particles, results for $t/\tau < 1$ were of interest. The shock passes over the particle bed at $t/\tau \sim 0.25$ for Mach 1.2 and Mach 1.7, so the run times of 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 1, 2, 3, 4, and 5 were selected to fully capture the motion. This would produce a better overall picture of particle resuspension with time. At this point, a better, more diverse attractive force array was implemented, where the attractive forces ranged from 5×10^{-9}N to 15×10^{-9}N, as opposed to the previous 0 to 10×10^{-9}N. The smaller attractive force was beneficial for exaggerating the dynamics as the particles more readily resuspend.

Collecting results at intermediate times required the addition of trackers within PartSimComplex, as well as a few on/off switches, as it became apparent that the code would have to perform two functions: 1) visual depictions of a characteristic flow and 2)

multiple runs of random arrangements. The newest version of the code would distinguish between the two, and present the results accordingly. PartBatchRun would run the code PartSimComplex as many times as required by the user, log the data, calculate the mean, maximum, minimum, range, standard deviation, as well as plus and minus one standard deviation, and plot the results for visual comparison. It can also generate a movie out of a particular arrangement, showing how particles resuspend.

PartBatchRun was created as a MATLAB function, which means a further code was wrapped around it, called ParticleResuspension which is listed in Appendix A.7. This code takes user inputs and determines which of the two functions the user wants: visuals or data collection. It also presents the option, with the visuals, to make an .avi file movie of the particle resuspension. The inputs for this code are listed in Table 16.

Table 16 – User inputs for ParticleResuspension

Input	**Description**
Columns	The initial number of columns.
Rows	The initial number of rows.
t/τ	The amount of time the particles are exposed to the flow, where the time the piston velocity takes to cross the flow is equal to 1.
Attractive Force Choice	Allows the user to input which attractive force of the four used within this text to use: NH2, COOH, CH3 or Test.
Plotting Run or Data Run	This allows the user to input if this is a plotting run or a data run. A plotting run generates images for one run. A data run produces statistical data for series of runs
Movie Run Switch	This allows the user to decide if they wish to make an .avi file movie of a single run.
Run Count	For a data run, the number of times the simulation will run.

Chapter 10 – Results

The final product allows for multiple runs with the same inputs. Since the attractive forces and grid arrangements are varied randomly, each run will provide different results. Using this tool, particle resuspension can now be examined in terms of varying attractive forces based on surface chemistry and varying grid size.

10.1 – 12x52 Grid, Mach 1.2

This is the smallest grid, both in height and length and thus has the fewest particles exposed to the piston flow following a Mach 1.2 shock. As shown in Figure 28, nearly all the particles using the CH3 attraction forces resuspend, while less than 1% of the particles using the COOH attraction forces resuspend. None of the particles using the NH2 attraction forces resuspend. The resuspended particles are mostly Situation Twos, where the few particles that do resuspend using the COOH attraction forces are Situation Twos. The complete results for CH3 are listed in Appendix B.1.

10.2 – 22x102 Grid, Mach 1.2

This is the longest grid, and thus has the most particles in the direction of the flow. As shown in Figure 29, nearly all the particles using the CH3 attraction forces resuspend, while less than 1% of the particles using the COOH attraction forces resuspend. Again, none of the particles using the NH2 attraction forces resuspend. The resuspended particles are mostly composed of situation twos, and in the COOH resuspension, there are more Situation Four particles resuspending than Situation Ones, and a comparable

number of Situation One and Three particles resuspending. The complete results for CH3 are listed in Appendix B.2. The complete results for COOH are listed in Appendix B.3.

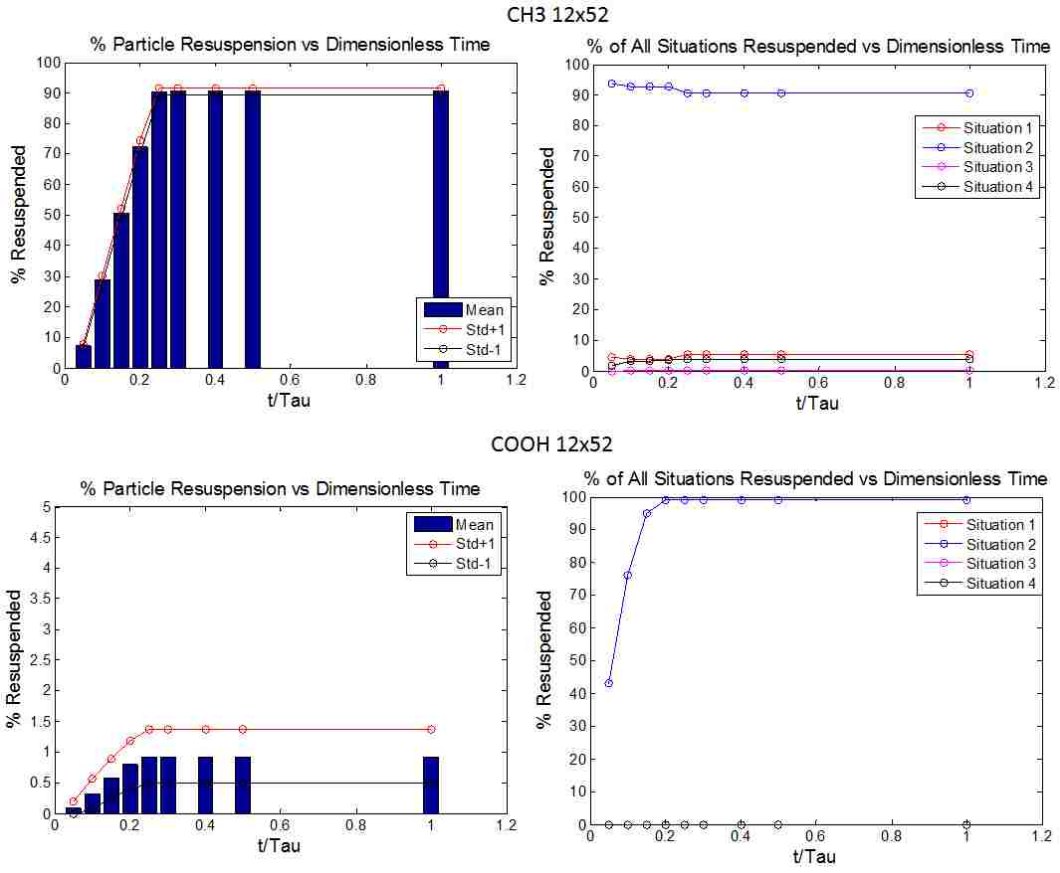


Figure 28 – Particle resuspension for 15x52 grid size with varying attractive forces vs dimensionless time.

10.3 – 32x92 Grid, Mach 1.2

This is the largest grid, and thus has the most particles resuspending overall. As shown in Figure 30. Nearly 100% of the particles using the CH3 attraction forces resuspend, while 2% of the particles using the COOH attraction forces resuspend. None of the particles using the NH2 attraction forces resuspend. As with the previous results, the resuspension is dominated by Situation Twos. However, uniquely to the COOH attraction forces, there

are more Situation Fours resuspending than Situation Ones. The complete results for CH3 are listed in Appendix B.4. The complete results for COOH are listed in Appendix B.5.

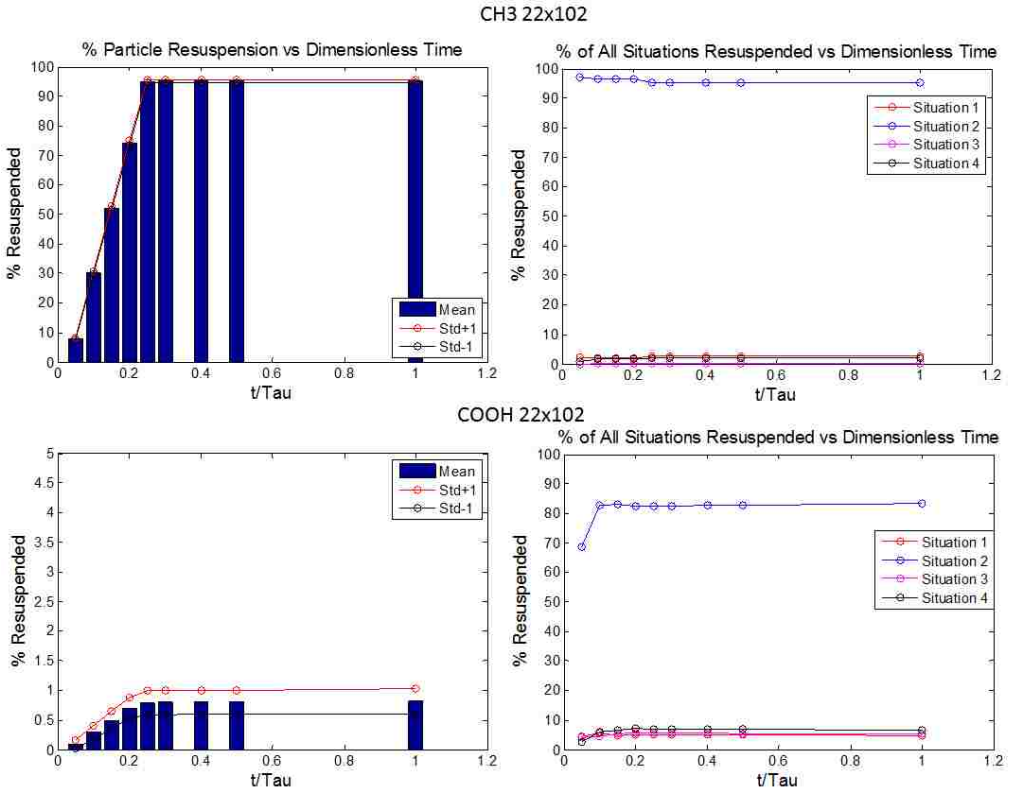


Figure 29 – Particle resuspension for 22x102 grid size with varying attractive forces vs dimensionless time.

10.4 – Trends

There are trends apparent across all three grid arrangements and attractive forces, most notably that the Situation Twos comprise a majority of the resuspended particles in each, followed by the remaining three situations. Another trend would be that the Situation Twos decrease percentage wise over time, as they are popular for the top rows, which are exposed to higher flows and thus more easily overcome the attractive forces, allowing particles to resuspend readily. However as time progresses, the easily resuspended

particles lift off, leaving strongly attracted particles, which make way for the Situation Ones, Threes and Fours. Situation Ones are also very common at the start of resuspension, then the trail off before rising again. This is due to the top layer of particles being composed of many Situation Ones, where the particles most readily resuspend.

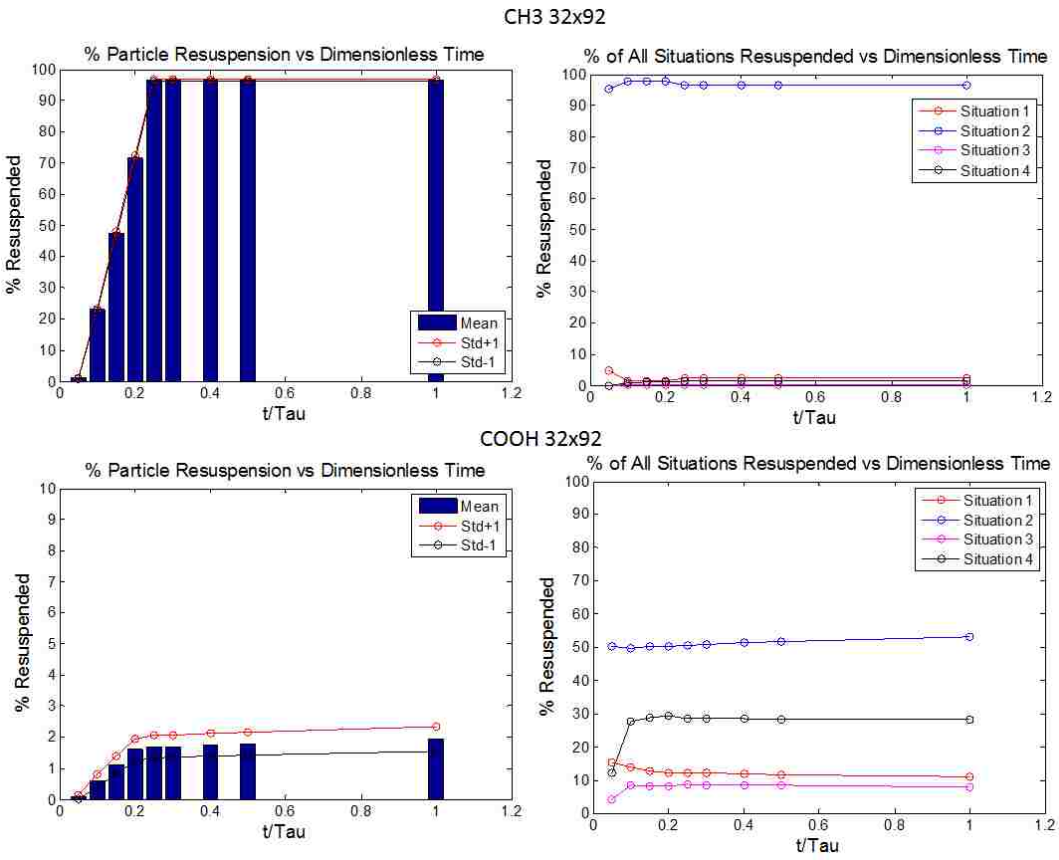


Figure 30 – Particle resuspension for 32x92 grid size with varying attractive forces vs dimensionless time.

10.5 – Results at Different Velocities

The code can run at varying velocities, as the piston flow was analyzed for two velocities, the flow following a Mach 1.2 shock and the flow following a Mach 1.7 shock. When the velocity profile for the Piston flow following a Mach 1.7 shock was used, 0.55τ was the time it took for the shock to cross the particles. The flow is significantly higher than the

piston flow following a Mach 1.2 shock, and thus more particles resuspend with the entirety of the CH3 particles resuspending regardless of grid size and a majority of the COOH and NH2 particles resuspending as well. These results can be compared in Table 17 and Table 18.

Table 17 – Percent resuspension for various grids and attractive forces for a Mach 1.2 shock, $t/\tau = 1$.

	% Resuspended	% Situation One	% Situation Two	% Situation Three	% Situation Four
12x52					
COOH	0.93	0	100	0	0
CH3	90.18	5.49	90.71	0.11	3.69
22x102					
COOH	0.82	4.86	83.21	5.26	6.66
CH3	95.12	2.70	95.34	0.04	1.91
32x92					
COOH	1.95	10.99	53.04	7.86	28.10
CH3	96.45	2.28	96.34	0.04	1.34

Table 18 – Percent resuspension for various grids and attractive forces for a Mach 1.7 shock, $t/\tau = 2$.

	% Resuspended	% Situation One	% Situation Two	% Situation Three	% Situation Four
12x52					
NH2	84.09	5.80	91.24	0.05	2.90
COOH	94.29	5.34	92.07	0.06	2.53
CH3	100	5.27	94.73	0	0
22x102					
NH2	91.91	2.80	95.80	0.02	1.38
COOH	97.07	2.70	96.00	0.02	1.28
CH3	100	2.66	97.34	0	0
32x92					
NH2	94.12	2.31	96.72	0.01	0.96
COOH	97.91	2.28	96.78	0.01	0.92
CH3	100	2.29	97.71	0	0

Chapter 11 – Observed Particle Behavior

There were a number of interesting observed particle behaviors.

Observation 1 – Most particles resuspend as Situation Two Particles. Situation Twos are exposed to the flow more prominently, and when they are resuspended, they usually generate another Situation Two particle immediately downstream. This creates a wave of particles that are resuspending a few particle diameters behind the location of the shock. The lag is due to the particle having to rise one radii before it is considered to have resuspended and expose the next particle downstream to the flow. They make up the majority of the top rows, with the last particle on each row becoming a Situation One, as seen in Figure 31.

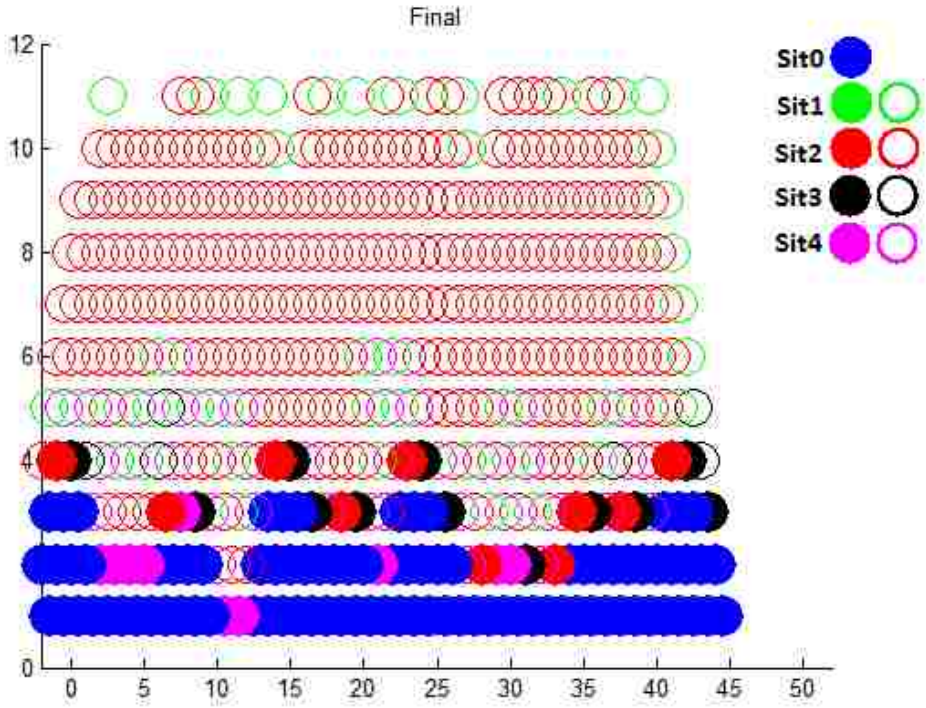


Figure 31 – Typical particle bed at the end of a simulation. Resuspended Situation Twos indicated by empty red circles

Observation 2 – There are fewer Situation Three's that resuspend. They are exposed to less of the flow and are the last exposed to the flow. When a particle immediately upstream resuspends, a Situation Three is transformed into a Situation One. If there are Situation Three particles, their resuspension usually occurs in the lower rows, where the velocity is lower, making it less likely for the upstream particles to resuspend as the piston flow passes over the particles.

Observation 3 – There are very few Situation Four particles being resuspended. They have little exposure to the flow, and particles on either side prevent them from rotating. They often transition to another Situation Two or Three, depending on the motion of the surrounding particles.

Observation 4 – The top several rows are the easiest to resuspend, because they are exposed to large enough velocity to overcome the attractive forces. The upper rows are exposed to higher velocity flow. They usually resuspend as a Situation Two, and they zipper off, resuspending from upstream to downstream.

Observation 5 – Situation One particles tended to form when a particle has a stronger attractive force. They were usually a Situation Two, until the particle immediately downstream resuspended. Now the Situation One is free to rotate and overcome its stronger attractive force and resuspend. The Situation Ones that are inside the particle bed (as opposed to the top row of particles), tended to be paired with a Situation Three particle downstream, or with a weakly attracted Situation Four particle. This is illustrated by the lone Situation One particle in the fourth row in Figure 32.

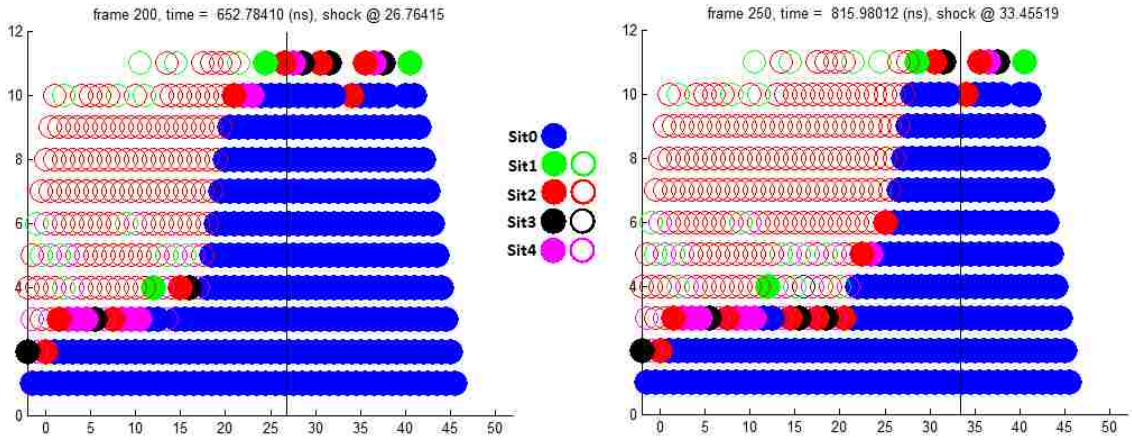


Figure 32 – Typical particle bed visualizations during shock passage that show a Situation One particle remaining on the fourth row. Empty circles indicate the Situation of the resuspended particle.

Observation 6 – The fraction of resuspended particles that are Situation Twos decreases with time. They are the overwhelming majority at the start of the simulation and then taper off slightly as other situations resuspend in the lower rows where strongly attracted particles are not so easily resuspended.

Observation 7 – The higher the particles extend into the flow, the higher the percentage of resuspended particles. However, if the particles only extend along the surface, and have more columns rather than adding rows, the percentage of the particles remains the same, indicating that there is a threshold height beneath which particles will not resuspend, regardless of the time exposed to the flow.

Observation 8 – If particles in lower rows did not resuspend shortly after the shock passing, attractive forces were too strong for the velocity at that elevation to overcome. They tended to come in pairs of Situations Two and Three.

Observation 9 – Particles that do not resuspend anchor the top of a mountain. Particles would erode from the back. Canyons are also created, as erosion from the front of one mountain would merge with erosion from the back of a neighboring mountain. Several mountains can develop, as shown in Figure 33.

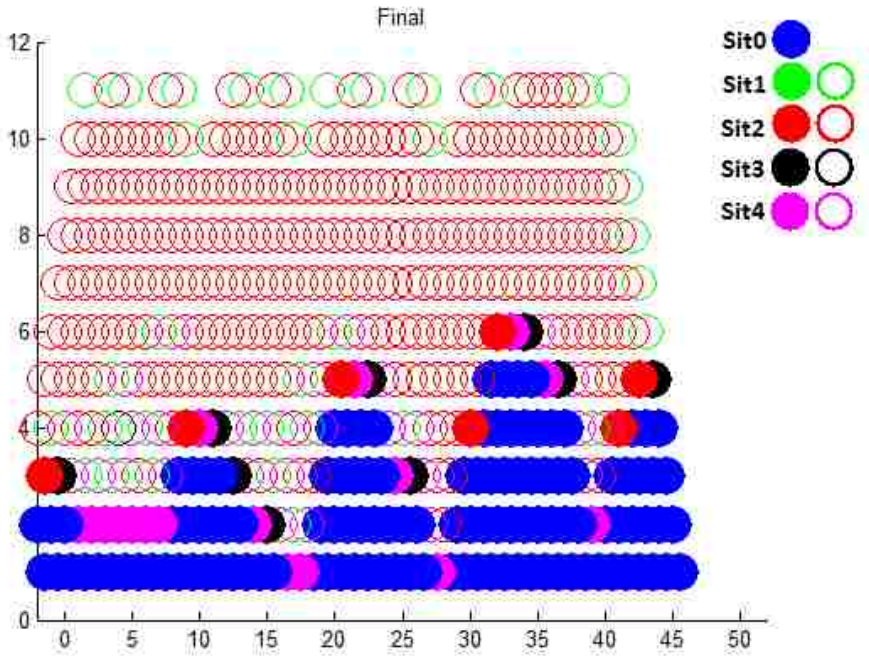


Figure 33 – Typical particle bed visualization showing remaining mountains developed by strongly attracted particles. Empty circles indicate the Situation of the resuspended particle.

Observation 10 – The Base Row of Particles rarely was exposed to the piston flow, and when it was, it was not exposed long enough for the particles to resuspend.

Observation 11 – The particle resuspension happens quickly compared to τ, the time of the passage of the piston flow over the bed of particles. Nearly all particle resuspension was completed by $\tau/4$, or the time for the shock to pass over the particle bed.

Chapter 12 – Conclusions

This model was developed to analyze particle resuspension in piston flow following a shock wave, in a two-dimensional set up with perfectly spherical particles all of the same size. The model classified each particle according to the location of adjacent particles, and based on that classification, specific force and moment equations were used to determine whether particles resuspended. This model was based on the previous models of Reeks, Reed, and Hall (Reeks et al. 1998, 2001), and used particle adhesion data from Truman et al. (2011).

Various multi-particle interactions and behaviors were observed. A majority of the particles resuspend, zippering off in the direction of the piston flow trailing the shock. Strongly attracted particles cause mountains and canyons to develop, caused by the erosion of nearby weakly-attracted particles. All particles resuspend above a certain height, where velocity is high enough for lift and drag to overcome the maximum attractive forces on the particles. The stronger the attractive forces between particles, the greater the velocity required to generate the necessary lift and drag forces to cause particle resuspension. Resuspension of a specific particle is dependent on particle placement within the grid and its surrounding particles that determine the attractive forces. The higher the piston velocity, the more particles will resuspend.

Within the simulation, each particle is unique, both in location and in attraction, and thus each simulation provides a different result. These results can be analyzed for trends regarding the influence of the attractive forces on particle resuspension, and the piston

velocity trailing the shock. This gives insight into how particles resuspend and the rate at which they resuspend, and what forces dictate this resuspension.

Chapter 13 – Future Work

While this analysis covers many of the basics to give a general idea of how particles should and can resuspend, it is not a complete model. There are some details that it does not cover, and some dimensions that it does not cover. These parts were all omitted due to assumptions made at the beginning of the analysis, when the situation was simplified down to the core and most important components.

These basic assumptions, which were described at the beginning of the model set up, all originated from the simplification that the particles form a perfect, two-dimensional, hexagonal grid. In real life, this assumption is false, as a perfect hexagonal grid requires perfect spheres of identical size and packed into a perfect grid. This means that Triangle ABC in Figure 34 may not be perfectly equilateral, meaning the angles used to make the above formulas and models could vary around a mean value of 60 degrees. This would change the strength of the forces, and would alter how the particles lifted off, as well as create new situations with voids within the grid. The contact forces would be much more difficult to compute.

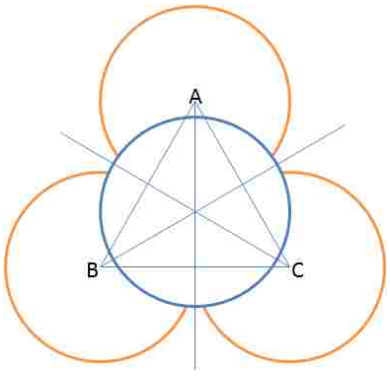


Figure 34 – Potential three-dimensional developments

It was assumed that the particles were in a two-dimensional pattern as opposed to a three-dimensional pattern. In theory, there would simply be new free-body diagrams that allowed for three-dimensional motion. Each particle (in a perfect, hexagonal grid where each particle was perfectly spherical and identical in size) would be touching as many as twelve other particles. Particles that are free to move have three adjacent particles below and up to six adjacent particles on the same level. This would mean that there are 2^6 possible situations for each particle. In the two-dimensional situation, the wind is assumed to only come from one direction, whereas in the three-dimensional simulation the wind can come from any angle in the horizontal plane.

Particle attraction is another way the problem was simplified for analysis. The attractions forces were assumed to be for a nano-rough surface. A nano-rough surface has points of contact and the elasticity and deformation of the particle do not need to be considered. A much more complicated model could developed from this to include particle elasticity and deformation.

The equations of motion were built using Reeks and Hall's (2001) second version of the Rock'n Roll model, where the velocity fluctuations do not create strong enough oscillations to affect particle motion. Particle motion is dictated primarily by drag and lift forces as the particles are pivoted about a point. As per their second model, once a particle detaches, it is no longer considered. A more detailed model would include the finding of their first Rock'n Roll model, where particles will remain within the sublayer, drifting downstream, until they reach resonance frequency and escape the sublayer. As the resonance frequency is dictated by particle motion due to turbulent forces, which are by their very nature unpredictable, this would be a probabilistic model.

The particles are assumed to resuspend due to the piston flow, and the piston flow only. This does not take into account particle collisions, nor does it take into account the shock itself. Igra and Falcovitz (2011) predicted that the lift and drag forces on a sphere were affected by sliding shocks. They found time-varying drag and lift forces due to shock interaction with the surface. Future work may take this into account and add these new forces acting on the particle during the passage of the shock.

References

Ahamdi, G. "Lift Force" *ME 437/457*. Clarkson University, New York, 2005. [http://web2.clarkson.edu/projects/fluidflow/courses/me637/1_4Lift.pdf Accessed 4/2/14]

Bernard, P.S., and Wallace, J.M. *Turbulent Flow: Analysis, Measurement and Prediction*, Wiley, 2002, Hoboken, New Jersey, p. 121.

Biasi, L., de los Reyes, A., Reeks, M.W., and de Santi, G.F. (2001) "Use of a simple model for the interpretation of experimental data on particle resuspension in turbulent flows" *Journal of Aerosol Science*. Vol. **32**, pp. 1175-1200.

Boor, B.E., Siegel, J.A., and Novoselac, A. (2011) "Development of an experimental methodology to determine monolayer and multilayer particle resuspension from indoor surfaces" *ASHRAE Transactions*. Vol. **117** Part **1**, pp. 434-441.

Fedorov, A.V., Fedorova, N.N., and Fedorchenko, L.A. (2002) "Numerical simulation of the shock wave interaction with a near-wall fine particle layer" Proceedings, 11th International Conference on Methods of Aerophysical Research, Novosibirsk, Russia, July 2002. pp. 45-50.

Gac, J., Sosnowsku, T., and Gradon, L. (2008) "Turbulent flow energy for aerosolization of powder particles" *Journal of Aerosol Science*. Vol. **39**, pp. 113-126.

Hontanñón, E., de los Reyes, A., and Capitão, J.A. (2000) "The CÆSAR code for aerosol resuspension in turbulent pipe flows: assessment against the STORM experiments" *Journal of Aerosol Science*. Vol. **31**, No. **9**, pp. 1061-1076.

Ibrahim, A.H., Dunn, P.F., and Brach, R.M., (2003) “Microparticle detachment from surfaces exposed to turbulent air flow: controlled experiments and modeling” *Journal of Aerosol Science*. Vol. **34**, pp. 765-782.

Ibrahim, A.H., Dunn, P.F., and Brach, R.M., (2004a) “Microparticle detachment from surfaces exposed to turbulent air flow: Effects of flow and particle deposition characteristics” *Journal of Aerosol Science*. Vol. **35**, pp. 805-821.

Ibrahim, A.H., Dunn, P.F., and Brach, R.M., (2004b) “Microparticle detachment from surfaces exposed to turbulent air flow: Microparticle motion after detachment” *Journal of Aerosol Science*. Vol. **35**, pp. 1189-1204.

Igra, D. and Falcovitz, J., (2001) “Lift and drag on a sphere above ground by a sliding shock.” *Proceedings of the Insitution of Mechanical Engineers, Part G: Journal of Aerospace Engineering.* Vol **226**, pp. 550-560.

Jacobs, G.B., Don, W.S., Dittmann, T. (2012) “High-Order resolution Eulerian-Lagrangian simulations of particle dispersion in the accelerated flow behind a moving shock” *Theoretical and Computational Fluid Dynamics*. Vol **26**, pp. 37-50.

Kassab, A.S., Ugaz, V.M., King, M.D., and Hassan, Y.A. (2012) “High resolution study of micrometer particle detachment on different surfaces” *Aerosol Science and Technology*. Vol. **47**, pp. 351-360.

Kim, Y., Gidwani, A., Wyslouzil, B.E., and Sohn, C.W. (2010) “Source term models for fine particle resuspension from indoor surfaces” *Building and Environment.* Vol. **45**, pp. 1854-1865.

Krauter, P. and Biermann, P. (2007) “Reaerosolization of fluidized spores in ventilation systems” *Applied and Environmental Microbiology*. Vol. **73** No. **7**, p. 2165-2172.

Lazaridis, M., Drossinos, Y., and Georgopoulos, P.G. (1998) “Turbulent resuspension of small nondeformable particles” *Journal of Colloid and Interface Science*. Vol. **204**, pp. 24-32.

Loosmore, G. (2003) “Evaluation and development of models for resuspension of aerosols at short times after deposition” *Atmospheric Environment*. Vol. **37**, pp. 639-647.

Mollinger, A.M., Nieuwstadt, F.T.M., and Bessem, J.M. (1995) “A new device to measure the lift force on a particle in the viscous sublayer” *Measurement Science Technology*. Vol. **6**, pp. 206-213.

Parmar, M.K., Haselbacher, A., and Balachandar, S. (2009a) “Prediction and modeling of shock-particle interaction” AIAA Paper 2009-1124.

Parmar, M.K., Haselbacher, A., and Balachandar, S. (2009b) “Modeling of unsteady forces on particles in compressible flow” AIAA Paper 2009-1125.

Parmar, M.K., Haselbacher, A., and Balachandar, S. (2010) “Improved drag correlation for spheres and application to shock tube experiments” *AIAA Journal*. Vol. **48** No. **6**, pp. 1273-1276.

Reeks, M.W., Reed, J. and Hall, D.(1988) “On the resuspension of small particles by a turbulent flow.” *Journal of Physics D. Applied Physics.* Vol. **21**, pp. 574-569.

Reeks, M.W. and Hall, D (2001) “Kinetic models for particle resuspension in turbulent flows: theory and measurement” *Journal of Aerosol Science*. Vol. **32**, pp. 1-31.

Soltani, M., and Ahmadi, G. (1995) “Particle detatchment from rough surfaces in turbulent flows” *Journal of Adhesion*. Vol. **51**, pp. 105-123

Suzuki, T., Sakamura, Y., Igra, O., Adachi, T., Kobayashi. S, Kotani, A, and Funawatashi, Y. (2006) “Shock tube study of particles’ motion behind a planar shock wave” *Measurement Science and Technology*. Vol. **16**, pp. 2431-2436.

Truman, C.R., Vorobieff,P., and Smyth, H.D.C., "Experimental and numerical studies of respirable particle transport from surfaces by acoustic/shock waves," Annual Report to DTRA Threat Agent Science-Agent Characterization, 2011, 8 pp.

Vainshtein, P., Ziskind, G., Fichman M., and Gutfinger, C. (1997) “Kinetic model of particle resuspension by drag force” *Physical Review Letters*. Vol. **8** No. **3**, pp. 551-554.

Wayne, P.J., Vorobieff, P., Smyth, H., Bernard, T., Corbin, C., Maloney, A., Conroy, J., White, R., Anderson, M., Kumar, S., Truman, C.R., and Srivastava, D. (2013) “Shock-driven particle transport off smooth and rough surfaces” *Journal of Fluids Engineering*. Vol. **135**, 061302.

White, F.M. (2006) *Viscous Fluid Flow, 3rd Edition.* McGraw Hill (2006), New York, NY. pp. 505-521.

Zhang, H., and Ahmadi, G. (2000) “Aerosol particle removal and re-entrainment in turbulent channel flows – a direct numerical simulation approach” *Journal of Adhesion*. Vol **74**, pp. 441-493.

Ziskind, G., Fichman M., and Gutfinger, C. (1998) “Effects of shear on particle motion near a surface – application to resuspension” *Journal of Aerosol Science*. Vol. **29**, pp. 323-338.

Ziskind, G., Fichman M., and Gutfinger, C. (2000) “Particle behavior on surfaces subjected to external excitations” *Journal of Aerosol Science*. Vol. **31** No. **6**, pp. 703-719.

Appendix A – MATLAB Codes

A.1 – MomentCalc

```
clear all

close all

clc


% This program calculates the values of velocity needed to start a

% particlerolling, when the moment is less than zero, as a positive

% moment indicates no motion.


% Assigns variables values.

r = 5*10^-6; %m

m = 2.4*4/3*pi*r^3*10^-3; %kg

g = 9.81; %m/s^2

Cd = 0.0337; %drag coef, calculated from Soltani et al. (1995)

Cl = 0.0509; %lift coef, calculated from Ahmadi (2005)

roh = 1.184149835; %kg/m^3

V = 0:1:200; %m/s

Fd = 1/2*roh*V.^2*Cd*pi*r^2; %N

Fl = 1/2*roh*V.^2*Cl*pi*r^2; %N


%% For the initial value of Fa = 20*10^-9N


Fa = 20*10^-9; %N


% Situation ONE
```

```
Mp1 = Fa*cosd(30)*r - Fd*r*cosd(30) - Fl*r*cosd(60) ;


% Situation TWO
Mp2 = Fa*cosd(30)*r + Fa*cosd(30)*r - Fl*r;


% Situation THREE
Mp3 = Fa*cosd(30)*r + Fa*cosd(30)*r*cosd(30) - Fl*r*cosd(60)...
    - Fd*r*cosd(60);


% Situation FOUR
Mp4 = Fa*cosd(30)*r + Fa*cosd(30)*r - Fl*r;


% plot the values for visual inspection
plot(Mp1, V, 'r')
hold on
grid
plot(Mp2, V, 'b')
plot(Mp3, V, 'g')
plot(Mp4, V, 'k')
plot([0,0],[0,200],'k')
legend('Sit1','Sit2','Sit3','Sit4')
xlabel('Moments, N/m')
ylabel('Lift Off Velocity, m/s')
axis([-2*10^-13,2*10^-13,0,200])
title('Velocity(m/s) vs Net Moment(N/m), FA = 20*10^-9N, NH2 Surface')
```

```
diff1 = abs(0-Mp1);

diff2 = abs(0-Mp2);

diff3 = abs(0-Mp3);

diff4 = abs(0-Mp4);


x1 = diff1(1);

y1 = 1;

x2 = diff2(1);

y2 = 1;

x3 = diff3(1);

y3 = 1;

x4 = diff4(1);

y4 = 1;


for i = 1:length(diff1)

    if diff1(i) < x1

        x1 = diff1(i);

        y1 = i;

    end

    if diff2(i) < x2

        x2 = diff2(i);

        y2 = i;

    end

    if diff3(i) < x3

        x3 = diff3(i);

        y3 = i;

    end

    if diff4(i) < x4
```

```
        x4 = diff4(i);
        y4 = i;
    end
end

y = [y1 y2 y3 y4]
plot([0,0,0,0],[y1,y2,y3,y4],'ko')

%% upper bound of adhesion forces. Fa = 5*10^-9N

Faa = 5*10^-9; %N

% Situation ONE
Mp1a = Faa*cosd(30)*r - Fd*r*cosd(30) - Fl*r*cosd(60);

% Situation TWO
Mp2a = Faa*cosd(30)*r + Faa*cosd(30)*r - Fl*r;

% Situation THREE
Mp3a = Faa*cosd(30)*r + Faa*cosd(30)*r*cosd(30) - Fl*r*cosd(60)...
    - Fd*r*cosd(60);

% Situation FOUR
Mp4a = Faa*cosd(30)*r + Faa*cosd(30)*r - Fl*r;

% plot
figure
```

```
plot(Mp1a, V, 'r')
hold on
grid
plot(Mp2a, V, 'b')
plot(Mp3a, V, 'g')
plot(Mp4a, V, 'k')
plot([0,0],[0,200],'k')
legend('Sit1','Sit2','Sit3','Sit4')
xlabel('Moments, N/m')
ylabel('Lift Off Velocity, m/s')
axis([-2*10^-13,2*10^-13,0,200])
title('Velocity(m/s) vs Net Moment(N/m), FA = 5*10^-9N, COOH Surface')

diff1 = abs(0-Mp1a);
diff2 = abs(0-Mp2a);
diff3 = abs(0-Mp3a);
diff4 = abs(0-Mp4a);

x1a = diff1(1);
y1a = 1;
x2a = diff2(1);
y2a = 1;
x3a = diff3(1);
y3a = 1;
x4a = diff4(1);
y4a = 1;

for i = 1:length(diff1)
```

```
    if diff1(i) < x1a
        x1a = diff1(i);
        y1a = i;
    end
    if diff2(i) < x2a
        x2a = diff2(i);
        y2a = i;
    end
    if diff3(i) < x3a
        x3a = diff3(i);
        y3a = i;
    end
    if diff4(i) < x4a
        x4a = diff4(i);
        y4a = i;
    end
end

ya = [y1a y2a y3a y4a]
plot([0,0,0,0],[y1a,y2a,y3a,y4a],'ko')

%% nano-rough bound of adhesion forces. Adhesion = 10*10^-9N

Fab = 0.5*10^-9; %N

% Situation ONE
Mp1b = Fab*cosd(30)*r - Fd*r*cosd(30) - Fl*r*cosd(60);
```

```
% Situation Two
Mp2b = Fab*cosd(30)*r + Fab*cosd(30)*r - Fl*r;

% Situation Three
Mp3b = Fab*cosd(30)*r + Fab*cosd(30)*r*cosd(30) - Fl*r*cosd(60)...
    - Fd*r*cosd(60);

% Situation Four
Mp4b = Fab*cosd(30)*r + Fab*cosd(30)*r - Fl*r;

% plot attempt
figure
plot(Mp1b, V, 'r')
hold on
grid
plot(Mp2b, V, 'b')
plot(Mp3b, V, 'g')
plot(Mp4b, V, 'k')
plot([0,0],[0,200],'k')
legend('Sit1','Sit2','Sit3','Sit4')
xlabel('Moments, N/m')
ylabel('Lift Off Velocity, m/s')
axis([-2*10^-13,2*10^-13,0,200])
title('Velocity(m/s) vs Net Moment(N/m), FA = 0.5*10^-9N, CH3 Surface')

diff1 = abs(0-Mp1b);
```

```
diff2 = abs(0-Mp2b);

diff3 = abs(0-Mp3b);

diff4 = abs(0-Mp4b);


x1b = diff1(1);

y1b = 1;

x2b = diff2(1);

y2b = 1;

x3b = diff3(1);

y3b = 1;

x4b = diff4(1);

y4b = 1;


for i = 1:length(diff1)
    if diff1(i) < x1b
        x1b = diff1(i);
        y1b = i;
    end
    if diff2(i) < x2b
        x2b = diff2(i);
        y2b = i;
    end
    if diff3(i) < x3b
        x3b = diff3(i);
        y3b = i;
    end
    if diff4(i) < x4b
        x4b = diff4(i);
```

```
        y4b = i;
    end
end


yb = [y1b y2b y3b y4b]
plot([0,0,0,0],[y1b,y2b,y3b,y4b],'ko')
```

A.2 – gridgen

```
function [gridd] = gridgen(rows,cols)
% GRIDGEN generates a grid of particles to run test with. Input is
% the base number of ROWS and COLS. The output is the grid of
% particles to be modeled. [gridd] = gridgen(rows,cols)

%% Generate Base
% Generates a random set of numbers in an [ROWS, COLS] matrix.
% Rounded to either a zero or a one, they indicate an empty or filled
% space.

gridd = rand(rows,cols);
gridd = round(gridd);
gridd = [zeros(1,cols); gridd];
gridd = [zeros(rows+1,1) gridd zeros(rows+1,rows+1)];
[r,c] = size(gridd);

cantmove = 8;
sit1 = 1;
sit2 = 2;
sit3 = 3;
sit4 = 4;

%% Fill in the Base
% Fills in the base so there are no zero-values within the grid and
% eliminates hanging particles, making sure each particle has a
% stable base of two particles to rest on.
```

```
for i = 1:r
    for j = 1:c
        if gridd(i,j) == 1;
            gridd(i+1,j) = 1;
            gridd(i+1,j+1) = 1;
        end
    end
end

%% Switch to -1 for empty space and 0 for particle
% Reallocate for ease.

for i =1:r+1;
    for j = 1:c
        if gridd(i,j) == 0
            gridd(i,j) = -1;
        else
            gridd(i,j) = 0;
        end
    end
end



%% Determine which Particles cannot move
% Determines Situation Zero particles.
```

```
for i = 2:r+1
    for j = 2:c
        if gridd(i-1,j) == 0
            gridd(i,j) = cantmove;
        end
        if gridd(i-1,j-1) == 0
            gridd(i,j) = cantmove;
        end
        if gridd (i-1,j) == cantmove
            gridd(i,j) = cantmove;
        end
        if gridd(i-1,j-1) == cantmove
            gridd(i,j) = cantmove;
        end
    end
end

%% Determine what Situation remaining particles are
% Determines the situations for the remaining particles, One
% through Four.

for i = 2:r+1
    for j = 2:c
        if gridd(i,j) == 0
            if gridd(i,j-1) == -1
                if gridd(i,j+1) == -1
                    gridd(i,j) = sit1;
                else
```

```
                    gridd(i,j) = sit2;
                end
            else
                if gridd(i,j+1) == -1
                    gridd(i,j) = sit3;
                else
                    gridd(i,j) = sit4;
                end
            end
        end
    end
end
```

A.3 – partmap

```
function partmap(gridd)
% PARTMAP visually displays the particles generated by GRIDGEN.

gridd2 = flipud(gridd); %flips gridd, so it can be easily mapped
[r,c] = size(gridd); %sizes gridd

figure
axis([-2,c,-1,c])
kk = 0;

hold on
for i = r:-1:1
    for j = c:-1:1
        if gridd2(i,j) == 8 %Plots Situation Zero Particles
            if floor(i/2) == i/2
                plot(j-kk,i,'.','MarkerSize',50)
            else
                    plot(j-kk,i,'.','MarkerSize',50)
            end
        end
        if gridd2(i,j) == 1 %Plots Situation One Particles
            if floor(i/2) == i/2
                plot(j-kk,i,'g.','MarkerSize',50)
            else
                plot(j-kk,i,'g.','MarkerSize',50)
            end
```

```
end
if gridd2(i,j) == 2 %Plots Situation Two Particles
    if floor(i/2) == i/2
        plot(j-kk,i,'r.','MarkerSize',50)
    else
        plot(j-kk,i,'r.','MarkerSize',50)
    end
end
if gridd2(i,j) == 3 %Plots Situation three Particles
    if floor(i/2) == i/2
        plot(j-kk,i,'k.','MarkerSize',50)
    else
        plot(j-kk,i,'k.','MarkerSize',50)
    end
end
if gridd2(i,j) == 4 %Plots Situation Four Particles
    if floor(i/2) == i/2
        plot(j-kk,i,'m.','MarkerSize',50)
    else
        plot(j-kk,i,'m.','MarkerSize',50)
    end
end
if gridd2(i,j) == -8 %Plots Resuspended Situation Ones
    if floor(i/2) == i/2
        plot(j-kk,i,'go','MarkerSize',15)
    else
        plot(j-kk,i,'go','MarkerSize',15)
    end
```

```
            end
            if gridd2(i,j) == -7 %Plots Resuspended Situation Twos
                if floor(i/2) == i/2
                    plot(j-kk,i,'ro','MarkerSize',15)
                else
                    plot(j-kk,i,'ro','MarkerSize',15)
                end
            end
            if gridd2(i,j) == -6 %Plots Resuspended Situation Threes
                if floor(i/2) == i/2
                    plot(j-kk,i,'ko','MarkerSize',15)
                else
                    plot(j-kk,i,'ko','MarkerSize',15)
                end
            end
            if gridd2(i,j) == -5 %Plots Resuspended Situation Fours
                if floor(i/2) == i/2
                    plot(j-kk,i,'mo','MarkerSize',15)
                else
                    plot(j-kk,i,'mo','MarkerSize',15)
                end
            end
        end
        kk = kk+.5;
    end
```

A.4 – PartSimSimple

```
clear all

close all

clc


rows = 10; %Row Input

cols = 10; %Column Input

tindex = 15; %Number of Plots


gridda = gridgen(rows,cols);

partmap(gridda)

axis([-2,cols+12,0,rows+2])

title('Initial')


%% Variable Assignment

% Assigns variables, mostly zero values for storage.

cantmove = 8;

sit1 = 1;

sit2 = 2;

sit3 = 3;

sit4 = 4;

lift = -9;

initial = 0;


[rows,cols] = size(gridda);

dely = gridda*0;

vely = gridda*0;
```

```
accy = gridda*0;

forcey = gridda*0;

forcex = gridda*0;

posy = gridda*0;


loggridda = zeros(rows,cols,tindex);

logdely = loggridda*0;

logvely = loggridda*0;

logaccy = loggridda*0;

logforcey = loggridda*0;

logforcex = loggridda*0;

logposy = loggridda*0;


delt = 10^-17;


%% Known Variables


Fa = 0.5*10^-9; %N, Attractive Force

r = 5*10^-6; %m, radius of the particle

m = 2.4*4/3*pi*r^3*10^-3; %kg, mass of the particle

g = 9.81; %m/s^2, gravity

Cd = 0.0337; %drag coef, calculated from Soltani et al. (1995)

Cl = 0.0509; %lift coef, calculated from Ahmadi (2005)

roh = 1.184149835; %kg/m^3, density of the air

V = 200; %m/s, velocity, constant value

Fd = 1/2*roh*V.^2*Cd*pi*r^2; %N, drag force

Fl = 1/2*roh*V.^2*Cl*pi*r^2; %N, lift force
```

```
%%



for t = 2:tindex+1
    for i = 1:rows
        for j = 1:cols


            %Situation One Calculations
            if gridda(i,j) == sit1
                % calculate force y
                forcey(i,j) = -Fa*cosd(30) + Fl;
                logforcey(i,j,t) = forcey(i,j);
                % determine acc
                accy(i,j) = forcey(i,j)/m;
                logaccy(i,j,t) = accy(i,j);
                % determine vel y
                vely(i,j) = logvely(i,j,t-1) + accy(i,j)*delt;
                logvely(i,j,t) = vely(i,j);
                % determine pos y
                posy(i,j) = logposy(i,j,t-1) + logvely(i,j,t-1) + ...
                    accy(i,j)*delt^2/2;
                logposy(i,j,t) = posy(i,j);
                %if pos y is greater than threshold, shift to resuspend
                if (logposy(i,j,t)-logposy(i,j,t-1)) > r/2
                    gridda(i,j) = lift + sit1;
                end
```

```
end

%Situation Two Calculations
if gridda(i,j) == sit2
    % calculate force y
    forcey(i,j) = - Fa*cosd(30)- Fa*cosd(30) + Fl;
    logforcey(i,j,t) = forcey(i,j);
    % determine acc
    accy(i,j) = forcey(i,j)/m;
    logaccy(i,j,t) = accy(i,j);
    % determine vel y
    vely(i,j) = logvely(i,j,t-1)+accy(i,j)*delt;
    logvely(i,j,t) = vely(i,j);
    % determine pos y
    posy(i,j) = logposy(i,j,t-1) + logvely(i,j,t-1) + ...
        accy(i,j)*delt^2/2;
    logposy(i,j,t) = posy(i,j);
    %if pos y is greater than threshold, shift to resuspend
    if (logposy(i,j,t)-logposy(i,j,t-1)) > r/2
        gridda(i,j) = lift + sit2;
    end
end

%Situation Three Calculations
if gridda(i,j) == sit3
    % calculate force y
    forcey(i,j) = - Fa*cosd(30) - Fa*cosd(30) + Fl;
    logforcey(i,j,t) = forcey(i,j);
```

```
        % determine acc
        accy(i,j) = forcey(i,j)/m;
        logaccy(i,j,t) = accy(i,j);
        % determine vel y
        vely(i,j) = logvely(i,j,t-1)+accy(i,j)*delt;
        logvely(i,j,t) = vely(i,j);
        % determine pos y
        posy(i,j) = logposy(i,j,t-1) + logvely(i,j,t-1) + ...
            accy(i,j)*delt^2/2;
        logposy(i,j,t) = posy(i,j);
        %if pos y is greater than threshold, shift to resuspend
        if (logposy(i,j,t)-logposy(i,j,t-1)) > r/2
            gridda(i,j) = lift + sit3;
        end
    end

    %Situation Four Calcuations
    if gridda(i,j) == sit4
        % calculate force y
        forcey(i,j) = - Fa*cosd(30) - Fa*cosd(30) + Fl;
        logforcey(i,j,t) = forcey(i,j);
        % determine acc
        accy(i,j) = forcey(i,j)/m;
        logaccy(i,j,t) = accy(i,j);
        % determine vel y
        vely(i,j) = logvely(i,j,t-1)+accy(i,j)*delt;
        logvely(i,j,t) = vely(i,j);
        % determine pos y
```

```
    posy(i,j) = logposy(i,j,t-1) + logvely(i,j,t-1) + ...
        accy(i,j)*delt^2/2;
    logposy(i,j,t) = posy(i,j);
    %if pos y is greater than threshold, shift to resuspend
    if posy(i,j) < 0
        posy(i,j) = 0;
    end
    if (logposy(i,j,t)-logposy(i,j,t-1)) > r/2
        gridda(i,j) = lift + sit4;
    end
end

% Redetermines Situations of Exposed Particles
if gridda(i,j) == cantmove
    if gridda(i-1,j) < -1
        gridda(i,j) = 0;
    end
    if gridda(i-1,j-1) < -1
        gridda(i,j) = 0;
    end
end
if gridda(i,j) == 0
    if gridda(i,j-1) < 0
        if gridda(i,j+1) < 0
            gridda(i,j) = sit1;
        else
            gridda(i,j) = sit2;
        end
```

```
                else
                    if gridda(i,j+1) < 0
                        gridda(i,j) = sit3;
                    else
                        gridda(i,j) = sit4;
                    end
                end
            end
            if gridda(i,j) > 0
                if gridda(i-1,j) > 0
                    gridda(i,j) = cantmove;
                end
                if gridda(i-1,j-1) > 0
                    gridda(i,j) = cantmove;
                end
            end
            loggridda(i,j,t) = gridda(i,j);
        end
    end
end



%Plots the steps
for i = 2:tindex
    partmap(loggridda(:,:,i))
    axis([-2,cols,0,rows])
end
```

```
title('Final')
```

A.5 – partmap2

```
function partmap2(gridd)

% PARTMAP2 visually displays the particles generated by GRIDGEN,
reduced.


gridd2 = flipud(gridd);

[r,c] = size(gridd);


kk = 0;


i8=0;

i1=0;

i2=0;

i3=0;

i4=0;

i_8=0;

i_3=0;

i_4=0;

i_5=0;


% Determines the situation and determines if the row is entirely made

% of particles that cannot move.

for i = r:-1:1

    if8 = 1;

    for j = c:-1:1

        if gridd2(i,j) == 8

            i8 = i8+1;  x8(i8) = j-kk;  y8(i8) = i;

        end
```

```
    if gridd2(i,j) == 1
        i1 = i1+1;  x1(i1) = j-kk;  y1(i1) = i;
    end
    if gridd2(i,j) == 2
        i2 = i2+1;  x2(i2) = j-kk;  y2(i2) = i;
    end
    if gridd2(i,j) == 3
        i3 = i3+1;  x3(i3) = j-kk;  y3(i3) = i;
    end
    if gridd2(i,j) == 4
        i4 = i4+1;  x4(i4) = j-kk;  y4(i4) = i;
    end
    if gridd2(i,j) == -8
        i_8 = i_8+1;  x_8(i_8) = j-kk;  y_8(i_8) = i;
    end
    if gridd2(i,j) == -7
        i_3 = i_3+1;  x_3(i_3) = j-kk;  y_3(i_3) = i;
    end
    if gridd2(i,j) == -6
        i_4 = i_4+1;  x_4(i_4) = j-kk;  y_4(i_4) = i;
    end
    if gridd2(i,j) == -5
        i_5 = i_5+1;  x_5(i_5) = j-kk;  y_5(i_5) = i;
    end
    if gridd2(i,j) ~= 8 && gridd2(i,j) ~= -1
        if8 = 0;
    end
end
```

```
    if i < (r-4) && if8 == 1
        h = i;
        break
    else
        h = i;
    end
    kk = kk+.5;

end



figure
axis([-2,c,(h-1),r])
hold on

% Plots based on values for if particles are there, what situation and
% if particles have resuspended.
if i8 > 0.1
    plot(x8,y8,'.','MarkerSize',50)
end
if i1 > 0.1
    plot(x1,y1,'g.','MarkerSize',50)
end
if i2 > 0.1
    plot(x2,y2,'r.','MarkerSize',50)
end
```

```
if i3 > 0.1
    plot(x3,y3,'k.','MarkerSize',50)
end
if i4 > 0.1
    plot(x4,y4,'m.','MarkerSize',50)
end
if i_8 > 0.1
    plot(x_8,y_8,'go','MarkerSize',15)
end
if i_3 > 0.1
    plot(x_3,y_3,'ro','MarkerSize',15)
end
if i_4 > 0.1
    plot(x_4,y_4,'ko','MarkerSize',15)
end
if i_5 > 0.1
    plot(x_5,y_5,'mo','MarkerSize',15)
end
```

A.5 – PartSimComplex

```
function [snapshotdata, partcounti, finalpart, cols, rows, ...
    totaltime] = PartSimComplex(rows,cols,timeselect,plotonoff,...
    movieonoff, attf)
% PARTSIMCOMPLEX is a simulation for particle simulation. Inputs are
% the number of ROWS and COLS for the simulation, the TIMESELECT (how
% long the simulation runs based on the piston velocity following the
% shock), PLOTONOFF, MOVIEONOFF indicators (tell whether or not plots
% or movies are generated), and attf (indicates which attractive force
% to use). Outputs are SNAPSHOTDATA (tracks the particle resuspension
% at specific times), PARTCOUNTI (the initial count of particles),
% FINALPART (the final count of unresuspended particles), ROWS (the
% final number of rows) and COLS (the final number of columns) as
% well as TOTALTIME (the time the simulation ran).

plotcount = 100; %how often the plots are displayed.
rowsstore = rows; %store for printing results
colsstore = cols; %store for printing results

load 0927hsbv % Load the velocity inputs from STARCCM+
load 0927hsbx % Load the x-coordinates for inputs STARCCM+

x = x*0.0254; %in to m conversion
v = v*0.3048; %ft/s to m/s conversion

[gridda] = gridgen(rows,cols); %calls GRIDGEN
```

```
[rows,cols] = size(gridda); %new values for rows/cols

%% Assigning Variables

if attf == 1
    Fa = rand([rows,cols])*(1*10^-9)+(0*10^-9);%CH3 Attr forces, N
elseif attf == 2
    Fa = rand([rows,cols])*(2*10^-9)+(4*10^-9);%COOH attr forces, N
elseif attf == 3
    Fa = rand([rows,cols])*(15*10^-9)+(10*10^-9);%NH2 attr forces, N
elseif attf == 4
    Fa = rand([rows,cols])*(1*10^-9)+(1*10^-9); %Test attr forces, N
end
r = 5*10^-6; %m, assigns the radius
m = 2.4*4/3*pi*r^3*10^-3; %kg, assigns the mass
g = 9.81; %m/s^2, gravity
Cd = 0.0337; %drag coef, calculated from Soltani et al. (1995)
Cl = 0.0509; %lift coef, calculated from Ahmadi (2005)
roh = 1.184149835; %kg/m^3, density of air.
Inertia = 2/5*m*r^2; %m^4, inertia for the particles

%% Variable Creation
% Creates variables used for the rest of the program, most of which
% are empty grids that will be filled in.

cantmove = 8;
sit1 = 1;
```

```
sit2 = 2;

sit3 = 3;

sit4 = 4;

lift = -9;

initial = 0;


cosd30 = cosd(30);

cosd60 = cosd(60);

sind30 = sind(30);


rint = zeros(rows,1);


respcount = 0;

sit1count = 0;

sit2count = 0;

sit3count = 0;

sit4count = 0;

resptrack = zeros(rows,cols);


dely = gridda*0;

vely = gridda*0;

accy = gridda*0;

forcey = gridda*0;

forcex = gridda*0;

posy = gridda*0;

alpha = gridda*0;

omega = gridda*0;
```

```
delta = gridda*0;


loggridda = zeros(rows,cols,2);
logdely = loggridda*0;
logvely = loggridda*0;
logaccy = loggridda*0;
logforcey = loggridda*0;
logforcex = loggridda*0;
logposy = loggridda*0;
logalpha = loggridda*0;
logomega = loggridda*0;
logdelta = loggridda*0;


%% Count the number of particles in simulation
% Counts the total number of particles within the simulation so
% they can be used for later analysis.


partcounti = 0;


for i = 1:rows
    for j = 1:cols
        if gridda(i,j) > 0
            partcounti = partcounti + 1;
        else
            partcounti;
        end
    end
```

```
end

%% Calculating the Time Index
% Determines how long the simulation should run and how many time
% steps should be taken. It is based on the fastest time to resuspend,
% that of a particle exposed to full flow, and then taking five
% timesteps to cover that resuspension.

Flmaxts = 1/2*roh*max(v).^2*Cl*pi*r^2; %Calculates max lift force
Fdmax = 1/2*roh*max(v).^2*Cd*pi*r^2; %calculates max drag force
forceyts = Flmaxts - m*g; %Calculales Y forces
accyts = forceyts/m; %Calculates acceration
timestep = sqrt((r*2)*2/accyts); %Calculates the time to resuspend
V_shock = 410; %m/s, the speed of the shock
Mpmax = (max(max(Fa))*cos(pi/6)*r - Fdmax*r*cos(pi/6) -...
    Flmaxts*r*cos(pi/6) + m*g*r*cos(pi/6)); %Calculates max moment
alpha = abs(Mpmax/(2/5*m*r^2)); %Calculates angular acceleration
timerot = sqrt(pi/6/alpha*2); %Calculates time to rotate

dis = cols*r*2; %distance the shock travels over particles
timedis = dis/max(v); %the time taken for the piston velocity
delt = timestep/5; %the length of a timestep
tindex = timedis/delt*timeselect; %how many steps taken

%% Calculating & Plotting the Velocity values at Row Heights
% Interpoltes the values for Velocity at given heights from STARCCM+
% values.
```

```
for i = 1:rows
    rint (i) = r+(rows-i)*r; %The row heights
end

V = interp1(x,v,rint); %the corresponding velocities
vprint = V; %stores Velocity values

if plotonoff == 1
    figure
    plot(v, x,'r.')
    hold on
    plot(V, rint)
    axis([0,50,0,0.0001])
    title('Velocity Interpolation')
    xlabel('Velocity, m/s')
    ylabel('Y-position, m')
    legend('STARCCM+','Interpolation')
end

%%

if plotonoff == 1
    if rows < 50
        partmap(gridda)
        axis([-2,cols,0,rows])
    else
```

```
        partmap2(gridda)
    end
    Frame(1)=getframe;
    title('Initial')
end



for i = rows:-1:1
    xcor(i,1) = (rows-i+1)*.5;
    for j = 1:cols
        xcor(i,j) = xcor(i,1)+j-1;
    end
end

xcor = xcor*2*r;

for t = 2: round(tindex)
    for i = 1:rows
        for j = 1:cols
            %determine current situation of the particle
            if gridda(i,j) > 0
                if gridda(i,j) == cantmove
                    gridda(i,j) = cantmove;
                elseif gridda(i,j-1) < 0
                    if gridda(i,j+1) < 0
                        gridda(i,j) = sit1;
                    else
```

```
            gridda(i,j) = sit2;
        end
    else
        if gridda(i,j+1) < 0
            gridda(i,j) = sit3;
        else
            gridda(i,j) = sit4;
        end
    end
end

% Determines the Velocity based on Shock Location
distance = V_shock*t*delt; %Calculates the shock location
if xcor(i,j) > distance % Verifies Shock Passage
    V(:) = 0; %if it has not passed, Velocity = 0.
end
Fd = 1/2*roh*V(i).^2*Cd*pi*r^2; %N %calculates drag forces
Fl = 1/2*roh*V(i).^2*Cl*pi*r^2; %N %calculates lift forces

% Situation One Calculations
if gridda(i,j) == sit1
    % calculate the moment m
    Mp1 = Fa(i,j)*cosd30*r - Fd*r*cosd30 - Fl*r*cosd60;
    % calculate force y
    forcey(i,j) = -Fa(i,j)*cosd30 - Fa(i,j)*cosd30 + Fl;
    logforcey(i,j,2) = forcey(i,j);
    if forcey(i,j) > 0
        % determine acc
```

```
            accy(i,j) = forcey(i,j)/m;
            logaccy(i,j,2) = accy(i,j);
            % determine vel y
            vely(i,j) = logvely(i,j,1)+accy(i,j)*delt;
            logvely(i,j,2) = vely(i,j);
            % determine pos y
            posy(i,j) = logposy(i,j,1) + logvely(i,j,1) + ...
                accy(i,j)*delt^2/2;
            logposy(i,j,2) = posy(i,j);
            %if pos y > threshold, shift to resuspend
            if (logposy(i,j,2)-logposy(i,j,1)) > r*2
                gridda(i,j) = lift + sit1;
                respcount = respcount + 1;
                sit1count = sit1count + 1;
                resptrack(i,j) = t;
            end
        elseif Mp1 < 0
            if logdelta(i,j,1) < -1*(pi/6)
                alpha(i,j) = 0;
                omega(i,j) = 0;
            else
                alpha(i,j) = Mp1/(Inertia);
                logalpha(i,j,2)=alpha(i,j);
                omega(i,j) = logomega(i,j,1) + alpha(i,j)*delt;
                logomega(i,j,2) = omega(i,j);
                delta(i,j) = logdelta(i,j,1) + ...
                    logomega(i,j,1)*delt + ...
                    alpha(i,j)*delt^2/2;
```

```
            logdelta(i,j,2) = delta(i,j);
        end
        if delta(i,j) < -1*(pi/6)
            % calculate force y
            forcey(i,j) = -Fa(i,j) + Fl;
            logforcey(i,j,2) = forcey(i,j);
            % determine acc
            accy(i,j) = forcey(i,j)/m;
            logaccy(i,j,2) = accy(i,j);
            % determine vel y
            vely(i,j) = logvely(i,j,1) + accy(i,j)*delt;
            logvely(i,j,2) = vely(i,j);
            % determine pos y
            posy(i,j) = logposy(i,j,1) + logvely(i,j,1)...
                + accy(i,j)*delt^2/2;
            logposy(i,j,2) = posy(i,j);
            %if pos y > threshold, shift to resuspend
            if (logposy(i,j,2)-logposy(i,j,1)) > r*2
                gridda(i,j) = lift + sit1;
                respcount = respcount + 1;
                sit2count = sit2count + 1;
                resptrack(i,j) = t;
            end
        end
        logomega(i,j,1) = logomega(i,j,2);
        logdelta(i,j,1) = logdelta(i,j,2);
    end
end
```

```
% Situation Two Calcluations
if gridda(i,j) == sit2
    Mp2b = Fa(i,j)*cosd30*r + Fa(i,j)*cosd30*r - Fl*r;
    % calculate force y
    forcey(i,j) = - Fa(i,j)*cosd30 - Fa(i,j)*cosd30 + Fl;
    logforcey(i,j,2) = forcey(i,j);
    % determine acc
    accy(i,j) = forcey(i,j)/m;
    logaccy(i,j,2) = accy(i,j);
    % determine vel y
    vely(i,j) = logvely(i,j,1) + accy(i,j)*delt;
    logvely(i,j,2) = vely(i,j);
    % determine pos y
    posy(i,j) = logposy(i,j,1)+logvely(i,j,1) + ...
        accy(i,j)*delt^2/2;
    logposy(i,j,2) = posy(i,j);
    %if pos y is greater than threshold, shift to resuspend
    if Mp2b < 0
        gridda(i,j) = lift + sit2;
        respcount = respcount + 1;
        sit2count = sit2count + 1;
        resptrack(i,j) = t;
    elseif (logposy(i,j,2)-logposy(i,j,1)) > r*2
        gridda(i,j) = lift + sit2;
        respcount = respcount + 1;
        sit2count = sit2count + 1;
        resptrack(i,j) = t;
```

```
    end
end


% Situation Three Calculations
if gridda(i,j) == sit3
    Mp3 = Fa(i,j)*cosd30*r + Fa(i,j)*cosd30*r*cosd30 - ...
        Fl*r*cosd60 - Fd*r*cosd60;
    % calculate force y
    forcey(i,j) = - Fa(i,j)*cosd(30) - Fa(i,j)*cosd(30)...
        + Fl;
    logforcey(i,j,2) = forcey(i,j);
    if forcey(i,j) > 0
        % determine acc
        accy(i,j) = forcey(i,j)/m;
        logaccy(i,j,2) = accy(i,j);
        % determine vel y
        vely(i,j) = logvely(i,j,1) + accy(i,j)*delt;
        logvely(i,j,2) = vely(i,j);
        % determine pos y
        posy(i,j) = logposy(i,j,1) + logvely(i,j,1) + ...
            accy(i,j)*delt^2/2;
        logposy(i,j,2) = posy(i,j);
        % if pos y > threshold, shift to resuspend
        if (logposy(i,j,2)-logposy(i,j,1)) > r*2
            gridda(i,j) = lift + sit3;
            respcount = respcount + 1;
            sit3count = sit3count + 1;
            resptrack(i,j) = t;
```

```
    end
elseif Mp3 < 0
    if logdelta(i,j,1) < -1*(pi/6)
        alpha(i,j) = 0;
        omega(i,j) = 0;
    else
        alpha(i,j) = Mp3/(Inertia);
        logalpha(i,j,2)=alpha(i,j);

        omega(i,j) = logomega(i,j,1) + alpha(i,j)*delt;
        logomega(i,j,2) = omega(i,j);

        delta(i,j) = logdelta(i,j,1) + ...
            logomega(i,j,1)*delt + ...
            alpha(i,j)*delt^2/2;
        logdelta(i,j,2) = delta(i,j);
    end
    if delta(i,j) < -1*(pi/6)
        % calculate force y
        forcey(i,j) = - Fa(i,j)*cosd(30) - ...
            Fa(i,j)*cosd(30) + Fl;
        logforcey(i,j,2) = forcey(i,j);
        % determine acc
        accy(i,j) = forcey(i,j)/m;
        logaccy(i,j,2) = accy(i,j);
        % determine vel y
        vely(i,j) = logvely(i,j,1) + accy(i,j)*delt;
        logvely(i,j,2) = vely(i,j);
```

```
            % determine pos y
            posy(i,j) = logposy(i,j,1) + logvely(i,j,1)...
                + accy(i,j)*delt^2/2;
            logposy(i,j,2) = posy(i,j);
            % if pos y > threshold, shift to resuspend
            if (logposy(i,j,2)-logposy(i,j,1)) > r*2
                gridda(i,j) = lift + sit3;
                respcount = respcount + 1;
                sit3count = sit3count + 1;
                resptrack(i,j) = t;
            end
        end
        logomega(i,j,1) = logomega(i,j,2);
        logdelta(i,j,1) = logdelta(i,j,2);
    end
end

% Situation Four Calculations
if gridda(i,j) == sit4
    Mp4b = Fa(i,j)*cosd30*r + Fa(i,j)*cosd30*r - Fl*r;
    % calculate force y
    forcey(i,j) = - Fa(i,j)*cosd30 - Fa(i,j)*cosd30 + Fl;
    logforcey(i,j,2) = forcey(i,j);
    % determine acc
    accy(i,j) = forcey(i,j)/m;
    logaccy(i,j,2) = accy(i,j);
    % determine vel y
    vely(i,j) = logvely(i,j,1) + accy(i,j)*delt;
```

```
    logvely(i,j,2) = vely(i,j);
    % determine pos y
    posy(i,j) = logposy(i,j,1) + logvely(i,j,1) + ...
        accy(i,j)*delt^2/2;
    logposy(i,j,2) = posy(i,j);
    % if pos y is greater than threshold, resuspend
    if Mp4b < 0
        gridda(i,j) = lift + sit4;
        respcount = respcount + 1;
        sit4count = sit4count + 1;
        resptrack(i,j) = t;
    elseif (logposy(i,j,2)-logposy(i,j,1)) > r*2
        gridda(i,j) = lift + sit4;
        respcount = respcount + 1;
        sit4count = sit4count + 1;
        resptrack(i,j) = t;
    end
end

% Determines situations for newly exposed particles
if gridda(i,j) == cantmove
    if gridda(i-1,j) < -1
        gridda(i,j) = 0;
    end
    if gridda(i-1,j-1) < -1
        gridda(i,j) = 0;
    end
end
```

```
if gridda(i,j) == 0
    if gridda(i,j-1) < 0
        if gridda(i,j+1) < 0
            gridda(i,j) = sit1;
        else
            gridda(i,j) = sit2;
        end
    else
        if gridda(i,j+1) < 0
            gridda(i,j) = sit3;
        else
            gridda(i,j) = sit4;
        end
    end
end
if gridda(i,j) > 0
    if gridda(i-1,j) > 0
        gridda(i,j) = cantmove;
    end
    if gridda(i-1,j-1) > 0
        gridda(i,j) = cantmove;
    end
end


%Log Results
loggridda(i,j,1) = gridda(i,j);
logdely(i,j,1) = logdely(i,j,2);
logvely(i,j,1) = logvely(i,j,2);
```

```
        logaccy(i,j,1) = logaccy(i,j,2);
        logforcey(i,j,1) = logforcey(i,j,2);
        logforcex(i,j,1) = logforcex(i,j,2);
        logposy(i,j,1) = logposy(i,j,2);
        V = vprint; %replace Velocity with Velocity profile

    end

end

% Logs results as a snapshot at very specific intervals, where one
% unit of time (timedis) is the amount of time it takes for the
% piston velocity to cross all of the particles, and then at 0.05,
% 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 1, 2, 3, 4, and 5 times
% that interval.
if t == round((timedis/delt*0.05))
    data0005 = [0.05 respcount sit1count sit2count...
        sit3count sit4count];
    snapshotdata(1,:) = data0005;
elseif t == round((timedis/delt*0.1))
    data0010 = [0.1 respcount sit1count sit2count...
        sit3count sit4count];
    snapshotdata(2,:) = data0010;
elseif t == round((timedis/delt*0.15))
    data0015 = [0.15 respcount sit1count sit2count...
        sit3count sit4count];
    snapshotdata(3,:) = data0015;
elseif t == round((timedis/delt*0.2))
```

```
    data0020 = [0.2 respcount sit1count sit2count...
        sit3count sit4count];
    snapshotdata(4,:) = data0020;
elseif t == round((timedis/delt*0.25))
    data0025 = [0.25 respcount sit1count sit2count...
        sit3count sit4count];
    snapshotdata(5,:) = data0025;
elseif t == round((timedis/delt*0.3))
    data0030 = [0.3 respcount sit1count sit2count...
        sit3count sit4count];
    snapshotdata(6,:) = data0030;
elseif t == round((timedis/delt*0.4))
    data0040 = [0.4 respcount sit1count sit2count...
        sit3count sit4count];
    snapshotdata(7,:) = data0040;
elseif t == round((timedis/delt*0.5))
    data0050 = [0.5 respcount sit1count sit2count...
        sit3count sit4count];
    snapshotdata(8,:) = data0050;
elseif t == round((timedis/delt))
    data0100 = [1 respcount sit1count sit2count...
        sit3count sit4count];
    snapshotdata(9,:) = data0100;
elseif t == round((timedis/delt*2))
    data0200 = [2 respcount sit1count sit2count...
        sit3count sit4count];
    snapshotdata(10,:) = data0200;
elseif t == round((timedis/delt*3))
```

```
    data0300 = [3 respcount sit1count sit2count...
        sit3count sit4count];
    snapshotdata(11,:) = data0300;
elseif t == round((timedis/delt*4))
    data0400 = [4 respcount sit1count sit2count...
        sit3count sit4count];
    snapshotdata(12,:) = data0400;
elseif t == round((timedis/delt*5))
    data0500 = [5 respcount sit1count sit2count...
        sit3count sit4count];
    snapshotdata(13,:) = data0500;
end


if plotonoff == 1
    % plot results every PLOTCOUNT # time steps
    if (rem(t,plotcount) == 0)
        if rows < 50
            partmap(loggridda(:,:,1))
            axis([-2,cols,0,rows])
        else
            partmap2(loggridda(:,:,1))
        end
        distpart = V_shock*t*delt/(2*r);
        hold on
        plot([distpart,distpart],[0,rows],'k')
        string = sprintf(...
            'frame %d, time = %10.5f (ns), shock @ %3.5f',...
            t, t*delt*(10^9),distpart);
```

```
            title(string)
            hold off
        end
    end
    if movieonoff == 1
        % Creates an avi file of each frames.
        if rows < 50
            partmap(loggridda(:,:,1))
            axis([-2,cols,0,rows])
        else
            partmap2(loggridda(:,:,1))
        end
        distpart = V_shock*t*delt/(2*r);
        hold on
        plot([distpart,distpart],[0,rows],'k')
        string = sprintf(...
            'frame %d, time = %10.5f (ns), shock @ %3.5f',...
            t, t*delt*(10^9),distpart);
        title(string)
        Frame(t) = getframe;
        hold off
        close
    end
end

% Particle calculations & total time
finalpart = partcounti-respcount;
perresp = respcount/partcounti*100;
```

```
totaltime = delt*tindex;


if plotonoff == 1
    if rows < 50
        partmap(loggridda(:,:,1))
        axis([-2,cols,0,rows])
    else
        partmap2(loggridda(:,:,1))
    end
    title('Final')
end


if movieonoff == 1
    movie(Frame)
    movie2avi(Frame,'resuspension.avi')
end


if plotonoff == 1
    % Report of Model Results
    disp('Summary')
    fprintf('The initial value for the rows: %d. \n', rowsstore)
    fprintf('The initial value for the columns: %d. \n', colsstore)
    fprintf('The final value for the rows: %d. \n', rows)
    fprintf('The final value for the columns: %d. \n', cols)
    fprintf('The initial number of particles: %d. \n', partcounti)
    fprintf('The final number of particles: %d. \n', finalpart)
    fprintf('The total number of resuspended particles: %d. \n',...
        respcount)
```

```
    fprintf(...
        'Number of Situation One resuspended particles: %d. \n', ...
        sit1count)
    fprintf(...
        'Number of Situation Two resuspended particles: %d. \n', ...
        sit2count)
    fprintf(...
        'Number of Situation Three resuspended particles: %d. \n', ...
        sit3count)
    fprintf(...
        'Number of Situation Four resuspended particles: %d. \n', ...
        sit4count)
    fprintf(...
        'Percentage of particles resuspended: %3.3f percent. \n', ...
        perresp)
    fprintf('The time elapsed: %2.13f seconds. \n', totaltime)
end
end
```

A.6 – PartBatch Run

```
function [snapshotdatafull, perall, meanall, maxall, minall, ...
    rangeall, stdall, partcountfull, finalpartfull] = ...
    PartBatchRun(runcount, rows, cols, timeselect, plotonoff, ...
    movieonoff, attf);

% PARTBATCHRUN performs multiple runs of the program PARTSIMCOMPLEX.
% Inputs are RUNCOUNT (the number of times PARTSIMCOMPLEX will be run),
% the number of ROWS and COLS for the simulation, the TIMESELECT (how
% long the simulation runs based on the piston velocity following the
% shock), PLOTONOFF and MOVIEONOFF indicators (tell whether or not
% plots or movies are generated) and ATTF (which determines the
% attractive force used). Outputs are SNAPSHOTDATAFULL (tracks the
% particle resuspension at  various specific times for all runs).
% There are also numerous outputs  of various runs at specific times.
% These outputs are PERRESP (the percent resuspended), MEANALL (the
% mean of resuspended particles), MAXALL (the maximum resuspended),
% MINALL (the minimum resuspended), STDALL (the standard deviation of
% the particle resuspension data,  PARTCOUNTFULL (the number of
% particles in each run), and FINALPARTFULL (the number of particles
% remaining at the end).

%% Initialize Values for Speed
partcountfull = zeros(runcount,1);
finalpartfull = partcountfull;
colsfull = partcountfull;
rowsfull = partcountfull;
```

```
totaltimefull = partcountfull;
colsave = cols;
rowsave = rows;

%% Running the Simulation RUNCOUNT # of times

for i = 1:runcount
    runn = i % Counts the number of runs for visual tracking.
    [snapshotdata, partcounti, finalpart, cols1, rows1, totaltime]...
        = PartSimComplex(rows, cols, timeselect, plotonoff, ...
        movieonoff, attf);
    snapshotdatafull(:,:,i) = snapshotdata(:,:); % Logs all data.
    partcountfull(i,1) = partcounti; % Logs initial particle count.
    finalpartfull(i,1) = finalpart; % Logs final particle count.
    colsfull(i,1) = cols1; % Logs colomns dimension of particles.
    rowsfull(i,1) = rows1; % Logs rows dimension of particles.
    totaltimefull(i,1) = totaltime; % Logs time of simulation.
    cols = colsave; % Resets Colomns for next iteration.
    rows = rowsave; % Resets Rows for next iteration.
end


%% Statistical Analysis, Initializing Matricies

xvals = snapshotdata(:,1,1);
resusp(:,:) = snapshotdatafull(:,2,:);
sit1co(:,:) = snapshotdatafull(:,3,:);
```

```
sit2co(:,:) = snapshotdatafull(:,4,:);

sit3co(:,:) = snapshotdatafull(:,5,:);

sit4co(:,:) = snapshotdatafull(:,6,:);


dimcount = size(resusp);


perresp = zeros(dimcount);

persit1 = perresp;

persit2 = perresp;

persit3 = perresp;

persit4 = perresp;


meanresp = zeros(dimcount(1),1);

meansit1 = meanresp;

meansit2 = meanresp;

meansit3 = meanresp;

meansit4 = meanresp;


maxresp = zeros(dimcount(1),1);

maxsit1 = maxresp;

maxsit2 = maxresp;

maxsit3 = maxresp;

maxsit4 = maxresp;

minresp = maxresp;

minsit1 = maxresp;

minsit2 = maxresp;

minsit3 = maxresp;
```

```
minsit4 = maxresp;


stdresp = zeros(dimcount(1),1);
stdsit1 = stdresp;
stdsit2 = stdresp;
stdsit3 = stdresp;
stdsit4 = stdresp;


%% Calculate Percentage


% Calculates Percentage Resuspended
for i = 1:runcount
    perresp(:,i) = resusp(:,i)/partcountfull(i,1)*100;
end


% Calculates Percentage of Resuspended that are Situation Ones
for i = 1:runcount
    for j = 1:dimcount(1)
        persit1(j,i) = sit1co(j,i)./resusp(j,i)*100;
        if resusp(j,i) == 0
            persit1(j,i) = 0;
        end
    end
end


% Calculates Percentage of Resuspended that are Situation Twos
for i = 1:runcount
```

```
    for j = 1:dimcount(1)
        persit2(j,i) = sit2co(j,i)./resusp(j,i)*100;
        if resusp(j,i) == 0
            persit2(j,i) = 0;
        end
    end
end

% Calculates Percentage of Resuspended that are Situation Threes
for i = 1:runcount
    for j = 1:dimcount(1)
        persit3(j,i) = sit3co(j,i)./resusp(j,i)*100;
        if resusp(j,i) == 0
            persit3(j,i) = 0;
        end
    end
end

% Calculates Percentage of Resuspended that are Situation Fours
for i = 1:runcount
    for j = 1:dimcount(1)
        persit4(j,i) = sit4co(j,i)./resusp(j,i)*100;
        if resusp(j,i) == 0
            persit4(j,i) = 0;
        end
    end
end
```

```
% Saves Results for Output.
perall(:,:,1) = perresp;
perall(:,:,2) = persit1;
perall(:,:,3) = persit2;
perall(:,:,4) = persit3;
perall(:,:,5) = persit4;

%% Averages

% Calculates the Average Percentage Resuspended.
for i = 1:dimcount(1)
    meanresp(i,1) = mean(perresp(i,:));
end

% Calculates the Average Percentage of Resuspended particles that are
% Situation Ones.
for i = 1:dimcount(1)
    meansit1(i,1) = mean(persit1(i,:));
end

% Calculates the Average Percentage of Resuspended particles that are
% Situation Twos.
for i = 1:dimcount(1)
    meansit2(i,1) = mean(persit2(i,:));
end

% Calculates the Average Percentage of Resuspended particles that are
```

```
% Situation Threes.
for i = 1:dimcount(1)
    meansit3(i,1) = mean(persit3(i,:));
end

% Calculates the Average Percentage of Resuspended particles that are
% Situation Fours.
for i = 1:dimcount(1)
    meansit4(i,1) = mean(persit4(i,:));
end

% Saves Results for Output.
meanall = [meanresp meansit1 meansit2 meansit3 meansit4];

%% Min Max and Range

% Determines the Min, Max and Range of Values for Percent Resuspended.
for i = 1:dimcount(1)
    maxresp(i,1) = max(perresp(i,:));
    minresp(i,1) = min(perresp(i,:));
end
rangeresp = maxresp - minresp;

% Determines the Min, Max and Range of Values for the Percentage of
% Resuspended values that are Situation Ones.
for i = 1:dimcount(1)
    maxsit1(i,1) = max(persit1(i,:));
```

```
    minsit1(i,1) = min(persit1(i,:));
end
rangesit1 = maxsit1 - minsit1;

% Determines the Min, Max and Range of Values for the Percentage of
% Resuspended values that are Situation Twos.
for i = 1:dimcount(1)
    maxsit2(i,1) = max(persit2(i,:));
    minsit2(i,1) = min(persit2(i,:));
end
rangesit2 = maxsit2 - minsit2;

% Determines the Min, Max and Range of Values for the Percentage of
% Resuspended values that are Situation Threes.
for i = 1:dimcount(1)
    maxsit3(i,1) = max(persit3(i,:));
    minsit3(i,1) = min(persit3(i,:));
end
rangesit3 = maxsit3-minsit3;

% Determines the Min, Max and Range of Values for the Percentage of
% Resuspended values that are Situation Fours.
for i = 1:dimcount(1)
    maxsit4(i,1) = max(persit4(i,:));
    minsit4(i,1) = min(persit4(i,:));
end
rangesit4 = maxsit4-minsit4;
```

```
% Saves Results for Output.
maxall = [maxresp maxsit1 maxsit2 maxsit3 maxsit4];
minall = [minresp minsit1 minsit2 minsit3 minsit4];
rangeall = [rangeresp rangesit1 rangesit2 rangesit3 rangesit4];

%% Standard Deviation

% Calculates the Standard Deviation of the Percentage Resuspended, as
% well as the +/- One Standard Deviation frpm the mean.
for i = 1:dimcount(1)
    stdresp(i,1) = std(perresp(i,:));
end
std1resp = meanresp + stdresp;
stdn1resp = meanresp - stdresp;

% Calculates the Standard Deviation for the Percentage of Resuspended
% values that are Situation Ones, as well as the +/- One Standard
% Deviation from the mean.
for i = 1:dimcount(1)
    stdsit1(i,1) = std(persit1(i,:));
end
std1sit1 = meansit1 + stdsit1;
stdn1sit1 = meansit1 - stdsit1;

% Calculates the Standard Deviation for the Percentage of Resuspended
% values that are Situation Twos, as well as the +/- One Standard
```

```
% Deviation from the mean.
for i = 1:dimcount(1)
    stdsit2(i,1) = std(persit2(i,:));
end
std1sit2 = meansit2 + stdsit2;
stdn1sit2 = meansit2 - stdsit2;

% Calculates the Standard Deviation for the Percentage of Resuspended
% values that are Situation Threes, as well as the +/- One Standard
% Deviation from the mean.
for i = 1:dimcount(1)
    stdsit3(i,1) = std(persit3(i,:));
end
std1sit3 = meansit3 + stdsit3;
stdn1sit3 = meansit3 - stdsit3;

% Calculates the Standard Deviation for the Percentage of Resuspended
% values that are Situation Fours, as well as the +/- One Standard
% Deviation from the mean.
for i = 1:dimcount(1)
    stdsit4(i,1) = std(persit4(i,:));
end
std1sit4 = meansit4 + stdsit4;
stdn1sit4 = meansit4 - stdsit4;

% Saves Results for Output.
stdalltemp = [stdresp stdsit1 stdsit2 stdsit3 stdsit4];
std1all = [std1resp std1sit1 std1sit2 std1sit3 std1sit4];
```

```
stdn1all = [stdn1resp stdn1sit1 stdn1sit2 stdn1sit3 stdn1sit4];
stdall(:,:,1) = stdalltemp;
stdall(:,:,2) = std1all;
stdall(:,:,3) = stdn1all;

%% Plots Zoomed Scaling

% Plots mean Percent Resuspended, as well as +/- One Standard
% Deviation.
figure
bar(xvals,meanresp)
hold on
plot(xvals,std1resp,'-ro')
plot(xvals,stdn1resp,'-ko')
title('% Particle Resuspension vs Dimensionless Time')
legend('Mean', 'Std+1', 'Std-1')
xlabel('t/Tau')
ylabel('% Resuspended')
%axis([0, max(xvals)*1.2, 0, min(100,max(std1resp)*1.1)])
hold off

% Plots the Percentage of Resuspended Values that are Situation Ones,
% as well as +/- One Standard Deviation.
figure
bar(xvals,meansit1)
hold on
plot(xvals,std1sit1,'-ro')
plot(xvals,stdn1sit1,'-ko')
```

```
title('% Situation One Particle Resuspension vs Dimensionless Time')
legend('Mean', 'Std+1', 'Std-1')
xlabel('t/Tau')
ylabel('% Resuspended')
%axis([0, max(xvals)*1.2, 0, min(100,max(std1sit1)*1.1)])
hold off

% Plots the Percentage of Resuspended Values that are Situation Twos,
% as well as +/- One Standard Deviation.
figure
bar(xvals,meansit2)
hold on
plot(xvals,std1sit2,'-ro')
plot(xvals,stdn1sit2,'-ko')
title('% Situation Two Particle Resuspension vs Dimensionless Time')
legend('Mean', 'Std+1', 'Std-1')
xlabel('t/Tau')
ylabel('% Resuspended')
%axis([0, max(xvals)*1.2, 0, min(100,max(std1sit2)*1.1)])
hold off

% Plots the Percentage of Resuspended Values that are Situation Threes,
% as well as +/- One Standard Deviation.
figure
bar(xvals,meansit3)
hold on
plot(xvals,std1sit3,'-ro')
plot(xvals,stdn1sit3,'-ko')
```

```
title('% Situation Three Particle Resuspension vs Dimensionless Time')
legend('Mean', 'Std+1', 'Std-1')
xlabel('t/Tau')
ylabel('% Resuspended')
%axis([0, max(xvals)*1.2, 0, min(100, max(std1sit3)*1.1)])
hold off

% Plots the Percentage of Resuspended Values that are Situation Fours,
% as well as +/- One Standard Deviation.
figure
bar(xvals,meansit4)
hold on
plot(xvals,std1sit4,'-ro')
plot(xvals,stdn1sit4,'-ko')
title('% Situation Four Particle Resuspension vs Dimensionless Time')
legend('Mean', 'Std+1', 'Std-1')
xlabel('t/Tau')
ylabel('% Resuspended')
%axis([0, max(xvals)*1.2, 0, min(100, max(std1sit4)*1.1)])
hold off

% PLots the Percentage Resuspended for All Situations against one
% another.
figure
plot(xvals,meansit1, '-ro')
hold on
plot(xvals,meansit2, '-bo')
plot(xvals,meansit3, '-mo')
```

```
plot(xvals,meansit4, '-ko')

title('% of All Situations Resuspended vs Dimensionless Time')

legend('Situation 1', 'Situation 2', 'Situation 3', 'Situation 4')

xlabel('t/Tau')

ylabel('% Resuspended')

%axis([0, max(xvals)*1.2, 0, min(100,max(meanall)*1.1)])

hold off


%% Plots 100% Scaling, same as above, scaling from 0 - 100% on Y AXIS


% Plots mean Percent Resuspended, as well as +/- One Standard
Deviation.

figure

bar(xvals,meanresp)

hold on

plot(xvals,std1resp,'-ro')

plot(xvals,stdn1resp,'-ko')

title('% Particle Resuspension vs Dimensionless Time')

legend('Mean', 'Std+1', 'Std-1')

xlabel('t/Tau')

ylabel('% Resuspended')

axis([0,max(xvals)*1.2,0,100])

hold off


% Plots the Percentage of Resuspended Values that are Situation Ones,

% as well as +/- One Standard Deviation.

figure

bar(xvals,meansit1)
```

```
hold on
plot(xvals,std1sit1,'-ro')
plot(xvals,stdn1sit1,'-ko')
title('% Situation One Particle Resuspension vs Dimensionless Time')
legend('Mean', 'Std+1', 'Std-1')
xlabel('t/Tau')
ylabel('% Resuspended')
axis([0,max(xvals)*1.2,0,100])
hold off

% Plots the Percentage of Resuspended Values that are Situation Twos,
% as well as +/- One Standard Deviation.
figure
bar(xvals,meansit2)
hold on
plot(xvals,std1sit2,'-ro')
plot(xvals,stdn1sit2,'-ko')
title('% Situation Two Particle Resuspension vs Dimensionless Time')
legend('Mean', 'Std+1', 'Std-1')
xlabel('t/Tau')
ylabel('% Resuspended')
axis([0,max(xvals)*1.2,0,100])
hold off

% Plots the Percentage of Resuspended Values that are Situation Threes,
% as well as +/- One Standard Deviation.
figure
bar(xvals,meansit3)
```

```
hold on
plot(xvals,std1sit3,'-ro')
plot(xvals,stdn1sit3,'-ko')
title('% Situation Three Particle Resuspension vs Dimensionless Time')
legend('Mean', 'Std+1', 'Std-1')
xlabel('t/Tau')
ylabel('% Resuspended')
axis([0,max(xvals)*1.2,0,100])
hold off

% Plots the Percentage of Resuspended Values that are Situation Fours,
% as well as +/- One Standard Deviation.
figure
bar(xvals,meansit4)
hold on
plot(xvals,std1sit4,'-ro')
plot(xvals,stdn1sit4,'-ko')
title('% Situation Four Particle Resuspension vs Dimensionless Time')
legend('Mean', 'Std+1', 'Std-1')
xlabel('t/Tau')
ylabel('% Resuspended')
axis([0,max(xvals)*1.2,0,100])
hold off

% PLots the Percentage Resuspended for All Situations against one
% another.
figure
plot(xvals,meansit1, '-ro')
```

```
hold on

plot(xvals,meansit2, '-bo')

plot(xvals,meansit3, '-mo')

plot(xvals,meansit4, '-ko')

title('% of All Situations Resuspended vs Dimensionless Time')

legend('Situation 1', 'Situation 2', 'Situation 3', 'Situation 4')

xlabel('t/Tau')

ylabel('% Resuspended')

axis([0,max(xvals)*1.2,0,100])

hold off


end
```

A.7 – ParticleResuspension

```
clear all
close all
clc

% Input the number of columns, and checks that the column input is an
% integer number.
colscheck = input(...
    'Input the number of columns as a positive integer: ');
while colscheck ~= round(colscheck) || colscheck < 0
    disp(...
        'ERROR! The number of columns MUST be a positive integer.')
    colscheck = input('Input the number of colomns: ');
end
cols = colscheck;

% Input the number of rows, and checks that the row input is an integer
% number.
disp(' ')
rowscheck = input('Input the number of rows as a positive integer: ');
while rowscheck ~= round(rowscheck) || rowscheck < 0
    disp('ERROR! The number of rows MUST be a positive integer.')
    rowscheck = input('Input the number of rows: ');
end
rows = rowscheck;

% Input the time scaling, and verifies that the time scaling is one of
```

```
% the accepted values for time scaling.

disp(' ')

disp('For the dimensionless time, tau, 1 is equal to the length of ')

disp('time it takes for the piston velocity to cross the particles.')

disp('Most resuspension occurs before tau = 1. The accepted values ')

disp('are 0.05, 0.1,0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 1, 2, 3, 4, and
5');

disp(' ')

timeselectcheck = input('Please enter one of the accepted values: ');

while timeselectcheck ~= 0.05 && timeselectcheck ~= 0.1 && ...

        timeselectcheck ~= 0.15 && timeselectcheck ~= 0.2 && ...

        timeselectcheck ~= 0.25 && timeselectcheck ~= 0.3 && ...

        timeselectcheck ~= 0.4  && timeselectcheck ~= 0.5 && ...

        timeselectcheck ~= 1 && timeselectcheck ~= 2 && ...

        timeselectcheck ~= 3 && timeselectcheck ~= 4 && ...

        timeselectcheck ~= 5 && timeselectcheck ~=10 && ...

        timeselectcheck ~= 15

    disp('ERROR! Tau MUST be one of the approved values.')

    timeselectcheck = input('Please enter an accepted value: ');

end

timeselect = timeselectcheck;


% Input the Attractive force to be used for the simulation and verifies

% it is one of the accepted values

disp(' ')

disp('This program allows four different random attractive forces to ')

disp('be used for simulation:')

disp('(1) is for CH3, 0-1nN')
```

```
disp('(2) is for COOH, 4-6nN')

disp('(3) is for NH2, 15-25nN')

disp('(4) is for testing, 1-2nN')

attf = input(...

    'Please enter one of the values for attractive force, 1-4: ');

while attf ~=1 && attf ~= 2 && attf ~= 3 && attf ~= 4

    disp('ERROR! Please enter a value 1-4')

    attf = input('Please enter an accepted value: ');

end



% Input whether or not this is a plotting run.

disp(' ')

disp('Is this a plotting run, or a data run?')

disp('A plotting run will generate one run that displays a series of')

disp('images of the particles in the layout as they resuspend. A data
')

disp('run will do multiple runs of randomly generated particles and')

disp('attractive forces, and look at the data at verious time scales.')

disp('1 = plotting run, 0 = data run')

disp(' ')

plotonoffcheck = input('Would you like this to be a plotting run: ');

while plotonoffcheck ~= 1 && plotonoffcheck ~=0

    disp('ERROR! The value MUST be either 1 or 0.')

    plotonoffcheck = input('Would you like to plot? Enter 1 or 0:  ');

end

plotonoff = plotonoffcheck;
```

```
% Input whether or not this is a movie run.

disp(' ')

disp('Do you want to generate a movie? This will automatically turn ')

disp('this into a plotting run. However it will generate one frame for
')

disp('every time step, so this is a rather time consuming process.')

disp(' ')

movieonoffcheck = input('1 = movie creation, 0 = no movie: ');

while movieonoffcheck ~=1 && movieonoffcheck ~= 0

    disp('ERROR! The value MUST be either 1 or 0.')

    movieonoffcheck = input('Would you like this to be a movie run: ');

end

movieonoff = movieonoffcheck;



% Verify that a movie run is what is required.

disp(' ')

if movieonoff == 1

    disp('Are you sure? This can generate upwards of 15,000 frames, ')

    disp('which is both time and memory consuming.')

    disp(' ')

    movieonoff = input('Reaffirm by entering 1 now: ');

    if movieonoff ~= 1

        movieonoff = 0;

    end

end

if movieonoff == 1

    plotonoff = 1;
```

```
end

if plotonoff == 1
    [snapshotdata, partcounti, finalpart, cols, rows, totaltime] = ...
        PartSimComplex(rows, cols, timeselect, plotonoff, movieonoff);
end

if movieonoff == 1
    disp(' ')
    disp('The movie file is an automatically generated .avi format ')
    disp('file. It will be called "resuspension.avi" and will save to')
    disp('the current directory when it is finished.')
end

if plotonoff == 0
    disp(' ')
    disp('For a batch run, the code will run repeatedly. The number')
    disp('of runs is selected here. A set of 100 runs seems to be ')
    disp('adequete for data analysis. This must be an integer.')
    disp(' ')
    runcountcheck = input('Input the number of runs: ');
    while runcountcheck ~= round(runcountcheck) || runcountcheck < 0
        disp('ERROR! The number of runs MUST be a positive integer.')
        runcountcheck = input('Input the number of runs: ');
    end
    runcount = runcountcheck;
    [snapshotdatafull, perresp, meanall, maxall, minall, rangeall, ...
        stdall, partcountfull, finalpartfull] = PartBatchRun(...
```

```
        runcount,rows, cols, timeselect, plotonoff, movieonoff,attf);
    disp('Runs Complete')
end
```

Appendix B – Results for Mach 1.2

B.1 – 12x52 CH3

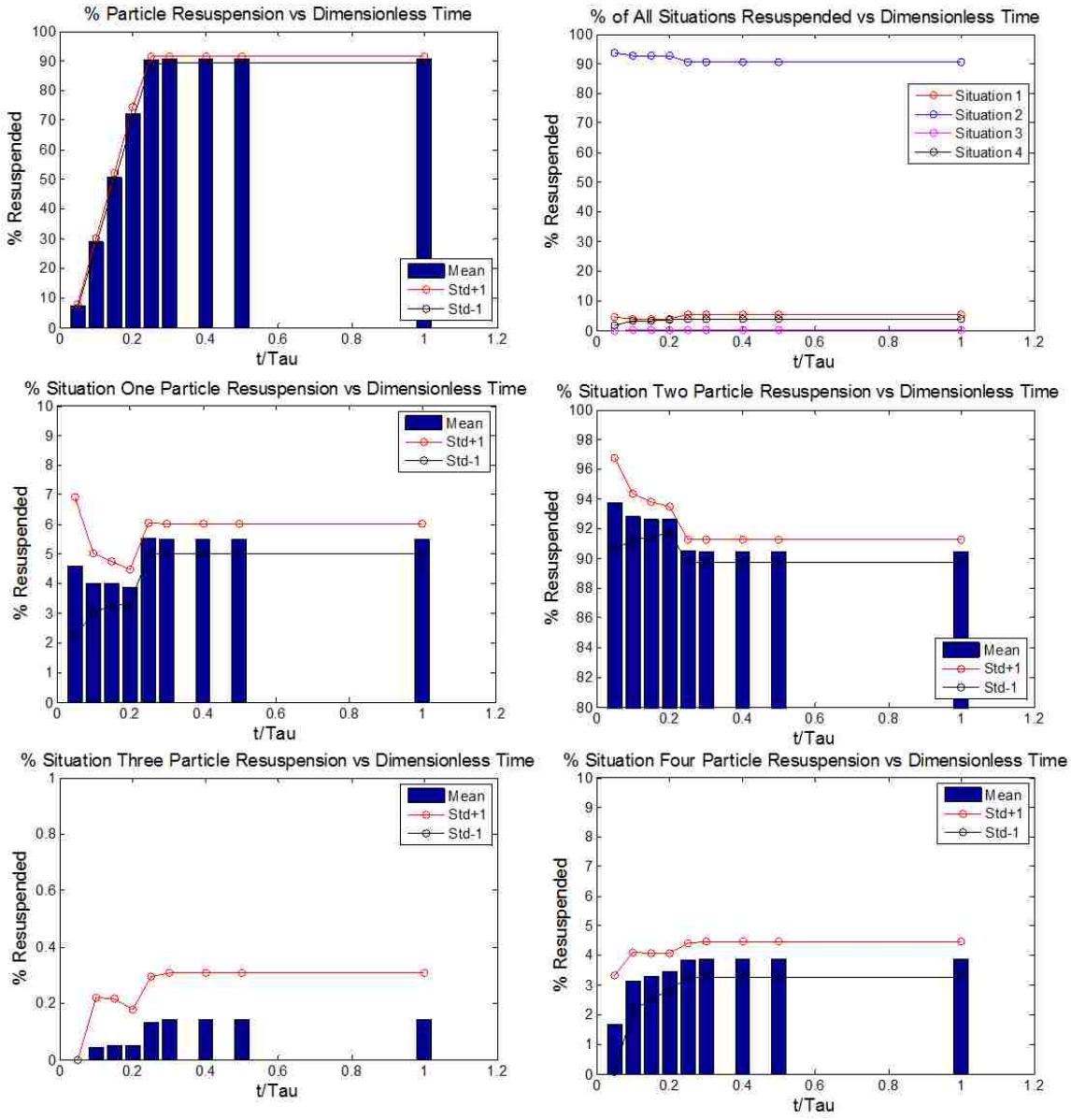


Figure B1 – Percent particle resuspension vs dimensionless time for CH3 with 12x52 grid size, using the piston flow follwing a Mach 1.2 shock. Note they are scaled for clarity.

B.2 – 22x102 CH3

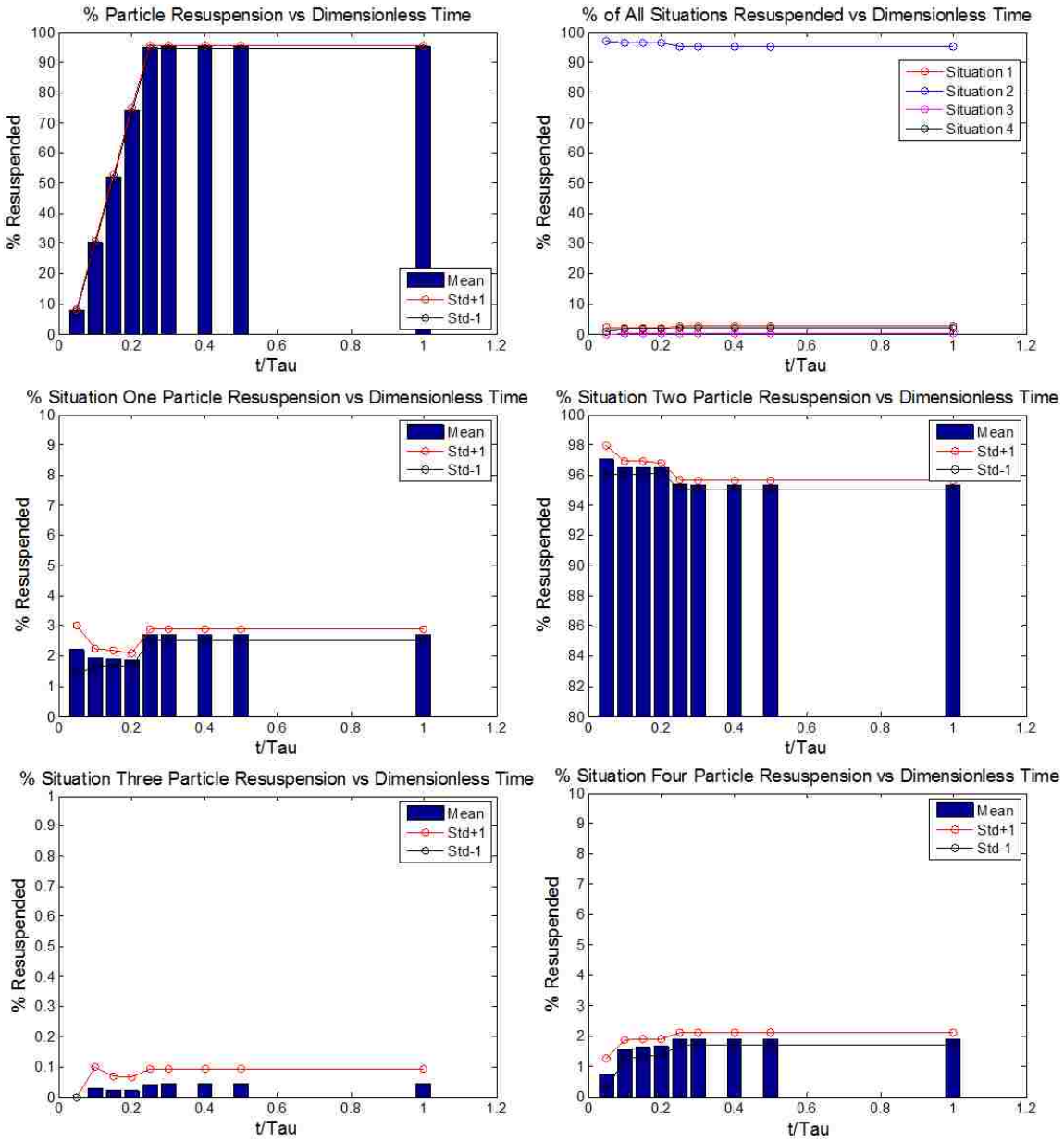


Figure B2 – Percent particle resuspension vs dimensionless time for CH3 with 22x102 grid size using the piston flow follwing a Mach 1.2 shock. Note they are scaled for clarity.

B.3 – 22x102 COOH

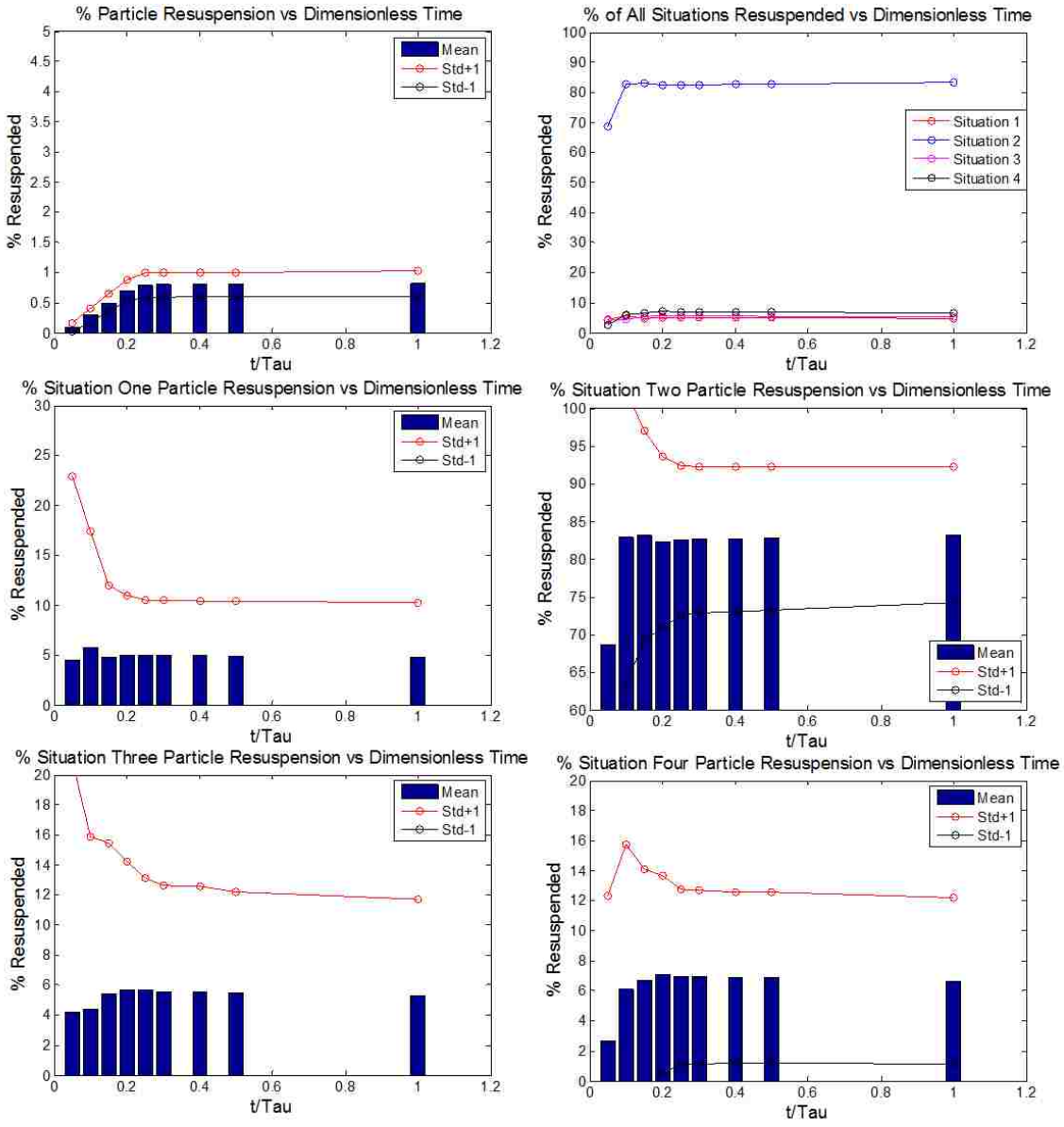


Figure B3 – Percent particle resuspension vs dimensionless time for COOH with 22x102 grid size using the piston flow follwing a Mach 1.2 shock Note they are scaled for clarity.

B.4 – 32x92 CH3

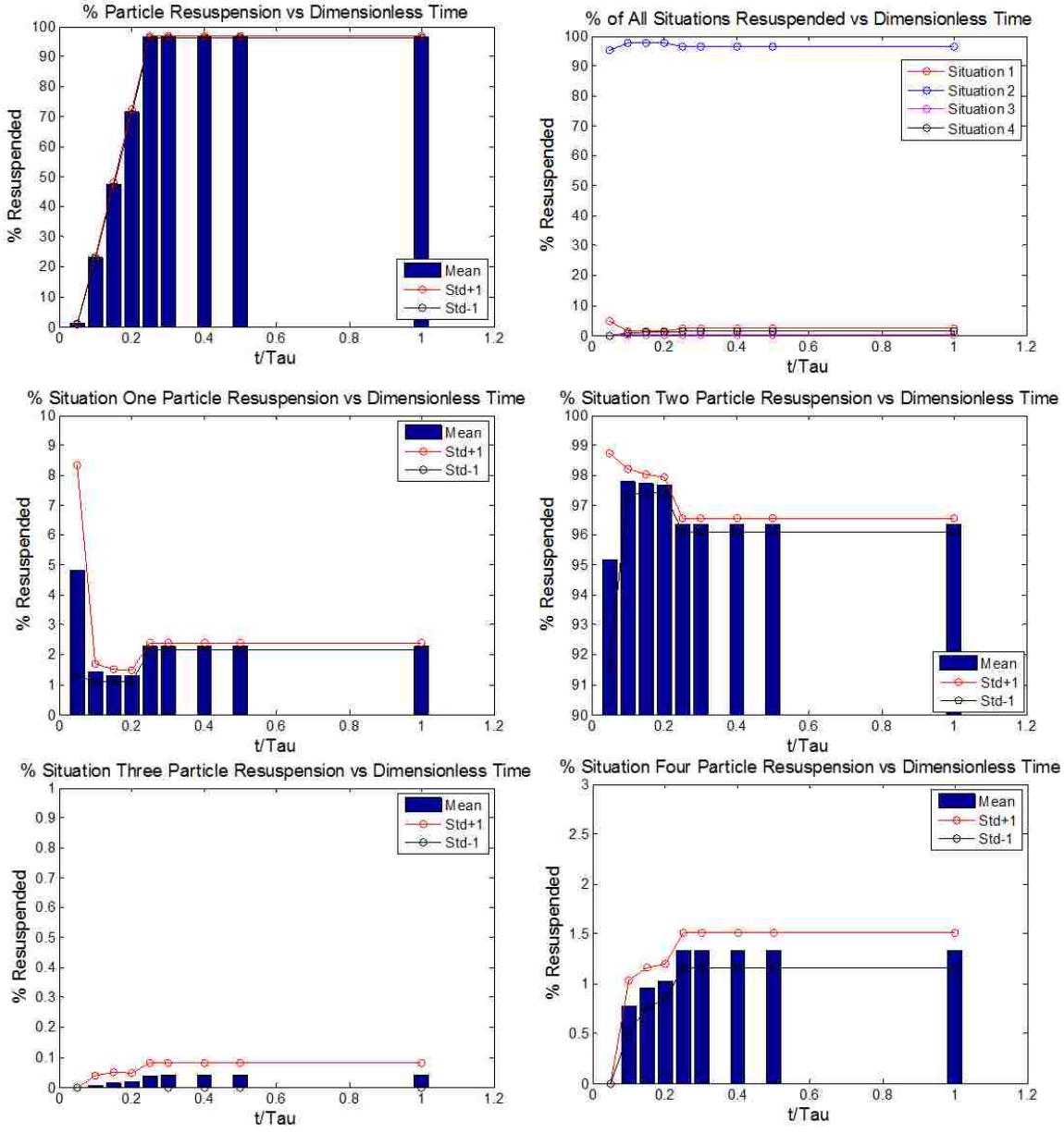


Figure B4 – Percent particle resuspension vs dimensionless time for CH3 with 32x92 grid size using the piston flow follwing a Mach 1.2 shock. Note they are scaled for clarity.

B.5 – 32x92 COOH

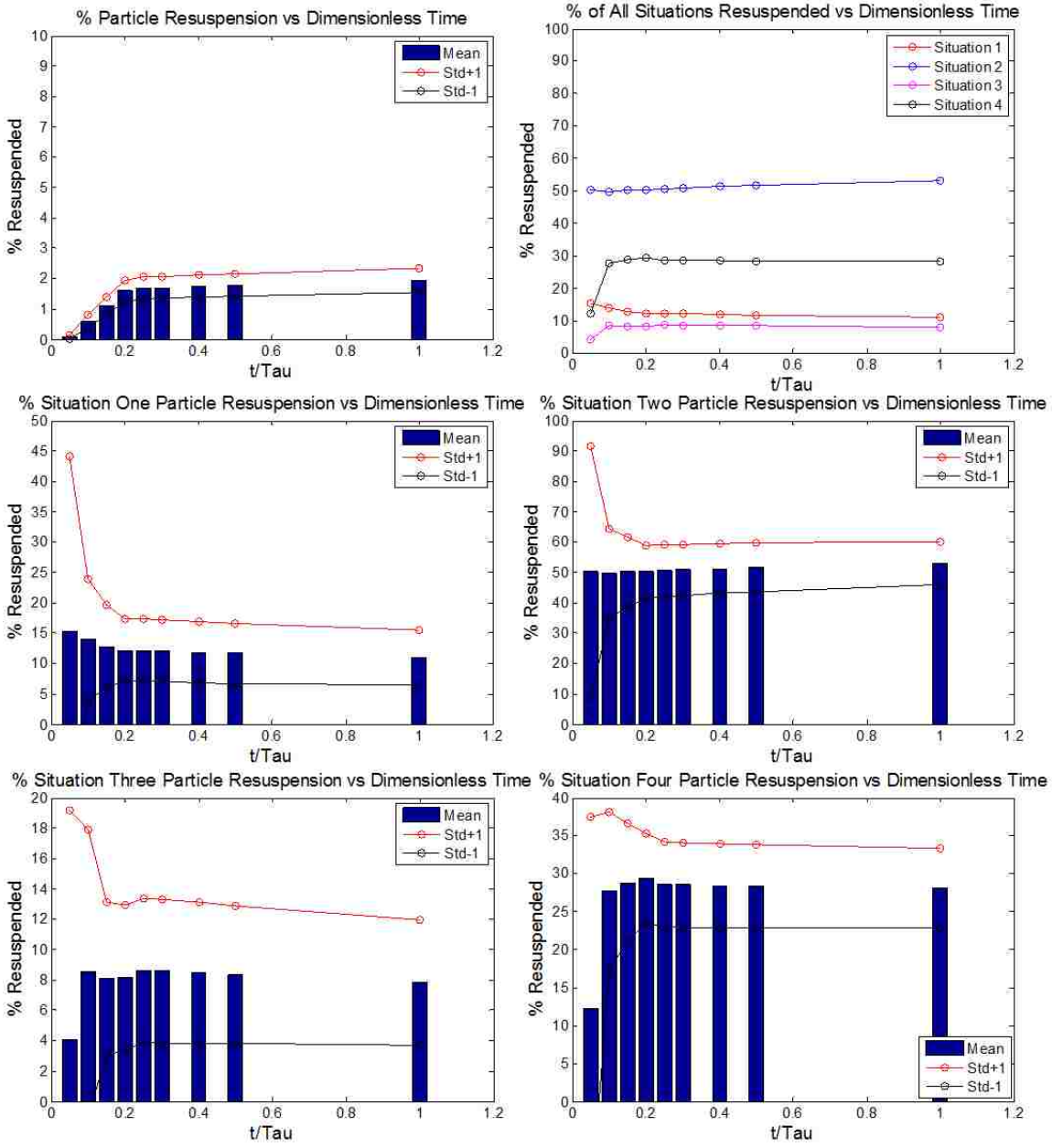


Figure B5 – Percent particle resuspension vs dimensionless time for COOH with 32x92 grid size using the piston flow follwing a Mach 1.2 shock. Note they are scaled for clarity.