Fall 2010

Artificial neural network for behavior learning from meso-scale simulations, application to multi-scale multimaterial flows

Christopher Lu
University of Iowa

ARTIFICIAL NEURAL NETWORK FOR BEHAVIOR LEARNING FROM MESO-SCALE SIMULATIONS, APPLICATION TO MULTI-SCALE MULTIMATERIAL FLOWS

by

Christopher Lu

A thesis submitted in partial fulfillment
of the requirements for the Master of Science
degree in Mechanical Engineering
in the Graduate College of
The University of Iowa

December 2010

Thesis Supervisor: Professor H.S. Udaykumar

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

____

MASTER'S THESIS

____

This is to certify that the Master's thesis of

Christopher Lu

has been approved by the Examining Committee for the thesis requirement for the Master of Science degree in Mechanical Engineering at the December 2010 graduation.

Thesis committee: ___________________________________
H.S. Udaykumar, Thesis Supervisor

____
Christoph Beckerman

____
Pablo Carrica

ACKNOWLEDGEMENTS

I would first like to thank my advisor Professor H.S. Udaykumar for his help and guidance. I also would like to thank my lab mates for providing me answers to countless questions which saved me hundreds of hours of literature search.

I need to express my gratitude to my parents and friends who have provided comfort and encouragement throughout my education. They have stood beside me aiding my academic and personal growth as well as encouraging me to pursue and attain goals I would have otherwise been unable to reach.

I would also like to acknowledge the support of the Mechanical Engineering Department, the College of Engineering at the University of Iowa and the Eglin Air Force Research Laboratory for supplying the materials and resources to make this possible.

ABSTRACT

This thesis attempts to develop a framework to affect such a coupling of scales by "learning" from selected computational experiments at the meso-scale and transmitting the "learned" behavior to the macro-scale. The "learning" is performed by means of an artificial neural network that is trained using data extracted from the meso-scale direct numerical simulations. In particular, this thesis describes the use of an Artificial Neural Network (hereafter abbreviated to ANN), to learn and predict the transient forces on a particle in a compressible flow field to produce an accurate model for shocked particulate-laden flows. In the multi-scale sense, the ANN learns meso-scale information of particle-fluid interactions requiring expensive computations; once the behavior is learnt, the ANN can be interrogated to obtain information by a macro-scale model to accurately produce results without continuing to perform expensive computations in direct numerical simulations. Particle data is collected from a compressible Eulerian-Lagrangian solver and provided to the ANN for a range of control parameters, such as Mach number, particle radii, particle-fluid density ratio, position, and volume fraction. Beginning with a simple single stationary particle case and progressing to moving particle laden clouds, the ANN is able to evolve and reproduce correlations between the control parameters and particle dynamics. The trained ANN is then used in computing the macro-scale flow behavior in a model of shocked dusty gas advection. The model predicts particle motion and other macro-scale phenomena in agreement with experimental observations and with a very large reduction in time and computational expense.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I:

INTRODUCTION

Background

In phenomena involving high-speed flows in multiphase materials, such as in dust explosions, condensation shocks, explosive debris transport, detonation in heterogenous media and a host of other phenomena, there are complex interactions that occur between propagating shocks and rarefaction waves, including carrier fluid-particle interactions and particle-particle interactions.(1)(2) Such flows are very difficult to visualize (due to the wide range of length scales and short time scales involved) and experimental measurements are difficult and expensive to obtain.(3) Therefore, to understand what happens in such violent environments requires the development of accurate computational models.

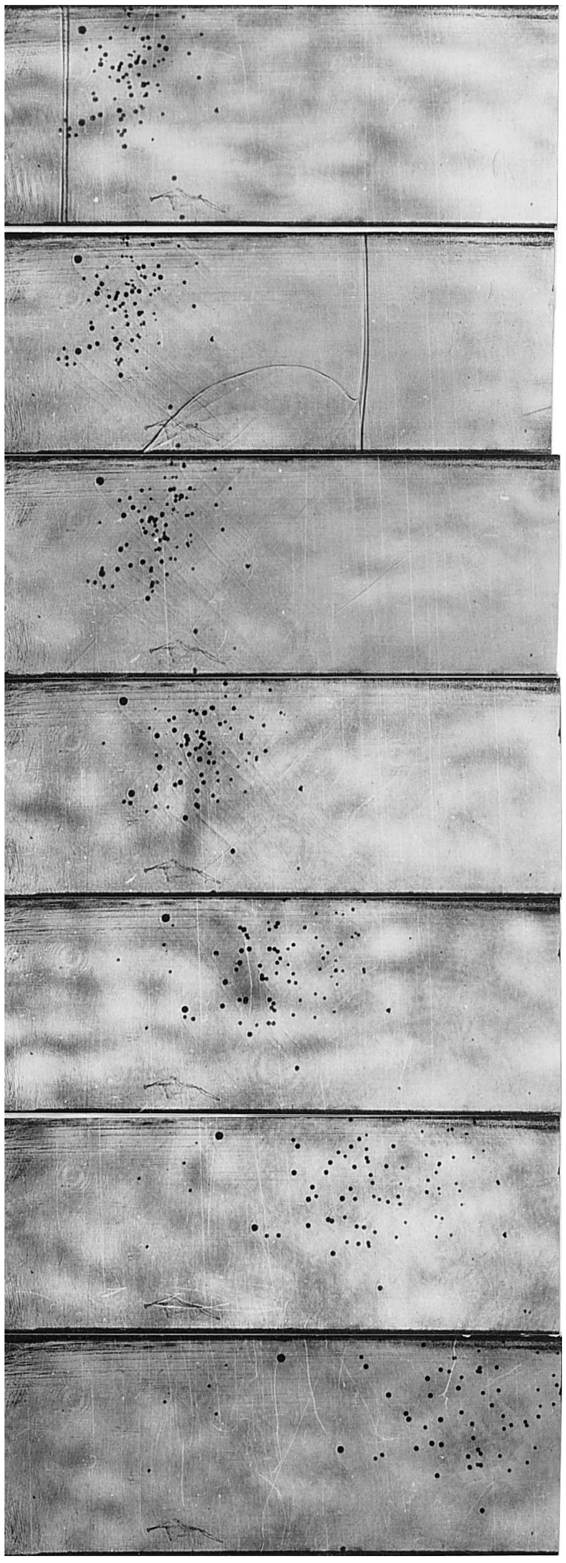
Computational speeds in fluid calculation have increased tremendously over the past few decades, along with advances in modeling and calculating complex multi-phase flows. (4) However, accurate modeling of complex multimaterial flows still presents stiff challenges due to at least the following reasons:

1. The full description of compressible multimaterial flows requires models to embed the interactions between the fluid and particles, and between particles themselves (5). Such models are empirical in nature. As mentioned above experimental investigation of shocked particle laden flow are rather challenging and the information derived from such experiment cover limited parameter ranges.

2. Even where experimental data for such flows are available, the manner in which the behavior of a mixture in described in a continuum setting can lead to loss of important physics.(6)
3. Resolution of transient shocked flows in itself demands rather heavy computational resources. Since the length scales of the discrete particles in a multi-material system and the time scales of response of the particulate phases may be vastly different from that of the bulk flow, resolving the dynamics of the individual components of the mixture is impossible. Therefore some overall (averaged or homogenized) behavior of the multi-material mixture needs to be modeled and computed.(7) While such averaged material representations may be sufficient for many engineering applications, there are some physical problems where the local behavior of the material, i.e. the detailed interactions between the (unresolved) individual phases in the mixture can become important and can influence the observed global dynamics. Examples of such sub-grid phenomena that can manifest at large scales and completely dominate the overall material behavior include: a) material failure/fracture/spall in ductile materials, where cracks can develop at grain boundaries and progress along specific defects in the material leading to the change of the overall (i.e. large scale) material response;(8) b) deflagration to detonation transition in a heterogeneous explosive, where detonation initiation is thought to occur at voids in the material or due to inter-granular interactions upon passage of a compression wave in the material;(9) or c)

particle-particle interactions and particle-carrier medium interactions in dusty gas flow leading to changes in the observed macroscopic behavior.(10)

An example in the last category is shown in Figure 1 and Figure 2.(1) The experiments from which the figure is obtained were performed by Boiko et al.(1) Here a cloud of particles (polystyrene, average particle diameter d_p of 80 microns) is hit by a shock wave (traveling from left to right) in a shock tube. The shock wave was initiated by a driver chamber while particles were thrown up into the chamber by an electromagnetic propelling device. The images and tracking data was obtained from a fast-acting laser visualization method known as multiframe shadow visualization.(1) The overall behavior of the particles subjected to the shock is very interesting; in particular, for the high particle volume fraction case the particle distribution assumes a triangular form as shown in Figure 2. The reason for formation of the triangular structure in the case of the heavily loaded mixture case (while the low particle loading does not produce a distinct structure) must hinge upon the interactions between the more densely packed particles. In order to reproduce the observed macro-scale distribution of the particles, the effects of the micro-scale interactions between the particles must be placed in a macro-scale description of the shocked particle-laden fluid.



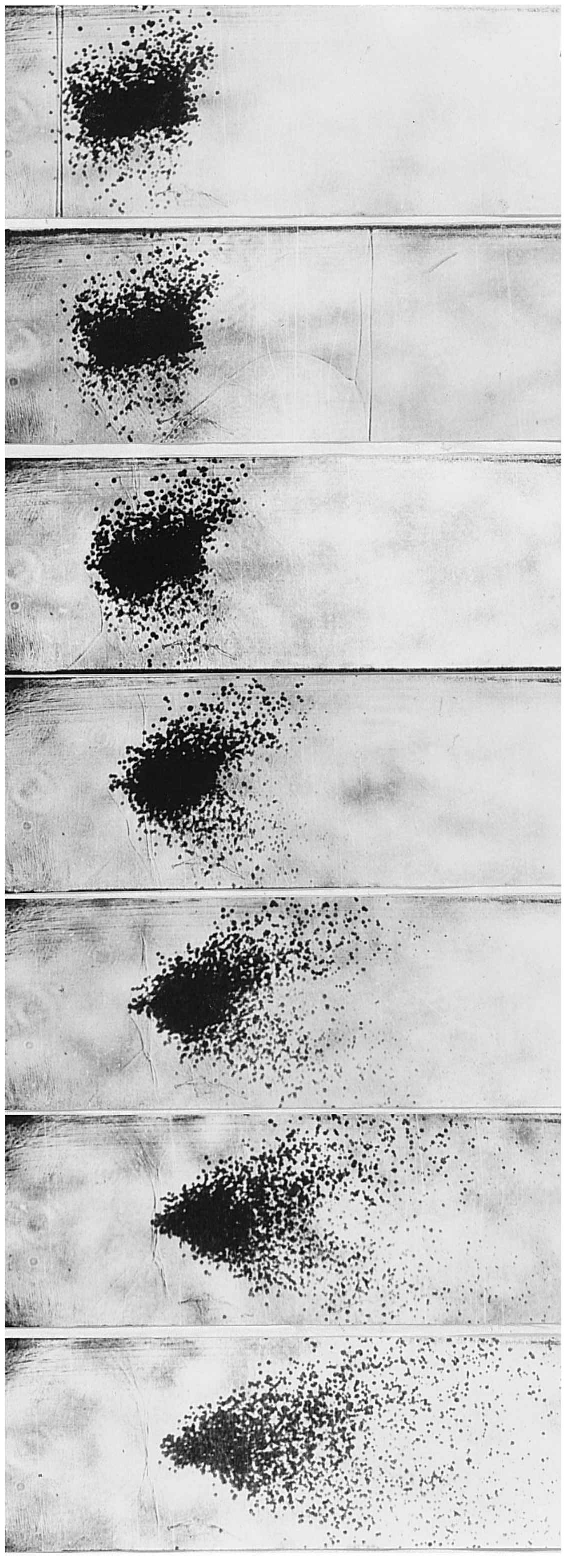


Figure 1: Experimental Image,

(Low φ_p - Boiko et al)(1)

Figure 2: Experimental Image,

(High φ_p - Boiko et al)(1)

The particle motions in a macro-scale particle-fluid mixture model traditionally follow from Newton's laws applied to the individual particles and reflect the force transmitted to the individual particles by the impinging shock.(11) This force will depend on the shock strength (Mach number, M), the density of the particle relative to the fluid ($\frac{\rho_p}{\rho_f}$), the volume fraction of the solid (φ_p) and the particle size (d_p). The key question is: how does one determine the dependency of the force on a given particle on each of these parameters?

The idea pursued in this thesis is that one can perform direct numerical simulations on small clusters of particles subject to a range of conditions in the parameter space defined above (consisting of M, $\frac{\rho_p}{\rho_f}, \varphi_p, d_p$) to learn about the behavior of "representative particles". For example, one can compute the drag versus time curves for particles based on such simulations as a function of the above four parameters.(12) Then one can encapsulate the dependence of the drag on time (t) as well as on the parameters in the form: $D(t) = f(M, \frac{\rho_p}{\rho_f}, \varphi_p, d_p, t)$, which is conventionally the route taken in establishing experimental correlations or drag laws. However, since the drag law to be derived is dependent in a rather complex way on multiple parameters, the resulting manifold in the parameter space that describes the drag law can be quite difficult to obtain. In this regard, the idea of employing a device to "learn" this law from a series of computational experiments becomes attractive. After all, organic systems, exemplified by human beings, learn rather complex patterns and assimilate them with ease; this is accomplished by utilizing the rather complex neural architectures residing in the human brain. The general concept of utilizing neural architectures to learn behaviors that can be transmitted to other systems opens the possibility of using ANNs for multiscale modeling. In the following we briefly review the concept of multiscale modeling (MSM) and various approaches used in such models. The current approach draws some central ideas from such methods but follows the route of ANN-based learning, which has been applied only in a few instances of multiscale modeling thus far. (13)

CHAPTER II:

MULTISCALE MODELING APPROACHES

As stated in Chapter I, the specific goal of this project is to develop a method utilizing an ANN to efficiently model shocked particle-laden flows. The configuration of interest is similar to Figure 1 and Figure 2, taken from Boiko et al., (1) where a cloud of particles is placed in a domain and a shock wake is allowed to pass through the cloud. To simulate the behavior of the particle-laden flows, in particular to capture the particle distribution that is observed in the experiments, it is necessary to adequately represent particle-shock and particle-particle interactions in a macro-scale model.(14) Thus, a connection between micro-scale dynamics and macro-scale dynamics must be established. In the past, empirical or semi-empirical correlations were obtained to connect the micro-scale dynamics to macro-scale parameters and the macro-scale model then employed these correlations;(15)(16) examples include drag laws for the particles, heat transfer correlations (capturing effects of unresolved features such as boundary layers), turbulent viscosity coefficients etc. Developing such correlations requires painstaking experimentation to cover parameter spaces; these experiments can be difficult and expensive, and sometimes, for phenomena occurring at small spatial or temporal time scales, even impossible to perform.(17) In recent times, the idea of multiscale modeling has emerged, spurred by the availability of large-scale computing platforms and improved numerical techniques.(18) In multiscale modeling, detailed computations are performed separately at the microscale and the information obtained from such simulations is “somehow” transferred to the macro-scale and conversely.(19) The key operating word here is “somehow”; the methods for affecting such interscale transfer are still being developed. Some candidate techniques are mentioned below.

Importance of and challenges to multiscale modeling

All phenomena in natural and engineered systems are intrinsically multi-scale. However, in describing these phenomena at the level of detail relevant to engineered systems, continuum scale dynamic laws are employed.(20) Assumptions are made regarding the effects of small spatial and short time scales (hereafter collectively called *“fine”* scales) on the observed large-scale (hereafter called *“coarse”* scale) dynamics. Familiar examples are the encapsulation of molecular interaction effects through thermodynamic equilibrium assumptions into macroscopic material properties, such as viscosity, thermal conductivity, surface tension etc. In traditional continuum mechanics, empirical models (typically called “closure” models) such as Newtonian fluids, Coulombic friction, Linear or nonlinear elastic material, etc. are common and highly successful in describing and predicting the coarse-scale behavior of the material.(21) However, there exist a substantial number of phenomena where, semi-empirical “closure” models are quite inaccurate or even entirely incorrect in their predictions.(21) Typically, in such problems, the fine scales have a disproportionate impact on the coarse scale flows, and therefore demand to be treated in sufficient detail. A classical example of this situation is the effect of boundary layers in flows with small viscosity. Prior to the advent of boundary layer theory classical inviscid hydrodynamic theory produced totally incorrect predictions of flows around solid obstacles, as exemplified by the d’Alembert paradox (22). These difficulties were removed and the theory of flight was placed on solid ground when the boundary layer concept removed the vortex sheet singularity at a solid surface by developing a detailed model of the small scale flow behavior within the boundary layer (23).

Fortunately, in the case of viscous flows, two distinct scales (the body length scale and the viscous length scale) are present, and it is possible to treat these scales separately and to match solutions between the two scales.(24) Examples of other problems (in many cases with singularities resting at the fine scale) where such scale separation exists include crack propagation phenomena in solids, microstructure growth in solidified materials, three-phase contact line motion in multiphase flow, instabilities leading to formation of fine scale structures, such as droplets in spray atomization, flows through porous media etc.(16)(25) (26) On the other end of the spectrum are multiscale phenomena where a separation of scales is not possible and a full description of the mechanics demands treatment of the phenomena at each scale and the interaction between scales. The archetype is fluid turbulence (27). Since the contribution of a continuum of scales is to be accounted for (i.e. turbulent spectra are intrinsically broad band), the modeling of fluid turbulence remains intractable for the foreseeable future.(28) Modelers will continue to rely on “closure” models for the description of fluid turbulence, at least for some range of spatio-temporal scales. (29)

From the standpoint of tractability with regard to multi-scale modeling, it is becoming increasingly feasible to tackle phenomena where a distinct separation of scales exists and where the physical characteristics and transport at these distinct scales can be adequately described and computed.(30) This has become possible, in part due to the rapid development of computer simulation techniques and hardware, particularly associated with large scale computing on multiprocessor systems. Examples of systems that have seen robust activity in multiscale modeling include flows in porous media (31)(32) , multiphase flows (33)(34)(30)(35), and flows in biological systems (36)(37). In these cases, the development of computational techniques and hardware has been assisted by the tandem development of advanced visualization techniques that have enabled quantitative characterization of the geometry of the smaller scales. Since the fine scales can be visualized and efficient methods to compute flows at the fine scale are available, direct numerical simulations can be performed at these scales. The challenge, then, is to extract information from the simulations at the fine scales (called *"restriction"* (38)) that are important to correctly describe the effects of the fine scale on the coarse scale dynamics. Various approaches have been developed in the literature for performing the restriction operation as will be described in the review of methods below. The equations at the coarse scale are computed based on the information provided by the fine scales. The sequence of computation then returns to the small scales which are then evolved again, with initial and boundary conditions supplied from the large scale (through a procedure called *"lifting"* (38)). Multi-scale computation of this type can proceed in a "concurrent" (18) manner (where fine and coarse scales are computed in an alternating sequence with full coupling) or in a "hierarchical" manner (where fine scale computations are performed separately with varying parameters and the results are encapsulated in a model for later use in a stand-alone coarse scale computation). While hierarchical modeling of multi-scale phenomena has great value and is perhaps the only practical recourse for many physical phenomena, concurrent computation has gained increasing attention, particularly with the wide availability of multiprocessor computing environments. This proposal seeks to proceed along the path of concurrent multiscale modeling using novel approaches for computation of the governing equations as well as for communicating between the different scales.

Approaches to multiscale modeling

Modeling codes in computational fluid dynamics, or CFD, are limited in speed mainly by the number of cells used in calculations.(39) To maintain accuracy, cell sizes must be kept small. However the need for CFD models with increasing domain sizes is on the rise. With the small grid sizes and increasing overall domain sizes, cell numbers approach hundreds of millions of grid cells; far more than a single computer processor can handle. Thus a new breed of CFD models has risen up with the idea of multiscale modeling.(13)(17)(40) A significant issue in multiscale modeling is the passage of information from the smaller meso-scales to the larger macro-scales. This process of multi-scale information exchange has given rise to different methods. One of the most promising methods is the patch dynamics method; where only “patches” of fine scale calculations are made then information is passed to the larger scale; the small scale to large scale passage is called lifting.(19) The main issue of lifting is deciding what to pass and how to use the information. This problem becomes even more important to multiscale modeling when multiple phases, such as rigid particles immersed in a fluid, are introduced.(1)(41)

Homogenization/ Up-scaling

The traditional approach to dealing with multiscale phenomena has taken the route that falls under the umbrella of “homogenization/mixture formulations/up-scaling” (42)(43). In such approaches the fine scale details are transmitted to the coarse scale (at which the simulations are conducted) via empirical/semi-empirical/analytical correlations reflecting effective mixture properties (conductivities, diffusivities, equations of state, drag, interphase flux terms etc.).(44) Homogenization-based approaches require specific properties of the small scale behavior, such as scale separation and periodicity.(45) While this treatment suffices for some problems, it suffers from a lack of generality. In addition, the homogenized model (typically embedded into the coarse scale via transfer functions or interphase interaction terms as sources in the governing equations) can often be problem specific or valid over restricted parameter spaces, restricting the range of applicability of the homogenized model. In any case, with this approach, typical of hierarchical multiscale models, the fine scale effects are not fully coupled to the coarse scale dynamics, leading primarily to a one-way coupled model, i.e. the fine scales affect the coarse scale but not vice-versa. Such "coarse-graining" of fine-scale models and the coupling of coarse-grained models to fine-scale models at specific boundaries separating the two sub-domains is quite popular in multiscale models coupling molecular and continuum dynamics computations(46)(47)(48) and in biomedical applications coupling lower-dimensional models to more detailed higher-dimensional models(36)(37)(26). In recent years, attention has turned to effecting two-way coupling between the fine and coarse scales, within and without the homogenization paradigm.(49) The methods adumbrated below provide a route to the type of modeling that is on the horizon, namely full treatment of physics and computation at each of the relevant scales that play important roles in the overall dynamics of the system. In each case the way in which the fine-scale dynamics is coupled to the coarse-scale is italicized for emphasis.

Embedding fine-scale features into global discretization

In the context of modeling flow through porous media, Hou and coworkers (15)(31)(50) and others (51) have developed a multiscale finite element procedure where the (fixed) fine scale variations are embedded into the coarse scale calculations by *incorporating the fine-scale variations into the finite element basis functions*. Thus, the fine scale features are implicitly included in the solution of the coarse scale equations through the global stiffness matrix. In order to include global effects (such as channel formation) in addition to fine-scale heterogeneity, into the porous medium model an adaptive multiscale model has also been proposed (51).

Wavelet-based multi-resolution analysis

Another approach to multiscale modeling in the porous medium literature is that performed Sahimi and coworkers (52)(44)(53). Their approach consists of *using the intrinsic multiresolution feature of wavelet transforms to direct computational resources to those regions that require fine-scale resolution*, while retaining coarse-scale representations where sufficient. This allows for efficient computation and transient adaptivity so that fine-scale features are captured to desired fidelity while interacting fully with coarse-scale computation. The natural multiscale basis functions embedded in wavelet representations and the data compression enabled in the wavelet domain is also exploited in discretizations with wavelet basis functions (54).

Equation-free modeling

A novel "equation-free" multiscale approach has been developed by Kevrekedis and coworkers [46][55]. *The method relies on "short bursts" of fine scale direct numerical simulations (typically using molecular dynamics) to determine the time evolution of coarse-scale quantities*. Since the short burst "numerical experiments" are used to evolve the coarse-scale quantities directly, no transfer functions or up-scaling of the fine-scale information is necessary and therefore the fine- to coarse-scale communication becomes equation-free. The coarse field is then time-stepped with a large (coarse) time step and the *coarse field is "lifted" (i.e. interpolated on to the fine mesh) to provide the initial and boundary conditions for the fine-scale computation*. The efficiency of the multi-scale methodology adopted in the equation-free framework comes from three sources: 1) The coarse-scale time step is much larger than the fine-scale time step ("gap-tooth"-ing [56]), 2) Fine-scale calculations are performed for short bursts, and 3) The fine-scale calculations are performed in discrete and non-contiguous spatial locations with grid sizes much smaller than the coarse grid size. This last strategy is called "patch dynamics" [19] [57][58] (see Figure 3 for illustration of the concept of patches). The spatially separated patches receive initial and boundary conditions through "restriction" from the coarse scale. Engquist and coworkers [38] have extended this approach to other applications in the form of a "heterogeneous multiscale method". In terms of generality the heterogeneous multiscale method holds promise as an approach that can be employed for solving problems involving transient phenomena such as interfaces embedded in the coarse as well as fine-scale fields.

Heterogeneous multiscale modeling

Based on the ideas set forth in the equation-free multiscale modeling strategy of Kevrekedis and coworkers, Engquist and others (38)(48)(59) propose a general and efficient methodology called Heterogeneous Multiscale Method (HMM). In this method, the basic ideas of using patch dynamics, gap-toothing, *lifting and restriction are employed to solve continuum equations at the coarse-scale by extracting fine-scale effects from "short burst", spatially sampled solutions at the fine scale.* Note that the whole notion of fine-scale computations to inform coarse-scale dynamics would be untenable if the efficiencies arising from temporal sampling (gap-toothing) as well as spatial sampling (patch dynamics) were not exploited. That is, the fine-scale solution is only obtained at a few spatial locations and for a short period of time. Example problems where HMM has been used that has direct bearing on the problem of interest to this thesis are the following:

1. Flame front propagation (59): In this problem the HMM approach is used to solve the coarse scale equations by advancing the micro-scale (which cannot be resolved by the coarse grid) combustion front using explicit interface tracking and direct numerical simulation of the front dynamics at the fine-scale. This is a Type A multiscale problem(38)(17), where the microscale solution provides boundary conditions for the macro-scale problem, i.e. the microscale effects are strong in a localized region, in the present case at the singularity represented by the flame front.
2. Crack propagation (60)**:** At the coarse scale the linear elasticity equations are solved along with an update of the crack tip position, while at the fine scale

the molecular dynamics solution in the vicinity of the crack tip is employed. This removes the singularity corresponding to the crack tip from the macro-scale (coarse) field. This is a Type A problem as well; fine scale effects are disproportionately critical in the vicinity of the crack tip.

3. Dynamics of complex fluids (48): At the coarse scales the standard Navier-stokes equations are solved; molecular dynamics is used at the fine scale to inform the coarse-scale constitutive laws where necessary. This includes regions that exhibit singularities at the coarse-scale, such as at moving contact lines, stress singularities at sharp corners etc. These represent Type A multiscale problems. In addition, the effect of complex molecular structure of fluids are included by considering macromolecules, modeled in the fine scale as dumbbell shaped particles with appropriate potentials in the MD calculations. The multiscale coupling then is used to inject the fluid stresses into the coarse-scale equations. This is a Type B multiscale problem (17), where the micro-scale problem provides constitutive relationships for the macro-scale field.

The problem of interest to this project is a Type B problem in the sense that we seek to obtain coarse-scale information on the constitutive properties and behavior of materials with subgrid texture and dynamics. The primary difference between the application of HMM to the Type B problem (# 3 above) and the problem to be addressed in the present proposal is that while in (48) the fine-scale model is an MD model, in the present case the fine-scale model is a continuum model. In this sense the situation of interest to this work is akin to the subgrid modeling of flows in porous media (15).

Application of Artificial Neural Networks to multiscale modeling

There have been significant advancements in the area of digital cognitive enhancements and artificial intelligence.(61) One particular application of artificial intelligence which closely parallels what we are seeking in this thesis is that of pattern recognition or knowledge assimilation for use in fluid dynamics.(62)(63) Essentially, we want to gather knowledge regarding how a particle in a cluster reacts to a shock that hits the cluster. We want to "learn" this behavior from computational experiments and then transmit this behavior to another simulation performed at a coarser scale. With this purpose in mind, a possible candidate approach for knowledge acquisition is an artificial neural network, or ANN, which is capable of learning a myriad of different behaviors. ANNs are capable of learning the complicated behavior of several variables by modifying a collection of weights attached to its "neurons".(64) In effect, the learning process is designed to mimic small parts of the human brain where learning takes place by modulating the strength of synaptic connections between individual neurons. In the human brain this process is incredibly fast, as the human brain is capable of processing the computational equivalent of over 10 petaflops.(65) The assimilation and recovery of knowledge from ANNs is not quite as spectacular. But the essential idea remains the same. The tedium in ANN applications (unlike in the human brain) comes from the need to train the ANN by providing it with sufficient samples of training data, so that the ANN can adequately construct (in its "mind's eye", so to speak) the hyper-surface (due to the multidimensional parameter space) representing the behavior of the system. The number of samples required to train the ANN depends on the complexity of the behavior to be represented and also depends on the complexity of the ANN itself.(66) Therefore, while the ANN provides learning and knowledge assimilation ability akin to a microcosm of the human brain, its use must be accompanied with sufficient care from the user in order to properly train the ANN in the desired parameter space. Once the ANN is trained however, knowledge recovery is rather rapid, and can be effected by interrogating the ANN. Utilizing the ability of the ANN to capture and represent complex behavior in a multidimensional parameter space in a CFD code would greatly improve the speed of calculations while still maintaining accuracy. The jump in speed would be obtained by the ANN's ability not only to learn and process information, but also because of its prediction capabilities.   The work in this thesis will seek to demonstrate these concepts by applying it to solve the problem of shock-impacted particle laden flows as pictured in Figure 1 and Figure 2 of Chapter 1.(1)

Macro-scale modeling and interscale coupling

At the macro-scale, information about what happens at the meso-scale is still needed. The largest problem in macro-scale modeling is to obtain meso-scale information in an efficient manner. The main idea utilized in our modeling is patch dynamics.(19) In patch dynamics, the idea is to only run direct numerical simulations (DNS) for fine scale or meso-scale models selected from the coarse scale or macro-scale model. A diagram of this concept is displayed in Figure 3.

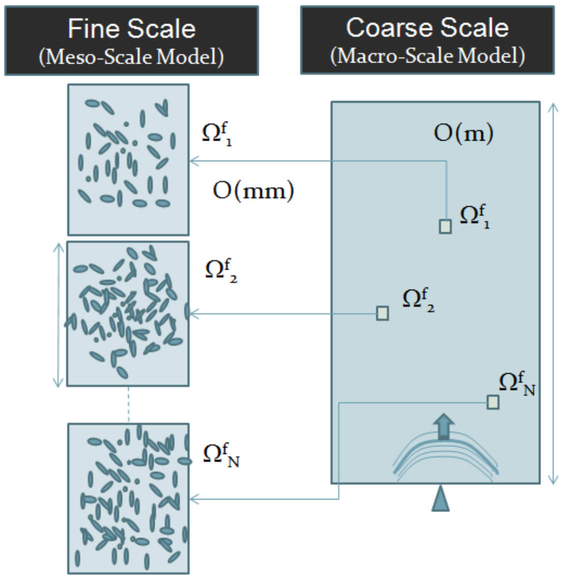


Figure 3: Patch Dynamics Concept

At the meso-scale, mixtures are considered to be suspended particles which require DNS, while at the macro-scale mixture may be assumed as homogeneous with characteristics obtained from the meso-scale. This way the effects of the meso-scale may be accounted for and the flow characteristics lifted from the smaller scale will be utilized without high computational expenses. In the case of flow through a porous media for instance, a detailed calculation at the meso-scale can provide the flow impedance of a small patch (i.e. RVE – representative volume element).(67) For shock-impacted particle laden flows the concern at the macro-scale is to predict the particle trajectory. In our case, the macro-scale modeling is only concerned with the evolution of particle motion. The particles at the macro-scale are advected in Lagrangian fashion and are treated as point particles. Provided with a particle’s mass, initial position and the force experienced by the particle, a Lagrangian model to track particle trajectory can be obtained. The meso-scale simulations then are charged with providing the force on the particle as a function of several control parameters, as mentioned previously. Using the concept of ANN based learning, the way in which meso-scale direct numerical simulations can be used to develop quantitative information on the forces on the particle, will be described in the following chapters.

CHAPTER III:

METHODOLOGY

The basic methodology behind behavior learning for shock-impacted particle laden flows is to develop some type of learning algorithm; in this case an ANN. Then it is shown that the ANN can properly learn from selected data sets (called “training sets”), and predict values for points in the parameter space (called the “testing set”) that were not provided in the training set. Note that ANNs perform well when used for interpolation, but poorly for extrapolation. Therefore the testing set typically must lie within the convex hull of the training set. In this chapter we briefly introduce the main principles of ANNs and the process of training and testing the ANN.

Artificial Neural Networks

ANNs are composed of a myriad of simple elements called neurons. The neuron by itself is a very simple device. In both a biological neuron and in an artificial neural network, each neuron has inputs coming into it; each neuron performs some process with the inputs, and then sends an output to other neurons. A single neuron is responsible for only one function and a representation of an artificial neuron can be seen in Figure 4.

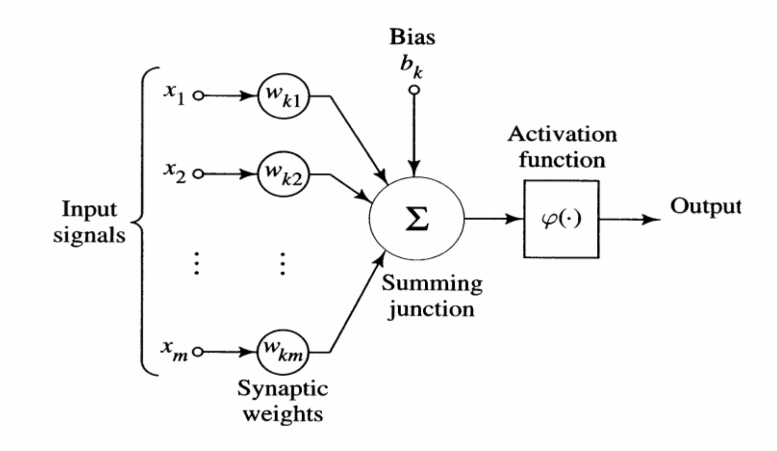


Figure 4: Artificial Neuron

The output for the single neuron illustrated above is described in by,

$$O_n = \varphi(b_k + \textstyle\sum_{i=1}^{m} w_{ki} * x_i) \qquad \text{Equation 1}$$

where O_n is the output, φ is the activation function, b_k is the threshold bias, w_k are the synaptic weights, x_i are the inputs from the previous layer of neurons, and i is the number of inputs. Every neuron has many connections going to and from other neurons. The information passed on to each neuron may have a very small or large effect on the final output. A threshold limit is used via a bias value in artificial neural networks to simulate the formation and degradation of inter-neuron connections as in the biological system. Large arrays of these neurons supply the ability to map out regions of parameter space defined by the input parameters. Each neuron is capable of editing weights supplied to it based upon the accuracy of the entire network. This enables the neural network to learn the behavior of data provided.

There are many different weighting schemes and update procedures. For the one utilized here, a complete interlayer network is used.(66) This connects each neuron of one layer to each neuron of the next layer. Every neuron in the network uses the same basis function to calculate its output. For the artificial neural network used in this work, the basis function is a sigmoid function. A sigmoid function is an S shaped curve. It can be a smooth logarithmic based curve or piecewise threshold type, as shown in Figure 5.

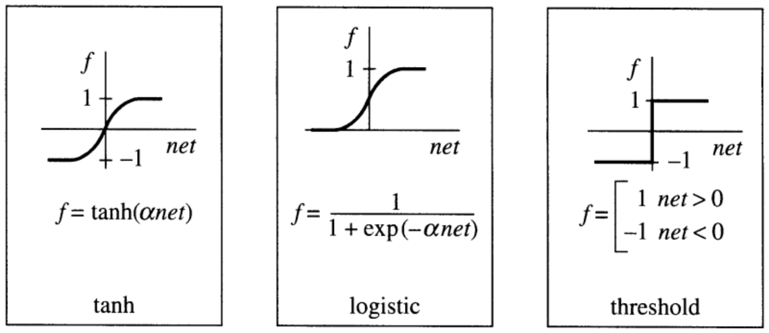


Figure 5: ANN Activation Functions

The activation function selected for our neural network is the function tanh(x). It is a fairly simple function whose derivative is always positive, making it a popular choice. Several ANNs use other basis functions, including simple step functions, periodic basis function such as sine and cosine, radial basis functions(68) such as Gaussian distributions, and even wavelets; the idea of using wavelets as a basis function (69) and to assist in learning was experimented as a tangent topic. A simple diagram of how the summation property to formulate a new hyper-surface by an ANN is shown in Figure 6. In this example, a network of two linear inputs, a bias threshold, four hidden layer neurons and an output neuron exists. This ANN is feed forward(68) because the direction of information travel is only forward. Using the inputs from the previous layer, each of the neurons in the hidden layer is formulating a simple decision scheme. For one neuron, the low values of the first input are useless and the high values are important while the second input is disregarded by not having a value above the bias threshold. For that hidden neuron, its own hyper plane would appear with one low end and one high end with a gentle sigmoid slope separating the two regions. This hyper plane developed will never be seen by the user and thus the data and the neuron interacting with it is “hidden”. The other three neurons behave in the same way treating one region as being more or less important as well as one input. However, when the four neurons are all summed together by the final output neuron, they create a hyper plane that shows a shared region of importance. A simple application of this network could be the amount or red and blue as inputs of a color mix and the final area would be purple. A more complicated example as Ahmadi et al. performed, include inputs of porosity and water saturation to predict permeability of porous media.(70)

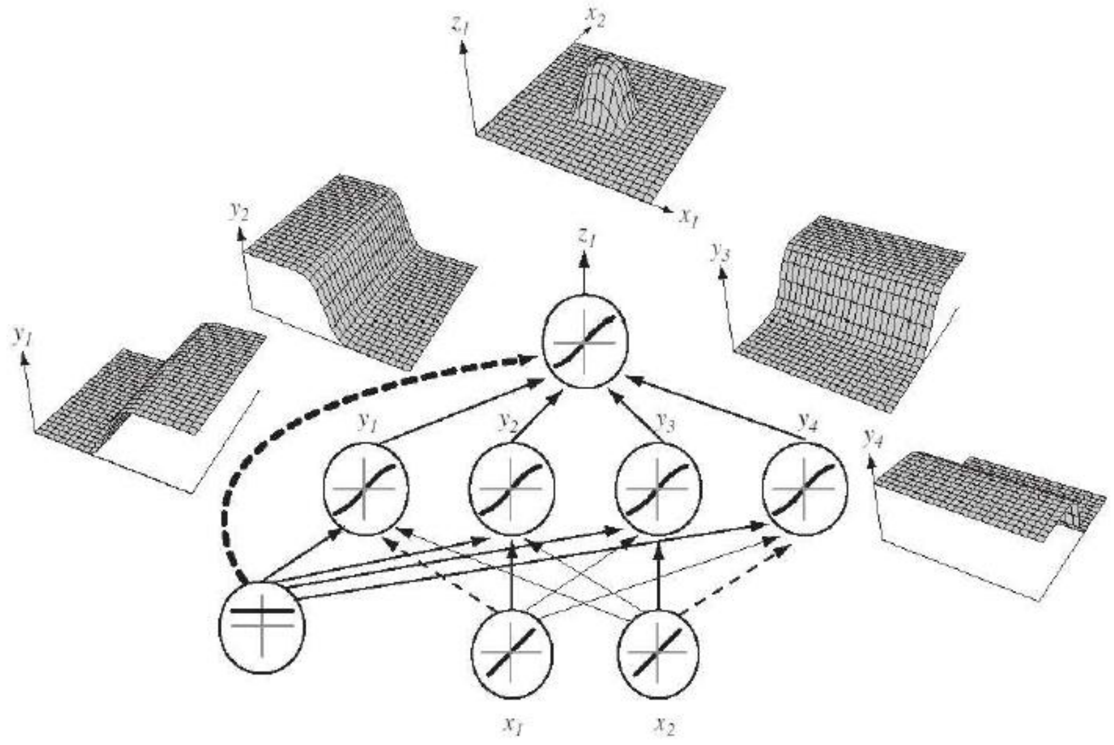


Figure 6: ANN Hyper-surface Development

Topography

The neural network used is a single hidden layer, feed-forward, back-propagation network. It possesses one hidden layer of neurons set between the input layer and output layer. The ANN is capable of expanding the number of hidden neurons based upon the complexity of the function the neural network is fitting. The input layer includes one bias neuron to facilitate different levels of activation for each hidden neuron. All of the input data was fed forward through the network in one direction without any neurons competing against each other; this is why the ANN is called a feed forward network. The last layer consists of outputs where a final prediction can be used to find an error in the prediction and adapt the weights to the previous layers allowing the ANN to learn. The basic network topography is show in Figure 7.

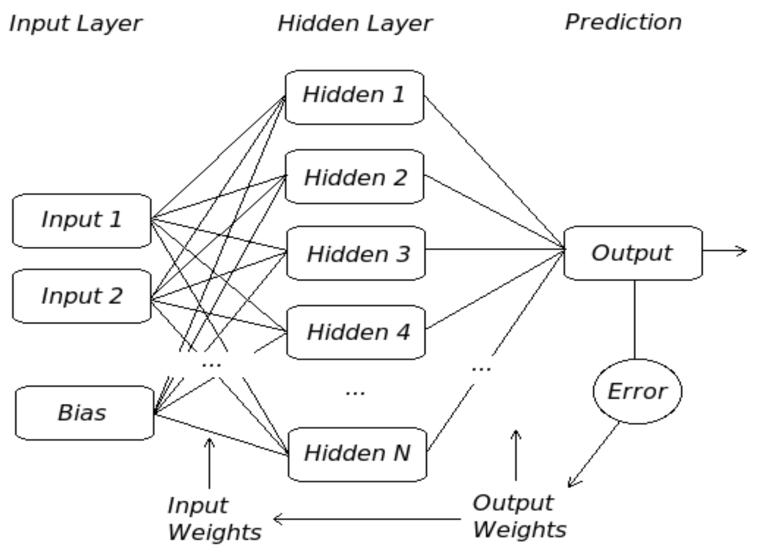


Figure 7: ANN Topography

Learning and Prediction

Once the ANN has been developed, it must go through two important phases before it will be capable of producing useful predictions. The first phase is the training phase where a set of data is provided and the ANN learns from the data. The algorithm used to learn and edit the weights for each neuron is called a back-propagation algorithm. Every neuron in the network contains the same basis function for processing data. For most cases, there is only one output neuron that sums all its inputs to arrive at a final prediction. A back-propagation algorithm (71) takes the predicted values and compares it to the expected values (i.e. to the target output for the given inputs in the training set). Depending on the error between the two, the weights for each neuron is edited.(68) The testing of the neural network is performed by making a random selection from the data set (until all the data are run through) and each data point is tested and used to train the neural network once per cycle. When the ANN is in training, it should be learning from every point in a data set otherwise learning will be biased. Every iteration step for an ANN consists of cycling through the total number of data points in a data set. The error produced on every iteration step can be plotted to show a convergence curve on how the ANN is being trained. One such convergence curve for the training of ANN is shown in Figure 8. Note that as the iterations increase the learning of the ANN saturates and convergence is declared at a pre-specified error tolerance or maximum iteration count.

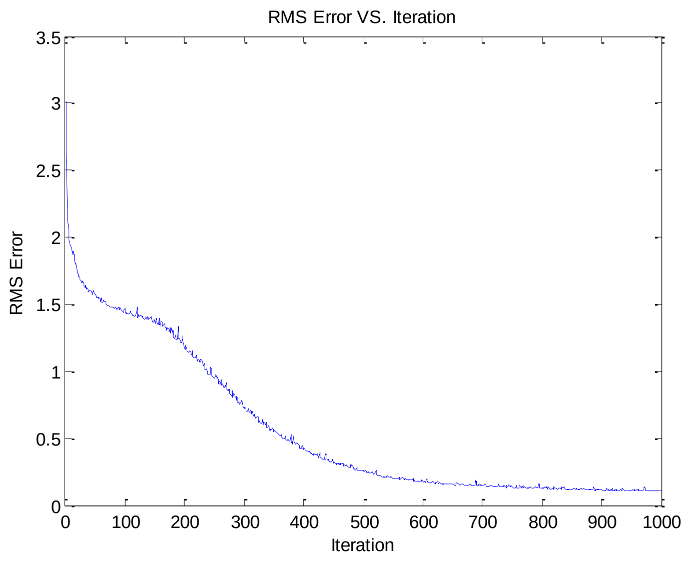


Figure 8: ANN Convergence

When the training phase is complete, an artificial neural network can be tested by querying with a testing set of input data. The resulting output from the ANN is compared against the desired output corresponding to the input parameters for that testing set. The ANN is believed to have successfully learned if the error produced for the testing set is below a desired tolerance. Querying an ANN at multiple points inside a domain allows us to obtain a final plot of what the ANN predicts. The performance of the ANN as a function approximation device is illustrated with some examples below.

Examples of ANN learning process

Logic gates

To illustrate the basic ideas of training and testing of an ANN, some simple examples of function learning are presented in the following. In the first example, the ANN is tested on logic gates by feeding in the inputs and target data. The

ANN then is trained and tested for accuracy. To train the ANN the input data is set as the value of the ordinate and the target output is the corresponding output value. Once the ANN has been trained, it is tested, i.e. the ANN is queried for values of the ordinate (as input data) of which, for most applications, the ANN was not trained. The resulting output of the ANN is the predicted data. In the case of training with logic gates, the ANN had 10 hidden neurons and was trained through 500 iterations. The inputs and output were Boolean variables while the ANN was allowed to use any real value; this allowed the ANN to use the sigmoid activation function and enables us to show a curved convergence path (as opposed to piecewise). The convergence for each logic gate is displayed in Figure 9. For the cases of AND, OR, and NOR, the areas of delimitation were linearly separable. In a 2 by 2 array of Boolean variables, the area of positive values for linearly separable region can be closed off with one line. This occurs for any function with a derivative with constant sign. For the XOR logic gate, the inputs are no longer linearly separable and therefore local minima or local maxima may occur. This requires a network of logic gates to define, and for an ANN the convergence curve is slow at first before it discovers this fact. This ability to handle apt to several regions of maxima and minima is what sets an ANN apart from an interpolation scheme.

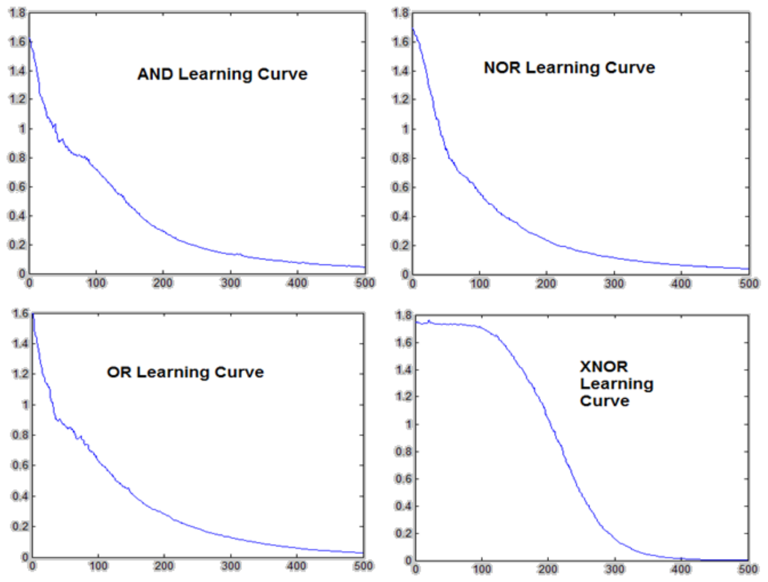


Figure 9: ANN convergence for logic gates

Table 1: Logic gates

AND	1 (true)	0 (false)
1 (true)	1 (true)	0 (false)
0 (false)	0 (false)	0 (false)

NOR	1 (true)	0 (false)
1 (true)	0 (false)	0 (false)
0 (false)	0 (false)	1 (true)

OR	1	0 (false)
1 (true)	1 (true)	1 (true)
0 (false)	1 (true)	0 (false)

XNOR	1 (true)	0 (false)
1 (true)	0 (false)	1 (true)
0 (false)	1 (true)	0 (false)

<u>The single-variable sine function</u>

In the next example, the ANN was provided discrete data corresponding to the functional form of a sine wave, shown in Figure 10. This single-variable function introduces the idea of multiple values for data input and correlation to the output. The ANN had 20 hidden neurons and was trained for 1000 iterations. The comparison for the training set data (red dashed line) and final prediction (blue solid line) can be seen in Figure 10. As seen from the figure, the ANN predicts the sine wave in good agreement with the actual sine function. It is important to note that the only difficulty that the ANN exhibits is at the maxima and minima of the curve where the predictions are not as accurate. The accuracy of the predictions can be improved by employing a greater number of neurons or using more sophisticated training methods. [72]

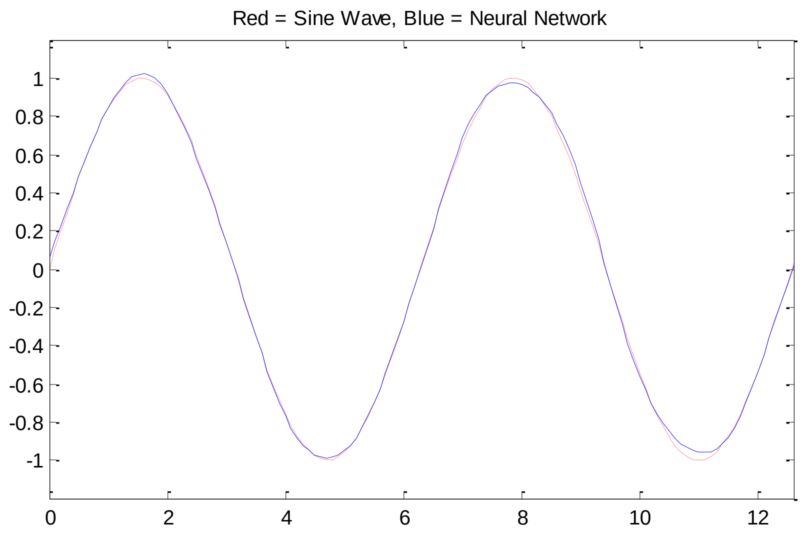


Figure 10: ANN Sine Wave Comparison

Multi-variable case (“Peaks”)

Next, we examine the performance of the ANN for a multi-variable function; a two-dimensional input space is considered, and the predicted manifold is then a surface. For multiple inputs the ANN is provided with each input in the form of an array and the prediction is plotted on a hyper-surface. The test case here is a MatLab standard test called “peaks”. This test case presents a fairly complex manifold to be learnt by the ANN, with multiple maxima and minima in the parameter space and fairly steep gradients. To insure accuracy the ANN learning was iterated 10000 times. The error of the ANN decreased dramatically in the beginning and then slowed down. There were small points in learning where the ANN had slight increases in error. These small jumps allowed for the learning of data regions that are not linearly separable. The 2-D surface known in MatLab as “peaks” is shown in Figure 11 with convergence - (a), ANN prediction - (b), training input – (c), and actual value (d). The powerful feature of the ANN is that increase in dimensionality of the parameter space can be carried out indefinitely (i.e. the hyperspace to be constructed can be of arbitrary dimensions), assuming the ANN is capable of adapting fast enough to what it needs to learn. Furthermore, once the ANN is trained for all available training data sets, if further data sets become available the ANN can be further trained by introducing these additional training data; therefore, refinements of the prediction capability of the ANN in selected areas of the parameter space can be carried out as necessary. These features lend versatility to the ANN and make it an attractive function approximator in comparison to standard regression techniques.

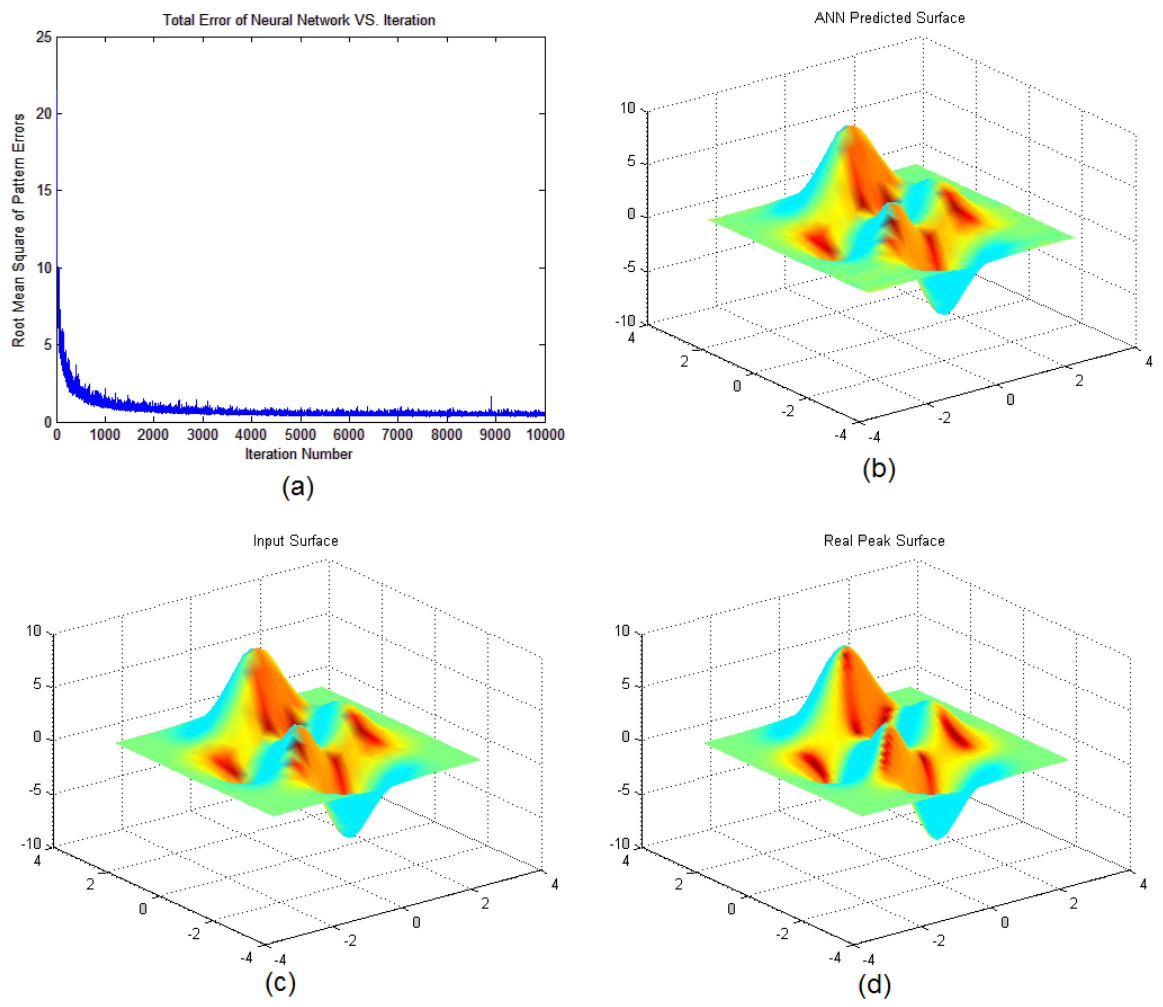


Figure 11: ANN learning of "Peaks"

<u>Learning a drag law</u>

When a planar shock wave hits a stationary spherical particle and passes over it, the drag force on the particle (i.e. force exerted on the particle) changes throughout shock passage. Once such drag versus time curve obtained by Tanno [73] in an experimental (shock tube) setup is displayed in Figure 12.

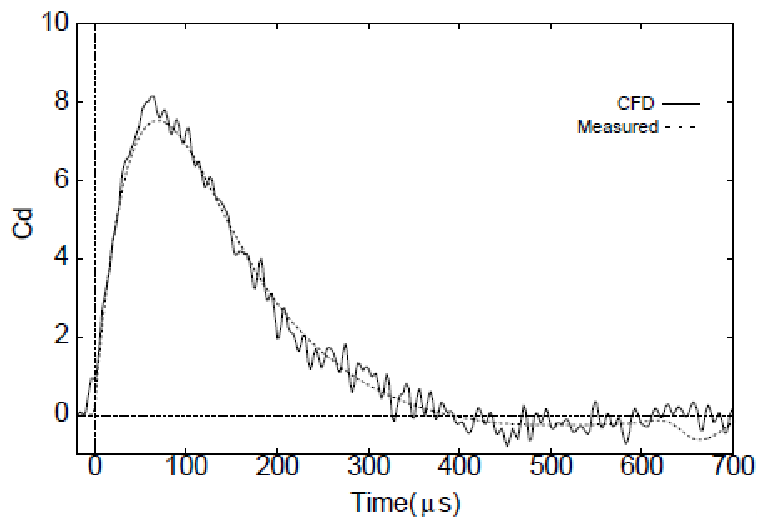


Figure 12: Transient Drag Curve[(73)]

<u>Empirical drag laws</u>

Empirical drag laws do not provide the transient drag experienced by the particle as the shock passes over it. Instead, some measure of steady drag is available that omits the details of the shock passage. With trained ANNs, however, one can retain the information on the drag versus time for a wide range of parameter space. Thus, information obtained from experiments or computations need not be discarded; it can be learned and retained as "knowledge" by the ANN. This does not imply that a large data set is stored. Once the ANN is trained the information on the drag versus time behavior is stored in the weights attached to the individual neurons in the ANN; the individual data sets used for training can then be discarded.

When attempting to accurately track the position of a particle, the forces, acceleration, and velocity changes over time. Almost no previous models of shock-impacted particle laden flows contain explicit transient drag force across a particle. Drag force laws with anything other than Reynolds Number as the dependent variable are few and far between. The ANN-based learning technique offers the possibility of retaining this information based on detailed experiments or numerical simulation around individual particles. Then, for developing such drag laws, the input space is spanned by the parameters characterizing the flow as well as the time of shock passage, so that the transient drag behavior can be learnt by the ANN and retained in the form: $D(t) = f(M, \frac{\rho_p}{\rho_f}, \varphi_p, d, t)$.

<u>“Lifting” information from meso-scale calculations</u>

The driving force behind particle motion in shock impacted particle laden flows is the drag force produced on the particle. Once a shock wave has passed over a particle, the subsequent trajectory of the particle can be determined from Newton’s law if the impulse provided to the particle by the shock is known. To model a particle’s trajectory at the macro-scale, information must be “lifted” from the meso-scale. To limit the amount of information passage between scales, only the most pertinent data is passed. A particle’s position, trajectory and velocity are dependent only on the initial location, mass and force applied. Since the force is transient in nature, its characteristics must be quantified. When viewing a shocked particle drag curve (Figure 12), it is evident that there is a peak point encountered and the drag force decays over a certain interval of time. These two values are maximum drag coefficient, $C_{d_{max}}$ and relaxation time, τ_r. Once the drag versus time curve is established and the $C_{d_{max}}$ and τ_r is known, the total impulse, I_t , can be computed as the area under the curve. For a standard drag curve (obtained from experiment or simulation), we can set τ_r to be represented by exponential decay and thus the impulse would be:

$$I_t = \int_{t_o}^{t_f} C_{d_{max}} * e^{-t/\tau_r} \qquad \text{Equation 2}$$

where I_t is the impulse, t_o is the impact time, t_f is the final time, $C_{d_{max}}$ is the maximum drag force, t is time, and τ_r is the relaxation time. It turns out that in macro-scale calculations, the quantity of interest is the I_t. In addition, since the application of I_t acts over a time characterized by τ_r, once these two values are known, the momentum change of a particle hit by a shock can be calculated. These two pieces of information are all that is needed to quantify a particle's trajectory in a macro-scale calculation. Thus, the ANN can be trained to learn these two quantities as functions of the input parameters.

Macro-scale calculations

Since the main idea behind using an ANN-based learning scheme was to create an "equation-free" lifting scheme,(40)(46)(55) macro-scale calculations can employ the information obtained from the ANN in effecting Lagrangian particle motion. Given the Mach number, $\frac{\rho_p}{\rho_f}$, and d_p, an ANN can predict $C_{d_{max}}$ and τ_r. These values can then be placed in a Lagrangian algorithm using Newton's second law and the particle trajectory calculated.

The above represents the main idea pursued in this thesis. The ANN is used to "lift" information from detailed meso-scale calculations (or perhaps even from experiments if they are available). The macro-scale calculations access the lifted information by simply querying the ANN (a procedure that rapidly provides information to the macro-scale on fairly complicated behavior of the particles as the meso-scale). The macro-scale calculations are then advanced further and information is accessed from the ANN at each step of the macro-calculation. This type of interaction between the micro-

and macro-scale, affected by trained ANNs represents a first step in developing a true multiscale simulation capability, in which it may be necessary to perform meso-scale simulations in tandem with macro-scale simulations, in a “patch dynamics” and “gap-tooth” framework.(19)(18) Then the ANN learning process will proceed alongside the querying process and the micro- and macro-calculations will need to be synchronized in some way. This rather elaborate setting for performing multiscale simulations will benefit from massively parallel computations on large processor clusters; the key issue then will be to efficiently orchestrate the micro- and macro-computations along with model assimilation using ANNs.

CHAPTER IV:
NUMERICS AND CALCULATIONS

Formulation

It is noted that the specific application of the methods developed in this thesis is the interaction of a shock wave and a dusty gas. We recall the reader to the illustration in Figure 1 and 2 of Chapter 1, where a shock traveling from left to right passes over a cloud of particles, conveying momentum to the particle clouds. While the low solid fraction cloud disperses in an amorphous fashion, the high volume fraction cloud assumes a triangular form. We seek to simulate this difference in behavior in a multiscale simulation framework. The key difference between the two cases mentioned above (and pictured in Figures 1 and 2 of Chapter 1) is that in the high-volume fraction case the inter-particle effects (such as shielding, shock reflections etc.) have a significant effect on the dynamics of the particle. Simple drag laws derived for single particles cannot be applied to obtain the behavior shown in Figure 2. To obtain quantification of these inter-particle interaction effects, detailed meso-scale calculations are performed on smaller clusters of particles and the behavior of a typical (representative) particle is to be learnt using the ANN. These meso-scale calculations are in the category of DNS, i.e. they are highly resolved. The computational setup for such simulations would require a domain large enough to contain the incident shockwave, the cloud of particles, bow shocks, and shock reflections without major wall interference. However, the grid size would need to be small enough to capture necessary details of shock-particle interaction, particle motion, shock wave dynamics, transient forces, and sharp interfaces. To accurately model at the meso-scale the physics of shock-impacted particle laden flows need to be understood.

Physics

There are two main things to note in the physics of shock impacted particle-laden flows, they are the jumps in properties across the shock wave and the shock wave particle interaction that occurs. Most of the other properties follow standard physics in fluid flow and multi-phase interaction. In our case we are particularly interested in the shock wave in air.

For shock-particle interaction, there are many other areas of consideration including the development of a bow shock that transmits much more drag to a particle than standard incompressible fluid flow would predict. Current drag laws for supersonic flow were obtained through experimental setups.(74) Some drag correlations (for spherical particles embedded in supersonic flow) are listed in Table 2. Some of the first numerical attempts to quantify drag forces were to use Stokes drag, this lead to the normal drag laws we have seen in regions of low Re as in Figure 13.(75) A comparison of drag laws at higher Re can be seen in Figure 14. Stokes drag disregarded turbulence, and at supersonic speeds, the kinematic boundary layer responsible for turbulence would not develop quick enough to have a significant effect on the drag.(76) However, when a shock wave encounters an interface, the steep jump in fluid properties produces a sharp jump in drag force. This steep jump is purely due to the pressure differences on the particle surface due to the passage of the shock.

Once the incident shock has passed over the surface and the reflected shock has formed a standing bow shock wave ahead of the particle a steady-state in the drag is reached. If the particle is free to move the final state of the particle is one of constant velocity and the drag on the particle goes to zero. All of these features of shock particle interaction must be captured by a drag law; this is obviously very difficult to do in empirical models. Thus, most previous work has resorted to using drag laws as functions of Re and Mach to determine drag such as the ones in Table 2. This type of drag laws does not explicitly define unsteady drag but rather an overall drag coefficient once the shock has already passed.

Table 2: Drag Laws

Paper Author(s):	**Coefficient of Drag Equation:**	
“Standard” Clift (1978) [77]	$C_D = \frac{3}{16} + \frac{24}{\mathrm{Re}}, \quad \text{for} \quad 0 < \mathrm{Re} < 0.01;$ $\log\left(C_D \frac{\mathrm{Re}}{24} - 1\right) = -0.881 + 0.82w - 0.05w^2, \quad \text{for} \quad 0.01 < \mathrm{Re} < 20;$ $\log\left(C_D \frac{\mathrm{Re}}{24} - 1\right) = -0.7133 + 0.6305w, \quad \text{for} \quad 20 < \mathrm{Re} < 260;$ $\log(C_D) = 1.6435 - 1.1242w + 0.1558w^2, \quad \text{for} \quad 260 < \mathrm{Re} < 1.50 \cdot 10^3;$ $\log(C_D) = -2.4571 + 2.5558w - 0.9295w^2 + 0.1049w^3, \quad \text{for} \quad 1.50 \cdot 10^3 < \mathrm{Re} < 1.20 \cdot 10^4;$ $\log(C_D) = -1.9181 + 0.6370w - 0.0636w^2, \quad \text{for} \quad 1.20 \cdot 10^4 < \mathrm{Re} < 4.40 \cdot 10^4;$ $\log(C_D) = -4.3390 + 1.5809w - 0.1546w^2, \quad \text{for} \quad 4.40 \cdot 10^4 < \mathrm{Re} < 3.38 \cdot 10^5;$ $C_D = 29.78 - 5.3w, \quad \text{for} \quad 3.38 \cdot 10^5 < \mathrm{Re} < 4.03 \cdot 10^5;$ $C_D = -0.49 + 0.1w, \quad \text{for} \quad 4.03 \cdot 10^5 < \mathrm{Re} < 10^6;$ $C_D = 0.19 - \frac{8 \cdot 10^4}{\mathrm{Re}}, \quad \text{for} \quad 10^6 < \mathrm{Re} < 10^8, \quad w = \log_{10}(\mathrm{Re}).$	Equation 3
Newton (as cited by **[78]**)	$C_d = 0.44$	Equation 4
Stokes (as cited by **[78]**)	$C_d = {}^{24}\!/_{Re}$	Equation 5
Oseen (as cited by **[78]**)	$C_d = \frac{24}{Re}(1 + \frac{3}{16}Re)$	Equation 6
Sommerfeld[41]	$C_d = 112 * Re^{-0.98}$	Equation 7
Boiko[1], **Fedorov**[79], **Khmel**[80]	$C_d = \left(1 + e^{\frac{-0.43}{M^{4.67}}}\right) * \left(0.38 + \frac{24}{Re} + \frac{4}{Re^{0.5}}\right)$	Equation 8
Saito[78]	$C_d = 0.48 - 28 * Re^{-0.85}$	Equation 9
Saito[81]	$C_d = \frac{2f}{\rho * U^2 * \pi * r^2}$	Equation 10
Kosinski[82]	$C_d = \frac{24}{Re}(1 + 0.183 * Re^{0.5}) + 0.42$	Equation 11
Kosinski[83]	$C_d = \frac{24}{Re}(1 + 0.15 * Re^{0.687})$	Equation 12
Ben-Dor[84][85]	$C_d = \frac{24}{Re}(1 + 0.15 * Re^{0.687}) + \frac{42}{1 + 425000 * Re^{-1.16}}$	Equation 13
Wang[86]	$C_d = 0.48 + 28 * Re^{-0.85}$	Equation 14
Igra[87]	$\log_{10}(\mathrm{C_d}) = 7.8231 - 5.8137(\log_{10}\mathrm{Re}) + 1.4129(\log_{10}\mathrm{Re})^2 - 0.1146(\log_{10}\mathrm{Re})^3$	Equation 15

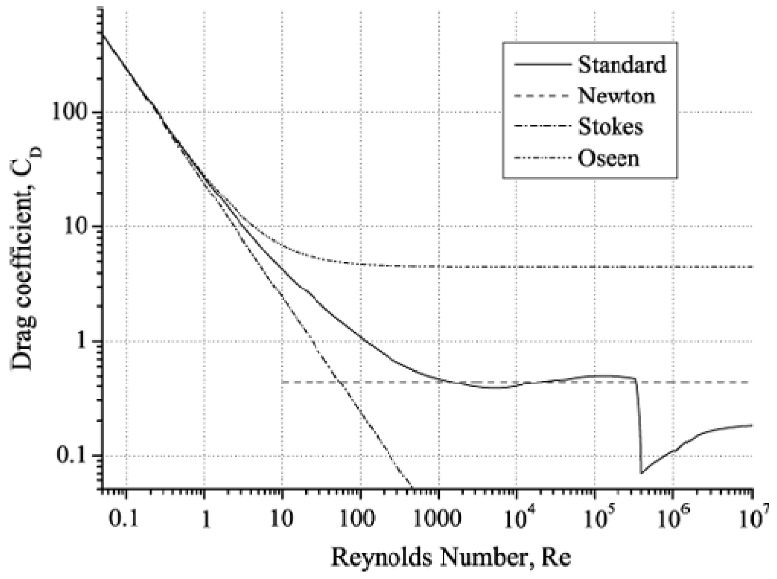


Figure 13: Comparison of “Standard”, Newton, Stokes and Oseen drag laws (78)

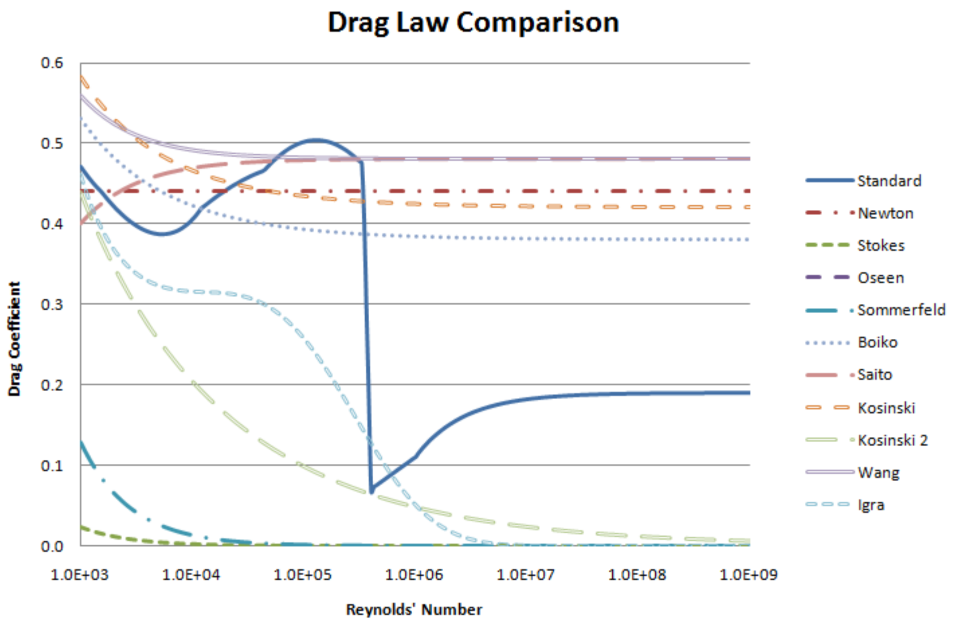


Figure 14: Comparison of drag laws

As seen above, there are a variety of drag laws producing essentially comparable magnitudes of drag. Interestingly they are all cast in terms of a Reynolds number, following conventional practice along the lines of Stokes and Oseen drag laws and subsequent corrections for incompressible flows. Due to the fact that drag on a particle is transient and drag law equations are heavily dependent on relative velocity, there is no method utilizing drag laws to explicitly predict the drag force on a particle. However, as shown below, when a particle is impacted by a shock the primary forces impelling the particle are inertial. In fact, for small enough particles (i.e. in the micron-range), shock passage is rapid enough that viscous effects can be neglected and the Euler equations can be employed to predict forces on the particles; this is the approach taken in this work; then, viscous effects come into play at much longer time scales. Thus, it is puzzling that all of the drag laws are cast in terms of a Reynolds number for a purely inertia-dominated system. In the present work, recognizing that a shock impinging on a particle conveys momentum to the particles purely due to inertial effects, the drag on the particle is obtained without recourse to the Reynolds number on the particle. The main physical effects that act on the particles in the meso-scale simulations are the pressure forces that arise due to the effect of the incident shock and the complex shock interactions that occur over short time scales. Secondly, the current ANN-based framework does not seek to develop a drag law as in Table 1, but assimilates the drag as a function implicit in the trained ANN.

Scaling and Variation

Here we reason that the application of the Euler equations is appropriate for the case of particles being impinged upon by a shock, a system that seeks to emulate the one shown in Boiko's experiments (Figures 1 and 2, Chapter 1). For a shock passing over a spherical particle, one can separate drag into inertial drag and viscous drag. This is because these two types of drag operate and two widely separated time scales. The inertial time scale can be estimated as:

$$\tau_{inertial} = \frac{d_p}{U_\infty} = \frac{d_p}{a} * \frac{a}{U_\infty} = \frac{d_p}{a} * \frac{1}{M} \qquad \text{Equation 16}$$

and the viscous time scale as:

$$\tau_{viscous} = \frac{{d_p}^2}{\nu} = \frac{d_p}{U_\infty} * \frac{d_p U_\infty}{\nu} = \frac{d_p}{U_\infty} * Re \qquad \text{Equation 17}$$

The ratio between the inertial and viscous time scale is:

$$\frac{\tau_{inertial}}{\tau_{viscous}} = \left(\frac{d_p}{a}\frac{1}{M}\right) * \left(\frac{U_\infty}{d_p}\frac{1}{Re}\right) = \left(\frac{U_\infty}{a}\frac{1}{M}\right) * \left(\frac{1}{Re}\right) = Re^{-1} \qquad \text{Equation 18}$$

where d_p is the particle diameter, U_∞ is the flow velocity, a is the speed of sound, M is the Mach number, ν is the kinematic viscosity, and Re is the Reynolds number. The Reynolds number is defined as the ratio of inertial forces to viscous forces. For high speed compressible flows, the Reynolds number is very large. It usually lies in the range of 10^5 to 10^6 even for small particles. The implication is that the effects of the viscosity of a fluid would not be significant until the shock is already 10^5 to 10^6 particle diameters away; thus in determining the motion of particles in the instants following shock impingement viscosity may be neglected and the driving force behind shocked particle motion is mainly inertial drag from the shock wave.

For the purpose of making comparisons, our simulations were kept fairly close to numerical calculations (78) (88) and experiments performed (1)(89) and published by others. As mentioned before the parameter space is defined by the Mach number, the particle volume fraction, the relative density of the particle to the fluid and time. Mach numbers were set between 1.2 and 4.0, $\frac{\rho_p}{\rho_f}$ was kept between 100 and 3100, and φ_p between 2.0% and 22.4% when large particle arrays were used. For larger particle arrays the setup is similar to the 41 particle cases; whose setup is seen in Figure 15. The shock wave was placed at 5 units from the left wall and traveled to the right.

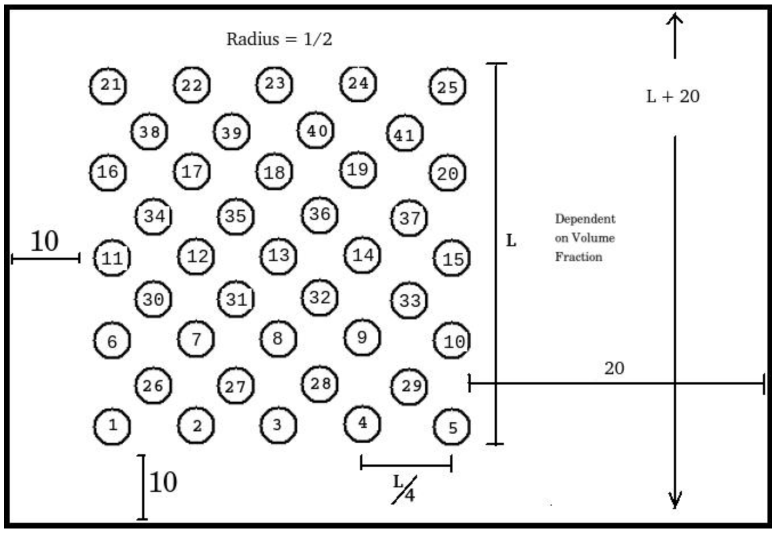


Figure 15: 41 Particle array setup

Assumptions

Because the physics of the problem certain assumptions can be made to simplify the problem without sacrificing accuracy of results. The following list contains the assumptions used and what they entail:

- The forces of gravity are negligible; the weight of each particle and movement affected by gravity and buoyancy are neglected in comparison to the drag forces.
- The fluid phase behaves as an ideal gas; the equation of state is the same as the ideal gas law.
- The gas and particles are calorically perfect; the specific heat values are constant for both phases.

- The solid particles are perfectly rigid; they undergo no deformation.
- There are no collisions; simulations stop when particle level-sets come in contact. In the macro-scale Lagrangian advection, particles are treated as points and may overlap.
- Thermal boundary layers do not develop in the time frame of shock-particle interaction; therefore adiabatic particle surfaces are assumed, thermal conductivity is set to zero.
- Kinetic boundary layers do not develop in the time domain; the model is inviscid, dynamic viscosity is ignored, no particle rotation occurs.
- Particle size is much large than molecular scale; Brownian motion is ignored, no random particle motion exists, intrinsic properties remain constant.
- Particles are inert; no chemical reaction occurs at boundaries.
- Remaining assumptions are case specific; e.g. moving/non-moving particles, perfect symmetry, etc.

Governing Equations

The method used solved a set of a governing set of hyperbolic equations for compressible fluid flow. These governing equations when simplified and placed in conservation form in Cartesian coordinates are:

$$\frac{\partial \vec{U}}{\partial t}+\frac{\partial \vec{F}}{\partial x}+\frac{\partial \vec{G}}{\partial y}+\frac{\partial \vec{H}}{\partial z}=\vec{S} \qquad \text{Equation 19}$$

where,

$$\vec{U}=\begin{pmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ \rho E\end{pmatrix},\ \vec{F}=\begin{pmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ \rho uw\\ u(\rho E+p)\end{pmatrix},\ \vec{G}=\begin{pmatrix}\rho v\\ \rho uv\\ \rho v^2+p\\ \rho vw\\ v(\rho E+p)\end{pmatrix}\ \text{and}\ \vec{H}=\begin{pmatrix}\rho w\\ \rho uw\\ \rho vv\\ \rho w^2+p\\ w(\rho E+p)\end{pmatrix}$$

In the equations above,

$$E=e+\frac{1}{2}\left(u^2+v^2+w^2\right) \qquad \text{Equation 20}$$

where E is the total internal energy and e is the specific internal energy. For the Euler equations in Cartesian coordinates, the source term $\vec{S}$, is set to zero. The extension of the methodology to the Navier-Stokes equations is fairly straightforward. Closure for the governing equations can be achieved by utilizing a stiffened equation of state,

$$P=\rho e(\gamma-1)-\gamma P_\infty \qquad \text{Equation 21}$$

where γ is the specific heat ratio and P_∞ is a material dependent constant. Under the assumption of an ideal gas, we would then have $P_\infty=0$ and $\gamma = c_p/c_v$.

For stiff fluids such as water, the specific heat ratio and the material dependent constant would assume the values of 5.5 GPa and 6.13 GPa, respectively. Lastly, from the definition of the speed of sound and using the stiffened equation of state, the speed of sound can be calculated by

$$c=\sqrt{\frac{\gamma(P+P_\infty)}{\rho}} \qquad \text{Equation 22}$$

Immersed Boundary Method

For the consideration of boundary conditions at an interface, an immersed boundary method is used. The algorithm used is an Eulerian-Lagrangian algorithm for interface tracking in three dimensions, otherwise known as ELAFINT3D. The ELAFINT3D code utilizes a sharp interface treatment method as described by Sambasivan.(90) The sharp interface treatment requires continuous tracking and representation for the interface surface. To represent the embedded interface surfaces, Level-sets were used, first introduced by Osher and Sethian.(91)(92) The level-set is simply an intersection between a defined level-set field and the working plane. The level-set field is advected using the level-set advection equation:

$$\frac{\partial \phi_l}{\partial t} = \vec{V}_l \cdot \vec{\nabla} \phi_l = 0 \qquad \text{Equation 23}$$

where φ_l represents the level-set and $\vec{V}_l$ represents the level-set velocity field for the l^{th} embedded surface. For the solution methodology, a fourth-order essentially non-oscillatory scheme for was used for spatial discretization and a fourth order Runge-Kutta time integration was used to solve the level-set advection equation. The value of the level-set field at φ_l any point is the signed normal distance from the l^{th} interface with $\varphi_l \leq 0$ inside the immersed boundary and $\varphi_l \geq 0$ outside. The interface is implicitly determined by the zero level-set field defined when $\varphi_l = 0$, and where the contours represent the l^{th} immersed boundary. The normal vector and the curvature at the interface can be computed from the level-set field by,

$$\vec{n}_l = \frac{\vec{\nabla} \phi_l}{\left\| \vec{\nabla} \phi_L \right\|} \text{ and } \kappa = -\nabla \qquad \text{Equation 24}$$

Boundary Conditions

To handle the jumps in the mass, momentum and energy fluxes along with the material properties across the interface, the tracked interface will have to be coupled with the flow solver to insure an accurate depiction. In the ghost fluid method, this translates to suitably populating the number of ghost points.(90) At the interface of a solid body immersed in a compressible flow, the following boundary conditions were applied for velocity, temperature and pressure fields. For no-penetration for normal velocity:

$$v_n = U_n \qquad \text{Equation 25}$$

where U_n is the center of mass velocity for the embedded rigid object. To satisfy the slip condition for the tangential velocity:

$$\frac{\partial v_{t_1}}{\partial n} = 0 \text{ and } \frac{\partial v_{t_2}}{\partial n} = 0 \qquad \text{Equation 26}$$

To satisfy the adiabatic temperature condition:

$$\frac{\partial T}{\partial n} = 0 \qquad \text{Equation 27}$$

To keep the normal force pressure balance:

$$\frac{\partial p}{\partial n} = \frac{\rho_s v_{t_1}^2}{R} - \rho_s a_n \qquad \text{Equation 28}$$

and

$$v_n = \vec{V} \cdot \hat{n},\ v_{t_1} = \vec{V} \cdot \hat{t}_1,\ v_{t_2} = \vec{V} \cdot \hat{t}_2 \qquad \text{Equation 29}$$

where v_n is the normal velocity, v_t is the tangential velocity in the interface referenced curvilinear coordinate, $\vec{V}$ is the velocity vector in the global Cartesian coordinate, $\hat{n}$, $\hat{t}_1$, $\hat{t}_2$ are the normal and tangential vectors, R is the radius of curvature and a_n is the acceleration of the interface; the set of boundary conditions that govern the behavior of the flow near the embedded solid body and must be enforced on the real fluid by suitably populating the corresponding ghost points. (90)

Verification

To insure the reliability of our code, the computed drag force obtained was non-dimensionalized using the same parameters as Drikakis et al. [(88)] The comparison of the non-dimensional drag force is shown in Figure 16. A visual comparison between the results obtained from the present approach and that of Drikakis et al. is shown in Figure 17 using isodensity lines. The transient drag curves produced by Drikakis et al. and those produced by the present calculations show minimal difference in peak magnitude and are rather similar, even though Drikakis et al. employed Navier-Stokes computations for rather modest Reynolds numbers for their calculations. The similarity of the drag behavior for the Euler and Navier-Stokes computations supports the present inviscid computations for the shock-particle interaction, particularly for the high Reynolds numbers that apply to the particles considered by Boiko et al and targeted in the present work.

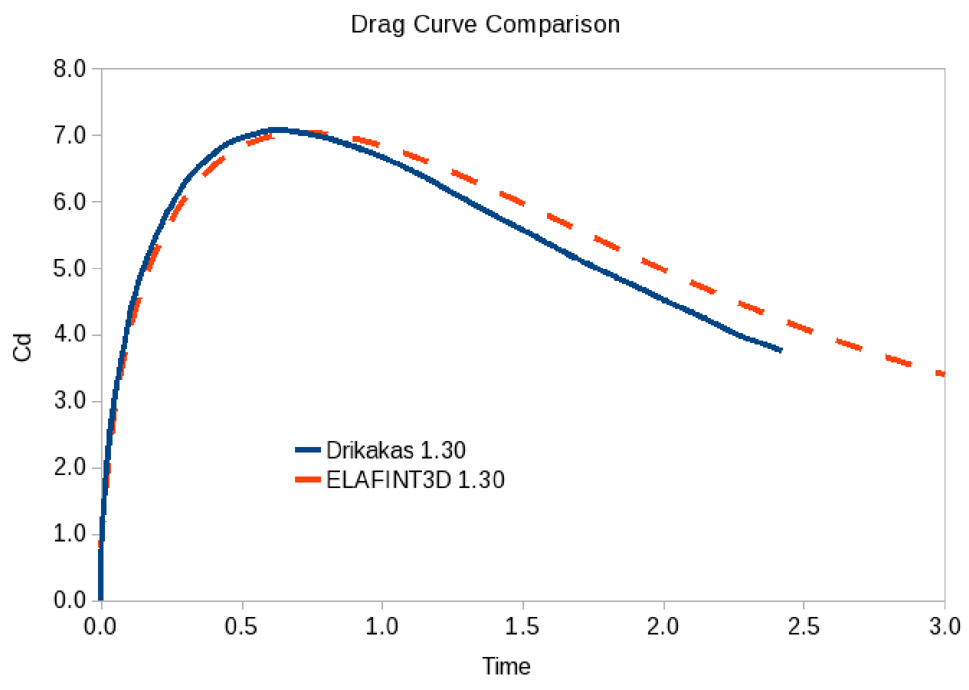


Figure 16: Numerical Drag Comparison

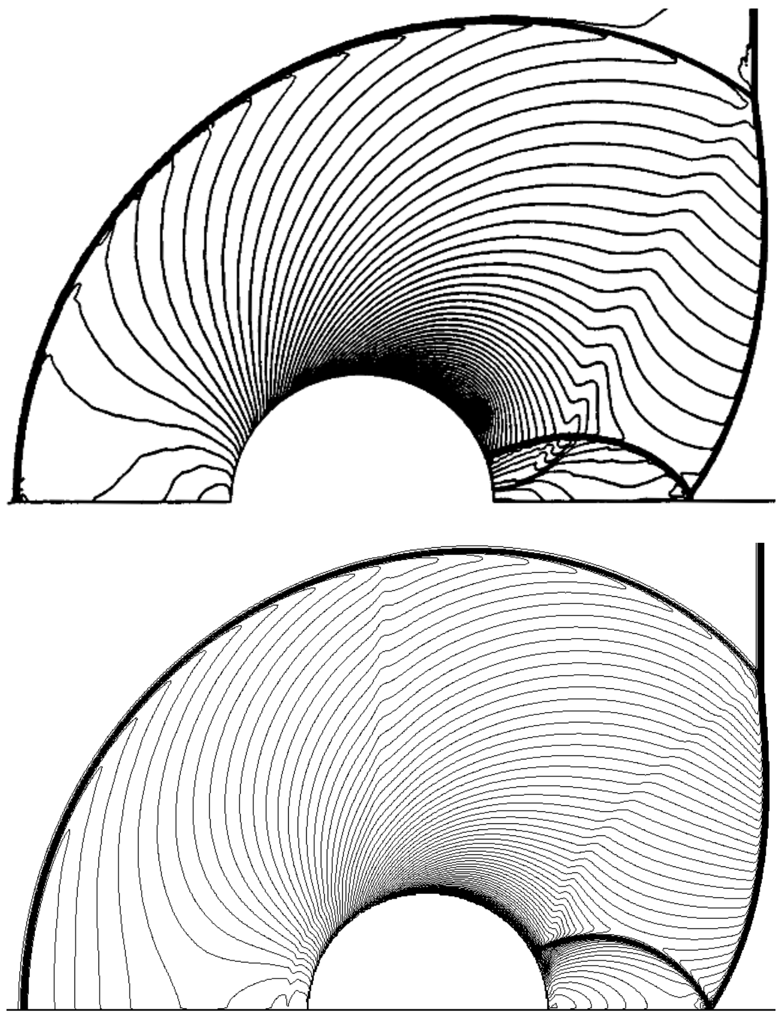


Figure 17: Isodensity Contours. Drikakis (top), ELAFINT3D (bottom)

In the next chapter, we employ the computational approach outlined above to compute shocked flows around single particles and particle clusters and describe the process of training the ANN to assimilate the particle drag function.

CHAPTER V:

RESULTS

Single Particle Cases

The ELAFINT3D code was first used to test a cylindrical particle in a fluid flow during varying conditions. This included experiments of post-shocked flow, a shocked stationary particle, and shocked moving particle. Later on, cases of shocked particle arrays with large number of particles were examined. The single particle tests were set up to illustrate the evolution of data processing the ANN needed to learn in an order of increasing complexity.

Post-Shocked Flow

The first experiment for testing the capabilities of the neural network was the case of post-shocked flow around a cylinder. The goal of the post shocked flow data set is to analyze whether or not the artificial neural network is capable of accurately predicting the drag curve which is simple, positive in value, and derivative is of constant sign. In the post shocked particle case, the particle is placed into a flow field after the shock wave has already passed. An example is shown as a Schlieren image where the shock is past the cylinder moving toward the right in Figure 18. No reflections or slip lines occur, thus the force on the cylinder is only due to the impending flow and bow shock development.

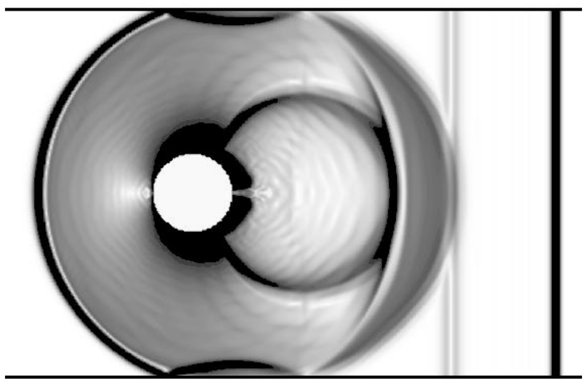


Figure 18: Post Shocked Particle Schlieren Image

For the post-shocked flow experiment, the cylindrical particle was allowed to move based on fluid forces. The shock wave was placed after the cylinder so that the horizontal force imparted was an effect of the fluid interaction and not the shock. The particle immediately begins to feel drag force and the incident shock wave is uninterrupted behind the particle. The boundary conditions set for this experiment are the same as all of the following single particle trials. The domain was set with the inlet on the left side and the outlet on the right. The top and bottom of the domain was set to a reflective boundary condition.

This experiment was analyzed at Mach numbers 1.4, 1.5, 1.7, 1.8, 2.0, and 2.2. The drag curve for Mach = 1.8 was left out so one could test the capabilities of the artificial neural network and its predictions. The drag forces were non-dimensionalized and calculated every time step. The curves produced by the ELAFINT3D code were made into the training data set provided to the artificial neural network. These curves are shown in Figure 19.

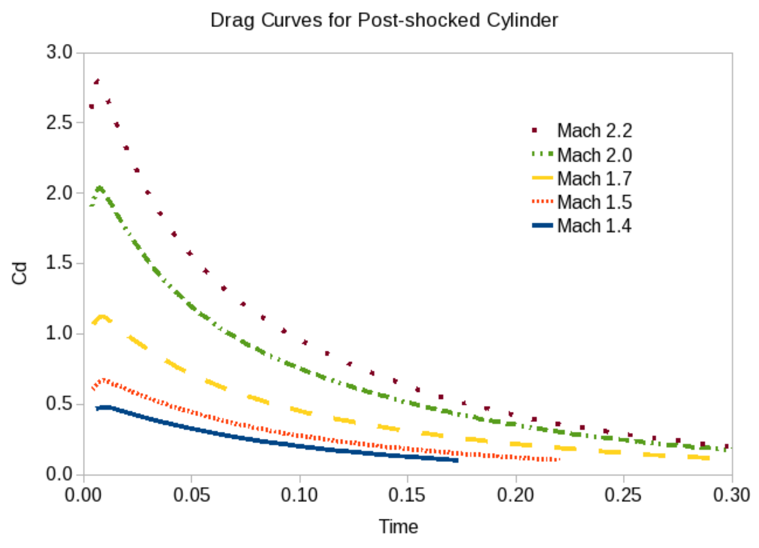


Figure 19: Drag Curves for Post-Shocked Cylinder

The neural network was setup to read the drag data and fit a curve to the data using 20 neurons and 1000 iterations. With the training data normalized, the neural network was capable of performing all the iterations in less than 30 seconds, while the time to run the full numerical cases, averaged around 40,000 seconds. The prediction of the drag curve at M = 1.8 as well as the remaining case of Mach 1.8 run through numerical methods are plotted together in Figure 20.

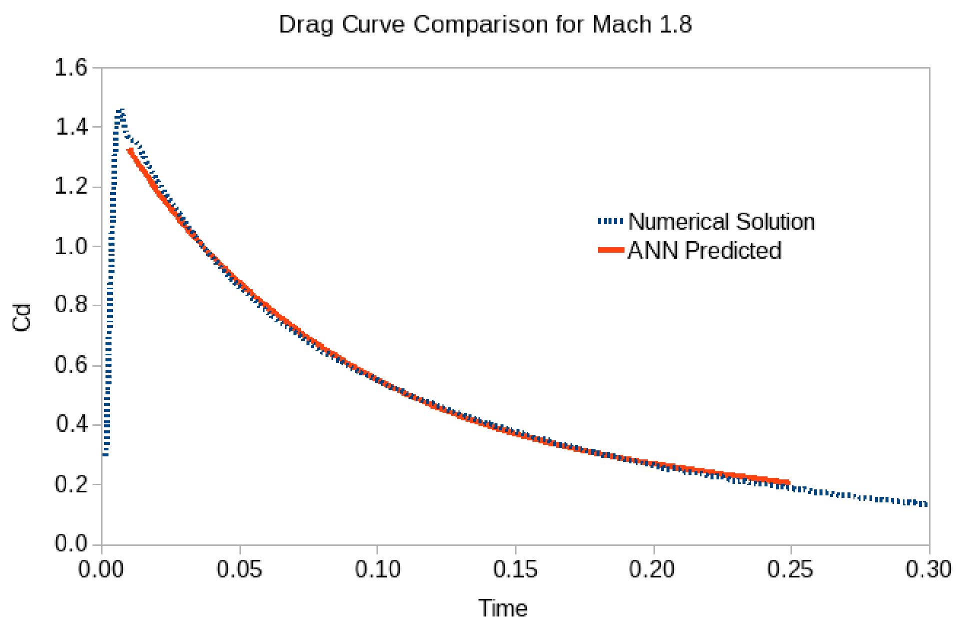


Figure 20: Post-shocked ANN prediction

The neural network produced a drag curve that agreed well with the computed drag curve. The simple nature of the drag curve compared to the time required by computational fluid dynamics illustrated the necessity of a faster method to extract such data. To better understand the prediction abilities of an artificial neural network and its application to fluid modeling, a more realistic example is chosen for the next section.

Stationary Particle

For the second test, the particle was held stationary and then hit with a shock. The boundary conditions were set the same as the post-shocked particle case except the lower wall set as symmetry. A grid domain of 500 by 250 cells was used for the drag curves calculated from the ELAFINT3D code. This was to match and verify the results by the ELAFINT3D code to those of Drikakis [(88)] as seen previously. The initial starting distance for the shock wave was set more than the radius of the cylinder away from the cylinder itself. The shock was allowed to impact the cylinder and continue to travel as data for horizontal force was recorded over time. A Schlieren image of the one of the cases is shown in Figure 21.

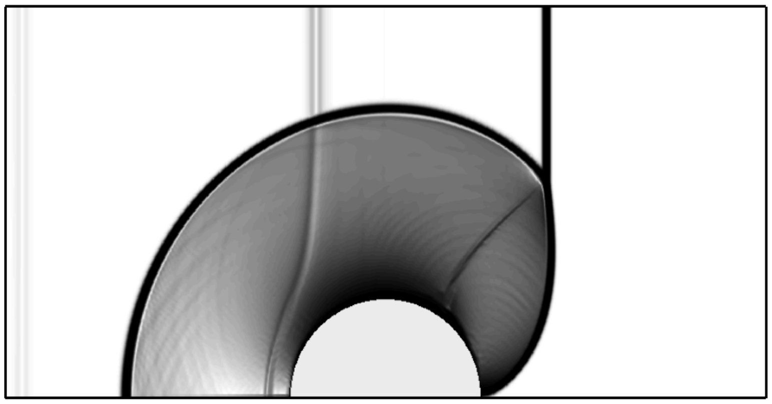


Figure 21: Stationary Particle Schlieren Image

With a smaller domain size, it would be reasonable to test the effect of grid size and the use of local mesh refinement. For the fine grid, the number of grid cells was increased by four times with the grid sizes half the original. For the local mesh refinement two levels of refinement were used to provide grid cells near the interface with edges a fourth of the original. It was discovered that both the finer grid structure and the use of local mesh refinement show some differences. The differences were rather negligible given the previous error for the neural network's prediction, and in the interest of time, the remaining cases were carried out with the original grid size. The resulting drag curves from the ELAFINT3D code at Mach numbers ranging from 1.1 to 2.6 are shown in Figure 22.

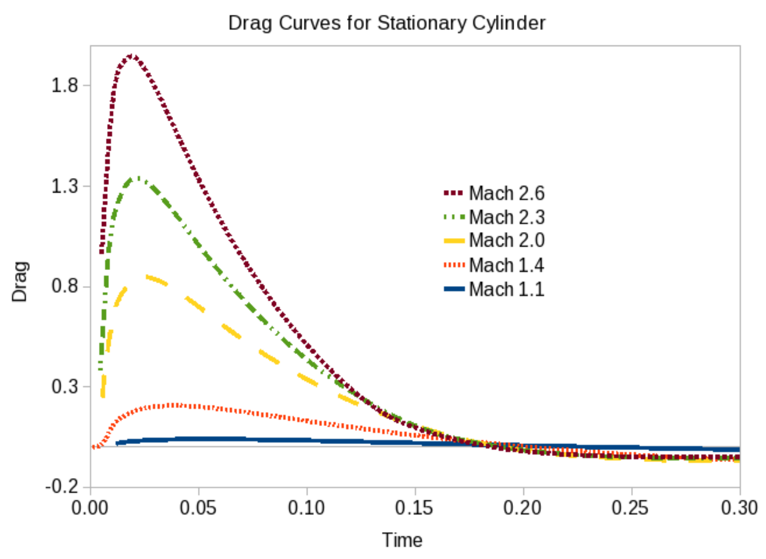


Figure 22: Drag Curves for Stationary Cylinder

The ANN was trained using this data set and the same number of neurons and number of iterations were used. The same order of computational time was observed as in the post-shocked flow calculations. The prediction curve of the neural network as well as the calculated transient drag curve is displayed in Figure 23.

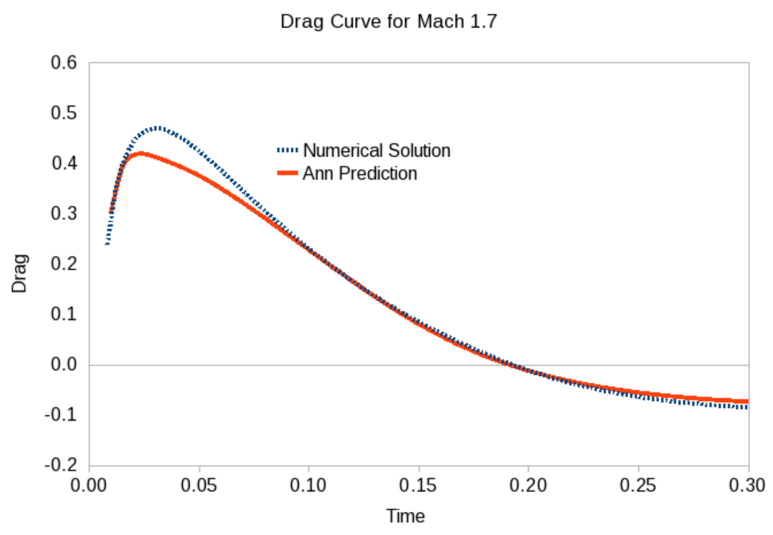


Figure 23: ANN Drag Prediction, Mach 1.7

The neural network was capable of matching the curve even into the negative force domain. The negative drag force arises when the incident normal shock traverses to the rear of the cylinder and a reflected bow shock has formed at the front of the cylinder, which leads to a higher pressure at the rear for a short period of time. However, in this case, the peak value of the drag was underestimated by the neural network. The cause of this is due to the neural network's activation function, and the summation of which is fitting a series of sigmoid functions to the curve. With data evenly distributed, a small number of data points exist near the peak. The unbalanced set causes the neural network to spend more time fitting to the rest of the curve than the peak. Another reason is that the neural network is attempting to fit with a global array, thus the overall prediction curve will be similar to a smoothing function and reduce peaks. The sharper the peak, the less likely the neural network will capture an accurate depiction. For a moving particle these sharper peaks do occur. Several solutions including the use of wavelet basis functions, neural network expansion, multi-resolution and segmentation exist; these will be discussed in detail later.

Moving Particle

For the moving particle problem, the boundary conditions, the initial conditions, domain size, and particle size remained unchanged from the previous experiment. The chosen Mach numbers allow for easier comparison to conditions used in various experiments.(73) The artificial neural network was set up to segment the drag curves in time to facilitate more customized fitting in the respective segments. This would allow for a better fit to the drag curve. The training data provided to the artificial neural network is shown in Figure 24.

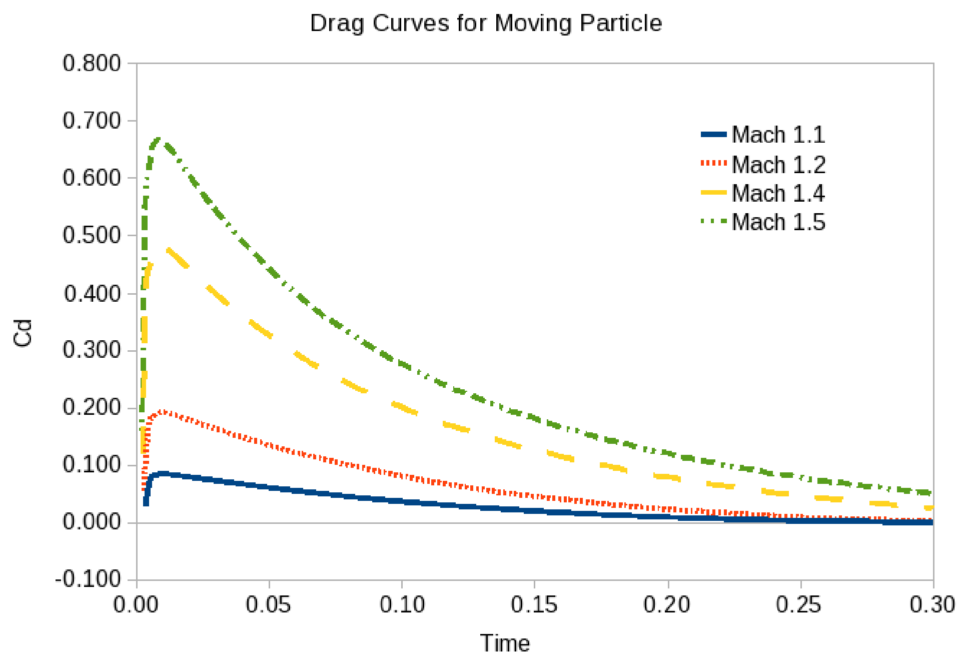


Figure 24: Moving Particle Drag Curves

The total training time for the neural network was still under 30 seconds because the amount of data per iteration for each partition of the neural network was reduced. The root mean square error was significantly reduced and was less than 0.5% for 700 data points in the later time section. The resulting prediction output was also segmented according to which partition of the artificial neural network was responsible for learning the curve characteristics of the function. The resulting 40 neuron, partitioned artificial neural network produced a remarkably good prediction as shown in Figure 25.

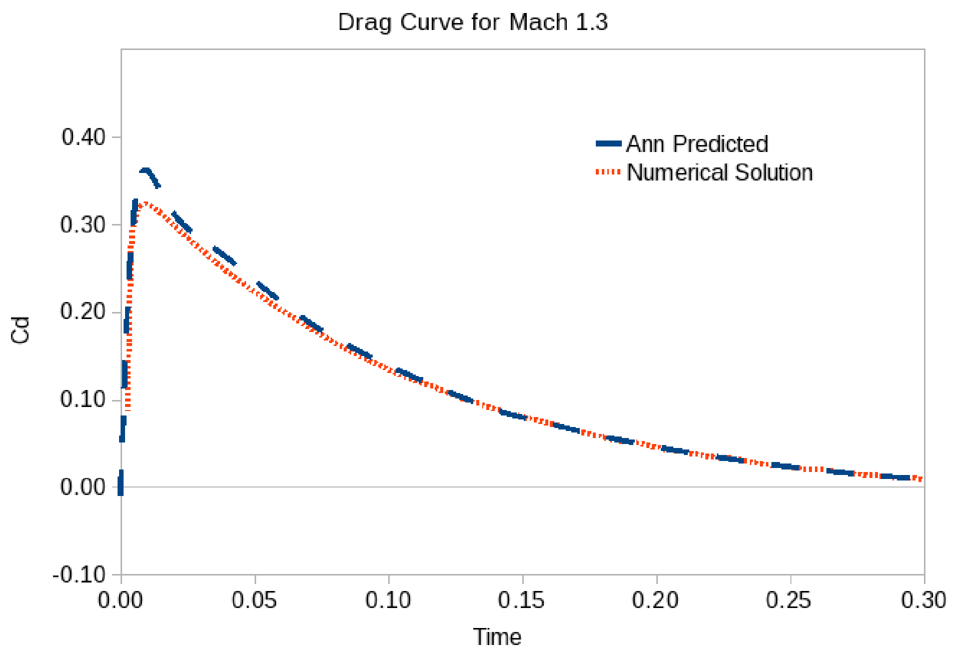


Figure 25: ANN Prediction, Mach 1.3

Multiple Particle Cases

The drag versus time curve for a single particle is fairly easily predicted by an artificial neural network with only one interacting shock wave. It may be necessary to implement another method of data assimilation to describe more complicated functions and drag curves. The previous experiments grew in difficulty to examine the different properties of supersonic fluid flow around a cylindrical particle. From the post-shocked experiment, the neural network observed that the drag increased as the Mach number increased and the drag went down over time. The drag force of the stationary particle displayed negative values. With the cylinder moving, the drag peaks became more prominent. A single neural network is able to derive the drag correlations from numerical methods given a single particle. When there is particle laden flow field, a new approach is needed to extract the drag correlations. The necessity to analyze the different scales using multi-resolution analysis may become apparent. This becomes even more important to data processing when the correlations are dependent on multiple parameters, as expected for the present problem. Another experiment was setup to test the ability of a multi-resolution augmented artificial neural network, or MRAANN, in learning more complicated flows.

Multiple Moving Particles

Here a dramatically different approach was used. Due to the increase in complexity for a particle array, multi-resolution analysis(93) was first performed on the data to examine if there are any correlations relating to each particle and varying φ_p. The boundary conditions on the top and bottom were changed to symmetry boundary conditions to simulate an infinite particle array. One constant Mach number was used; for all the experiments of moving particles, M = 1.22 which is commonly used.(73)(89) Ten particles were staggered, in the arrangement in Figure 26.

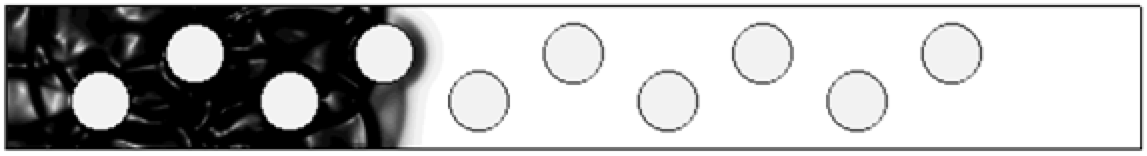


Figure 26: Schlieren Image of 10 Particle Array

To test other more complicated correlations for drag forces, more variables were introduced. In the multiple particle experiment, the MRAANN would need to learn the drag correlations for ten different particles, and different φ_p. To change the φ_p, the radii of the cylinders varied. The radii used were 0.25, .030, 0.35, and 0.40 units. Particles were set one ahead of each other 1 unit apart. The boundary conditions on the top and bottom edges were set as symmetry conditions to simulate an infinite vertical array. With the cylinders set close together, reflected shock waves and expansion waves were sent in every direction. The results are several drag curves with many peaks and troughs; this type of transient drag curve is typical of arrays and arises due to the multiple shock reflections and interactions between these shocks and the particles.(94) The drag curve for the third particle in the array is shown in Figure 27.

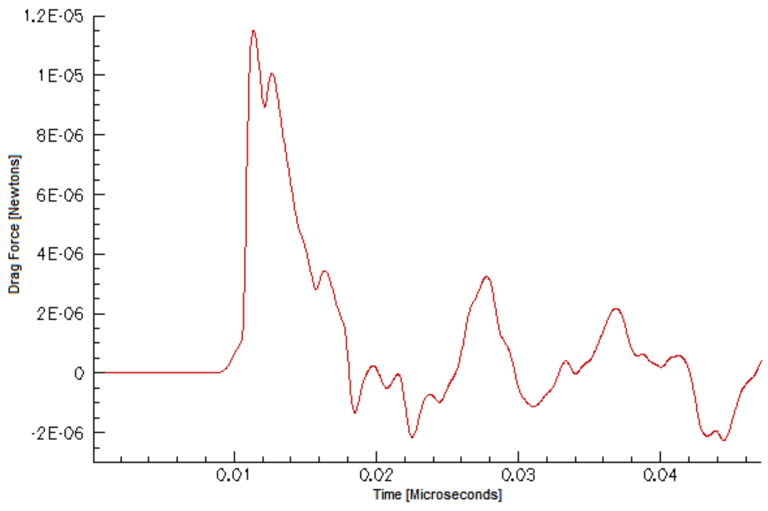


Figure 27: Drag Curve for 3rd Particle in Array

To aid the learning process for the neural network, the multi-resolution analysis process performed 6 levels of multi-resolution transforms, using the algorithm developed by Neal Grieb in his MS thesis (95). Many styles of multi-resolution neural network have been attempted.(53)(96) The basic idea of multi-resolution neural networks is to partition the ANN to allocate neurons for more complex learning. Boubez and Peskin (96) demonstrate Receptive Field Partitioning in their work. For ours, the ANN expansion would occur globally. The ANN would begin with an initial set of weights and neurons to learn the coarsest level and then pass those weights to new neurons to help learn the next transformation level. A pseudo-code of this process is provided in the appendix. The coarsest level of the multi-resolution transform of the drag curve shown in Figure 25 is shown in Figure 28 as a dashed line. The neural network was given the coarse training data and was able to match it well. The predicted curve is shown in Figure 28 as a solid line.

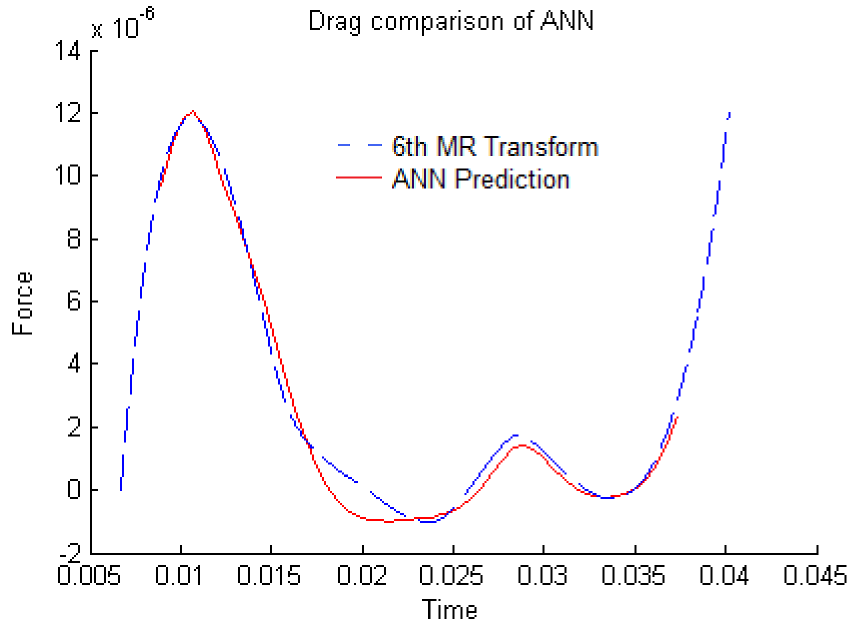


Figure 28: MRT and ANN Predicted Drag Curve

Each time the neural network learned a level, the training speed of the neural network would increase. The amount of time it took to learn a new segment decreased along with the error also. Shown in Figure 29 are the learning curves of the artificial neural network without any segmentation or multi-resolution analysis represented by the dashed line and the learning curve with segmentation and multi-resolution analysis represented by the solid line.

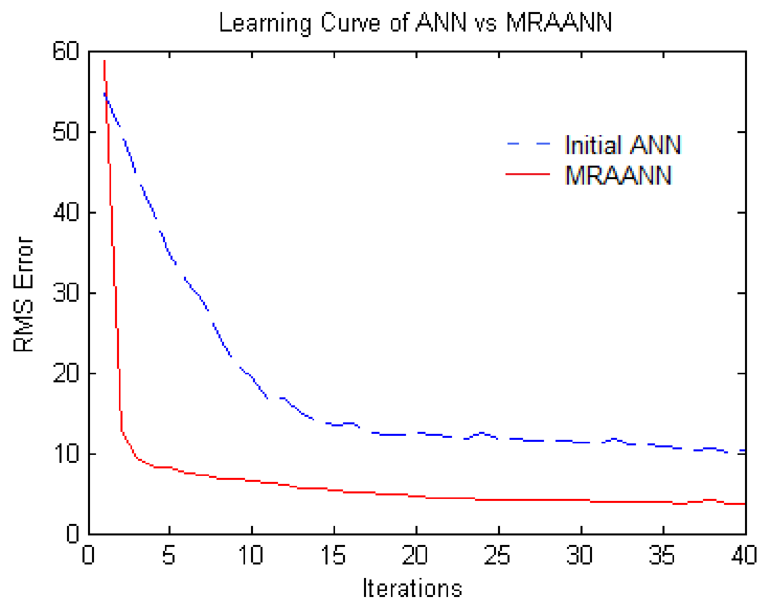


Figure 29: Learning Improvement

One can see that there was a decrease in error aided by the multi-resolution analysis (MRA). Once the MRAANN was able improve upon the function fitting abilities, it needed to be tested in full. 6 levels of multi resolution transforms were performed with 32 neural network segments. This correlates to roughly 2,000 iterations with 500 hidden neurons. By probing the MRAANN, a 3d hyper surface was able to be constructed to visually display the drag correlation with both time and φ_p. This surface is displayed in Figure 30.

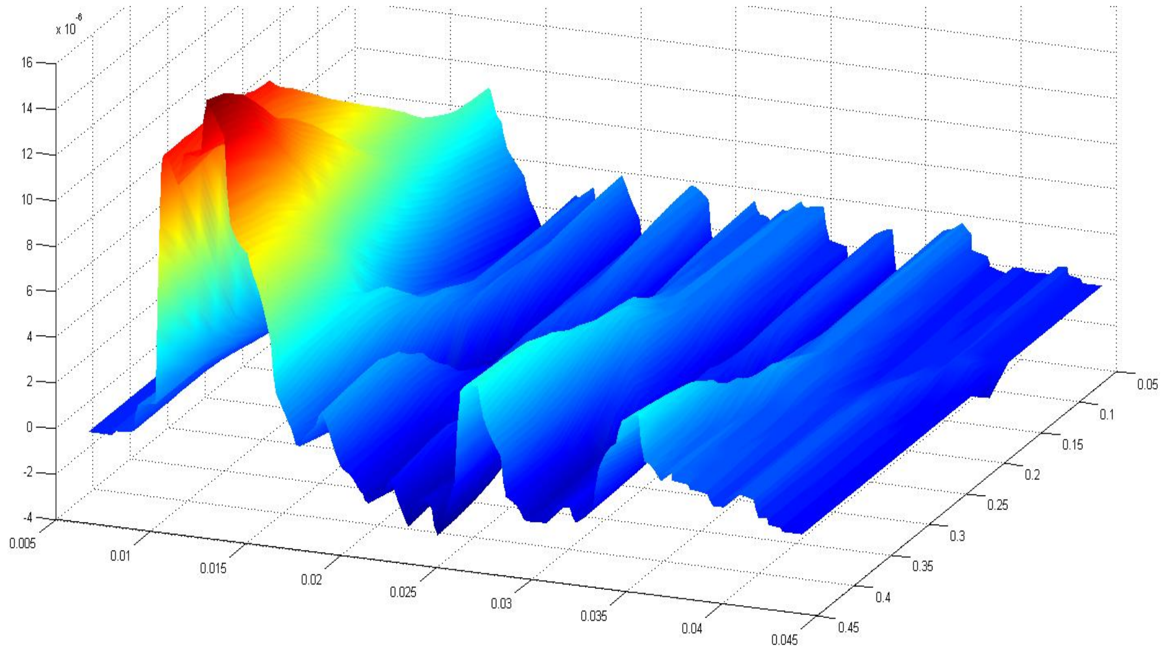


Figure 30: 3D Drag Curve Hyper Surface

The particle of radius 0.3 was left out of the training set for the MRAANN to predict what the drag curve would be. Three sets of training data was provided to the MRAANN to learn at each of the levels, for radii of 0.25, 0.35, and 0.40. The training data, numerical solution, and predicted drag curve are displayed together to observe overall characteristics in Figure 31. The training sets are displayed as the solid lines and show the progression in drag development. The dotted line is the actual drag curve obtained from the flow simulation while the dashed line is the prediction from the MRAANN. With all the data together, one can observe the difficulty in learning the complex behavior of particle shock interaction.

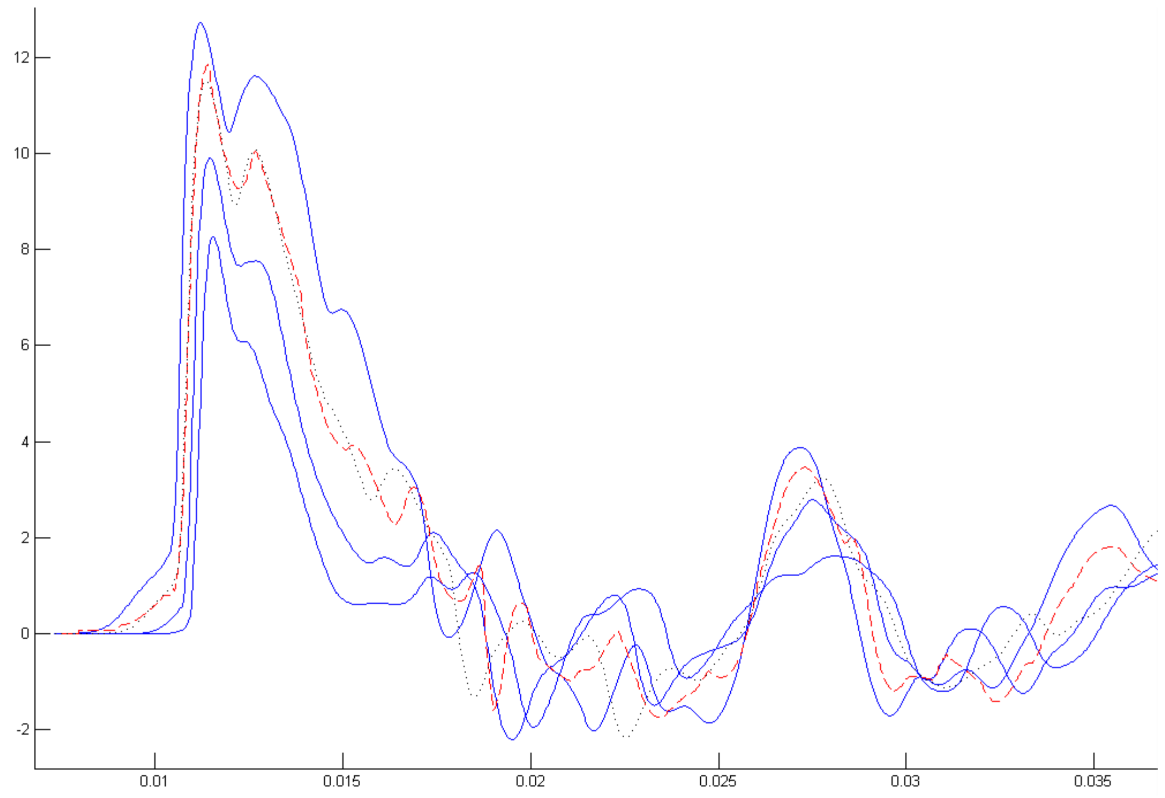


Figure 31: Training Data, ANN Prediction (dashed) and Actual Drag Curve (dotted)

Without seeing the data prior to the ANN prediction, it is difficult to tell which of the broken lines the original set is and which the ANN prediction is. The MRAANN is remarkably accurate in reproducing the data set with intermediate peaks and troughs included. There are many variables affecting the drag curve that cannot be explicitly defined by any function. Even with the complex behavior entailed by several reflected shocks in a regular array of spheres, the MRAANN is capable of adapting to the local maxima and minima of the drag versus time curves. However, note that in a dusty gas, particles are not regularly arranged and the question remains how one could extract data for such disordered arrangements. The solution to this problem will be addressed in the following.

Multiple Particle Clouds

In order to obtain a general drag curve with characteristics that could be applied to any particle embedded in a cloud, there needed to be data obtained from many particles in many possible arrangements. The best way to obtain data like this was to run simulations of randomly seeded clouds and to define a “representative particle (RP)” embedded in the flow; much as in the case of “representative elementary volumes” (RVEs) employed in volume-averaged formulations of multiphase flows One way to define such representative particles is to locate them at the center of a cloud of particles; this avoids edge effects and wave reflections from domain boundaries. The representative particles for one particular case are illustrated by the outline in Figure 30. To ensure the proper tracking of the same centralized particles, a particle array was first formed and then the particle centers were perturbed. The boundary conditions were set to simulate a shock tube for comparison to the works Boiko et al.(1), Tanno et al.(73) and Sun et al.(89) The left edge of the domain was set as an inlet, the right edge an outlet, and both the top and bottom edges were set as reflective boundaries. An example of the flow can be seen in Figure 32.

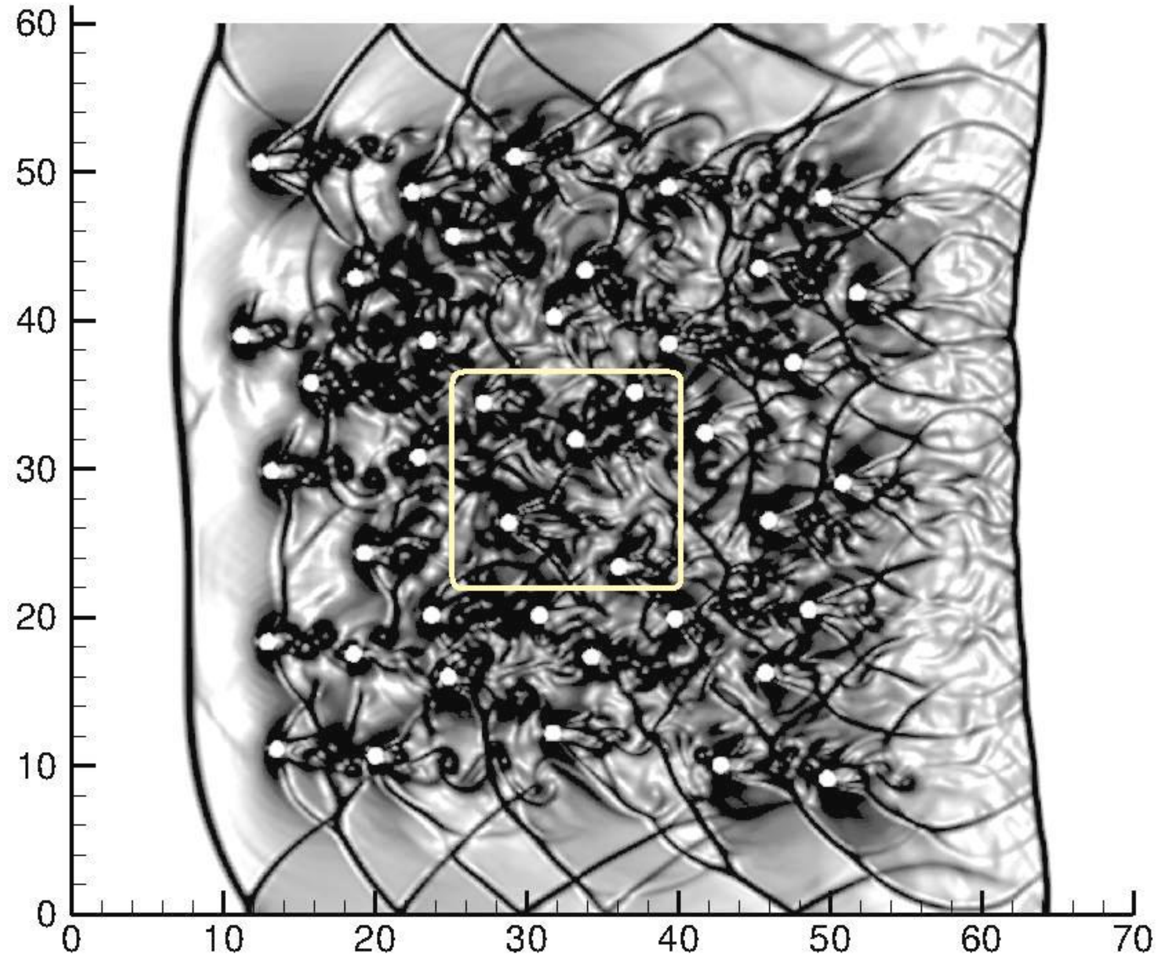


Figure 32: Schlieren Image and RPs of Shocked Cloud (Φ_{vol} = 22.4%)

The particles in this case number 41, each are seeded in a respective location where a 4 by 4 grid of 16 particles is embedded in 5 by 5 grid of 25 particles as seen in Figure 15. This enabled the users to easily code the location of each particle, yet create an array where every particle is staggered off the one directly in front. The slight randomization completed the task of attempting to simulate a random dispersal of particles while still being able to easily track a few. The few that were important enough were the particles embedded directly in the center of the array. The center particles experience a much more randomized collision of reflected shocks by the few rows and columns of particles behind and to each side. The drag curves for these particles were extracted by the integration of pressure over the level set boundary. The drag curves of 5 particles from the center of the cloud were then averaged. The results of the averaging of the drag curves for the RPs can be seen as the bold curve in Figure 33.

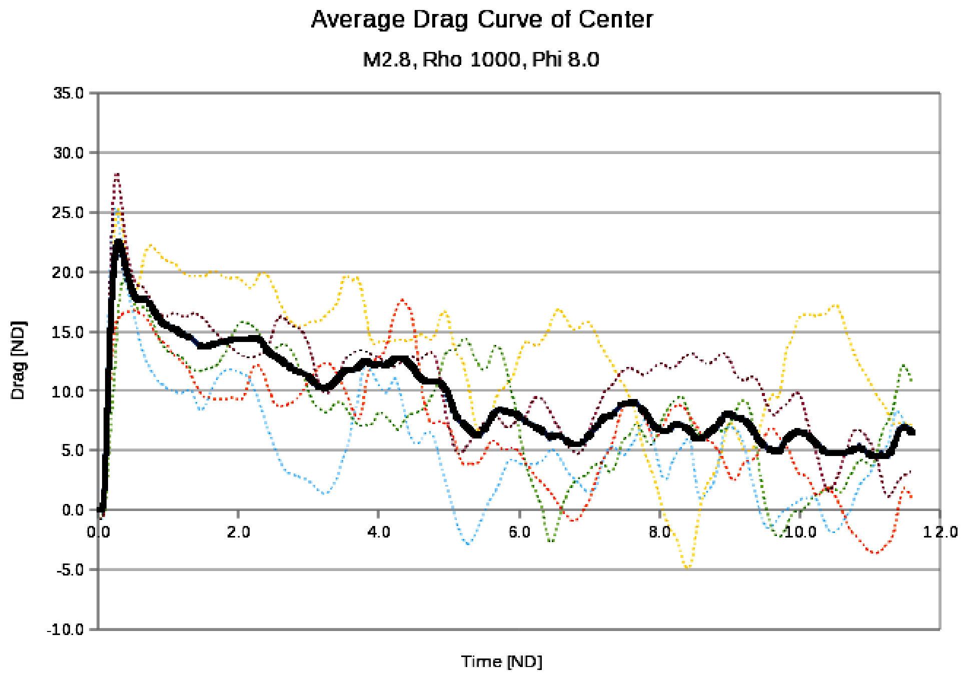


Figure 33: Averaged drag curve (41 random particle array, Mach 2.8)

Apart from the Mach number, the other parameters that can affect the behavior of particles in a cloud include the volume fraction of particles, the particle density relative to the fluid, particle shape, collisions between particles and viscous effects as controlled by the Reynolds number. The last three effects are not considered in this work as they are expected to have secondary effects in the initial phase of shock-particle interactions. Of the three parameters considered, namely Mach number (M), particle density ratio ($\frac{\rho_p}{\rho_f}$) and volume fractionφ_p, the effects of the φ_p variable are much more easily verified by direct viewing of the flow field. Upon comparison of the shape of the incident shock already passed over the particle arrays in Figure 32 and Figure 34, there is a very definitive concavity resulting from the passage over the more dense cloud. Such an obvious change (due to the higher impedance to shock propagation presented by the denser cloud) in the flow field would imply an equally large effect in drag force and thus the motion of the particles inside the cloud. Even though all other parameters were set the same, the dense cloud case in Figure 34 depicts more of a compressing of the cloud along the direction of flow.

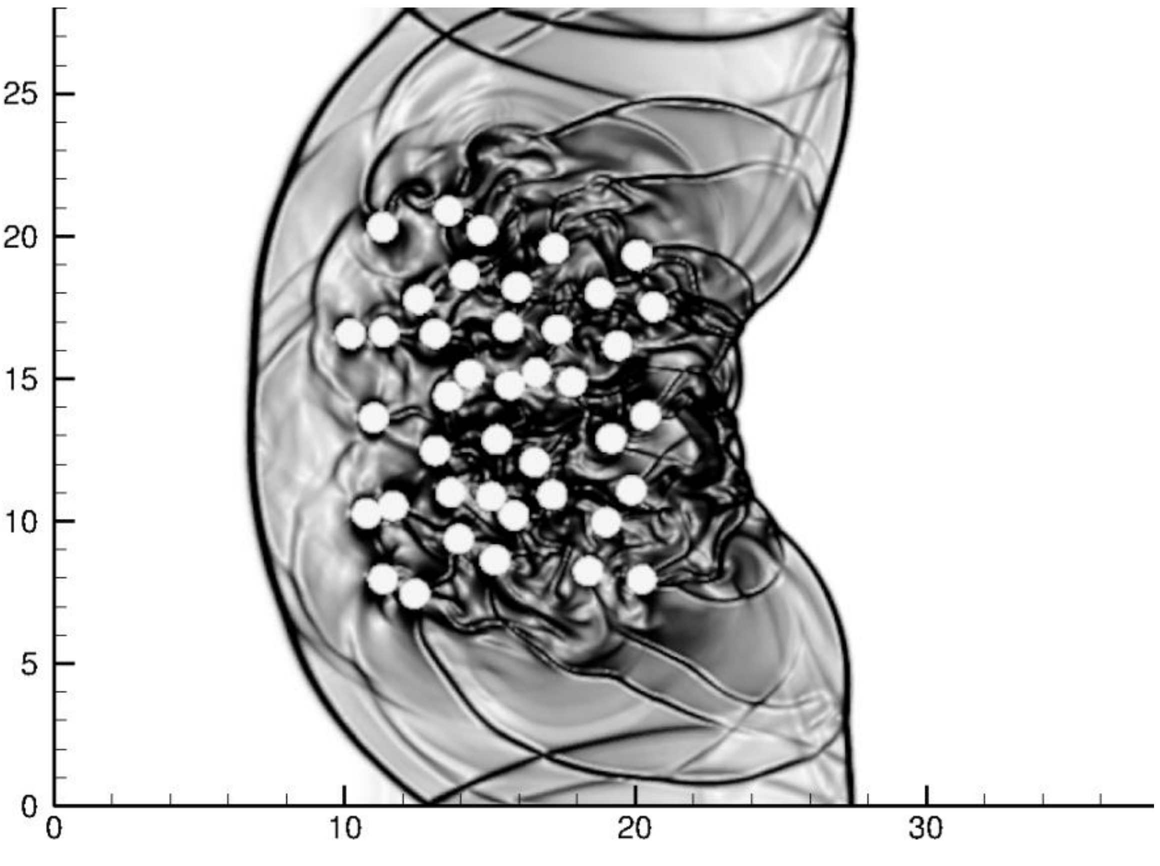


Figure 34: Schlieren image of shocked 41 particle cloud (φ_p = 22.4%)

Volume fraction, φ_p, is an important parameter that is different throughout any dust cloud and changes over time. It is only one of a few parameters that we chose to vary that we deemed to have the largest affect on particle motion. A comparison of the averaged drag curves (for the representative particle) for varied φ_p can be seen in Figure 35.

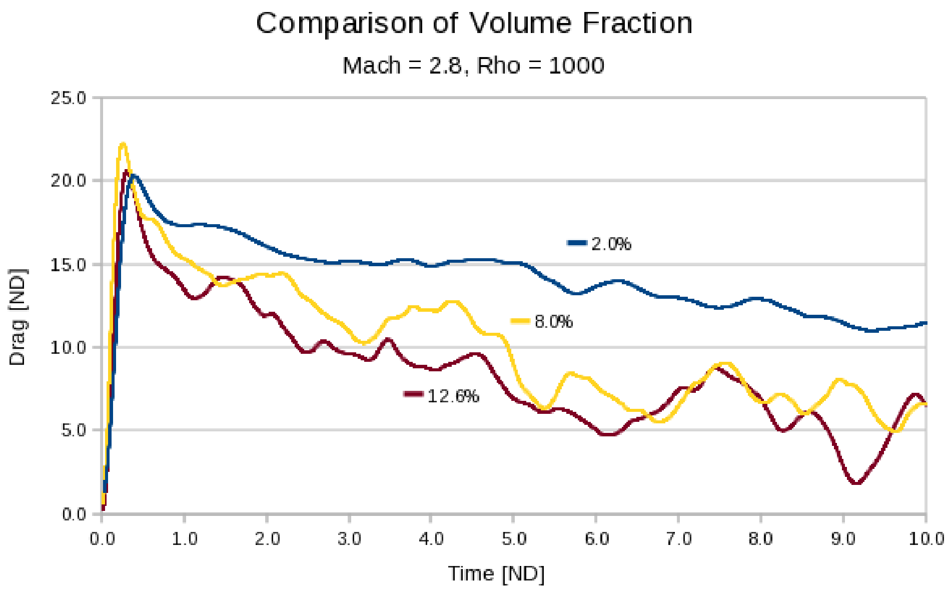


Figure 35: Comparison of the effect of φ_p

It may be deduced that the increase of φ_p decreases the impulse I_t delivered by the shock on a particle. The $C_{d_{max}}$ force experienced remains fairly constant but the decay of the force diminishes much more rapidly. This is due to the decreases in strength of the shock waves impinging on the RPs in the center of the cloud. Another more obvious variable that affects the drag force felt by shocked particles is the Mach number. In Figure 36 the effect of the Mach number is very obvious. As the Mach number increases, the $C_{d_{max}}$ felt rises dramatically.

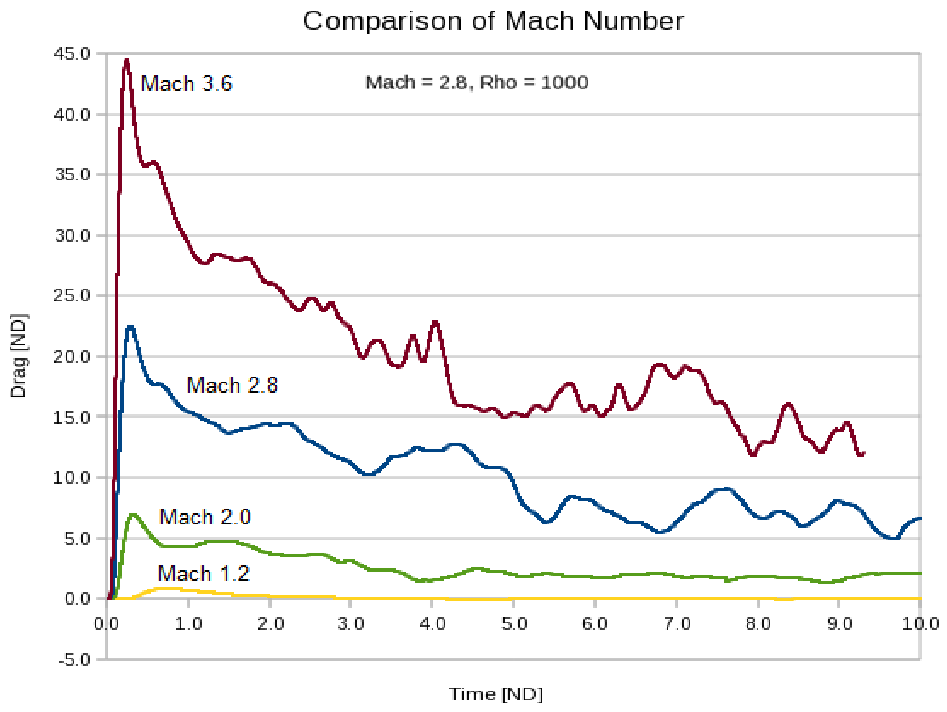


Figure 36: Comparison of the effect of Mach number

This dramatic rise is directly correlated to the shock strength in the high Mach number flow. It is the flow velocity that in turns defines the Reynolds number that most fluid codes based on the non-dimensional parameters. This is why so many drag laws depend on the Reynolds number, but the Mach number is a far more relevant parameter in the initial shock-particle interaction phase. It has already been shown that the main separating factor between Reynolds and Mach number is viscosity. It has also been shown that viscosity does not affect the drag force on a particle this early. This is why we believe that developing the variation of the drag on an RP with respect to the Mach number in an inviscid model is a better parameter for lifting drag information from the meso-scale to the macro-scale.

The next parameter examined in our experiments was that of the mass of the particle. In terms of non-dimensional variables, this correlates to the mass of the particle as a ratio of the density of the solid particle and the fluid surrounding it. Most of the experimental models of shock-particle interactions employed spheres made of acrylic and copper.(1) The medium used was air, and because of those models we chose to set $\frac{\rho_p}{\rho_f}$ near 1000. To encapsulate motion a little easier, we mainly varied $\frac{\rho_p}{\rho_f}$ lower. The comparison of drag forces are displayed in Figure 37. The data obtained shows correlations per each variable, the ANN will hopefully connect them together and engage its application to multiscale modeling.

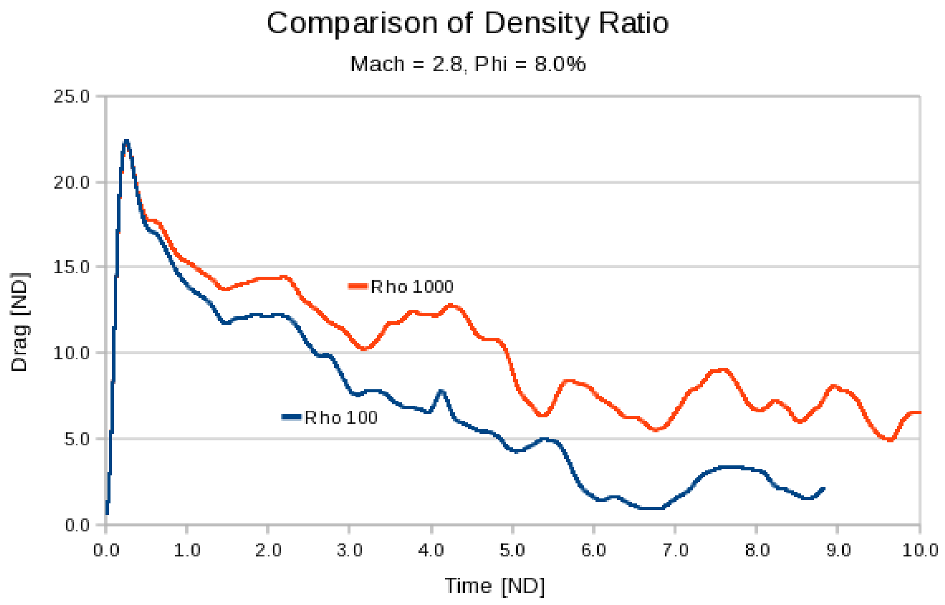


Figure 37: Comparison of the effect of $\frac{\rho_p}{\rho_f}$

CHAPTER VI:
APPLICATION OF ANN-BASED LEARNING TO MULTI-SCALE COMPUTATIONS

Information passage

To utilize the correlations obtained previously to use in multi-scale modeling, information must be lifted from the meso-scale. The transient drag curve is quite a lot of information to pass between scaling levels in multiscale modeling. With regard to the drag curves acquired, there were two important parameters needed for particle motion. Both Kosinska [97] and Kosinski [82] [83] showed that the linear motion of a rigid body even those immersed in shock waves can be found directly from Newton's second law and derivations of it. Newton's second law directly correlates force to mass and acceleration. To determine the speed and position we would need to know the momentum transferred and the rate of momentum transferred. The momentum and rate of transfer can be found as an expression of I_t and τ_r.

From Equation 11, we have a simple method of determining the total impulse, I_t, a particle would experience over time. Integrating Equation 11 from instant of shock impact to long times (when the inertia delivered by the shock has equilibrated particle motion, but still short enough that viscous effects can be neglected) results in:

$$I_t = C_{d_{max}} * \tau_r \qquad \text{Equation 30}$$

The maximum drag, $C_{d_{max}}$, is easily acquired, thus the next step be to fit the relaxation time, τ_r, to the drag curve of each case the ANN will learn from for Mach number, $\frac{\rho_p}{\rho_f}$ and φ_p.

Single particle motion

For each case presented, the quantified values for particle motion, $C_{d_{max}}$ and τ_r to attain I_t were found. The value of I_t and τ_r were found by numerical integration and fitting an exponential decay function by minimizing the error between the drag curve and the exponential function. One such fitting with the impulse highlighted can be seen in Figure 38.

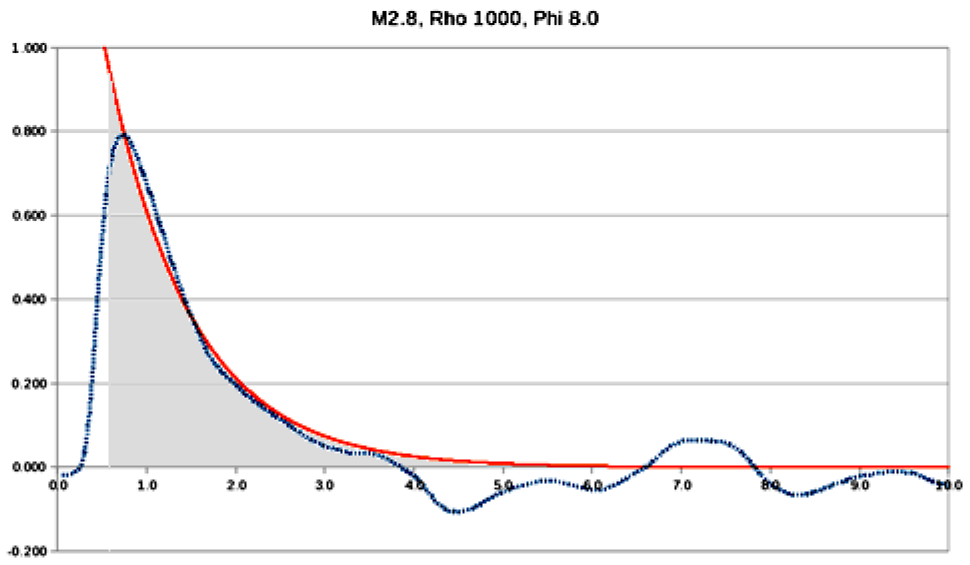


Figure 38: Exponential fitting to drag curve

We began with the data from our single particle cases because the drag curve fitting was simpler and straight forward without large errors due to the oscillations in drag. From that data we fed it into the ANN and obtained a hyper-surface to incorporate each variable. The 3 dimensional breakdown between two of the variables and there target parameter can be seen in the plots of Figure 39 and Figure 40

.

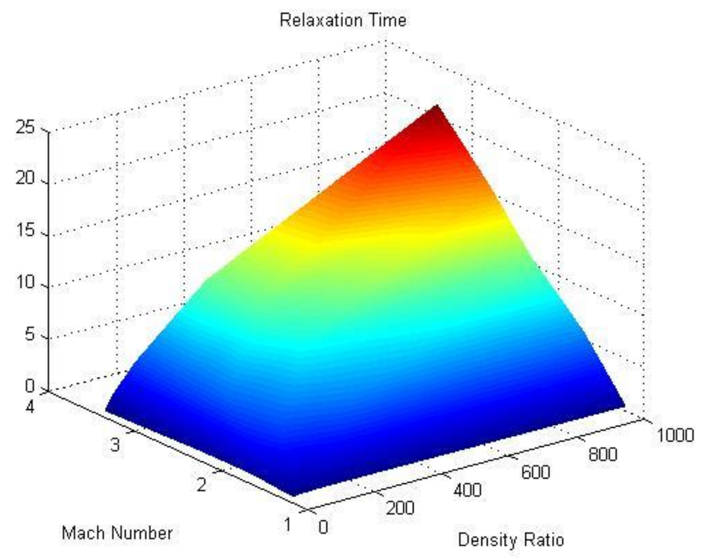


Figure 39: ANN predicted hyper-surface for relaxation

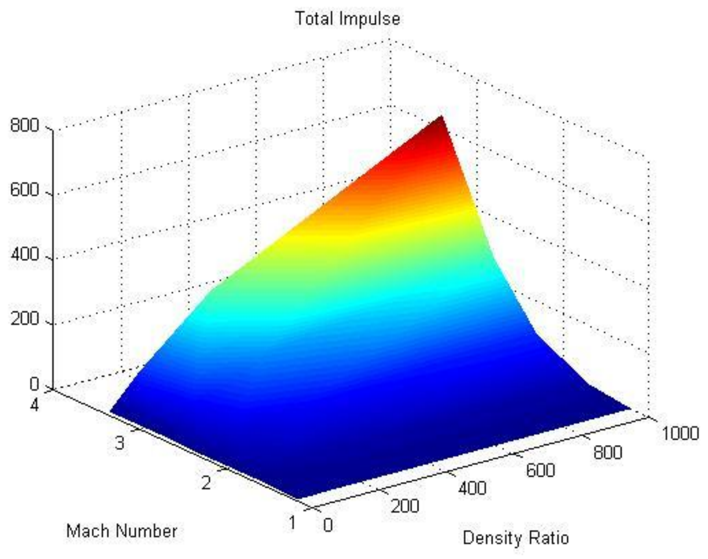


Figure 40: ANN predicted hyper-surface for I_t

It has already been determined, and one can see from the plots, that both the I_t and τ_r increase with Mach number. This has been shown many times before by other researchers. (78) (73) (89) It is interesting that the value of I_t actually gets steeper as the Mach number increases, making it a high order relationship. It is also important to note that both I_t and τ_r seem to approach zero near Mach 1. One can accredit that to the dramatic decrease in drag once relative velocity falls below the supersonic range. As for the effect that $\frac{\rho_p}{\rho_f}$ has, both I_t and τ_r level off toward higher values, as $\frac{\rho_p}{\rho_f}$ approaches the representation of an infinitely massive or stationary particle. For when $\frac{\rho_p}{\rho_f}$ approaches zero, both I_t and τ_r approach zero as a very small particle's motion should nearly behave the same as the fluid.

General Particle Motion

It is easy to see how a single particle would behave, but with multiple particles there occurs many complex reflective shock waves. To ensure that the general behavior of shocked particles is accurately learned by a neural network, data needs to be collected from many particles in a random orientation. Therefore the effect of a specific array setup would be diluted and more general values could be collected. The values of $C_{d_{max}}$ force and τ_r are still the two most important parameters that can be directly obtained from the ELAFINT3D code. For particle motion that occurs in a dusty gas, another input parameter should be taken into consideration. The value of φ_p plays a particularly important part in shock-impacted particle laden flows to learn how much the $C_{d_{max}}$, τ_r and I_t is affected by the φ_p in a multiple particle cloud, 45 different cases were performed using initial values recorded in Table 3. Each case had 41 particles placed in a staggered array and then randomly perturbed to simulate a dusty gas while still being set at a standard interval to better capture the affects of φ_p.

Table 3: Input Parameters of Training Cases

Mach	$\frac{\rho_p}{\rho_f}$	Φ_{vol} [%]	Mach	$\frac{\rho_p}{\rho_f}$	Φ_{vol} [%]	Mach	$\frac{\rho_p}{\rho_f}$	Φ_{vol} [%]
1.2	100	2.0	2.0	1000	8.0	2.8	1000	22.4
1.2	100	8.0	2.0	1000	12.6	2.8	3100	8.0
1.2	100	12.6	2.0	1000	22.4	3.2	310	8.0
1.2	310	22.4	2.4	100	8.0	3.2	310	22.4
1.2	1000	2.0	2.4	100	22.4	3.6	100	2.0
1.2	1000	8.0	2.4	310	2.0	3.6	100	8.0
1.2	1000	12.6	2.8	31	8.0	3.6	100	12.6
1.6	31	2.0	2.8	100	2.0	3.6	1000	2.0
1.6	310	8.0	2.8	100	8.0	3.6	1000	8.0
2.0	100	2.0	2.8	100	12.6	3.6	1000	22.4
2.0	100	8.0	2.8	100	22.4	4.0	100	22.4
2.0	100	12.6	2.8	310	8.0	4.0	310	2.0
2.0	100	22.4	2.8	1000	2.0	4.0	1000	8.0
2.0	310	12.6	2.8	1000	8.0	4.4	1000	8.0
2.0	1000	2.0	2.8	1000	12.6	4.4	3100	12.6

The ANN was trained twice, once for $C_{d_{max}}$ and once for τ_r. The value of I_t is implied by the application of these two variables. The training period lasted for 5000 iterations with 25 neurons and the convergence curve is seen in Figure 41. The relationship of Mach number and φ_p versus$C_{d_{max}}$ can be seen in Figure 42, Mach number and φ_p versus τ_r, in Figure 43, and Mach number and φ_p versus I_t in Figure 44.

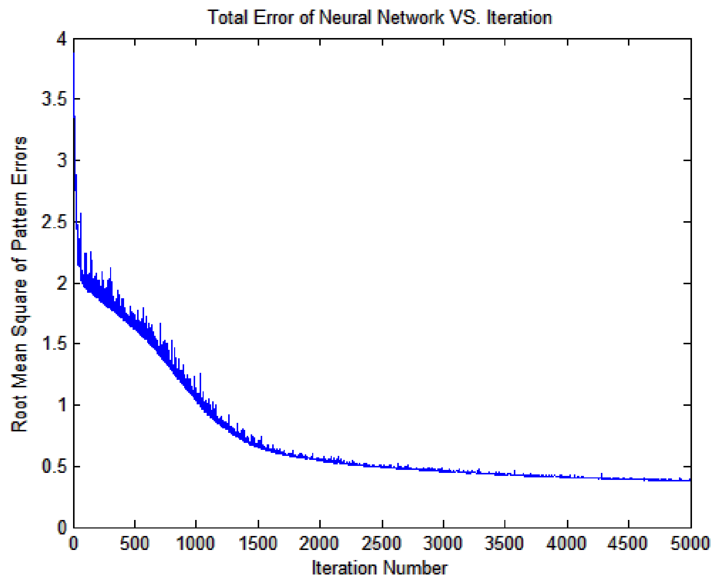


Figure 41: ANN error convergence for mulitiple particles.

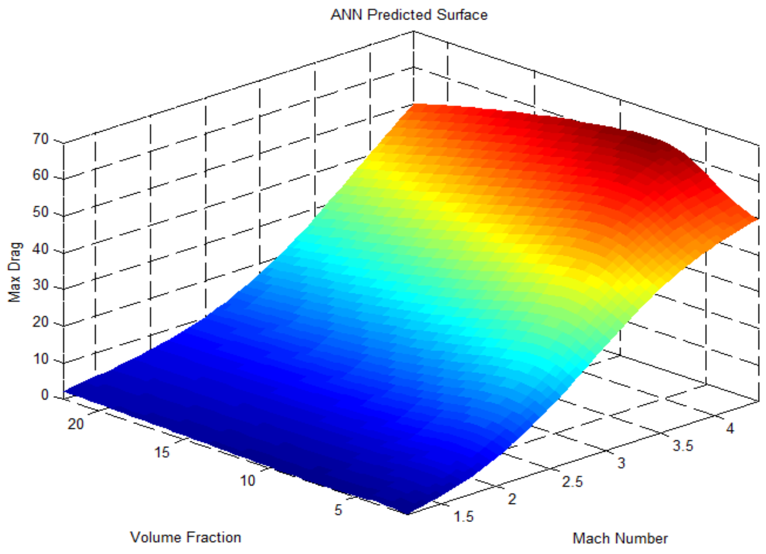


Figure 42: ANN predicted hyper surface for $C_{d_{max}}$

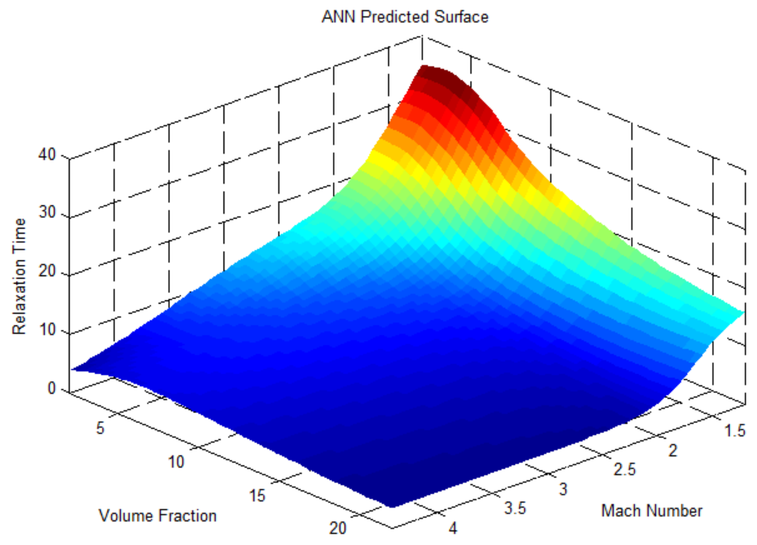


Figure 43: ANN Predicted hyper surface for τ_r

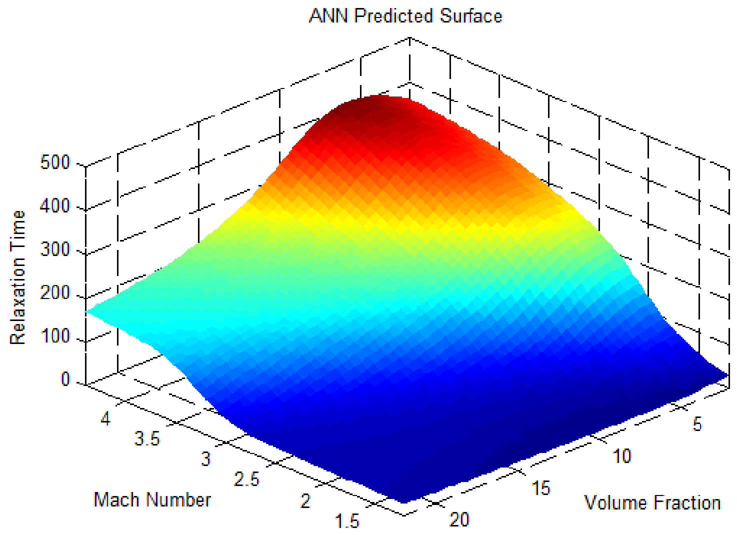


Figure 44: ANN Predicted hyper surface for I_t

It becomes obvious that the major contributor for $C_{d_{max}}$ is the Mach number. The φ_p does not seem to affect $C_{d_{max}}$ at low Mach numbers. As for τ_r, both Mach number and φ_p have great affects. At low Mach numbers, the τ_r greatly increases. Drag force at subsonic velocities are relatively slow to apply. The φ_p has a major affect only at low

Mach numbers. At higher Mach numbers the effect of φ_p goes away, soon thereafter, one may assume that the particles are no longer going to be shielded by particles, but actually hit by them. The surface trend for I_t is somewhat expected, the general trend being that I_t increases as the φ_p decreases and Mach number increases. For the averaged data for all cases, refer to Table A1 in the appendix.

Of course there exist errors in our model that arise from many areas, for example the averaging of multiple individually shocked particles. Very little error was displayed in the single particle cases because there was no random particle-shock interaction from reflections. Testing consisted of randomly selecting a single data point and removing it from the training set. The ANN would be reset and learn the new training set without the point being tested. The ANN was then queued at the test point and was then checked for error. The testing phase consisted of testing a few points by this method; with only one point missing on a multidimensional map, visualization is difficult to show. Testing by selection and removal showed errors all under 2%. For the multiple particle cases, errors ranged greatly. During the training phase of the multiple particle cases, the average error for the training data was less than 1%. However, due to the unsteady curvature and some areas of inconsistent trends, the average error for the prediction of randomly removed and tested points inside the ANN prediction curve for I_t were 7.3%. The largest error for the tested cases resulted from the Mach 4.4 cases which are also responsible for the extra bump on the plots of $C_{d_{max}}$ and I_t. When cases where the Mach number was 4.0 or above was left out and tested for, errors between 12.2% and 14.6% would occur.

Lagrangian Advection

Now that we have a trained ANN with the correlation of Mach number, $\frac{\rho_p}{\rho_f}$ and φ_p to I_t on a particle, we can use it to predict how a shock impacted particle will move. Restating what was mentioned before, one may use the $C_{d_{max}}$ and τ_r to recreate a drag curve represented by exponential decay from the $C_{d_{max}}$ at the "point" of impact. With a defined drag curve, the trajectory of a particle can be predicted by simple Lagrangian advection using Newton's first law of motion. We performed this advection scheme with case data to match previous experiments of Boiko. Our data was limiting to simulating values of φ_p down to 2.0 percent due to the constraints of domain size, particles placed ever 15 diameters away would have produced over 5 million grid cells. The result of using data from the ANN and this Lagrangian advection scheme can be seen as the solid line alongside the experimental work of Boiko et al.$^{(1)}$ in Figure 45.

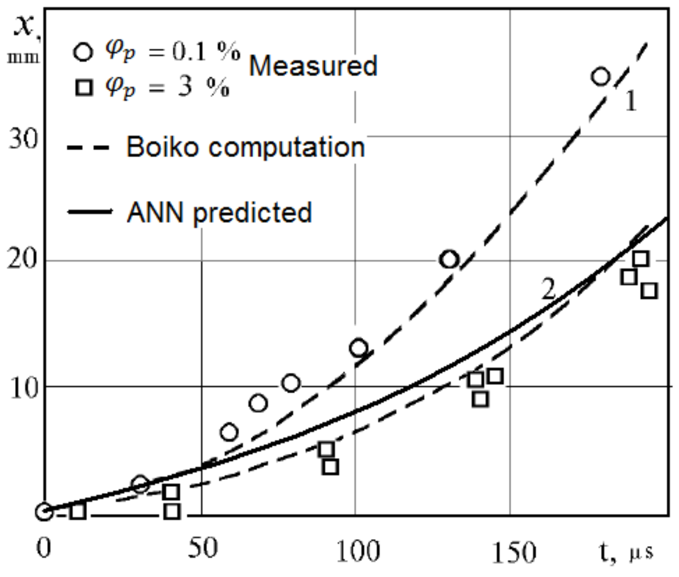


Figure 45: Comparison of Lagrangian Advection

The symbols are directly from experimentation, the dashed line is Boiko's computation, and the solid line is our Lagrangian advection using lifted behavior learned by the ANN. To insure the proper values of I_t were used, an numerical integration scheme was used on our data. This lead to a slightly higher initial peak and exponential representation tends to decay a slightly faster than normal as seen in Figure 38. The largest error is near the beginning where the initial impact of our model is piecewise and thus sharper. However, the ANN and Lagrangian advection model is a close representation of how a particle moves.

Macro-Scale Phenomena

Now that we are able to predict movement of a particle using data from the meso-scale, we should be able to adapt that to multiple particles at the macro-scale. With the ELAFINT3D code running on a serial machine with limited random access memory, we are currently limited to particle clouds of less than 180 particles. To maintain the same staggered and perturbed setup, we chose to model a cloud using 145 particles. This particular case ran with a domain size of 60 by 70 d_p and the smallest cells having a grid size of 0.03 d_p approaching 4 million cells and thus reaching the limit of memory on the machine. To arrive at this point, the model ran more than 25 non-dimensional units of time for about a wall clock time of three weeks. A Schlieren image of this case can be seen in Figure 46. We were able to then use the drag data from each particle to advect their location further. The result was merely the compression of the cloud moved a few domains along the flow direction.

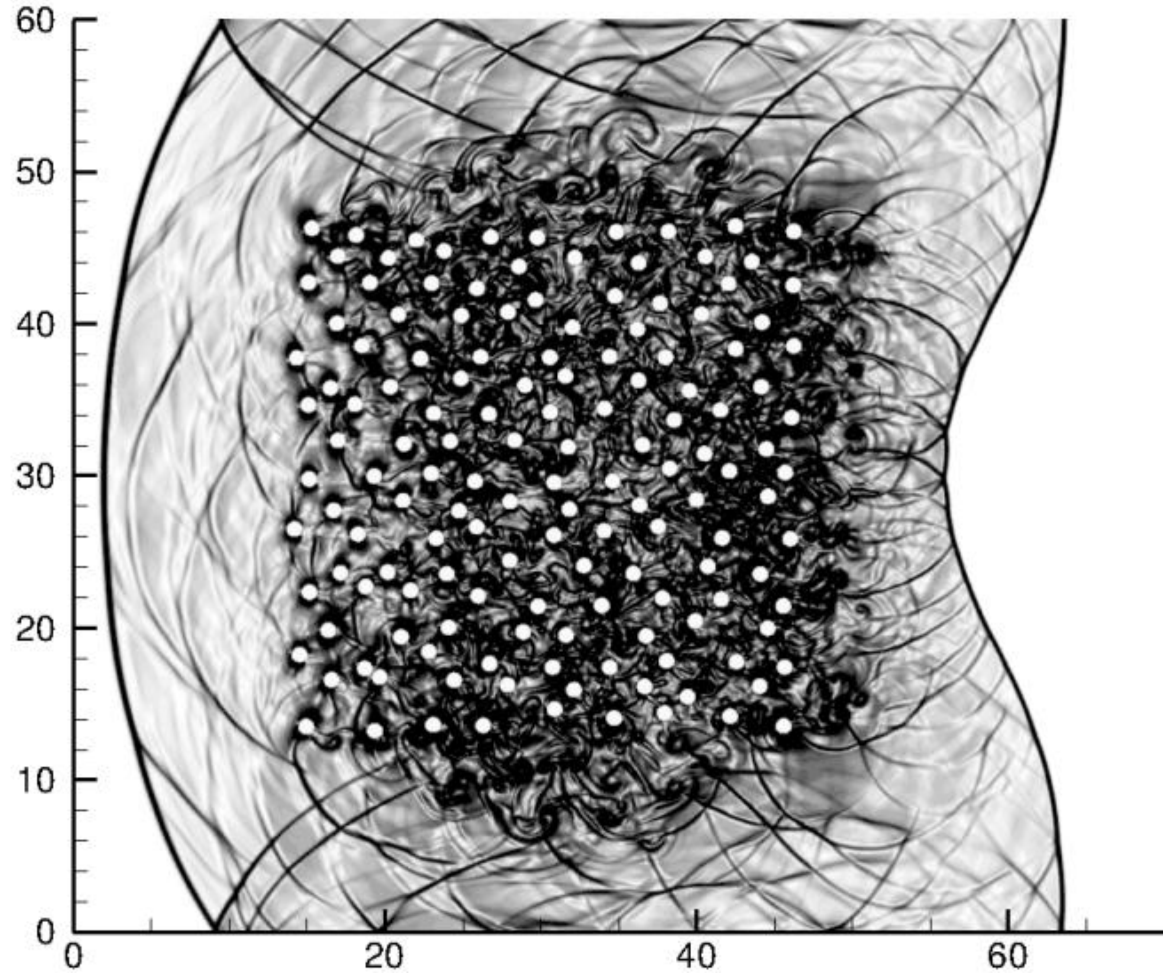


Figure 46: Schlieren image of large particle cloud

(145 particles, Mach = 2.0, φ_p = 8.0%)

There are no experimental demonstrations of this type of cloud; however, in the experimental work of Boiko et al. (1) a macro-scale phenomenon of larger and denser dusty gas clouds emerges. Referring once again to Figure 2 (Chapter 1), one can observe the formation of a sideways "V". This is a phenomenon that arises only in the cases where the dust clouds are sufficiently dense. It is not observable Figure 1 or in the other cases performed by Boiko et al.(1) as seen in Figure 47. In each of the cases presented, a thin band of particles is shocked.

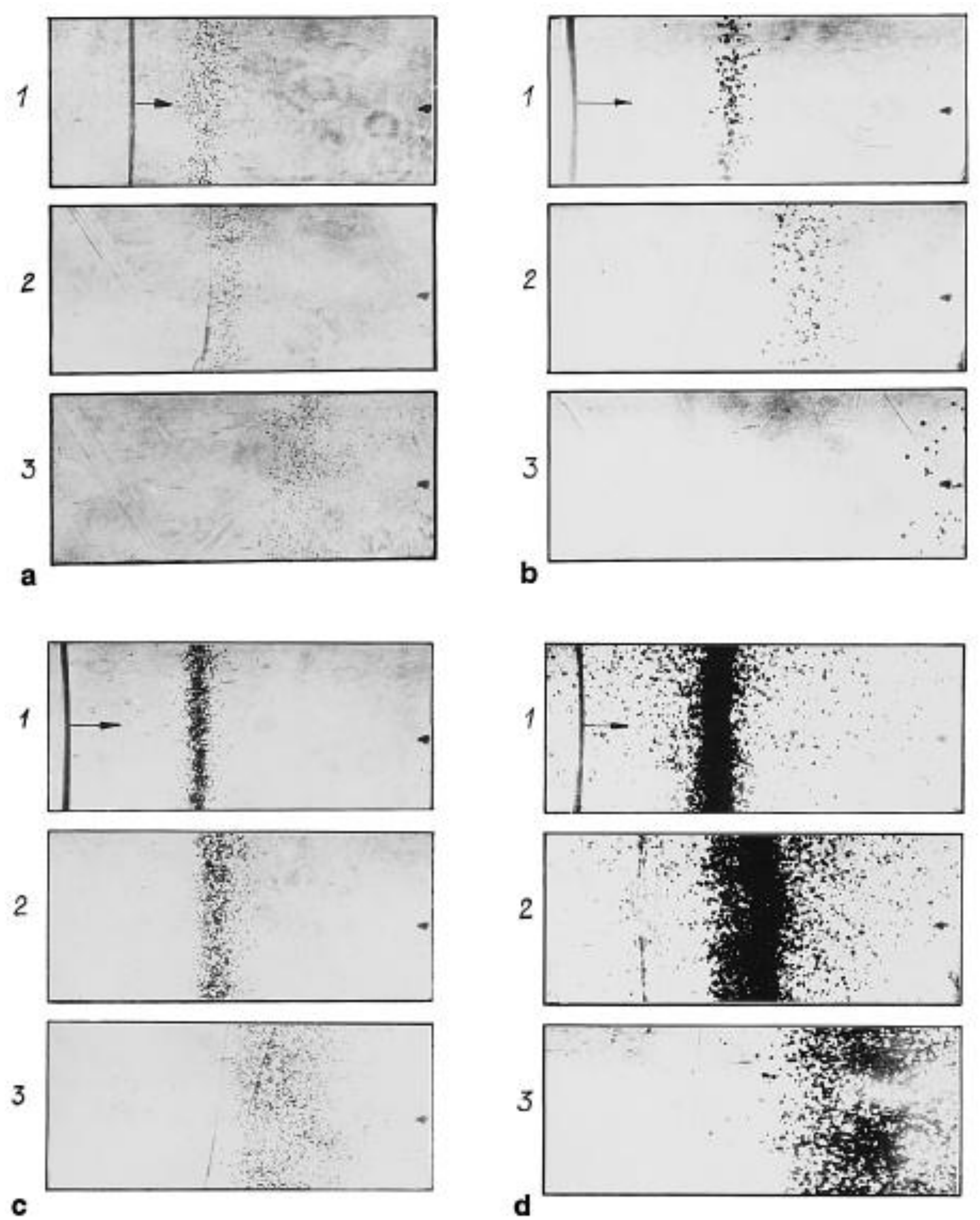


Figure 47: Shock wave particle interaction (Boiko et al.)(1)

In any case, the Mach number and the $\frac{\rho_p}{\rho_f}$ remains virtually the same throughout the whole domain, yet the particles obviously move at different velocities, which mean they would have different values of I_t. The first two factors that arise that may affect I_t are φ_p and the shielding of the shock wave by particles ahead in the flow domain. Shielding is of course directly related to φ_p as well as the total number of particle in the domain. Since the total number of particles in a domain is an extrinsic property of a dust cloud, it is not advisable to use it. This would have to be utilized via a decrease in Mach number. Another case was performed in where a triangular dust cloud was shocked to observe the effect of shielding. Both this case and others found that within a certain φ_p, the total number of particles does not drastically affect particle motion.(1)(80)(85) Thus the main contributing factor to the variance in particle motion in a single domain with constant Mach number and $\frac{\rho_p}{\rho_f}$, is φ_p. Knowing this, a much larger simulation can be performed with drag forces obtained from an ANN which learned from cases with varying φ_p.

Macro-scale simulation

For the macro-scale simulation, we used a Lagrangian advection scheme to move particles based on the drag force obtained from $C_{d_{max}}$ and τ_r predicted by an ANN given Mach number, $\frac{\rho_p}{\rho_f}$ and φ_p. The Mach number and $\frac{\rho_p}{\rho_f}$ was predefined while the φ_p was calculated based on the area fraction (in 2D) occupied by the particles, computed for a box of 20 by 20 diameters in the level set field with a domain size of 1024 by 512 grid points as seen in Figure 48, Figure 49 and Figure 50.

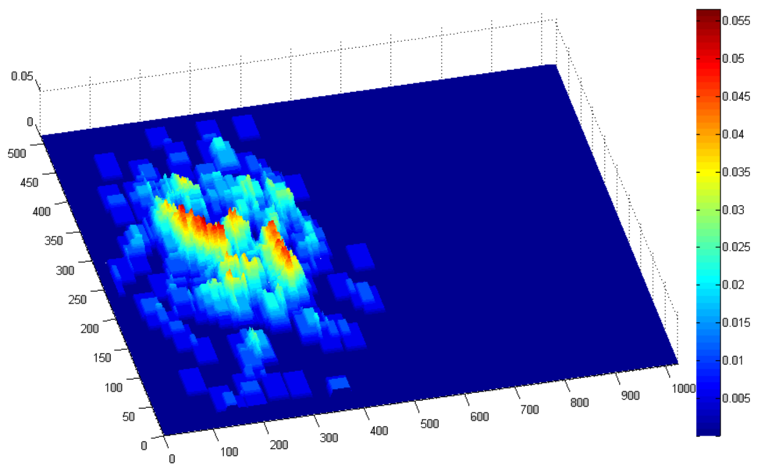


Figure 48: φ_p for sparse dust cloud

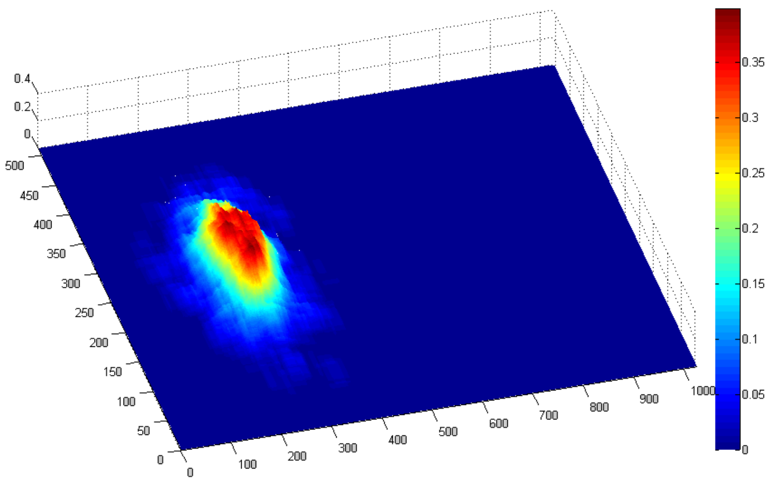


Figure 49: φ_p for dense dust cloud

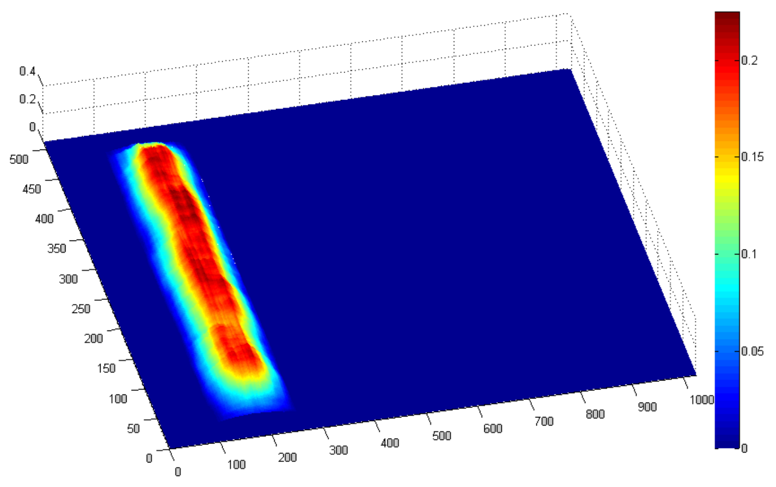


Figure 50: φ_p for dust band

Assumptions

To aid in modeling the shock-impacted particle-laden flow, certain assumptions were made. The assumptions and their implications are as follows:

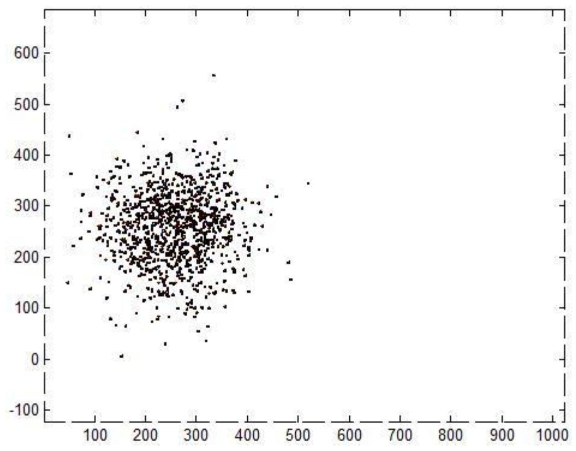
- Particles are normally distributed in both the x and y directions for dust clouds. They are completely still at beginning with no velocity components.
- All small scale forces are neglected. This ignores the effects of buoyancy, gravity, electromagnetic forces, chemical attraction, or Brownian motion.
- Particles are treated as points. No collisions occur in calculations and particles may over lap or pass through each other. To stimulate y direction and small forces, a random purturbment of location was included.
- φ_p is computed by summing the surrounding first level set, particle volumes overlapping are neglected.
- $C_{d_{max}}$ force and τ_r computed by ANN in the first step, thus this is as if the incident shock impacted every particle at the same time

- Largest variable factor in drag is φ_p, particles with very high φ_p barely move, due to lack of collisions.
- For an example of the algorithm, a pseudo code has been provided in the attachments.

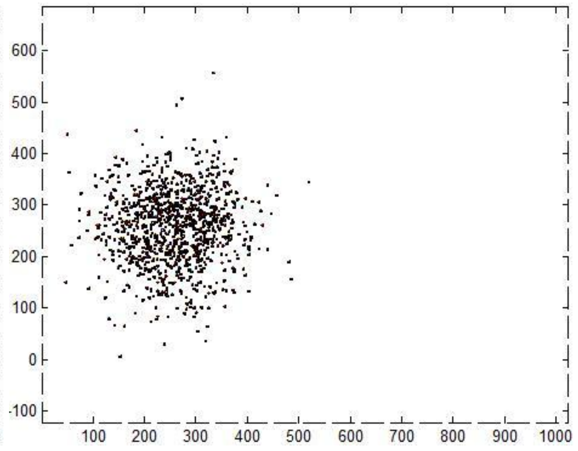
General Motion

To ascertain that indeed the formation of the "V" shaped phenomenon is due to that of the variation in φ_p several macro-scale models were performed. They included simulations that were drag law based, with low φ_p, with high φ_p, and with a uniform band φ_p. For the case based on a drag law, the "standard" drag law found in Table 2 was used to determine the force an each particle. This straightforward method and the assumptions made above caused every point to move roughly the same amount as seen in Figure 51.

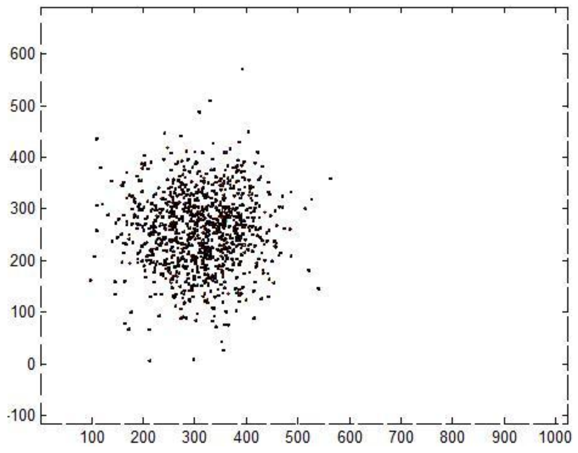
For the sparse dust cloud case, Figure 52, similar actions occurred due to a small variance inφ_p. Demonstrated experimentally, little difference in movement also occurs in Figure 1, of Boiko's experiments. With φ_p and other parameters all the same, each particle should experience the same motion. When the density of particle is increased such as in Figure 53, a "V" phenomenon would appear as seen in Figure 2 by Boiko et al.[1] The formation of this phenomenon occurs only at the macro-scale when there is a wide range in φ_p. In Boiko's experiment one can observe a block of particles in the middle. Just a block alone is not capable of producing a strong enough variance in particle velocity to form a "V". The simulation in Figure 46 demonstrated no large differences in I_t or velocity. When a band a particles was used, as in Figure 54, more particles were spread out just behind the cloud. This is most obvious in Figure 54d and Figure 47d frame 2 where the left side of the block is evidently denser than the right.



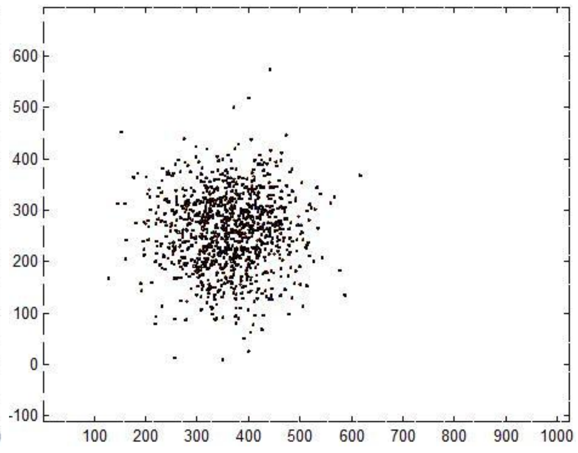
(a) Time = 0.0



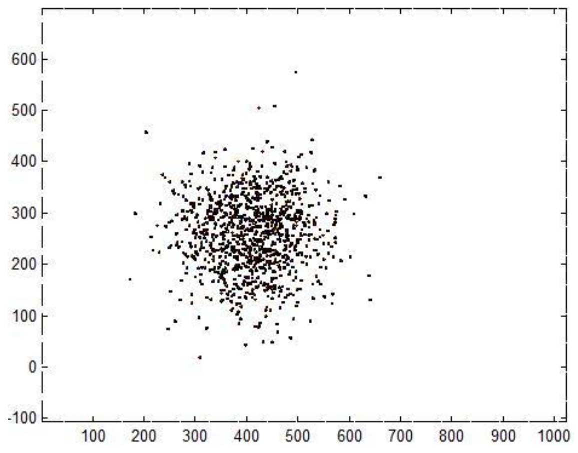
(b) Time = 200



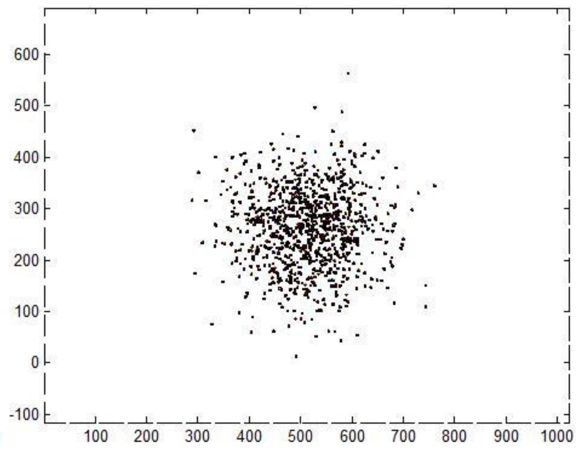
(c) Time = 400



(d) Time = 600

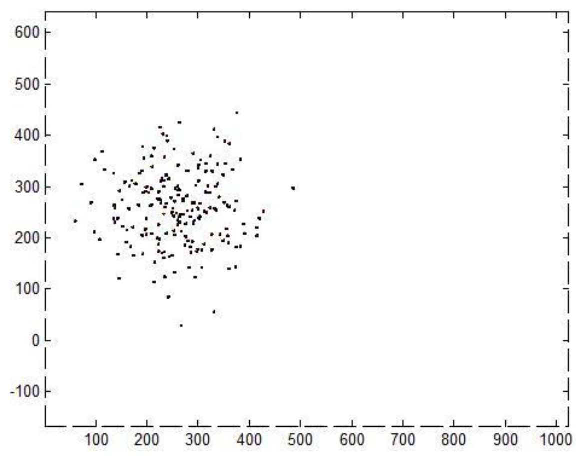


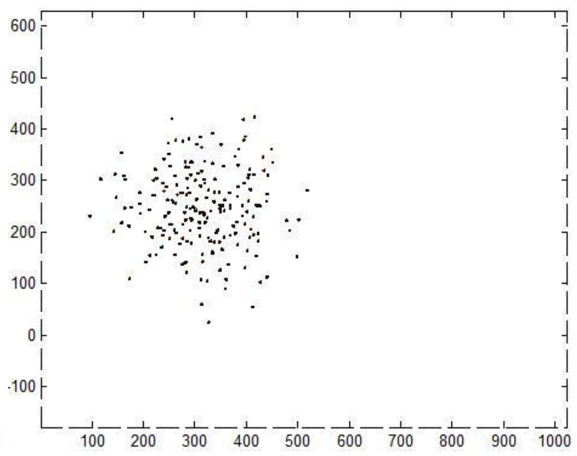
(e) Time = 800

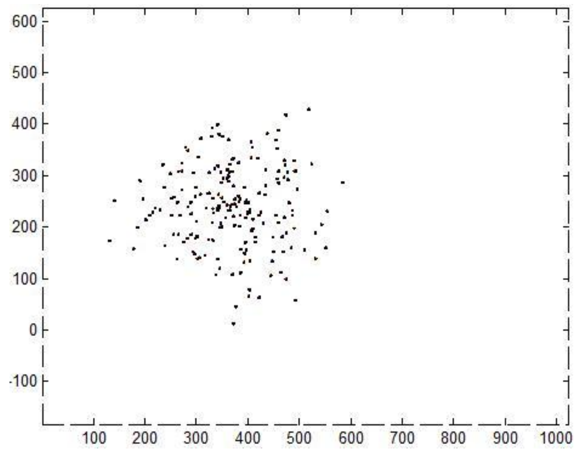


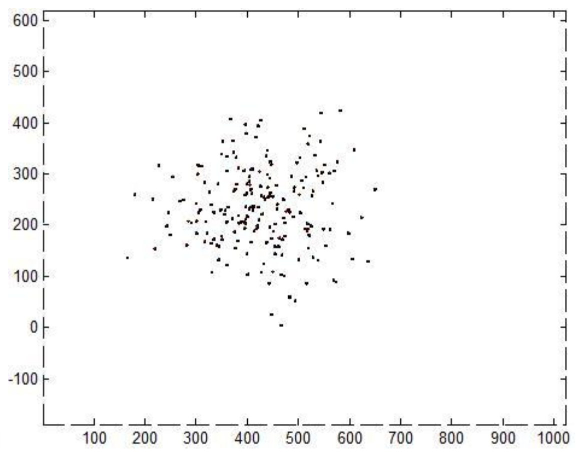
(f) Time = 1000

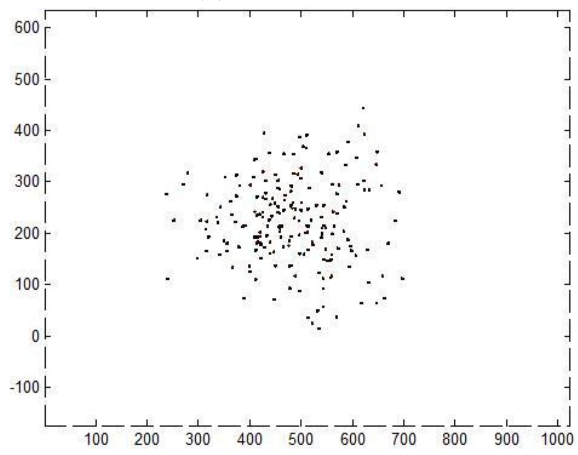
Figure 51: Shock-impacted particle laden flow simulation
(1000 particles, Mach = 2.0, drag law based)


(a) Time = 0.0


(b) Time = 200


(c) Time = 400


(d) Time = 600


(e) Time = 800

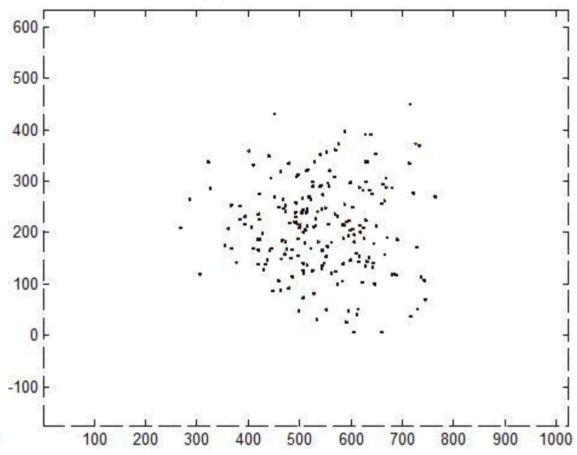
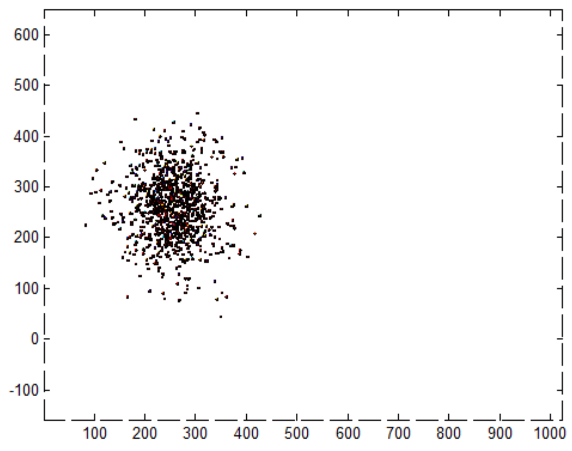
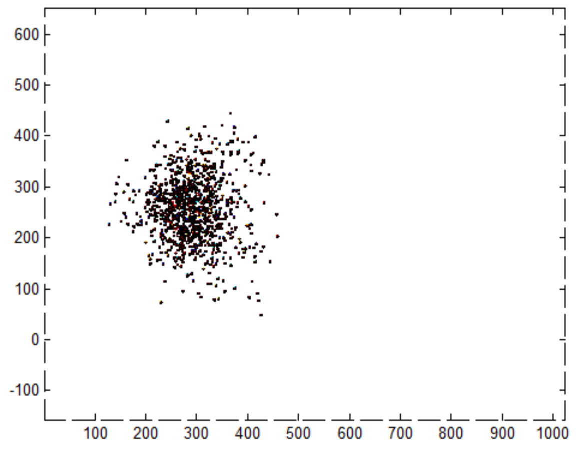
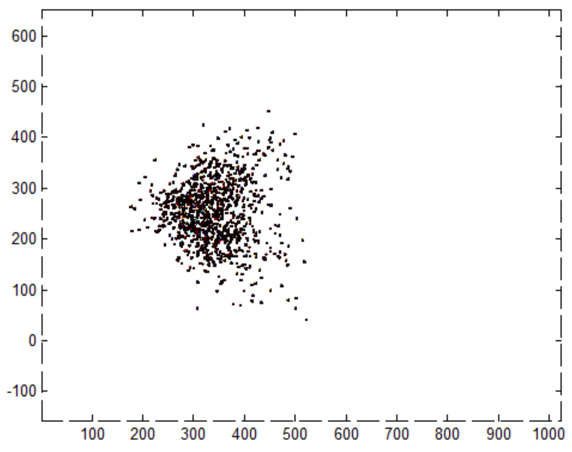
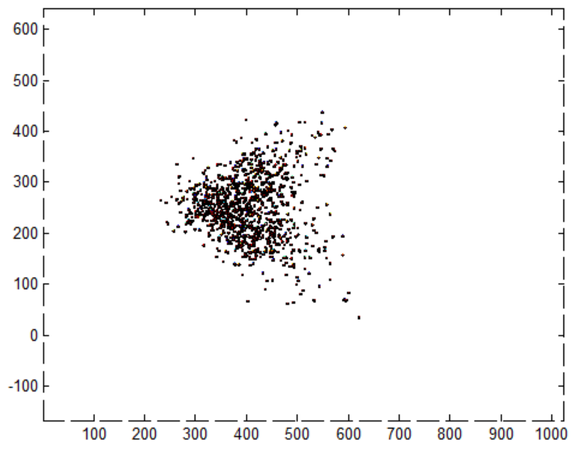

(f) Time = 1000

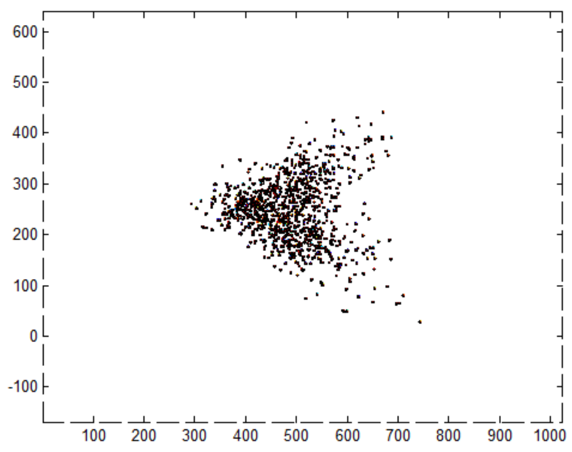
Figure 52: Shock-impacted particle laden flow simulation
(200 particles, Mach = 2.0, φ_p based ANN)


(a) Time = 0.0


(b) Time = 300


(c) Time = 600


(d) Time = 900


(e) Time = 1200

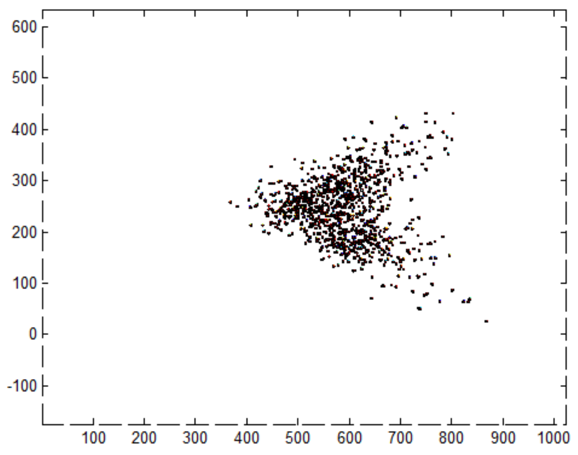
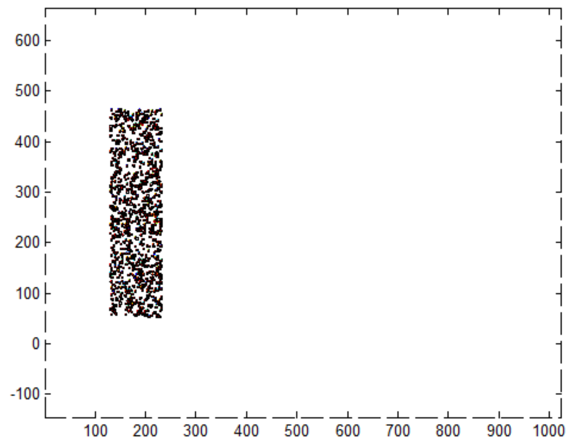
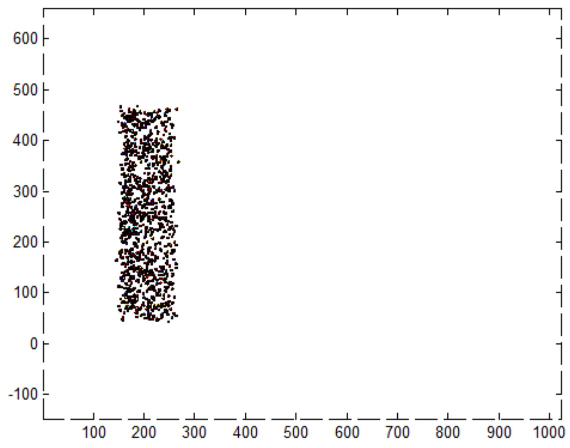

(f) Time = 1500

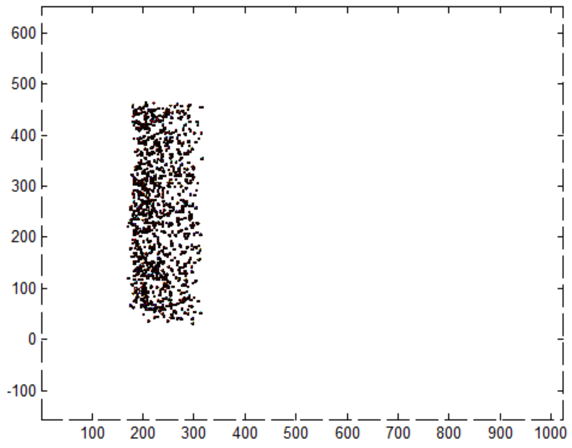
Figure 53: Shock-impacted particle laden flow simulation
(1000 particles, Mach = 2.0, φ_p based ANN)



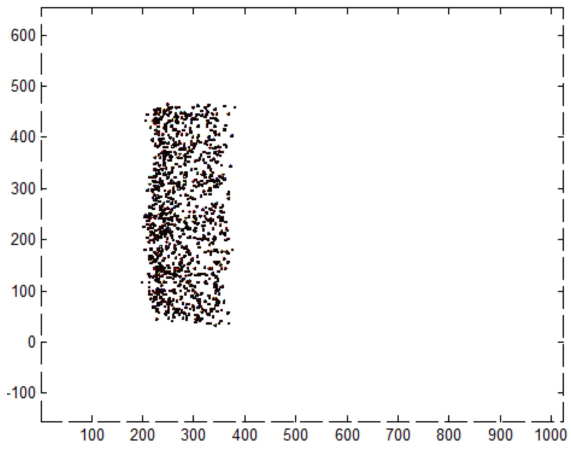
(a) Time = 0.0



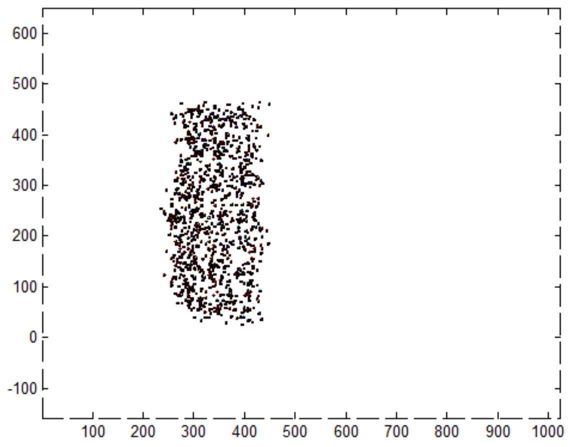
(b) Time = 150



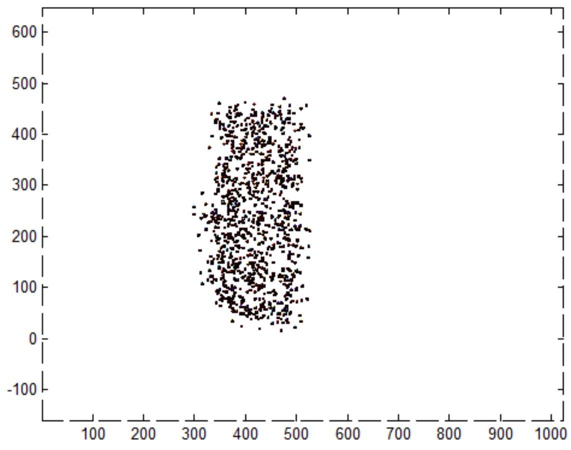
(c) Time = 300



(d) Time = 450



(e) Time = 600



(f) Time = 750

Figure 54: Shock-impacted particle laden flow simulation
(1000 particles, Mach = 2.0, φ_p based ANN)

CHAPTER VII:

CONCLUSIONS AND RECCOMENDATIONS

Conclusions

The objective of this thesis was to efficiently model the interaction of a shock wave and a dusty gas. We wanted to accomplish this by formulating an algorithm to learn the behavior of meso-scale simulations. We successfully set up and used a feed-forward back propagation artificial neural network to learn the drag curves from single and multiple particle cases. For application to multiscale modeling, we chose important characteristics from the meso-scale simulations to be “lifted” in to a macro-scale simulation. The values of $C_{d_{max}}$ and τ_r formulated a transient representation of I_t. The values of I_t learned using the ANN and had an average error of less than 0.5% in training and 2.0% in testing for single particles and less than 1.0% and 8.0% for multiple particles. The multiple particle cases provided more variance in the data of each particle separately than the variance of the ANN learning. The learned behaviors by the ANN were successfully used in macro-scale simulations. Though the method used did not contain any collision models and many simplifying assumptions were made, the different macro-scale simulations demonstrated the great improvement of using an ANN and multiscale methods over traditional methods using predefined drag laws. There remain some limitations on the general problem that could be accurately modeled, mostly those dependent on the scale separation as well as human intervention for the selection of necessary data to be lifted, however, the general method of behavior learning and data lifting was successful.

Computational Savings

The largest simulation that was performed for this thesis was that of a shock impacted 145 particle cloud. This simulation took a wall clock time of nearly 3 weeks on a serial processor and used 16 gigabytes of random access memory prior to stalling out near 30 non-dimensional units of time between 3 million and 4 million cells. To run a simulation with 1000 particles in a domain nearly 125 times larger it would take the same machine, assuming limitless memory, roughly 50 years. The ANN learned behavior in the macro-scale simulation performed such a task in less than one minute. The data processed was 45 cases each lasting a few days but is capable of running in parallel. In either case, with the use of behavioral learning of 'lifted' meso-scale variables, much time was saved. Disregarding a lifting of single variables, the ANN could still learn the entirety of the drag curve for complicated scenarios if aided by multi-resolution analysis.

Multi-Resolution Augmented ANN

In the case where multi-resolution analysis was performed, there was a significant drop in error after the multi-resolution which can be explained by two reasons. The first reason is that the number of data points decreased, allowing the neural network to perform half the number of calculations as well as decrease the summation of errors over all the data points. The second reason is the softer gradient decent the back propagation algorithm was performed on a curve that does not have large slopes present. The most drastic improvement is time needed for all the iterations to reach a convergence point.

Although the total number of iterations was kept the same, when and what to iterate the learning algorithm for the neural network to identify flow behavior was much more selective. The capturing of flow characteristics would be very difficult for a single artificial neural network. The MRAANN decomposed the drag curves to functions that could be learned very quickly and then to reconstruct the original drag curve. A single neural network assigning different sections to different partitions of hidden layers is allowed to grow. Each level of transformation doubles the number of iterations per level, but reduces the time to calculate the updating scheme.

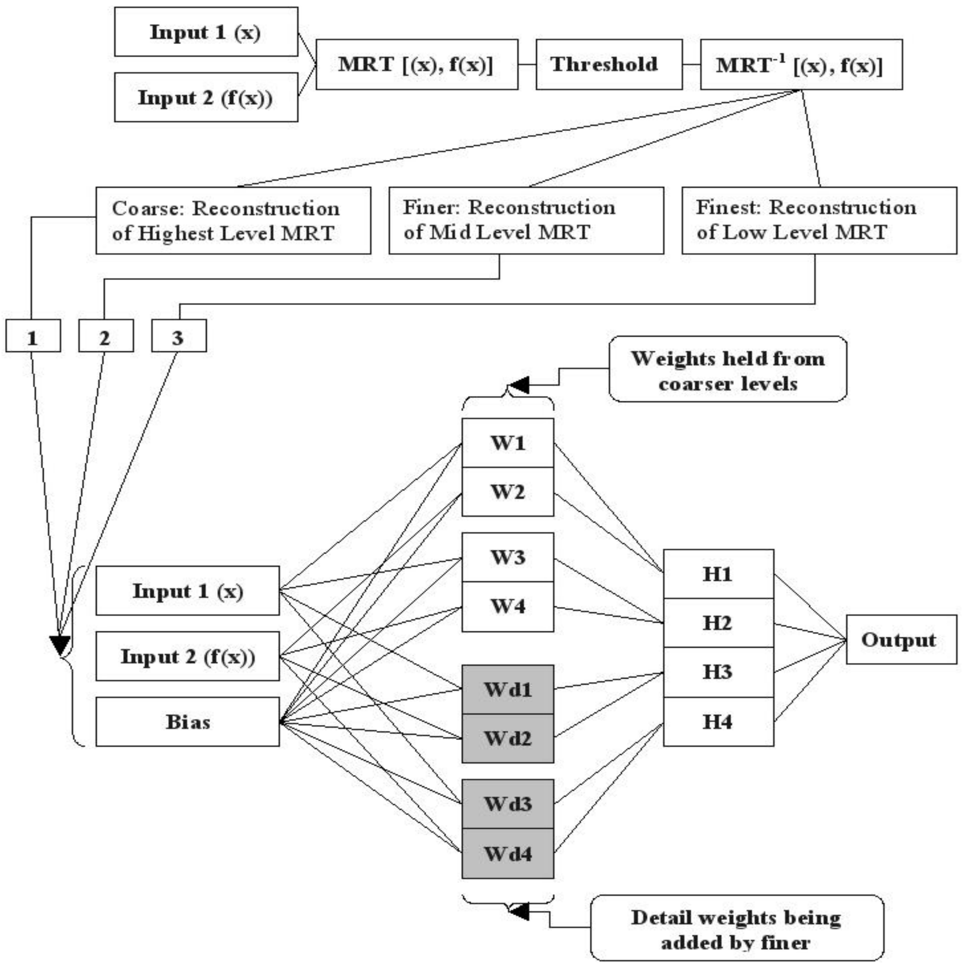

Input 1 (x)
Input 2 (f(x))
MRT [(x), f(x)]
Threshold
MRT^{-1} [(x), f(x)]
Coarse: Reconstruction of Highest Level MRT
Finer: Reconstruction of Mid Level MRT
Finest: Reconstruction of Low Level MRT
1
2
3
Weights held from coarser levels
W1
W2
W3
W4
Wd1
Wd2
Wd3
Wd4
Input 1 (x)
Input 2 (f(x))
Bias
H1
H2
H3
H4
Output
Detail weights being added by finer

Figure 55: Multi-Resolution Augmented Artificial Neural Network

The theoretical number of calculations saved with the MRA approaches ½ as the number of transformed levels approached infinity.(97) This would reduce the number of iterations per level to less than one. The only way around this is to increase the number of iterations. However, because the error is also decreasing, to achieve the same overall accuracy, the MRAANN is able to perform a smaller number of iterations for the same accuracy.

The segmentation of a neural network would greatly increase the accuracy, but there may not be a correlation learned in multiple dimensions. A single data point is learned easily, but it will not produce the ability to predict flow behavior. Although the MRA is capable of reducing time and increasing accuracy, it is not capable of making predictions alone. The MRAANN still requires human interaction to determine the best set of variables to balance speed and accuracy. A method to avoid this and increase speed further is to assign more neurons, segments, or iterations based upon the amount of noise present and the value of the detail coefficients. Overall, the implementation of a MRAANN in data processing and fluid modeling has already been shown to be promising. However, due to the nature of multiscale modeling, the learning an ANN needs to undergo only has to contain data for characteristics that need to be lifted. For the most part, these characteristics, such as $C_{d_{max}}$ and τ_r need only a simple learning algorithm. The use of multi-resolution analysis at this scale is deemed unnecessary.

Recommendations and Future Work

Future recommendations for work in this area mainly entail the proper selection, setup and lifting data of the training cases or the design of an ANN for more complex learning. One may directly place the neural network into a fluid code to aid shock impacted particle laden flow simulations carry out calculations for particle motion after the shock wave has passed over including particle collisions. Currently there exists the necessity for human intervention for proper data lifting and restriction. In the future, the use of wavelets and multi-resolution analysis may be used to select and utilize data passed between scales. For ANN improvement, many methods exist, such as the MRAANN or a new method coined by Ghaboussi et al. (98) as “autoprogressive training”

Autoprogressive training

In autoprogressive training, the ANN is training for global information as usual and from any type of testing or simulation procedure. The specific examples are then used to train the ANN and set the basis for learning. This base level of learning is very similar to the coarse level training in the MRAANN. However, the when the weights are ‘frozen’ in autoprogressive training, they remain frozen while additional neuron are added and the next case is learned.(98) There are no duplicated weights, resolution refinement, or segmented data sets as in the MRAANN. In an autoprogressive trained artificial neural network, the hidden layers expand continuously and progressively as the dataset becomes more complicated. There should be minimal human intervention in the learning phase and expansion of the network. Further analysis phases which introduce more data can easily be updated and added without retraining the ANN for the whole data set.

This work represents a first step in constructing a technique to couple scales together in a multiscale framework. The main contribution of this thesis is to develop the capability to use an ANN to “lift” information from detailed meso-scale calculations (or perhaps even from experiments if they are available). The macro-scale calculations access the lifted information by simply querying the ANN (a procedure that rapidly provides information to the macro-scale on fairly complicated behavior of the particles as the meso-scale). The macro-scale calculations are then advanced further and information is accessed from the ANN at each step of the macro-calculation. This type of interaction between the micro- and macro-scale, affected by trained ANNs represents a first step in developing a true multiscale simulation capability, in which it may be necessary to perform meso-scale simulations in tandem with macro-scale simulations, in a “patch dynamics” and “gap-tooth” framework.(19)(18) Then the ANN learning process will proceed alongside the querying process and the micro- and macro-calculations will need to be synchronized in some way. This rather elaborate setting for performing multiscale simulations will benefit from massively parallel computations on large processor clusters; the key issue then will be to efficiently orchestrate the micro- and macro-computations along with model assimilation using ANNs.   These ideas are being pursued in ongoing work.

REFERENCES

1. Boiko, V.M., Kiselev, V.P., Kiselev, S.P., Papyrin, A.N., Poplavsky, S.V., Fomin, V.M. Shockwave Interaction with a Cloud of Particles. Novosibirsk, Russia : Institute of Theoretical and Applied Mechanics, 1997, ShockWaves, Vol. 7, pp. 275-285.

2. Clayton T. Crowe, Martin Sommerfeld, Yutaka Tsuji. Multiphase flows with droplets and particles. s.l. : CRC Press, 1988.

3. Baer, M.R., Nunziato, J.W. A two-phase mixture theory for the deflagration-to-detonation transition (ddt) in reactive granular materials. 6, 1986, International Journal of Multiphase Flow, Vol. 12, pp. 861-889. 0301-9322(86)90033-9.

4. Verberg, A. J., Ladd, C.R. Lattice-Boltzmann Simulations of Particle-Fluid Suspensions. 5, Springer Netherlands : Springer Netherlands, Dec 22, 2004, Journal of Statistical Physics, Vol. 104, pp. 1191-1251. A:1010414013942.

5. Smereka, J. A. Sethian and Peter. Level set methods for fluid interfaces. s.l. : Annual Reviews, Jan 2003, Annual Review of Fluid Mechanics, Vol. 35, pp. 341-372. annurev.fluid.35.101101.161105.

6. Iverson, R.P. Denlinger, R.M. Flow of variably fluidized granular masses across three-dimensional terrain with numerical predictions and experimental tests. 131, Vancouver : American Geophysical Union, Jan 10, 2001, Journal of geophysical research, Vol. 106, pp. 553-566. AGU2000JB900330.

7. Xu, B.H., Yu, A.B. Numerical simulation of the gas-solid flow in a fluidized bed by combining discrete particle method with computational fluid dynamics. 16, s.l. : Elsevier, Aug 1997, Chemical Engineering Science, Vol. 52, pp. 2785-2809. S0009-2509(97)00081-X.

8. Oscarson, J.H., Graff, K.F. Spall fracture and dynamic response of materials. Columbus : Storming Media, 1968.

9. Gokhale, S.S., Krier, H. Modeling of unsteady two-phase reactive flow in porous beds of propellant. 1, 1982, Progress in Energy and Combustion Science, Vol. 8, pp. 1-39. 0360-1285(82)90007-7.

10. Tamburello, D.A., Amitay, M. Active manipulation of a particle-laden jet. 9, s.l. : Elsevier, Sept 2008, International Journal of Multiphase Flow, Vol. 34, pp. 829-851. j.ijmultiphaseflow.2008.02.006.

11. Kleinstreuer, C. Modern Fluid Mechanics: Intermediate Theory and Applications. s.l. : Springer, 2009, Fluid Mechanics and Its Applications, Vol. 87, p. 620. 9781402086694.

12. Loth, E. Numerical approaches for motion of dispersed particles, droplets and bubbles. s.l. : Pergamon, 2000, Progress in Energy and Combustion Science, Vol. 26, pp. 161-223.

13. Unger, J.F, Konke, C. Coupling of scales in a multiscale simulation using neural networks. 21, s.l. : Elsevier, Nov 2008, Computers & Structures, Vol. 86, pp. 1994-2003. j.compstruc.2008.05.004.

14. Balachandar, S. A scaling analysis for point-particle approaches to turbulent multiphase flows. 9, s.l. : Elsevier, Sept 2009, International Journal of Multiphase Flow, Vol. 35, pp. 801-810. j.ijmultiphaseflow.2009.02.013.

15. Hou, T.Y., Wu, X.H. A multiscale finite element method for elliptic problems in composite materials and porous media. 1, Pasadena : s.n., 1997, Journal of computational physics, Vol. 134, pp. 169-189.

16. Collis, M.W., Lele, A.K., Mackley, M.R., Graham, R.S. Constriction flows of monodisperse linear entangled polymers: Multiscale modeling and flow visualization. 2, 2005, Journal of Rheology, Vol. 49, pp. 501-522. 1.1849180.

17. Weinan, E., Engquist, B., Li, X., Ren, W., Vanden-Eijnden, E. The Heterogeneous Multiscale Method: A Review. 3, Princeton : Springer, June 2007, Communications in Computaitonal Physics, Vol. 2, pp. 367-450.

18. Nuggehally, M.A., Shephard, M.S., Picu, R.C., Fish, J. Adaptive model selection procedure for concurrent multiscale problems. s.l. : Begell House, 2007, International Journal for Multiscale Computational Engineering, Vol. 5, pp. 369-386. IntJMultCompEng.v5.i5.20.

19. Hyman, J.M. Patch Dynamics for Multiscale Problems. 3, Los Alamos : IEEE, May 2005, Computing in Science and Engineering, Vol. 7, pp. 47-53. MCSE.2005.57.

20. Ge, W., Chen, F., Gao, J., Huang, J., Liu, X. Analytical multi-scale method for multi-phase complex systems in process engineering--Bridging reductionism and holism. 13, July 2007, Chemical Engineering Science, Vol. 62, pp. 3346-3377. j.ces.2007.02.049.

21. Liu, W.K., Karpov, E.G., Zhang, S., Park, H.S. An introduction to computational nanomechanics and materials. 17, s.l. : Elsevier, May 7, 2004, Computer Methods in Applied Mechanics and Engineering, Vol. 193, pp. 1529-1578. j.cma.2003.12.008.

22. Batchelor, G.K. An Introduction to Fluid Dynamics. Cambridge : Cambridge University Press, 2000. p. 635. 978-0521663960.

23. Rosenhead, L. Laminar Boundary Layers. s.l. : Dover Publications, 1988. p. 687. 9780486656465.

24. Ebrahimi, F., Sahimi, M. Grid coarsening, simulation of transport processes in, and scale-up of heterogeneous media: Application of multiresolution wavelet transformations. 8, s.l. : Elsevier, Aug 2006, Advances in Disordered Materials, Vol. 38. j.mechmat.2005.06.013.

25. Aarnes, J.E., Krogstad, S., Lie, K.A. A Hierarchical Multiscale Method for Two-Phase Flow based upon Mixed Finite Elements and Nonuniform Coarse Grids. 2, Norway : Sintef Ict, 2006, Multiscale Modeling and Simulation, Vol. 5, pp. 337-363.

26. Migliavacca, F., Balossino, R., Pennati, G., Dubini, G., Hsia, T.Y., Leval, M.R., Bove, E.L. Multiscale modelling in biofluidynamics: application to reconstructive paediatric cardiac surgery. 6, s.l. : Elsevier, 2006, Journal of Biomechanics, Vol. 39, pp. 1010-1020. j.jbiomech.2005.02.021.

27. Batchelor, G.K. The Theory of Homogeneous Turbulence. Cambridge : Cambridge University Press, 1982. 9780521041171.

28. Wilcox, DC. Turbulence modeling for CFD. Canada : DCW Industries, 2006.

29. Pierre, S. Large eddy simulation for incompressible flows: an introduction. 9783540263449, s.l. : Springer Science & Business, 2006.

30. Glotzer, S.C., Paul, W. Molecular and mesoscale simulation methods for polymer materials. s.l. : Annual Reviews, 2002, Annual Review of Materials Research, Vol. 32, pp. 401-436. annurev.matsci.32.010802.112213.

31. Efendiev, Y., Hou, T. Multiscale Finite Element Methods for Porous Media Flows and Their Applicaitons. Pasadena : CalTech, Feb 11, 2006, Applied Numerical Mathematics, Vol. 57, pp. 577-596.

32. Lunati, I., Jenny, P. Multiscale finite-volume method for compressible multiphase flow in porous media. 2, Zürich : Elsevier, Aug 10, 2005, Journal of Computational Physics, Vol. 216, pp. 616-636. j.jcp.2006.01.001.

33. Abgrall, R., Perrier, V. Asymptotic Expansion of a Multiscale Numerical Scheme for Compressible Multiphase Flow. 1, Bordeaux : SIAM, 2006, SIAM Journal on Multiscale Modeling and Simulation, Vol. 5, pp. 84-115. 050623851.

34. Dzwinel, W., Yuen, D.A., Boryczko, K. Bridging diverse physical scales with the discrete-particle paradigm in modeling colloidal dynamics with mesoscopic

features. 7, s.l. : Elsevier, 2006, Chemical engineering science, Vol. 61, pp. 2169-2185. j.ces.2004.01.075.

35. Mäkipere, K., Zamankhan, P. Simulation of Fiber Suspensions—A Multiscale Approach. 4, New York : American Society of Mechanical Engineers, 2007, Journal of Fluids Engineering, Vol. 129, pp. 446-456. 1.2567952 .

36. Fernández, M.A., Milisic, V., Quarteroni, A. Analysis of a geometrical multiscale blood flow. 1, Février : SIAM, 2005, Multiscale Modeling and Simulation, Vol. 4, pp. 215-236. 030602010.

37. Lagana, K., Balossino, R., Migliavacca, F., Pennati, G., Bove, E.L., Leval, M.R., Dubini, G. Multiscale modeling of the cardiovascular system: application to the study of pulmonary and coronary perfusions in the univentricular circulation. 5, s.l. : Elsevier, May 2005, Journal of Biomechanics, Vol. 38, pp. 1129-1141.

38. Weinan, E., Engquist, B., Huang, Z. Heterogeneous Multiscale Method: A General Methodology for Multiscale Modeling. Department of Mathematics, Princeton University. New Jersey : Princeton, 2003. pp. 1-3. PhysRevB.67.092101.

39. Versteeg, H.K., Malalasekera, W. An introduction to computational fluid dynamics: the finite volume method. s.l. : Prentice Hall, 2007.

40. Kevrekidis, I.G., Gear, C.W., Hummer, G. Equation-Free: The Computer-Aided Analysis of Complex Multiscale Systems. 7, Princeton : AIChE, July 2004, AIChE Journal, Vol. 50, pp. 1346-1355. aic.10106.

41. Sommerfeld, M. The Unsteadiness of Shock Waves Propagating Through Gas-Particle Mixtures. Aachen, FRG : Springer-Verlag, 1985, Experiments in Fluids, Vol. 3, pp. 197-206.

42. Gavrilyuk, S., Saurel, R. Mathematical and numerical modeling of two-phase compressible flows with micro-inertia. 1, s.l. : Academic Press, 2002, Journal of Computational Physics, Vol. 175, pp. 326-360. 0021-9991.

43. Pelanti, M., LeVeque, R.J. High-Resolution Finite Volume Methods for Dusty Gas Jets and Plumes. 4, s.l. : SIAM, 2006, SIAM Journal on Scientific Computing, Vol. 28, pp. 1335-1360. 050635018.

44. Sahimi, M., Mehrabi, A. Percolation and flow in geological formations: upscaling from microscopic to megascopic scales. 1, Amsterdam : Elsevier, 1999, Physica. A, Statistical mechanics and its applications, Vol. 266, pp. 136-152. 10029424.

45. Dorobantu, M., Engquist, B. Wavelet-based numerical homogenization. 2, s.l. : Society for Industrial and Applied Mathematics, April 1998, SIAM Journal of Numerical Analysis, Vol. 35, pp. 540-559. 00361429.

46. Chen, L., Debenedetti, P.G., Gear, C.W., Kevrekidis, I.G. From Molecular Dynamics to Coarse Self-Similar Solutions: A Simple Example Using Equation-Free Computation. 1, New Jersey : Elsevier, July 1, 2004, Journal of Non-Newtonian Fluid Mechanics, Vol. 120, pp. 215-223. j.jnnfm.2003.12.007.

47. Delgado-Buscalioni, D. and Coveney, P.V. Continuum-particle hybrid coupling for mass, momentum, and energy transfers in unsteady fluid flow. 4, s.l. : American Physical Society, Apr 2003, Physical Review, Vol. 67, p. 046704. PhysRevE.67.046704.

48. Weiqing, R., Weinan, E. Heterogeneous multiscale method for the modeling of complex fluids and micro-fluidics. s.l. : Elsevier, 2005, Journal of Computational Physics, Vol. 204, pp. 1-26. j.jcp.2004.10.001.

49. Praprotnik, M., Site, L.D., Kremer, K. Adaptive resolution molecular-dynamics simulation: changing the degrees of freedom on the fly. Ackermannweg : American Institute of Physics, Dec 14, 2006, The Journal of Chemical Physics, Vol. 123, p. 224106. 1.2132286.

50. Hou, T.Y. Multiscale modelling and computation of fluid flow. 8, s.l. : John Wiley & Sons, 2005, International Journal for Numerical Methods in Fluids, Vol. 47, pp. 707-719. fld.866 .

51. Durlofsky, L.J., Efendiev, Y., Ginting, V. An adaptive local-global multiscale finite volume element method for two-phase flow simulations. 3, s.l. : Elsevier, March 2007, Advances in Water Resources, Vol. 30, pp. 576-588. j.advwatres.2006.04.002.

52. Ebrahimi, F., Sahimi, M. Grid coarsening, simulation of transport processes in, and scale-up of heterogeneous media: Application of multiresolution wavelet transformations . 8, s.l. : Elsevier Ltd, August 2006, Advances in Disordered Materials, Vol. 38, pp. 772-785. j.mechmat.2005.06.013.

53. Sahimi, M. Fractal-wavelet neural-network approach to characterization and upscaling of fractured reservoirs. 8, s.l. : Elsevier, Oct 2000, Computers & Geosciences, Vol. 26, pp. 877-905. S0098-3004(00)00028-5.

54. Rastigejev, Y.A., Paolucci, S. Wavelet-based adaptive multiresolution computation of viscous reactive flows. 7, Notre Dame : Wiley, Mar 6, 2006, International Journal for Numerical Methods in Fluids, Vol. 52, pp. 749-784. fld.1202.

55. Erban, R., Kevrekidis, I.G., Othmer, H.G. An equation-free computational approach for extracting population-level behavior from individual-based models of biological dispersal. 1, Mar 1, 2006, Physica D: Nonlinear Phenomena, Vol. 215, pp. 1-24. j.physd.2006.01.008.

56. Hyman, J.M. Patch Dynamics for Multiscale Problems. Multiphysics, Los Alamos National Laboratory. Los Alamos : IEEE CS and the AIP, 2005. p. 7.

57. Li, S.G., Liu, Q., Afshari, S. An object-oriented hierarchical patch dynamics paradigm (HPDP) for modeling complex groundwater systems across multiple-scales. 5, May 2006, Environmental Modelling & Software, Vol. 21, pp. 744-749. j.envsoft.2005.11.001.

58. Samaey, G., Kevrekidis, I.G., Roose, D. Patch dynamics with buffers for homogenization problems. 1, s.l. : Elsevier, Mar 20, 2006, Journal of Computational Physics, Vol. 3, pp. 264-287. j.jcp.2005.08.010.

59. Engquist, W., Sun, Y., Bjorn, J. Heterogeneous Multiscale Methods for Interface Tracking of Combustion Fronts. 2, s.l. : Society for Industrial and Applied Mathematics, 2006, Multiscale Modeling & Simulation, Vol. 5, pp. 532-563. 050624844.

60. Weinan, E., Huang, Z. A Dynamic Atomistic-Continuum Method for the Simulation of Crystalline Materials. 2002, Journal of Computational Physics, Vol. 182, pp. 234-261. jcph.2002.7164.

61. Nilsson, N.J. Artificial intelligence: a new synthesis. s.l. : Morgan Kaufmann, 1998.

62. Hocevar, M., Sirok, B., Grabec, I. A Turbulent-Wake estimation using radial basis function neural networks. s.l. : Springer, 2005, Flow, Turbulence and Combustion, Vol. 74, pp. 291-308.

63. Giralt, F., Arenas, A., Ferre-Gine, J., Rallo, R. The simulation and interpretation of free turbulence with a cognitive neural system. 7, s.l. : American Institute of Phisics, July 2000, Physics of fluids, Vol. 12, p. 1826.

64. Krose, B., Schmat, P. An introduction to Neural Networks. 8th. Amsterdam : The University of Amsterdam, 1996. p. 135.

65. Song, Y.K., Borton, D.A., Park, S., Patterson, W.R., Bull, C.W. Microelectronics Neurosensor Arrays. [ed.] IEEE. Providence, RI : IEEE, 2008. Electron Devices Meeting. p. 1. 4244.

66. Mehratra, K., Mohan, C.K., Ranka, S. Elements of Artificial Neural Networks. Cambridge : Massachusetts Institute of Technology, 1996.

67. Bear, J. Dynamics of Fluids in Porous Media. New York : Dover, 1972. 0-486-65675-6.

68. Fausett, L.V. Fundamentals of Neural Networks. [ed.] D. Fowley. Upper Saddle River : Prentice-Hall, 1994. 0-13-334186-0.

69. Zhang, J., Walter, G.G., Miao, Y., Lee, W.N. Wavelet neural networks for function learning. 6, s.l. : Institute of Electrical and Electronics, 1995, IEEE Transactions on Signal Processing, Vol. 43, p. 1485. 1053-587X.

70. Ahmadi, M., Saemi, M. and Asghari, K. Estimation of the Reservoir Permeability by Petrophysical Information Using Intelligent Systems. 14, s.l. : Taylor and Francis, Jan 1, 2008, Petroleum Science and Technology, Vol. 26, pp. 1656-1667. 10916460701675173 .

71. VanOoyen, A., Nienhuis, B. Improving the convergence of the back-propagation algorithm. 1, 1992, Neural Networks, Vol. 5, pp. 465-471. 0893-6080(92)90008-7.

72. Bakshi, B.R., Stephanopoulos, G. Wave-net: a multiresolution, hierarchical neural network with localized learning. 1, s.l. : American Institute of Chemical Engineers, 1993, AIChE Journal, Vol. 39, pp. 57-81. aic.690390108.

73. Tanno, H., Itoh, K., Saito, T., Abe, A., Takayama, K. Interaction of a Shock Wave with a Sphere Suspended in a Verticle Tube. 3, Berlin : Springer, Sept 3, 2003, Shock Waves, Vol. 13, pp. 191-200. s00193-003-0209-y.

74. Mazumder, M.K., Kirsch, K.J. Flow tracing fidelity of scattering aerosol in laser Doppler velocimetry. 4, 1975 : Optics Info Base, 1975, Applied optics, Vol. 14, pp. 894-901. AO.14.000894.

75. Rogak, S.N., Flagan, R.C. Stokes drag on self-similar clusters of spheres. 1, s.l. : Elsevier, Jan 1990, Journal of Colloid and Interface Science, Vol. 134, pp. 206-218. 0021-9797(90)90268-S.

76. Robey, H.F., Zhou, Y., Buckingham, A.C., Keiter, P. The time scale for the transition to turbulence in a High Reynolds number accelerated flow. s.l. : American Institute of Physics, 2003, Physics of Plasmas, Vol. 10, pp. 614-623.

77. Clift, R., Grace, J.R. and Weber, M.E. Bubbles, drops, and particles. s.l. : Academic Press, 1978.

78. Saito, T., Marumoto, M., Takayama, K. Numerical Investigations of Shock Waves in Gas-Particle Mixtures. Sendai : Springer-Verlag, August 2003, Shock Waves, Vol. 13, pp. 299-322. s00193-003-0217-y.

79. Fedorov, A.V., Kharlamova, Y.V., Khmel, T.A. Reflection of a Shock Wave in a Dusty Cloud. 1, Novosibirsk : Springer Science, Jan 2007, Combustion, Explosion, and Shock Waves, Vol. 43, pp. 104-113.

80. Khmel, A.V., Fedorov, T.A. Interaction of a Shock Wave with a Cloud of Aluminum Particles in a Channel. 2, Novosibirsk : Plenum Publishing, March 2002, Combustion, Explosion, and Shock Waves, Vol. 38, pp. 206-214.

81. Saito, T. Numerical Analysis of Dusty-Gas Flows. Sendai : Elsevier Science, Sept. 2002, Journal of Computional Physics, Vol. 176, pp. 129-144. jcph.2001.6971.

82. Kosinski, P. Numerical Analysis of Shock Wave Interaction with a Cloud of Particles in a Channel with Bends. Bergen : Elsevier, Jan. 23, 2007, Heat and Fluid Flow, Vol. 28, pp. 1136-1143. j.ijheatfluidflow.2006.11.003.

83. Kosinski, P. On Shock Wave Propagation in a Branched Channel with Particles. [ed.] O. Igra. 1, Bergen : Springer-Verlag, Feb 14, 2006, Shock Waves, Vol. 15, pp. 13-20. s00193-005-0001-2.

84. Ben-Dor, G., Igra, O., Wang, L.Shock Wave Reflections in Dust-Gas Suspensions. [ed.] J. Eaton. New York : ASME, March 2001, Journal of Fluids Engineering, Vol. 123, pp. 145-153. 10.1115/1.1331558.

85. Ben-Dor, G., Mond, M., Igra, O., Martsiano, Y. A Nondimensional Analysis of Dusty Shock Waves in Steady Flows. 1, Negev Beer Sheva : KSME, Feb. 10, 1988, KSME Journal, Vol. 2, pp. 28-34.

86. Wang, B.Y., Wu, Q.S., Wang, C., Igra, O., Falcovitz, J. Shock Wave Diffraction by a Square Cavity Filled with Dusty Gas. [ed.] O. Igra. s.l. : Springer-Verlag, Oct 25, 2001, Shock Waves, Vol. 11, pp. 7-14.

87. Igra, O., Takayama, K. Shock Tube Study for the Drag Coefficient of a Sphere in a Non-Stationary Flow. A, London : The Royal Society, August 9, 1993, Proc. R. Soc. Lond., Vol. 442, pp. 231-247.

88. Drikakis, O., Ofengeim, D., Timofeev, E., Voionovich, P. Computation of Non-Stationary Shock-Wave / Cylinder Interaction Using Adaptive-Grid Methods. 6, Manchester : Academic Press Limited, August 1997, Journal of Fluids and Structures, Vol. 11, pp. 665-692. jfls.1997.0101.

89. Sun, M., Saito, T., Takayama, K., Tanno, H. Unsteady Drag on a Sphere by Shock Wave Loading. 1, Berlin : Springer, Aug 20, 2005, Shock Waves, Vol. 14, pp. 3-9. s00193-004-0235-4.

90. Sambasivan, S., Udaykumar, H.S. An Evaluation of Ghost-Fluid Methods for Strong Shock Interations with Immersed Solid Interfaces. Department of Mechanical Engineering, College of Engineering. Iowa City : University of Iowa, 2009.

91. Osher, S., Sethian, J.A. Fronts Propagating with Curvature Dependent Speed Algorithms Based on Hamilton-Jacobi Formulations. Berkley : Elsevier, 1988, Journal of Computational Physics, Vol. 79, pp. 12-49.

92. Sethian, J.A. Level-Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science. 2nd. Cambrige : Cambrige University Press, 1999.

93. Harti, A. Discrete multi-resolution analysis and generalized wavelets. 1, s.l. : Elsevier, May 1993, Applied Numerical Mathematics, Vol. 12, pp. 153-192. 0168-9274(93)90117-A.

94. Price, S.J. A Review of Theoretical Models for Fluidelastic Instability of Cylinder Arrays in Cross-Flow. 5, Montreal : Academic Press, July 1995, Journal of Fluids and Structures, Vol. 3, pp. 463-518 . jfls.1995.1028.

95. Grieb, N.P. Multiresolution Analysis For Adaptive Refinement Of Multiphase Flow Computations. Mechanical Engineering, College of Engineering. Iowa City : University of Iowa, 20010. Masters Thesis.

96. Boubez, T.I., Peskin, R.L. Multiresolution neural networks. s.l. : International society for optical engineering, 1994. pp. 649-660. QA403.3.w354.

97. Kosinska, A. Interaction of debris with a solid obstacle: Numerical analysis. 1, May 15, 2010, Journal of Hazardous Materials, Vol. 117, pp. 602-612. j.jhazmat.2009.12.075.

98. Kleinberg, J., Tardos, E. Algorithm Design. Boston : Addison Wesley, 2005. 0321295358.

99. Ghaboussi, J., Pecknold, D.A., Zhang, M.F., Haj-Ali, R.M. Autoprogressive training of neural network constitutive models. 1, s.l. : John Wiley & Sons, Ltd., 1998, International Journal for Numerical Methods in Engineering, Vol. 42, pp. 105-126. 10.100.

APPENDIX

Averaged and Smoothed Case data

Table A1: Data "lifted" from multiple particle cases

Mach	$\frac{\rho_p}{\rho_f}$	φ_p	$C_{d_{max}}$	τ_r	I_t		Mach	$\frac{\rho_p}{\rho_f}$	φ_p	$C_{d_{max}}$	τ_r	I_t
1.2	100	2.0	0.97	33.60	32.59		2.8	100	2.0	20.56	4.86	99.92
1.2	100	8.0	0.82	30.10	24.68		2.8	100	8.0	23.55	2.53	59.58
1.2	100	12.6	0.78	27.60	21.53		2.8	100	12.6	21.45	2.22	47.62
1.2	310	22.4	0.70	25.34	17.74		2.8	100	22.4	14.30	2.08	29.74
1.2	1000	2.0	0.88	30.10	26.49		2.8	310	8.0	19.44	5.51	107.11
1.2	1000	8.0	0.82	27.70	22.71		2.8	1000	2.0	21.10	12.87	271.56
1.2	1000	12.6	0.74	25.40	18.80		2.8	1000	8.0	23.60	9.53	224.91
1.6	31	2.0	3.38	16.90	57.12		2.8	1000	12.6	21.50	7.12	153.08
1.6	310	8.0	3.61	14.21	51.30		2.8	1000	22.4	14.14	5.21	73.67
2.0	100	2.0	7.51	10.52	79.01		2.8	3100	8.0	19.47	7.35	143.10
2.0	100	8.0	7.20	11.30	81.36		3.2	310	8.0	33.95	4.22	143.27
2.0	100	12.6	6.75	10.90	73.58		3.2	310	22.4	23.76	2.76	65.58
2.0	100	22.4	5.60	9.90	55.44		3.6	100	2.0	39.54	3.25	128.51
2.0	310	12.6	7.79	11.96	93.17		3.6	100	8.0	46.27	1.62	74.96
2.0	1000	2.0	7.55	12.63	95.36		3.6	100	12.6	41.42	1.46	60.47
2.0	1000	8.0	6.19	10.42	64.50		3.6	1000	2.0	42.07	8.24	346.66
2.0	1000	12.6	6.84	9.31	63.68		3.6	1000	8.0	46.24	6.41	296.40
2.0	1000	22.4	6.30	8.26	52.04		3.6	1000	12.6	41.32	4.96	204.95
2.4	100	8.0	15.05	3.48	52.37		4.0	100	22.4	41.84	1.26	52.72
2.4	100	22.4	10.90	2.57	28.01		4.0	310	2.0	54.20	4.75	257.45
2.4	310	2.0	13.40	6.12	82.01		4.0	1000	8.0	60.66	5.83	353.65
2.8	31	8.0	19.50	1.55	30.23		4.4	1000	8.0	70.30	4.68	329.00

MatLab Pseudo Code: Artificial neural network training

```
%% Defined Constants
number_of_iterations = 500  ; %keep less than 5000
number_of_neurons    = 5    ; %keep less than 50, or NaN error
output_learning_rate = .02  ; %should be less than .03 or really 'jumpy'
input_learning_rate  = .5   ; %should be less than .5
acceptable_error     = .0001; %convergance error

%% Data Input and Standardization
bias = ones(size(aquila_training,1),1);
training_input = [AQUILA_AIRFOIL(1) bias];
average_input = mean(training_input);
stdev_input = std(training_input);
training_input = (training_input(:,:)-average_input(:,1))/stdev_input(:,1);

%% Data Target and Standardization
training_target = AQUILA_AIRFOIL(2);
average_output = mean(training_target);
stdev_output = std(training_target);
training_target = (training_target(:,:)-average_output(:,1))/stdev_output(:,1);
training_target = training_target';

%% Allocating Weights and Error Array
inputs = size(training_input,2); % num of weights is twice num of neurons
input_weight = randn(inputs,number_of_neurons)/10; % small weights
output_weight = randn(1,number_of_neurons)/10;
error_plot = zeros(1, number_of_iterations);

%% MAIN LOOP: ANN Algorithim
for i = 1:number_of_iterations

    % secondary loop to evaluate each input/output set
    for j = 1:size(training_input,1)

        n = ceil(rand*size(training_input,1));

        % sigmoid activation function with derivitive = (1-tanh^2)
        activation_function = (tanh(training_input(n,:)*input_weight))';

        % Backpropagation:
        prediction = activation_function'*output_weight';
        error = prediction-training_target(n,1);
        delta_output = error.*output_learning_rate.*activation_function;
        output_weight = output_weight-delta_output';
        delta_input= input_learning_rate.*error.*output_weight'.*(1-
(activation_function.^2))*training_input(n,:); % d/dx tanh
        input_weight = input_weight - delta_input';
    end

    % Visual Output
    prediction = output_weight*tanh(training_input*input_weight)';
    final_error = prediction'-training_target;
    error_plot(i) = (sum(final_error.^2))^0.5;
    figure(1); plot(error_plot)

    % Converged Solution
    if error_plot(i) < acceptable_error
        fprintf('converged after %d iterations.\n',i);
        IN_WT = input_weight;
        OUT_WT = output_weight;
        break
    end
end
```

MatLab Pseudo Code: Multi-Resolution Augmented ANN

```
%% Starting on this level, going down
[Training_Data ,Target_Data, startvalue] = MULTI-RESOLUTION_ANALYSIS(Data);
starting_level = size(outputs,2);
Levels_To_Learn = size(outputs,2);

%% MAIN LOOP
for level = 0:Levels_To_Learn-1

    for i = 1:2^level

        if level == 0 % Initializes Weights
            number_of_inputs = size(inputs,2)+1; % plus bias
            input_weight = randn(number_of_inputs,number_of_neurons)/10; % small weights
            output_weight = randn(1,number_of_neurons)/10;

        else % Extracts and Segments Data from MRA
            seg = (max(Training_Data (:,1))-min(Training_Data (:,1)))/(2^level);
            clear inputs outputs
            cnt = 0;
            section(1)  = (seg*(i-1)+min(Training_Data (:,1)));
            section(2)  = (seg*(i  )+min(Training_Data (:,1)));

            for j = 1:length(Training_Data)
                if (seg*(i-1)+min(Training_Data (:,1))) < Training_Data (j)      &&
                                  Training_Data (j) < (seg*(i)+min(Training_Data (:,1)))
                    cnt = cnt+1;
                    inputs(cnt,1)  = Training_Data (j,1);
                    inputs(cnt,2)  = Training_Data (j,2);
                    outputs(cnt,1)  = Target_Data (j,starting_level-level);
                end
            end
            clear input_weight output_weight

            for j = 1:number_of_neurons
                input_weight(:,j)   = new_inwt(:,number_of_neurons*(i-1)+j);
                output_weight(1,j) = new_outwt(1,number_of_neurons*(i-1)+j);
            end
        end

        [final, IN_WT, OUT_WT] = ARTIFICIAL_NEURAL_NETWORK(inputs, outputs,    &
                                  input_weight, output_weight, iterations);

        if level == 0 % Allocates Arrays
            ALL_inwts   = zeros(2,number_of_neurons);
            ALL_outwts = zeros(1,number_of_neurons);
            new_inwt   = zeros(2,number_of_neurons);
            new_outwt = zeros(1,number_of_neurons);
        end

        for j = 1:number_of_neurons % Appends Weights
            ALL_inwts(:,number_of_neurons*(i-1)+j)   = IN_WT(:,j);
            ALL_outwts(1,number_of_neurons*(i-1)+j)  = OUT_WT(1,j);
        End

        for i = 1:size(ALL_inwts,2) % Expands Weights
            for j=1:number_of_inputs
                new_inwt(j, 2*i-1)  = ALL_inwts(j, i);
                new_inwt(j, 2*i)  = randn()/10;
            end
            new_outwt(1,2*i-1)  = ALL_outwts(1, i);
            new_outwt(1,2*i)  = randn()/10;
        end
    end
end
```

MatLab Pseudo Code: ANN based shock impacted particle advection

```
%% Defined Constants
NP    = 1000; % Number of Particles
xsize = 1024;
ysize = 512 ;
denr  = 1000;
time  = 0.0 ;
step  = 0.1 ; % Step Size
endt  = 1001; % End Time
Mach  = 2.0 ;

%% Allocating Arrays
cen = zeros(2,NP); % X and Y Particle Center
vel = zeros(2,NP); % X and Y Particle Velocity
acc = zeros(2,NP); % X and Y Particle Acceleration
phi = zeros(NP  ); % Volume Fraction
max = zeros(NP  ); % ANN Max Force Prediction
tau = zeros(NP  ); % ANN Relaxation Time Prediction
lvlset = zeros(xsize, ysize); % Level Set Field
volfrc = zeros(xsize, ysize); % Volume Fraction Field

%% Seeding and Initializing Level Set Field
for i = 1:NP
    cen(1, i) = int( randn()*(ysize)+ysize/2) ); % x
    cen(2, i) = int( randn()*(ysize)+ysize/2) ); % y
    for x = -1:1
        for y = -1:1
            lvlset( cen(1,i)+x , cen(2,i)+y ) = 1;
        end
    end
end

%% Calculates Local Volume Fraction Field
for i = 1:xsize
    for j = 1:ysize
        volfrc(i,j) = sum(lvlset( i-30:i+30 , j-30:j+30 ))/(60^2);
    end
end

%% Sets Volume Fraction to Particle
for i = 1:NP
    phi(i) = volfrac( cen(1,i) , cen(2,i) );
end

tau(i) = ARTIFICIAL_NEURAL_NETWORK_TAU(Mach, denr, phi(i), time); % Already Trained ANN
max(i) = ARTIFICIAL_NEURAL_NETWORK_MAX(Mach, denr, phi(i), time);

%% Main Loop: ADVECTION SCHEME
while time < endtime

    for i = 1 : NP
        %   frc(1, i) = MRA_ANN(Mach, phi(i), time) % without lifting
        frc(1, i) = max(i)*exp(-time/tau(i));
        frc(2, i) = max(i)*exp(-time/tau(i))*rand(.1);
        acc(1,i) = frc(1,i) / mass; % mass = denr * den_f * vol
        acc(2,i) = frc(2,i) / mass;
        vel(1,i) = vel(1,i) + acc(1,i) * step;
        vel(2,i) = vel(2,i) + acc(2,i) * step;
        cen(1,i) = cen(1,i) + vel(1,i) * step;
        cen(2,i) = cen(2,i) + vel(2,i) * step;
    end

    time = time + step;

    if (mod(time, 10) == 0)
        LEVELSET_IMAGE_PROCESS(cen, lvlset);
    end
end
```