7-15-2011

Design Optimization of Submerged Jet Nozzles for Enhanced Mixing

Edgard Espinosa
Florida International University, eespi002@fiu.edu

DOI: 10.25148/etd.FI11120607

Recommended Citation

Espinosa, Edgard, "Design Optimization of Submerged Jet Nozzles for Enhanced Mixing" (2011). *FIU Electronic Theses and Dissertations*. 501.
https://digitalcommons.fiu.edu/etd/501

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

DESIGN OPTIMIZATION OF SUBMERGED JET NOZZLES FOR ENHANCED MIXING

A thesis submitted in partial fulfillment of the
requirements for the degree of
MASTER OF SCIENCE
in
MECHANICAL ENGINEERING
by
Edgard Espinosa

2011

To: Dean Amir Mirmiran
College of Engineering and Computing

This thesis, written by Edgard Espinosa, and entitled Design Optimization of Submerged Jet Nozzles for Enhanced Mixing, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this thesis and recommend that it be approved.

Dwayne McDaniel

Leonel E. Lagos

Igor Tsukanov, Co-Major Professor

George S. Dulikravich, Co-Major Professor

Date of Defense: July 15, 2011

The thesis of Edgard Espinosa is approved.

Dean Amir Mirmiran
College of Engineering and Computing

Dean Lakshmi N. Reddi
University Graduate School

Florida International University, 2011

DEDICATION

This thesis is dedicated to my loving parents, for being so supportive and encouraging. I only hope that I've made them proud.

ACKNOWLEDGMENTS

This work was supported by the United States Department of Energy.

I would like to acknowledge and express my gratitude to the following individuals who have assisted me and encouraged me with my research endeavors.

Thank you, Dr. George S. Dulikravich for your supervision and guidance. I truly appreciate your support with my research, work, and life pursuits. I am thankful for your continuous interest in the development of this research. I am also grateful for the resources you have provided me to complete my thesis.

Thank you, Dr Leonel E. Lagos for your support during my fellowship with the United States Department of Energy Fellowship Program. You have provided me with many opportunities which have encouraged me to cultivate my "ganas" and push my limits. I am much appreciative.

To my committee members, Dr. Igor Tsukanov and Dr. Dwayne McDaniel, thank you for your advice and support. Thank you for making yourself available at a moment's notice when I required your assistance. Dr. McDaniel, I genuinely admire your scientific knowledge. Find here an expression of my gratitude.

To Dr. Seckin Gokaltun and members of the CFD team, Mr. Rinaldo Gonzalez and Mr. Stephen Wood, I appreciate our one-on-one discussions and working with you. I found all of our talks fruitful. Working with all of you has brought clarity to this challenging field of Computational Fluid Dynamics (CFD). Thank you for your support.

I cannot thank Mr. Stephen Wood enough for his assistance with the Multidisciplinary Analysis, Inverse Design, Robust Optimization and Control Laboratory (MAIDROC) cluster. I've lost count on the many times Mr. Wood has provided me assistance with computer systems issues onsite and while working remotely.

My final thanks go to the people I have previously not mentioned, who have been in any way involved with this research.

ABSTRACT OF THE THESIS

DESIGN OPTIMIZATION OF SUBMERGED JET NOZZLES FOR ENHANCED MIXING

by

Edgard Espinosa

Florida International University, 2011

Miami, Florida

Professor George S. Dulikravich, Co-Major Professor

Professor Igor Tsukanov, Co-Major Professor

The purpose of this thesis was to identify the optimal design parameters for a jet nozzle which obtains a local maximum shear stress while maximizing the average shear stress on the floor of a fluid filled system. This research examined how geometric parameters of a jet nozzle, such as the nozzle's angle, height, and orifice, influence the shear stress created on the bottom surface of a tank. Simulations were run using a Computational Fluid Dynamics (CFD) software package to determine shear stress values for a parameterized geometric domain including the jet nozzle. A response surface was created based on the shear stress values obtained from 112 simulated designs. A multi-objective optimization software utilized the response surface to generate designs with the best combination of parameters to achieve maximum shear stress and maximum average shear stress. The optimal configuration of parameters achieved larger shear stress values over a commercially available design.

TABLE OF CONTENTS

CHAPTER PAGE

I. INTRODUCTION

II. COMPUTATIONAL REVIEW

III. METHOD OF SOLUTION

IV. RESULTS AND DISCUSSION

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE PAGE

CHAPTER I

INTRODUCTION

1.1 Motivation

Savannah River Site (SRS) is exploring the use of a dilute-chemistry acid for the cleaning of high-level waste tanks in preparation for final tank closure. Waste storage tanks at legacy waste sites hold radioactive material generated from the production of nuclear weapons. These waste tanks are reaching their design life and degradation of the structure creates hazardous conditions to the surrounding environment. The initial process in cleaning out the waste tank is removing the bulk volume of waste leaving only the sludge heel to be removed prior to tank closure. Figure 1 shows a diagram of a typical tank used at SRS.

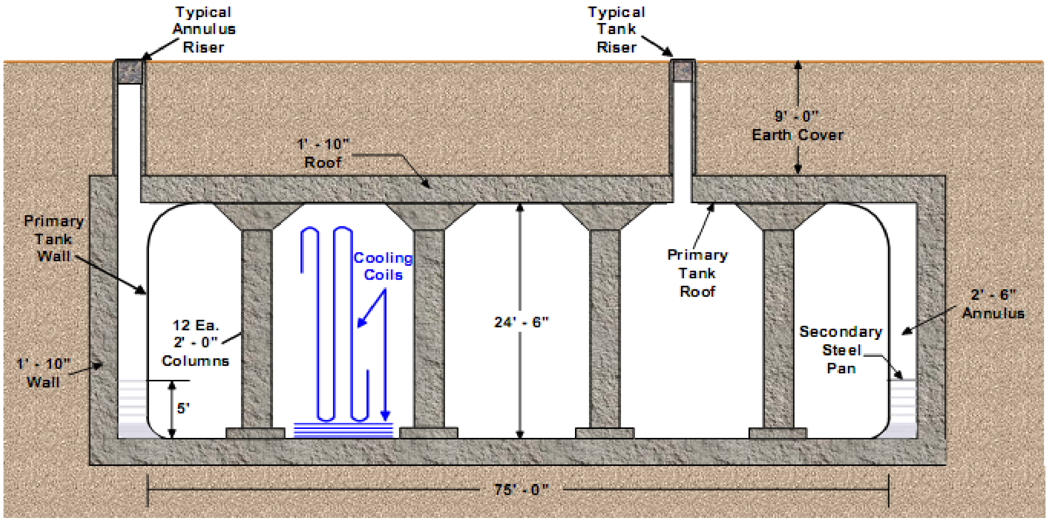


Figure 1 Schematic of radioactive waste tank [1]

Methods for removing the sludge heel that have been investigated include the use of a dilute chemistry acid. A private contractor, NuVision Engineering, has developed a technology using a pulsed-jet nozzle, which utilizes the dilute chemistry acid for the removal of the sludge heel waste.

1.2 Background

Oxalic acid has been used for sludge dissolution [2]. The use of acids, however, poses an added consideration, corrosion of the waste storage tank. Corrosion of these carbon steel tanks degrades the integrity of the structure, resulting in loss of wall thickness [3]. The addition of other acids to oxalic acid for dilution purposes has also been investigated. For example, oxalic/nitric acid mixtures were found to dissolve major metal species, such iron oxides and oxy-hydroxides, thus, further reducing sludge heel [4]. However, the quantity of sludge removed was less than expected [2]. The removal of sludge heel depends strongly on mass flow rate of the liquid impinging the surface. Jet nozzles are used to apply shearing forces to the problematic areas of high sludge build up. Figure 2 illustrates the impingement by a jet nozzle, characterized by three regions.

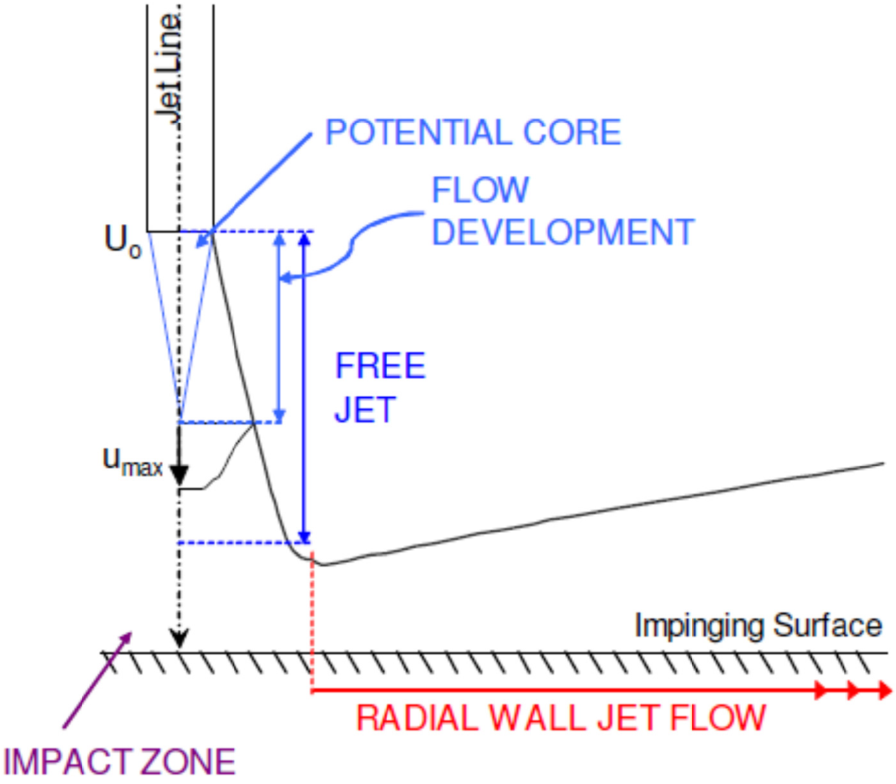


Figure 2 Diagram of three region of jet impingement [5]

First, the free jet is the fluid flow from the nozzle, where it may expand freely. The potential core is within the free jet and is where the flow maintains its exit velocity. It is surrounded by a mixing layer, which undergoes shear and induces turbulence. Next, a rapid decrease in axial velocity causes an increase in static pressure in the impact zone, also known as the region of impingement. Finally, the radial wall jet forms, expanding radially from the impact zone. At present, the study of impinging jets is restricted to primarily computational modeling.

However, the increase of mass flow rate alone does not guarantee an increased rate of sludge removal. Scientists have examined effective jet mixing. Jet mixing is an essential component for achieving substantial agitation used in wastewater treatment, chemical and biochemical industries. Of recent, scientists are relying more on jet mixers to accomplish mixing in large storage tanks and underground tanks [6].

The designs of jet nozzles have been examined for more efficient methods for mixing. Numerous investigators have carried out experimental studies on jet mixers in tanks using different tank geometries, nozzle positions, and nozzle diameters. The development from these arrangements has led many scientists to devise correlations which can determine the mixing time. Many of the experimental work previously conducted focused on the reduction of the mixing time. Scientists have accepted the value of mixing time when 95% or 99% of the concentration of a tracer is dispersed. Some of the early work on mixing time was conducted by Fox and Gex [7]. Patwardhan and Gaikwad [8] also observed the effect of nozzle orientation, i.e., 0^0, 30^0, 45^0, & 90^0, producing results showing that 45^0 mixes slightly better. It enabled the jet to spread more freely, entraining the surrounding liquid in its jet length. This jet entrains some of the surrounding liquid and creates a circulation pattern within the tanks, thus, leading to mixing of its content.

The position of the nozzles has also been considered extensively in achieving effective mixing time. Parvareh et al. [9] examined the nozzle location at various positions around the bottom of a rectangular tank. Additional CFD work has advanced the work on the nozzle's position (location and orientation). It has been reported that the optimum angle of injection at the bottom of the tank was found to be 30^0 (from the horizontal) at a height of 150mm which gave the shortest mixing time. Other research regarding jet mixers have investigated alternating pulse jet mixing. A CFD model developed by Ranade [10] examined, in addition to a steady jet, single and double nozzle sequenced, pulsed jet mixing.

1.3 Research Objectives

Sludge heel in radioactive waste storage tanks tends to adhere to side walls, bottom floor, and other areas within the tank. SRS has used a 1 wt% acid solution to dissolve sludge heel. This research investigates the use of a jet nozzle to inject this acid solution for enhancing sludge heel removal. A simulated environment was used where the performance of jet nozzles were evaluated by modifying the arrangement of the following parameters:

- Nozzle height
- Angle of the nozzle below the horizon
- Dimension of radii (left radius and right radius) of the nozzle orifice
    - The nozzle orifice is defined by two independently variable radii that are aligned on the horizontal centerline of the nozzle

- Distance between radii

The evaluation criterion is the jet nozzle's ability to achieve a local maximum shear stress while maximizing the average shear stress on the floor of the radioactive waste storage tank. Shear stress enhances the removal of sludge heel for effective decommissioning and decontamination of waste storage tanks before final closure.

1.4 Thesis Structure

Chapter I provides a brief introduction to the problem statement. A review of jet nozzle technology and methods for the removal of sludge waste is given. Finally, the objectives addressed in this thesis are presented.

Chapter II provides a brief review of fluid mechanics and turbulence modeling, including the governing equations, the Reynolds-Averaged Navier-Stokes equations, and Numerical Methods. A review of optimization methods is also included.

Chapter III describes the given domain and methodologies used to arrive at the solutions. A genetic algorithm in modeFrontier was used to optimize solutions to determine the best set of parameters. Finally, a grid convergence study is also presented to examine the variations in results as changes in grid size were made. Results from the simulations and comparisons from two optimizations studies are presented in Chapter IV. Discrepancies between real designs and derived designs are presented and addressed. In addition, a sensitivity analysis addresses the changes in the solutions if modifications were made to the placement of the jet nozzles.

A summary of the thesis with concluding remarks and proposed future work is described in Chapter V.

CHAPTER II

COMPUTATIONAL REVIEW

2.1 Governing Equations

The governing equations of fluid dynamics include the steady-state incompressible Navier-Stokes Equations. The ANSYS FLUENT solver was utilized to solve the governing equations for the problem in this thesis.

The continuity equation describes the conservation of mass:

$$\nabla \cdot \mathbf{u} = 0 \tag{2.1}$$

The conservation of momentum equation, for a steady incompressible fluid with dynamic viscosity (μ), in the absence of body forces is:

$$\rho(\mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla P + \mu \nabla^2 \mathbf{u} \tag{2.2}$$

where $\mathbf{u} \cdot \nabla \mathbf{u}$ is the convective acceleration term and $\mu \nabla^2 \mathbf{u}$ is the diffusive term.

2.1.1 RANS Equations: Statistical Time-Averaging of Flow

In Equation (2.2), when $\mathbf{u} \cdot \nabla \mathbf{u}$ gets larger than the diffusive term, the flow becomes unstable and large flow structures break up into smaller and smaller eddies until they are diffused into heat by viscous effects. This is referred to as turbulence. Turbulence is a three-dimensional, highly non-linear, time-dependent phenomenon. Turbulent flows are characterized by fluctuating velocity fields. When fluctuations exist in flow, the overall velocity vector (u) can be defined by a mean velocity $(\bar{u})$ and a fluctuating component about the mean (u'), as the following,

$$u = \bar{u} + u' \tag{2.3}$$

Inserting Eq. (2.3) into the governing equations yields the 3-D Reynolds Averaged Navier-Stokes (RANS) equation:

$$\rho\left[\frac{\partial}{\partial x}(\bar{u}^2) + \frac{\partial}{\partial y}(\overline{uv}) + \frac{\partial}{\partial z}(\overline{uw})\right] \tag{2.4}$$

$$= -\frac{\partial \bar{p}}{\partial x} + \left[\frac{\partial}{\partial x}\left(\mu\frac{\partial \bar{u}}{\partial x} - \rho\overline{u'^2}\right) + \frac{\partial}{\partial y}\left(\mu\frac{\partial \bar{u}}{\partial y} - \rho\overline{u'v'}\right) + \frac{\partial}{\partial x}\left(\mu\frac{\partial \bar{u}}{\partial z} - \rho u'w'\right)\right]$$

Although the average of the velocity fluctuations (u', v', w') will always be zero, the average of the product of these fluctuations is not necessarily zero. This product may contribute significantly to the overall motion of flow. The term, $\left[\frac{\partial}{\partial x}\left(\rho\overline{u'^2}\right) + \frac{\partial}{\partial y}(\rho\overline{u'v'}) + \frac{\partial}{\partial z}(\rho\overline{u'w'})\right]$, known as the Reynolds stresses are the source of turbulence. There are an insufficient number of equations for the number of unknowns because the velocity fluctuations are not computed directly. Therefore, modeling the kinetic energy (k) and dissipation rate (ϵ) is done to overcome this issue. The k - ϵ method is widely used for many engineering applications, and provides reasonable accuracy for a wide array of flow geometries.

2.1.2 Turbulence Model Explanation - Renormalization group (RNG) k- ϵ

Derived from the Navier-Stokes equations using a rigorous statistical technique called renormalization group theory [11], the RNG model includes the following refinements to the standard k - ϵ model:

- The RNG models have an additional term in its ϵ equation that significantly improves accuracy for rapidly strained flows.
- The swirl effects on turbulence are included in RNG model, thus, enhancing accuracy of swirling flows.
- RNG models provide an analytical formula for turbulent Prandtl numbers, while standard k - ϵ models utilize user-specified, constant values.
- The RNG model consists of constants different from those in the standard k - ϵ model, and additional terms and functions in the transport equations for k and ϵ.

The RNG model was initially developed for quantum mechanics problems. It is a systematic procedure for isolating phenomena which exhibit disparate scales and self-similarity. Derivation of the RNG model rose from the Fourier transformation of the Navier-Stokes, which provides access to scale and frequency information. Fourier analysis requires either an infinite medium or periodic boundary conditions in a finite domain. A random forcing function is added to the Navier-Stokes equations to sustain turbulence. Forster et al. [12] showed that a power law form of the forcing function could produce a Kolmogorov $k^{-5/3}$ energy spectrum, an inertial range where k is the wave number. Yakhot and Orszag [11] and Dannevik, Yakhot, and Orszag [13] related the rate of energy input to the rate at which turbulence energy is dissipated, ϵ, providing a route to closure modeling and turbulence statistics.

The RNG model was applied to hydrodynamics by Forster et al. [12], and to turbulent analysis by Yakhot and Orszag [11]. Standard k-ϵ models are typically used in high Reynolds number scenarios. The RNG model provides an analytically-derived differential formula for effective viscosity that accounts for low-Reynolds number effects. The idea revolves around small scales, which are presumed to be responsible for dissipation of turbulence energy and are more universal than large scales. It is assumed that the small scales depend on the rate of energy input at high enough Reynolds numbers, and not on the details of how the energy is produced. Turbulence production is a result of mean shear and body forces, such as wall boundary layers, rotation, and buoyancy.

2.2 Numerical Methods

CFD is used to mathematically express the dynamics of flow. CFD utilizes computational power to perform numerical calculations which simulate experimental conditions. Numerical methods utilize discretization schemes which transform the conservation equations into algebraic ones. For any CFD calculation there is a computational grid, which divides the solution domain into elements where the parameters are computed.

The finite volume method (FVM) discretizes partial differential equations (PDE) into algebraic ones. For this method, the physical space is divided into many small sub-domains called control volumes. The shapes of these “cells” vary. The PDEs are recast on these cells and approximated by the nodal values or central values of the control volumes. The commercially available software package, ANSYS FLUENT, utilizes finite volume method, by which the nodes are at the center of the finite volumes, and for which the conservation equations are discretized into their integral form.

By default ANSYS FLUENT stores discrete values of the scalar φ at the cell centers. However, face values (φ_f), are required for the convection terms in the discretized transport equations, and must be interpolated from the cell center values. This is accomplished using an upwind scheme. Upwinding refers to the derivation of φ_f from quantities in the cell upstream, or upwind, relative to the direction of the normal velocity, v_n. Second-order accuracy was used to discretize the diffusion terms of the transport equations. Quantities at cell faces were computed using a multidimensional linear reconstruction approach. With this approach, higher-order accuracy is achieved at cell faces through Taylor series expansion of the cell-centered solution about the cell centroid. Thus when second-order upwinding is selected, the face value φ_f must be computed. Face values are required for the convection terms in the Navier-Stokes equations and must be interpolated from cell center values. The following expression is used to determine φ_f:

$$\varphi_f = \varphi + \nabla\varphi \cdot \vec{r} \tag{2.5}$$

where φ and $\nabla\varphi$ are the cell-centered value and its gradient in the upstream cell and $\vec{r}$ is the displacement vector from the upstream cell centroid to the face.

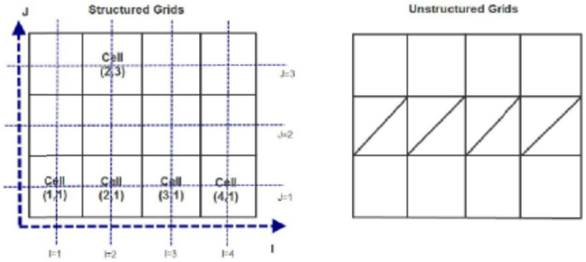
2.2.1 Solver Algorithm

The convective terms of the momentum equation are non-linear. The applied pressure field in the momentum equation must satisfy the continuity equation. Coupling of the momentum and continuity equations allows deriving the correct pressure. The Pressure-Implicitly with Splitting of Operators (PISO) algorithm was implemented into the solver to iteratively solve for this pressure. PISO is based on the higher degree of the approximate relation between the corrections for pressure and velocity. For improved efficiency, the PISO algorithm performs two additional corrections in addition to the correction of the Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) family of algorithms. These are the neighbor correction and the skewness correction.

2.2.2 Computational Grids

Flow domains are divided into much smaller domains called elements, or cells. Common sub-domains are quadrilaterals and triangles (2D) and hexahedra and tetrahedral (3D). The governing equations are discretized inside each of these elements to solve the approximated set of equations. The collection of cells is termed a mesh or grid.

Grids are classified as structured or unstructured based on the connectivity of the grid elements. The connectivity of a structured grid can be expressed as a two or three dimensional array. Thus, only quadrilaterals or hexahedra elements may be used, as shown in Figure 3. The regularity of the connectivity allows for space conservation. An unstructured grid, on the other hand, is characterized by irregular connectivity.



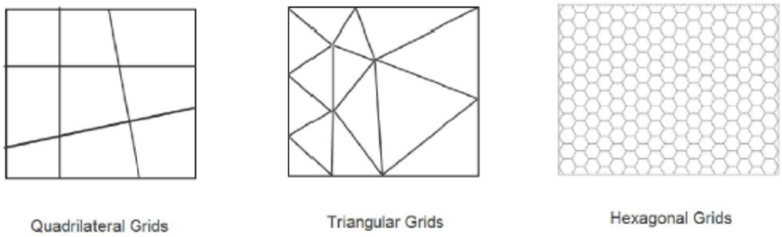


Figure 3 Details of structured and non-structured computational grids [13]

Computational storage requirements are much larger in comparison to structured grids. Depending on the grid type, an appropriate algorithm must be utilized for grid generation. For structured grids, the algebraic method and elliptic method are the two common methods. For unstructured grids, the Delaunay method and Quadtree/Octree methods are commonly used. A grid consists of points, faces, and volumes, as shown in Figure 4.

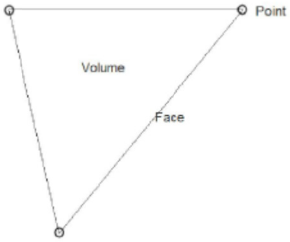


Figure 4 Tri-element [14]

The velocity and pressure are both stored in the center of each cell for unstructured grids. The velocity and pressure on the cell fact is interpolated by the cell central values of the two neighboring cells, as shown in Figure 5.

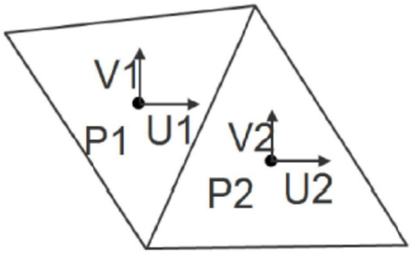


Figure 5 Two neighboring grid cells [15]

An iterative method is used to solve the Navier-Stokes equations. Arbitrary velocity and pressure fields are assigned to the cell center, and Navier-Stokes equations are solved to obtain a new velocity and pressure field. The velocity and pressure are coupled using the PISO algorithm. The corrected values are used to start a new iteration until a satisfactory velocity and pressure field are obtained.

Over recent years, numerical simulation has become an approach to tackling engineering and science questions in addition to the practical approach and theoretical approach. Numerical simulation is characterized by the discretization of mathematical equations; these equations are considered at only a finite number of selected points.

2.3 Optimization

Optimization has been applied to many facets that require making a decision amongst several choices. The decision of investing scarce resources such as time and money and getting the most out of them are issues everyone deals with. This is an example of how optimization has been applied to real-world situations.

To make the best choices, however, tools are required which aid in the decision-making process.

Engineering is not an exception to the topic of optimization. Incidentally it is through the use of engineering disciplines that we search to find some of the optimal choices and solutions to our problems. Computational modeling is a means by which science and engineering problems may be optimized through the use of algorithms to achieve better solutions. It is no exaggeration to say almost all research activities in engineering, science, and industry today involve a certain amount of modeling, data analysis, computer simulations, and optimization [15]. However, it is the discipline which drives the variations amongst the different problems focused on.

In the optimization process, the designer selects the most appropriate algorithm that will essentially find the optimum design. The algorithms are a sequence of operations based on mathematical equations and mimic the process of selection. The search to arrive at the optimum design can be affected by the uncertainty in considering all the possible variables and factors that may play a role. Therefore, seeking an optimal design requires that the process involved in the search, the algorithm, be reliable.

Modern methods of engineering design rely on the results of computer simulations. Designers have used the advances of technology to introduce complex systems and structures to the simulation process, and in turn have created a high demand for accuracy. Running a single simulation could cost several hours or even days of runtime. This process could become an inconvenience when a batch of hundreds of simulations is needed. However, these costly simulations are necessary to derive at accurate, optimal designs.

2.3.1 Optimization Algorithms

The selection of the optimization algorithm is a very important step to ensure the optimal solution has been reached. There are many optimization algorithms available, but it is important to know that a single algorithm may not solve all problems that may arise. Optimization algorithms fall under two types: deterministic methods or evolutionary methods. Deterministic methods have a strong mathematical background; they are also called gradient-based methods. Examples include Steepest Descent or Gauss-Newton method, which use the derivative to move to the minimum or maximum, depending on the objective. With these deterministic methods, if you start at the same location to perform a search, you will arrive to the same location. The other category of optimization algorithms is the evolutionary method. This method is population-based and uses members of a population to interact with one another to identify the fittest member of that population. Evolutionary type methods are classified as stochastic. Some methods under this type have strong ties to nature's programming, i.e. genetics and bird flocking. The creators of these methods have observed how nature operates and have created methods which simulate these natural processes. Stochastic methods usually start with a population where each member competes against one another and thrives on the variation between each member to obtain the optimum value. The manner in which the initial population of designs is determined is random and at each stage within the algorithm further randomization is introduced. An example of such a method is genetic algorithm.

2.3.2 Choice of Algorithms

The choice of algorithm has an influence on the quality and accuracy of the results. The type of algorithm chosen depends on the type of problem, nature of the algorithm, the desired quality of solutions, the availability of the algorithm implementation, and the expertise of the decision-makers [16]. It is not always possible to have all the desired resources available. With the assortment of many algorithms available and with some algorithms being more suitable may affect achieving success, due to not having them readily available to implement in a given system. However, the level of expertise in using these algorithms has a greater influence on the quality of work. The knowledge and experience used with these algorithms may be more valuable in selecting the most appropriate algorithm available than having the best algorithm.

2.4 Evolutionary Optimization Methods

The evolutionary type method considered here is based on nature programming, for example within genetics and bird flocking. The creators of these methods have observed how nature operates and have created methods to simulate the process. Stochastic methods usually start with a large set of members of a population where each competes against one another to obtain the lowest value. These methods thrive on the variation between each member to another to compete for the optimum value.

2.4.1 Genetic Algorithms

To arrive at the design which contains the most effective set of parameters, this research carried out an optimization study using evolutionary methods. This method is considered a heuristic method which does not rely on mathematical background [17]. The creators of these methods have observed and developed a sequence of operations which mimic how nature advances in each generation of a population to arrive at the fittest members. These methods usually start with a large population stochastically generated where each member of the population competes against one until an optimal member of the population has been determined. Genetic algorithms are an example of heuristic optimization. Each generation of the optimized population, the pool of designs, seeks to produce improved members or designs, from one generation to the next. This is accomplished by a three-stage process found within genetic algorithm programming: selection, crossover, and mutation.

After the initial population of design is generated, a fitness score is assigned to each member, or design, of the population. This fitness score is given based on how effective the design is at achieving the objective. In the present research the objective is to obtain maximum values of shear stress and a large value for average area shear stress on the bottom of the surface of a tank. For the total population of designs, a percentage is allotted to the fitness scores. Using this percentage, the algorithm uses a roulette wheel selection scheme for selecting the designs which make it to the next stage in the genetic algorithm. A description of this would be a pie chart and each fitness score is assigned a percentage. This percentage value is proportional to the fitness score and does not guarantee that the design with a high fitness score will be selected to proceed to the next stage; it only provides a probability that it will survive to the next step in the process.

The following step in the process, termed crossover, involves the exchange of "genetic" information between designs in the population. The crossover operation essentially produces new designs by swapping "genes" or design parameters, between two designs. Each gene or design parameter is assigned a number from 0 to 1. In the algorithm, the user specifies a value, such as 0.4. If the number specified for each gene is greater than 0.4, then those genes get swapped between the parent designs that are mating. However, if the gene value is less than 0.4, the bits between the mating parents do not get swapped.

The next step in the genetic algorithm process is mutation. The user must specify a probability for the genes to change. The crossover and mutation steps in the genetic algorithm introduce additional randomness into the process. Several iterations of the genetic algorithm process are required to reach the optimum solution.

2.4.2 Particle Swarm

An alternative optimization process to genetic algorithms is particle swarm which is based on methodologies of natural programming. The algorithm was developed by simplifying the behavior and methodology in bird flocking and fish schooling [18]. Particle Swarm is an evolutionary method which has similar ties to genetic algorithm. However, there are features of the programming that demonstrates its distinction to the genetic algorithm method, thus allowing it to be considered as an alternative to genetic algorithm.

There is a psychological description underlying in the particle swarm algorithm. The algorithm was created in 1995 by a psychologist and an electrical engineer [19]. Much like how we have sociability rank and individual rank amongst a group of people, particle swarm utilizes this idea to search for the best value amongst the entire population. As humans, if we tend to follow our own individuality, an individual has the probability to create but too much of this could retract our willingness to learn about others. On the other hand engaging with others heightens the learning process, accelerating within social ranks. Focusing too much on sociability hinders our new idea-making ability. Particle Swarm uses this theory to compare the best value of each iteration and the best value of the entire population up until the move forward to the global minimum.

Using the analogy of bird flocking, imagine a two dimensional grid which provides the location (x,y) and velocities (vx, vy) of birds. The birds' purpose is to search and find the best suitable nesting place. This is achieved by recording the location and velocity of each member in the population. During each iteration, the members determine the best location so far in the population and the best location in that particular iteration. It is then understood that two values are recorded for each iteration.

Each member then moves to the best nesting place by modifying its position by using the following information:

- the current position (x,y)
- the current velocities (vx, vy)
- the distance between the current position, and the best value so far and the best value in the population for that iteration.

Then the iterative procedure follows as:

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{2.6}$$

$$v_i^{k+1} = \alpha v_i^k + \beta r_{li}(p_i - x_i^k) + \beta r_{2i}(p_g - x_i^k) \qquad (2.7)$$

where:

x_i is the i-th individual vector of parameters, the current position at iteration k

$v_i = 0$, for $k = 0$, v is the velocity and k is the iteration counter

r_{1i} and r_{2i} are random numbers with uniform distribution between 0 and 1.

p_i is the best value found for the vector x_i, best value so far.

p_g is the best value found for the entire population, for the iteration.

$0 < \alpha < 1$; $0 < \beta < 1$ [17]

Reviewing equation 2.6, the term on the left hand of the equal sign represents the new position. The first term on the right hand side of this equation is the inertia of the member and it must decrease as the population determines the best nesting place. The second term on the right hand side represents individuality and the third represents sociability.

CHAPTER III

METHOD OF SOLUTION

3.1 Working Domain

A domain with dimension 84" X 84" X 12" was created using the Design Modular, part of the ANSYS Workbench. At the (-x) coordinate direction of the sidewall of the geometry, a small section of the pipeline attached to the nozzle and the nozzle itself was placed protruding into the domain. Figure 6 shows an image of the discretized domain.

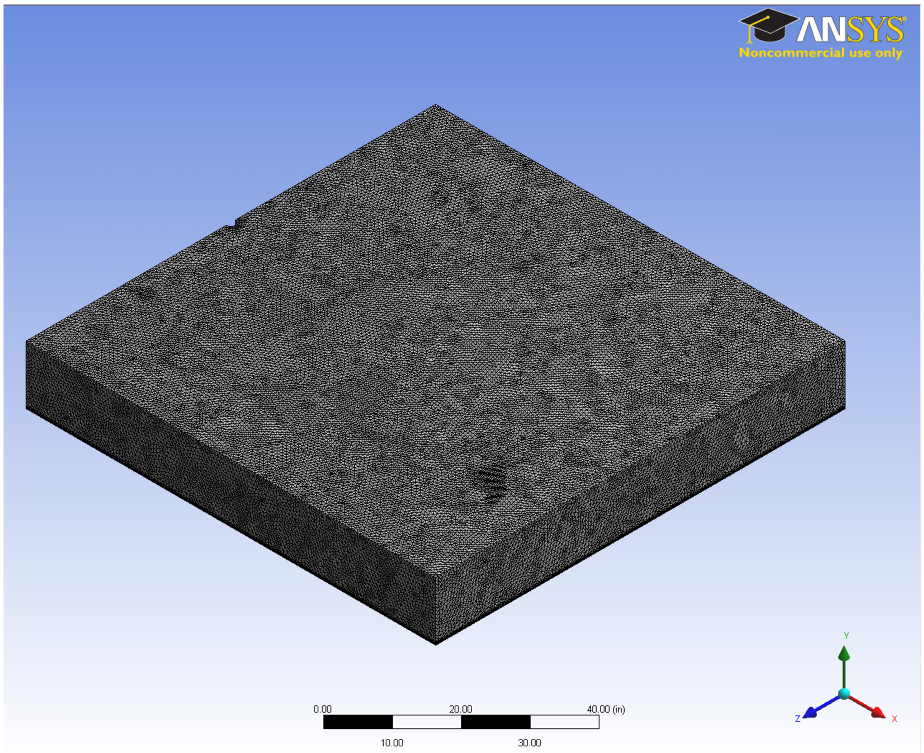


Figure 6 Unstructured grid

The grid of the domain was completed by the use of the ANSYS Workbench Mesher. The characteristics of the geometry and grid are as follows:

- Domain: 84" x 84" x 12"
- Unstructured grid containing 1,428,651 tetrahedral elements
- Skewness ratio was 0.8391/ 0.96
- Body sizing was specified to 0.6 in.
- A face sizing of 0.6 in was applied to the bottom wall of the tank
- Six inflated boundary layers were applied to bottom wall

Figure 7 illustrates the six inflated boundary layers on the bottom wall.

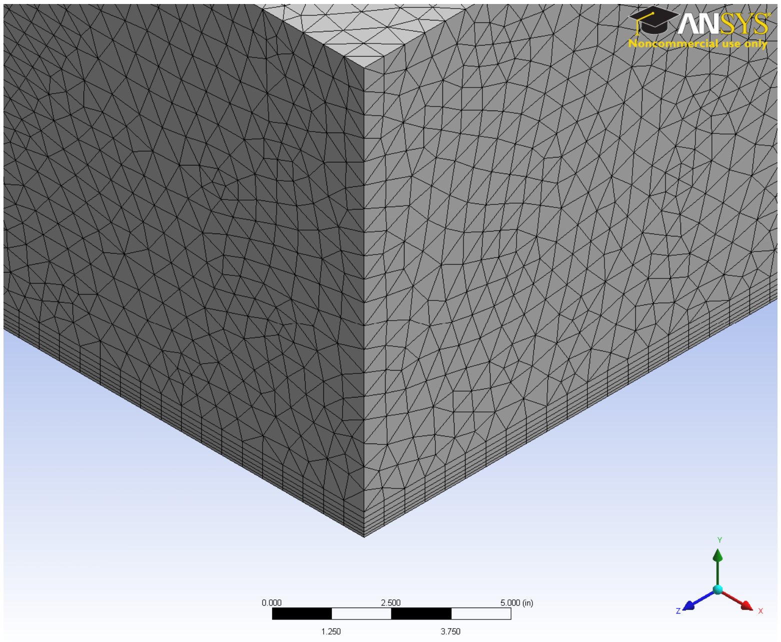


Figure 7 Inflated boundary added to capture the viscous effects

3.2 Parameterization of the Nozzle Design

ANSYS Workbench allows for the unique capability of parameterizing the nozzle geometry. This feature aids in generating the designs needed. Five parameters were used to characterize the nozzle design. The following ranges were used to constrain the parametric analysis:

- Angle : (-30°) - (- 60°)
- Left radius: 0.381 cm – 1.02 cm
- Right radius: 0.381 cm – 1.02 cm
- Distance between the radii: 1.40 cm – 2.79 cm

- The height of the nozzle from the bottom of the floor: 12.7 cm – 17.9 cm

Figure 8 provides a visual representation how the angle changes between designs.

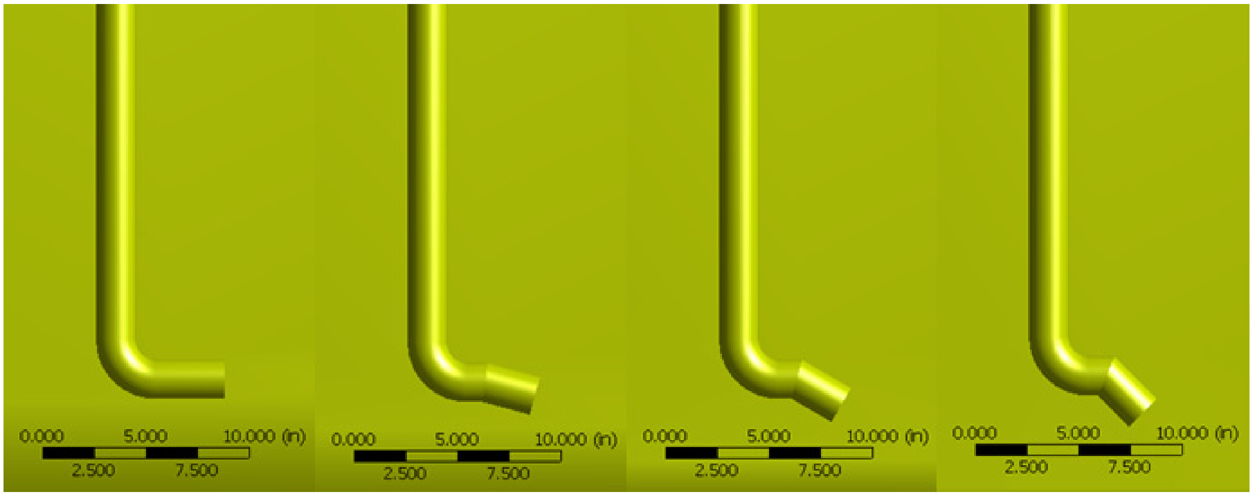


Figure 8 The nozzle is parameterized to allow it to rotate from (-30°) – (-60°) from the horizontal plane

The detailed image in Figure 9 illustrates the three parameters which make up the orifice.

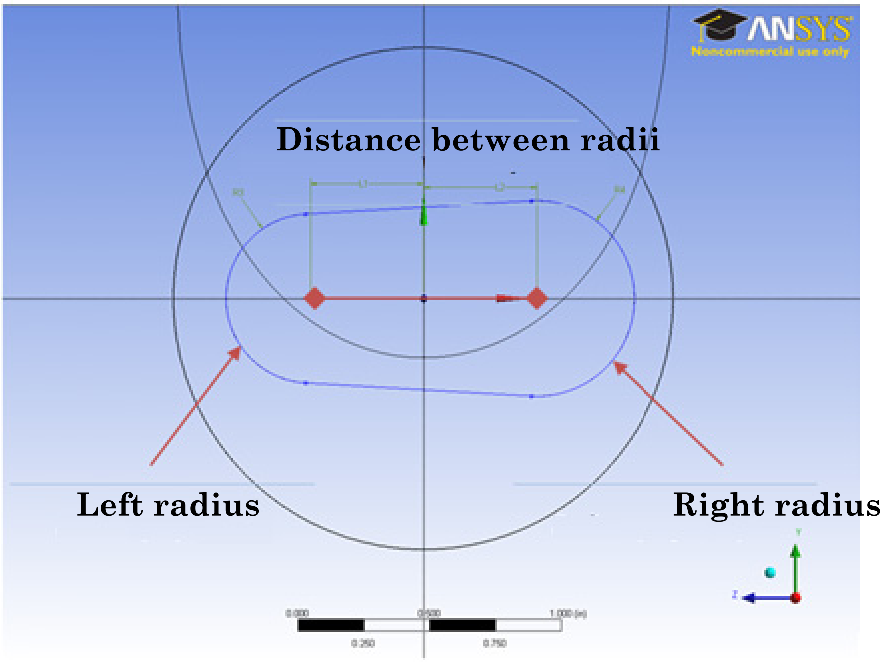


Figure 9 Parameters which defines the nozzle orifice

Figure 10 illustrates examples of the various orifices that can be created by the variation of the parameter pertaining to the orifice.

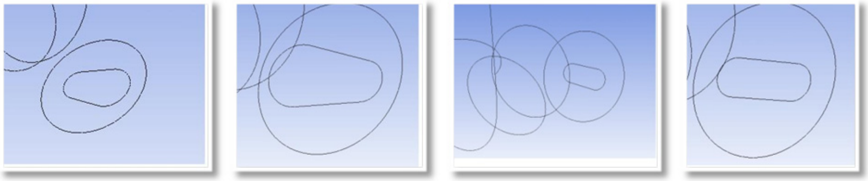


Figure 10 Examples of different nozzle orifices. Each orifice is defined by two design parameters.

Sobol's algorithm, provided in Appendix C, was used to randomize the combination of each design with the five different parameters. This algorithm was used to uniformly distribute nozzle designs. Using the algorithm, the nozzle designs fill the design space uniformly.

3.3 Utilization of the ANSYS FLUENT Solver

The CFD package used to solve the RANS equations is ANSYS FLUENT. Within ANSYS FLUENT, the turbulence model specified was RNG. The default coefficients in the RNG $k - \epsilon$ model used are listed below:

- C1- ϵ: 1.44
- C2- ϵ:1.92
- TKE Prandtl Number: 1
- TDR Prandtl Number: 1.3

Material properties and boundary conditions were specified in ANSYS FLUENT. The geometric domain was defined as a single-phase system which consisted of water (ρ=978 kg/m^3). The velocity specified at the inlet was selected to equal the discharge velocity produced by NuVision's technology. This velocity, 30.48 m/s, was used as a control to assess how the optimal design performed against the commercial design.

3.4 modeFrontier- Genetic Algorithm

In modeFrontier, the multi-objective optimization software, the Non-dominated Sorting Genetic Algorithm II (NSGA-II) was chosen to execute the optimization portion of the research.

NSGA-II is a fast and elitist multi-objective evolutionary algorithm. Within modeFrontier, an excel sheet was imported which contained all the parameters and associated shear stress values. One hundred real designs were used to create a response surface. The interpolating method used by modeFrontier for the response surface was radial basis function (RBF). RBFs have been developed for scattered multivariate data interpolation [20]. A study has shown that RBFs perform better than other known interpolation methods i.e. Kriging. The comparative study concluded that RBFs are more accurate and robust for different problem size, sample size, and for nonlinearity.

3.5 Grid Convergence Index

Numerical error is defined as the difference between error of the discrete solution and the exact solution. The discrete solution, *f(h)*, is a function of *h*, the specified grid spacing; *C*, a constant; and *p*, the order of convergence, and is defined as

$$f(h) = C\, h^{p} \tag{3.1}$$

A grid refinement study determines this order of convergence, *p*, also known as the rate of convergence. This is based on the solutions of three grid types (i.e., fine, medium, coarse) and is given by:

$$p = \ln\frac{\left(\frac{f_3 - f_2}{f_2 - f_1}\right)}{\ln(r)} \tag{3.2}$$

where:

f_i is the value of the function at specified grid size (i=1,2,3)

r is the grid refinement ratio.

Accuracy requires that the numerical solution fall within the asymptotic range of convergence [21]. The asymptotic range of convergence is reached when

$$C = \frac{discrete\ solution\ error}{h^p} \tag{3.3}$$

reaches constancy for various grid spacing and errors.

Roache [22] proposed a consistent approach to report results of grid refinement studies, by the use of the grid convergence index (GCI). This method is based upon a grid refinement error estimator derived from the Richardson Extrapolation, defined as the following:

$$error\ estimator = \frac{\epsilon}{r^p - 1} \tag{3.4}$$

where

ε is the relative error between successive grids

r is the grid refinement ratio

It provides a measure of the percentage the discrete solution is from the asymptotic numerical solution to determine *C*, in equation 3.3. A small GCI indicates the solution is within the asymptotic range. The GCI for a “fine” grid is determined by:

$$GCI_{fine} = \frac{F_s|\varepsilon|}{(r^p - 1)} \tag{3.5}$$

where:

F_s is the factor of safety

ε is the relative error between successive grids

r is the grid refinement ratio

A factor of safety of 1.25 is recommended for comparisons of three grids [23]. In instances where many CFD simulations are required, a coarse gird is used to compute the GCI study. This is defined by:

$$GCI_{coarse} = \frac{F_e|\varepsilon|r^p}{(r^p - 1)} \tag{3.6}$$

It is important that each grid level yields solutions that are within the asymptotic range of convergence for the computed solution. This can be checked by observing two GCI values as computed over three grids,

$$GCI_{23} = r^{p} GCI_{12} \tag{3.7}$$

3.6 Grid Convergence Study

The following procedure is carried out in conducting the grid convergence study. CFD tools are used to determine the wall shear stress for the bottom surface of the representative domain. The jet flow from the nozzle was computed on three grids, each with 1.5 times the body spacing specified in the grid. The table below indicates the grid information and the resulting wall shear stress computed from the solutions. Each solution was carried for 1000 iterations. Grid spacing was normalized with respect to the spacing of the finest grid.

Table 1 The values of maximum shear stress calculated from various grid sizes

Grid	Grid Size	Normalized Grid Spacing	Wall Shear Stress [Pa]
1	1	1	2670
2	1.5	2	2303
3	2.25	4	1413

As the grid spacing reduces, the wall shear stress approaches an asymptotic zero-grid spacing value. The order of convergence was determined to be, p = 2.18. A second order solution is assume, thus the theoretical order of convergence is p = 2. The difference is likely due to grid stretching, grid quality, non-linearities in the solution, presence of shocks, or turbulence modeling. The Richardson extrapolation, shown below

$$f_0 \cong f_1 + \frac{(f_1 - f_2) r^p}{r^p - 1} \tag{3.8}$$

is then applied using the two finest grids to obtain an estimate for the value of the wall shear stress at zero grid spacing for the bottom wall, determined to be 2774 Pa. The grid convergence index for the fine grid solution is then computed. A factor of safety of F_S=1.25 is used since three grids were used to estimate p, the convergence rate. Using equations 3.5, the GCI for grids 1 and 2 is 4.87%, and 13.7% for grids 2 and 3. The solutions are then checked to determine if they are in the asymptotic range of convergence. Using equation 3.7, the percentage the measure of the values for wall shear stress is 1.16 which indicates that the values obtained are within the asymptotic range of convergence. Based on this study the wall shear stress is estimated to be 2775 Pa with an error band of 4.87%. The calculated error found in this GCI study may be associated with the CFD errors cause by boundary conditions and specified flow conditions. Error may have been introduced due to the grid spacing between the inlet boundary of the nozzle and grid spacing of the domain; this may have led to discretization errors. In addition, the simulation needed to be stopped eventually but iterative convergence errors may exists because the simulation was stopped ahead.

Figures 11, 12, 13 are contour plots that demonstrate the formation of the shear stress as the cells were increased in the three grids that were investigated.

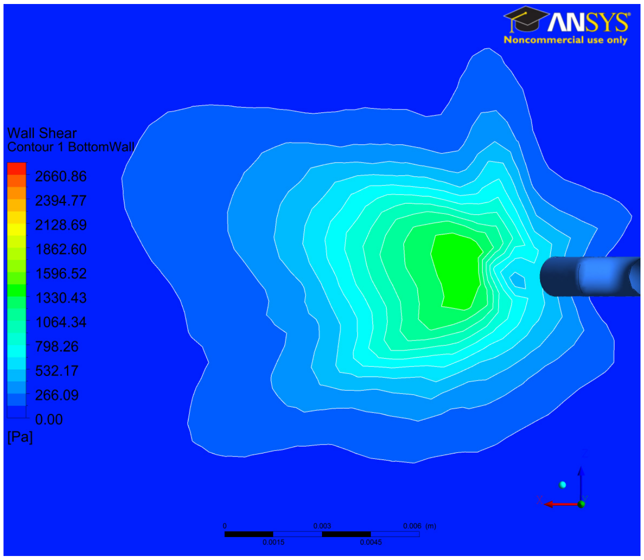


Figure 11 Coarse Grid: 553,521 tetrahedral elements

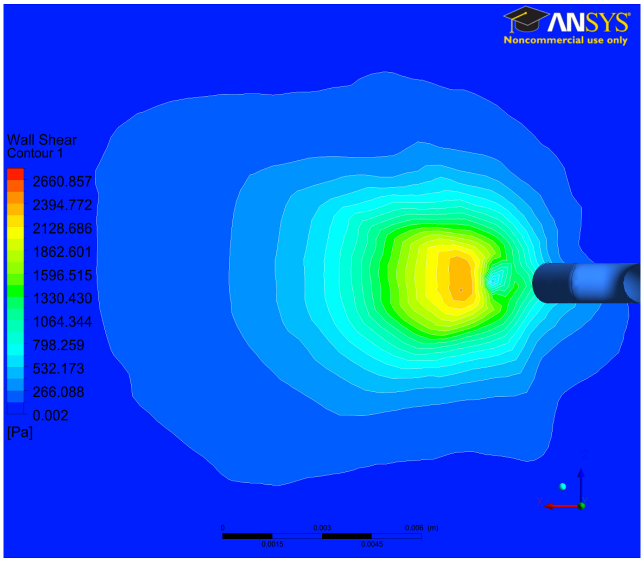


Figure 12 Medium Grid: 1,428, 651 tetrahedral elements

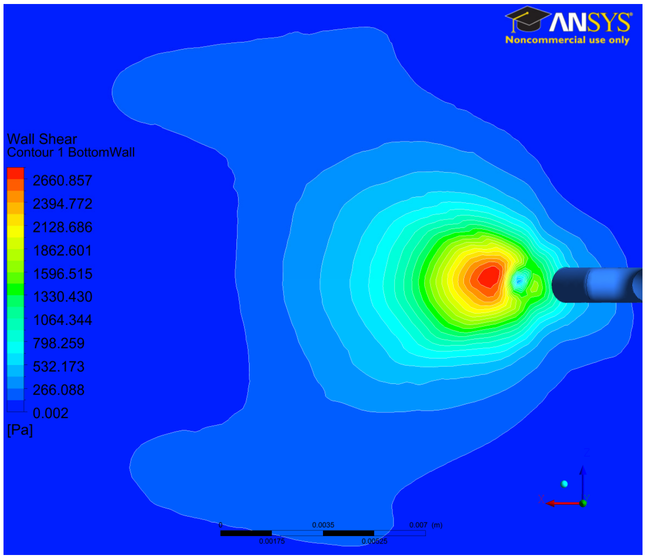


Figure 13 Fine Grid: 3,298, 902 tetrahedral elements

CHAPTER IV

RESULTS AND DISCUSSION

4.1 Creation of Sobol Designs

This research aims to use modifications in system parameters to enhance the performance of currently used nozzles for the removal of sludge heel waste fixed to the bottom of storage tanks. The performance of the new nozzle designs was evaluated for their effectiveness in achieving two objectives: (1) maximizing the local shear stress and (2) increasing the overall shear stress on the bottom floor of the waste tank. The initial process began with the generation of 100 randomized designs. For the creation of the matrix of designs, Sobol's algorithm was implemented to randomize the parameters that would describe each design. Sobol's algorithm promotes uniqueness across each design.

4.1.1 Generating a Pool of Candidate Designs

Each of the 100 initial real designs was solved for 1000 iterations using the ANSYS FLUENT solver within the ANSYS Workbench. After the completion of each simulation, the maximum value for shear stress and average area shear stress calculated at the bottom of the tank were recorded for each design. The maximum shear stresses from the 100 initial real test cases are shown in Table 2.

Table 2 Five designs resulting in the highest maximum shear stress values from the initial 100 designs.

Test Case	Angle(deg.) Below the Horizon	Left Radius (cm)	Right Radius (cm)	Nozzle Height (cm)	Distance Between Radii (cm)	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)
31	54.51	1.0	0.5	13.6	1.0	**2899**	46
61	54.98	0.8	0.9	14.5	1.1	**2876**	51
85	53.81	0.9	0.6	12.7	1.1	**2826**	45
19	46.08	0.9	0.9	13.1	1.4	**2747**	57
11	51.70	0.7	0.7	15.4	1.2	**2697**	44

Likewise, Table 3 shows the designs that produced the highest values for maximum average shear stress.

Table 3 Five designs resulting in the highest average area shear stress values from the initial 100 designs

Test Case	Angle(deg.) Below the Horizon	Left Radius (cm)	Right Radius (cm)	Nozzle Height (cm)	Distance Between Radii (cm)	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)
73	48.19	0.5	1.0	14.9	0.9	2291	**73**
82	27.56	1.0	0.9	14.6	1.3	1417	**60**
41	52.17	0.9	0.8	17.5	0.8	1891	**59**
66	34.36	0.9	1.0	16.1	1.2	1669	**58**
52	39.05	0.9	0.9	13.4	1.3	2291	**58**

The overall maximum shear stress value generated from the 100 initial real designs is just under 3000 Pa. It is expected that the optimized design would be a hybrid design of parameter values from Tables 2 and 3, producing greater shear stress values.

Figure 14 provides the contour plot of Test Case 31 which illustrates the shear stress created with the design parameters shown in Table 2.

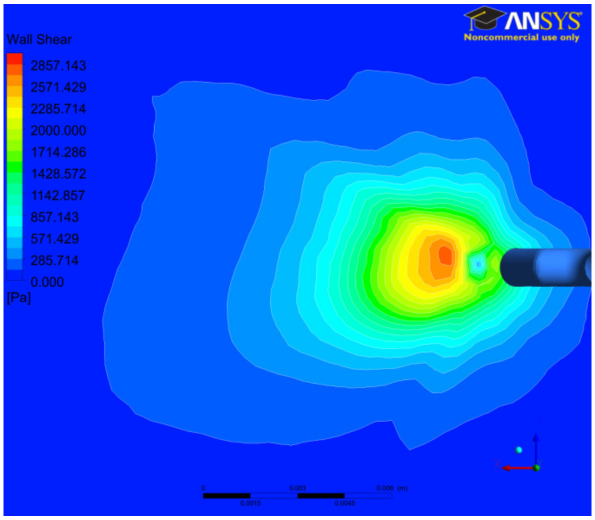


Figure 14 TestCase_031 Contour of shear stress distribution using parameters that provided the maximum shear stress.

4.2 Creating a Response Surface of the 100 Initial Designs

From the initial 100 real designs, each with 5 input variables and 2 objective output variables, a response surface was created, which provides a surrogate model that describes the behavior of these initial designs, referred to as ‘parent’ designs. The parent designs establish the boundary from the limits of the parameters. 50,000 virtual (offspring) designs were generated iteratively via the NSGA II, thus, improving the parent's results. Each of the 500 generations produced 100 new virtual designs aimed to satisfy the two objectives for shear stress.

The scatter plot in Figure 15 includes the 100 initial designs, each denoted by a green box. The response surface was created based on these 100 designs. The incorporation of genetic algorithm in modeFrontier yield improved designs from the initial 100 real designs. The objective of the software was to maximize shear stress values produced from the five parameters for each of these 100 real designs.

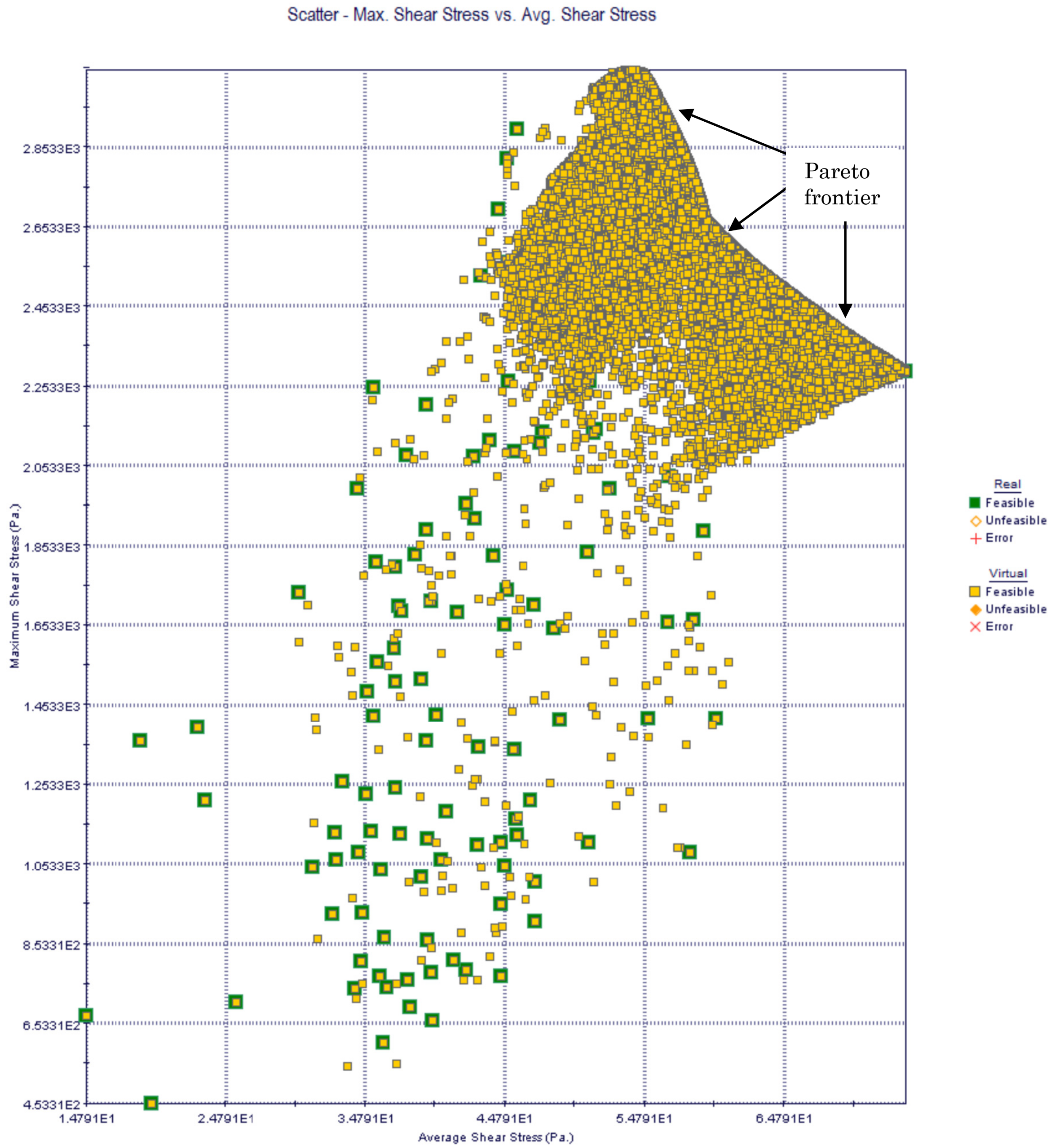


Figure 15 Scatter plot of the real (100) and virtual (50,000) designs

Figure 15 shows a “cloud” formation from the concentration of virtual designs. This “cloud” tends to move towards the upper-right direction of the plot. This is an indication that the values of the objectives, maximum shear stress and maximum average shear stress, are increasing for the virtual designs. The designs which align themselves at the outer boundary of this "cloud" are termed Pareto designs. These designs are classified as being the best designs. The boundary along which the designs align themselves is called the Pareto frontier.

4.3 Verification of the Response Surface

A process to verify the response surface was required to determine whether the surrogate model captured the behavior of the initial 100 real designs. The error between the real designs, solved in ANSYS FLUENT, and the predicted virtual designs, produced by the genetic algorithm executed by modeFrontier, is a good indicator of how accurate the surrogate model is.

Four virtual designs, along with their shear stress values, were randomly chosen from the designs created by modeFrontier. These chosen designs were used to verify the response surface by comparing the predicted values against the real design values for maximum shear stress and average shear stress. Table 4 provides the designs and the associated parameters used for verification purposes and Table 5 shows the calculation of the error between the predicted values and the actual calculated values of the simulations.

Table 4 Designs used to verify the response surface

mF ID	Angle(deg.) Below the Horizontal	Left Radius (cm)	Right Radius (cm)	Nozzle Height (cm)	Distance Between Radii (cm)	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)
1957	50.73	0.6	1.0	14.7	0.9	2493	65
5090	50.37	1.0	1.0	12.7	1.1	2966	56
6748	54.98	1.0	1.0	12.7	1.0	3108	55
8199	49.13	0.5	1.0	14.9	0.9	2362	70

Table 5 Error estimation of the shear stress between modeFrontier designs and ANSYS FLUENT designs

	modeFrontier		**ANSYS FLUENT**		**% Error**	
ID	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)	Maximum Shear Stress	Average Shear Stress
1957	2493	65	2348	46	**6.2**	**41**
5090	2966	56	3213	61	**7.6**	**7.9**
6748	3108	55	3240	57	**4.0**	**4.3**
8199	2367	70	2311	80	**2.4**	**12**

Test case ID 1957 had a noticeably large error (41%) for the average shear stress. Improving the accuracy of the response surface would eliminate these large shear stress discrepancies between virtual designs and real designs. Additional designs were randomized using the Sobol's algorithm. The following real designs in Table 6 were added to refine the response surface.

Table 6 Additional designs added to the initial 100 real designs to refine the response surface

Test Case	Angle(deg.) Below the Horizontal	Left Radius (cm.)	Right Radius (cm.)	Nozzle Height (cm.)	Distance Between Radii (cm.)	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)
102	29.47	0.4	0.9	14.7	0.7	782	40
103	37.20	0.8	0.5	15.4	0.7	1039	36
104	48.51	0.8	0.6	17.3	0.8	1595	37
105	50.85	0.6	0.5	16.4	1.4	1688	37
106	39.60	0.6	0.7	16.8	0.9	1130	37
107	30.23	0.4	1.0	14.9	1.1	1741	45
108	27.42	0.8	0.9	16.3	0.8	1008	47
109	38.67	0.6	0.7	12.8	0.9	1714	39
110	48.04	1.0	0.5	14.7	1.3	2532	50

Again, the modeFrontier virtual designs that were used to verify the response surface are presented in Table 7.

Table 7 Designs used to verify response surface after TestCase_102 through TestCase_110 were added using genetic algorithm

mF ID	Angle(deg.) Below the Horizontal	Left Radius (cm.)	Right Radius (cm.)	Nozzle Height (cm.)	Distance Between Radii (cm.)	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)
284	28.61	0.8	0.7	16.2	1.3	1123	56
4682	54.92	0.9	0.9	12.7	1.1	3040	52
6897	52.15	0.9	1.0	14.6	1.0	2645	52
4109	49.63	0.5	1.0	14.9	0.9	2372	70

This process is necessary to determine whether the addition of real designs (Table 6) improved the accuracy in the response surface. Table 8 shows the errors associated with the additional designs added to the response surface.

Table 8 Calculation of the error between real designs and virtual designs after adding design #102 to #110 in developing the response surface using the genetic algorithm

	modeFrontier		**ANSYS FLUENT**		**% Error**	
mF ID	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)	Maximum Shear Stress	Average Shear Stress
284	1123	55	1022	46	**9.8**	**0.2**
4109	2372	69	2347	45	**1.0**	**55**
4682	3040	52	3241	56	**6.2**	**6.9**
6897	2645	52	2780	55	**4.9**	**5.2**

In examining Table 8, noticeable change was not observed in the improvement of the response surface. A significant tool is to produce a robust response surface, which can lead to identification of a design that is the optimum in generating maximum shear stress and maximum average shear stress on the bottom surface of the tank. In turn, refining the local area in the response surface where observed optimal designs are located is of great interest.

4.4 Use of Pareto Designs to Refine the Local Area of Optimized Designs

The refinement of the region where the optimal design is found is discussed in the following section. Using the Parallel Coordinates chart, shown in Figure 16, is an effective tool for identifying the best designs, referred to as the Pareto designs. The Parallel Coordinates chart is provided with tabs that can be adjusted vertically to remove designs that are of no interest to the designer. In Figure 16, the shear stress values for the real and virtual designs are plotted on the two rightmost axes. Raising the bottom tabs removes the designs that exhibit low values for maximum shear stress and average shear stress, and is increased until the desired values are obtained.

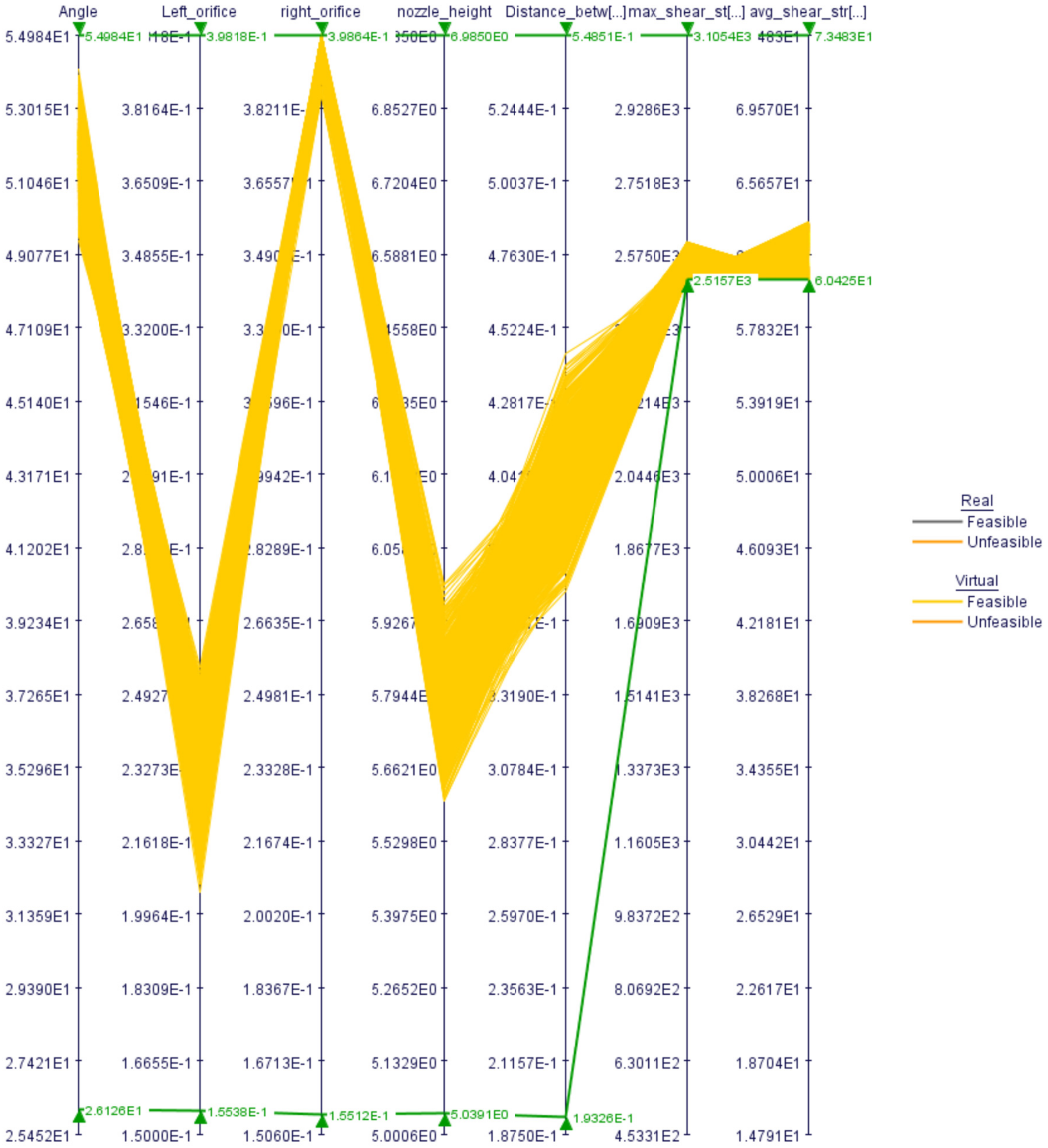


Figure 16 Parallel Coordinates charts. Plot of 50,000 designs; removing designs of no interest.

Both tabs are raised simultaneously until the optimal design is reached; that is, large values for both objectives are obtained. After the designs with low shear stress values were filtered out, four designs were selected to further refine the response surface. Referring to Figure 17, the designs denoted were those chosen to refine the response surface; ID 5749, ID 40277, ID 40286, and ID 58308.

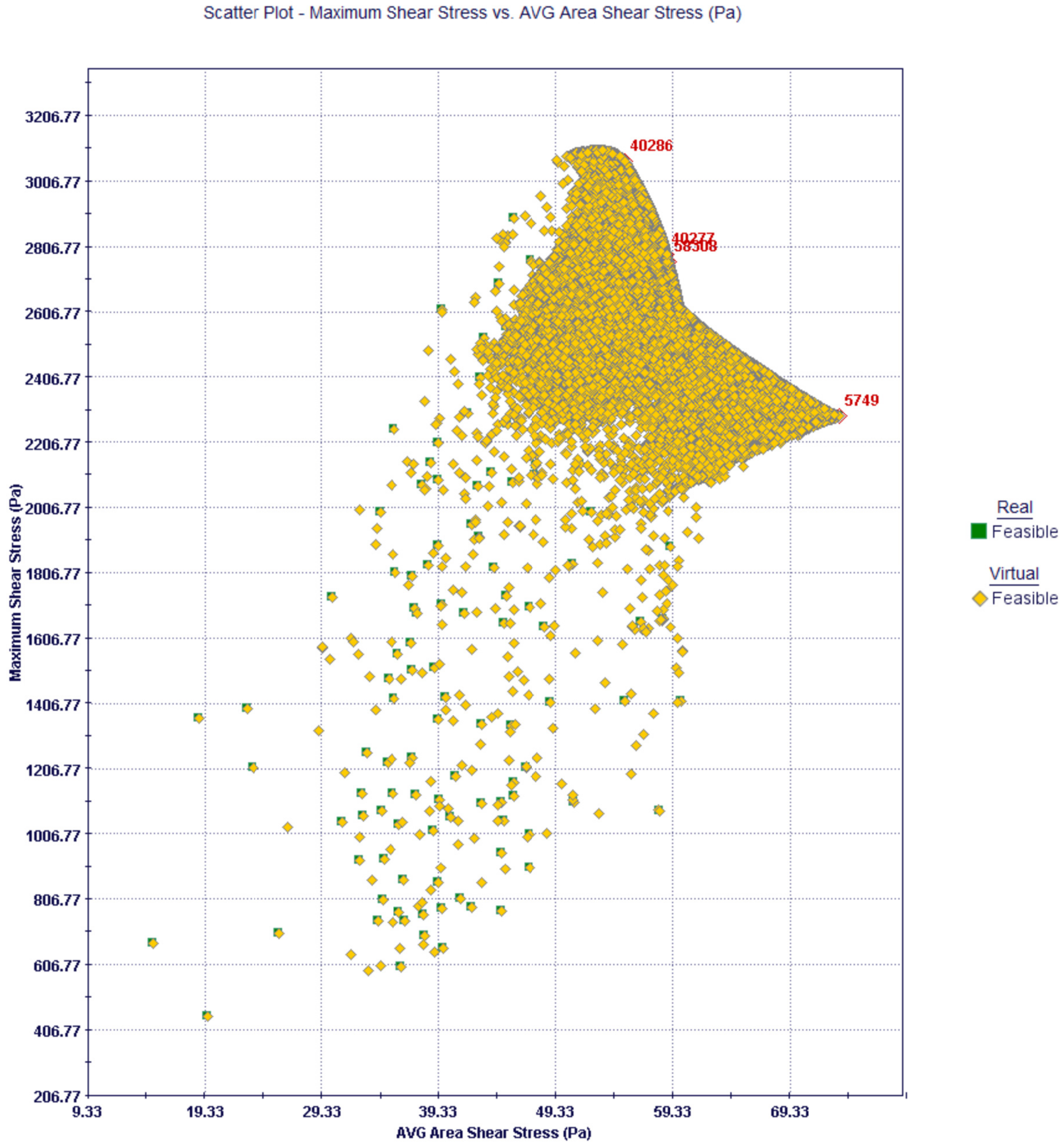


Figure 17 Identifying the designs chosen from the Pareto frontier

The values of the Pareto designs are listed in Table 9.

Table 9 Pareto designs chosen to verify the response surface

Test Case	Angle(deg.) Below the Horizon	Left Radius (cm)	Right Radius (cm)	Nozzle Height (cm)	Distance Between Radii (cm)	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)
5749	48.19	0.5	1.0	14.9	0.9	2291	73
40277	47.10	1.0	1.0	13.4	1.4	2788	59
40286	54.98	1.0	1.0	12.7	1.3	3070	55
58308	46.75	1.0	1.0	13.5	1.4	2766	59

After the inclusion of these designs in Table 9 to generate an updated response surface, a new population of offspring designs was created. Four of the virtual designs were selected to verify the response surface. Table 10 provides the values obtained by modeFrontier. It can be observed that virtual designs did not result in the expected values produced by ANSYS FLUENT simulations. The region where the optimal designs are located may not be well defined, and may require additional real design points for improved accuracy.

Table 10 Calculation of percentage error between ANSYS values (real) vs. modeFrontier(virtual) values

	modeFrontier		**ANSYS FLUENT**		**% Error**	
mF ID	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)	Maximum Shear Stress	Average Shear Stress
5749	2291	73	2083	43.84	**10**	**68**
40277	2788	58	3150	66.75	**-12**	**-12**
40286	3070	55	3382	64.55	**-9.2**	**-15**
58308	2766	59	2925	67.05	**-5.4**	**-12**

4.5 Conducting a Sensitivity Analysis on the Pareto Designs

The aim of this thesis is to present jet nozzles which produce large shear stresses on the bottom surface of a waste storage tank for the efficient cleanup of radioactive waste. The placement of such a nozzle inside the waste tank requires it be inserted through specific access points. The installation process is a task in itself and is a cause for undesirable errors. When installing these nozzles the centerline connecting the two radii should be parallel with level ground. This section investigates how a slight rotational shift of the nozzle's alignment from the horizon may cause unexpected results. Using the Pareto designs in Table 9, a sensitivity analysis was performed to evaluate changes in shear stress generated due to rotational shifts of ±5 degrees from the horizon. Table 11 provides a comparison between nozzles with proper alignment and those same nozzles with a (-5) deg rotational shift.

Table 11 Pareto design with the nozzle rotated (-5) degrees

mF ID	ANSYS FLUENT Maximum Shear Stress (Pa)	ANSYS FLUENT Average Shear Stress (Pa)	- 5 deg. design Maximum Shear Stress (Pa)	- 5 deg. design Average Shear Stress (Pa)	% Error Error Associated w/ Maximum Shear Stress	% Error Error Associated w/ Average Shear Stress
5749	2083	44	2194	44	**5.3**	**0.2**
40277	3150	67	2933	67	**-6.9**	**0.1**
40286	3382	64	3447	66	**1.9**	**2.3**
58308	2925	67	2879	67	**-1.7**	**0.5**

The values in the table show discrepancies between the shear stresses. This indicates that prescribed tolerances must be followed for the proper and accurate installation of these nozzles. Table 12 provides the values from the sensitivity study at a rotated angle of (+5) degrees. From the shear stress values of Table 11 and Table 12, design ID 5749 is very sensitive to shifts from the horizontal. With a +5 degree change, ID 5749 has an error of 12.8%. The other Pareto designs investigated, ID 40277, ID 40286, and ID 58308, are less sensitive to changes in alignment.

Table 12 Pareto design with the nozzle rotated (+5) degrees

mF ID	ANSYS FLUENT Maximum Shear Stress (Pa)	ANSYS FLUENT Average Shear Stress (Pa)	+ 5 deg. design Maximum Shear Stress (Pa)	+ 5 deg. design Average Shear Stress (Pa)	% Error Error Associated w/ Maximum Shear Stress	% Error Error Associated w/ Average Shear Stress
5749	2083	44	2349	45	**13**	**0.2**
40277	3150	67	2951	67	**-6.3**	**0.1**
40286	3383	65	3329	65	**-1.6**	**2.2**
58308	2925	67	2908	67	**-0.6**	**0.5**

4.6 A Comparison Between the Pareto Designs and a Commercial Design

The performance of the jet nozzle produced by modeFrontier was compared with a commercially available nozzle design developed by NuVision Engineering. This design has a circular orifice with diameter of 7.67 cm. All the nozzle designs used in this optimization study are distinct from one another. The optimized nozzles have an orifice defined by two independently variable radii that are aligned on the horizontal centerline of the nozzle and separated by a variable distance. The radii are connected by straight segments at a point of tangency. Figure 18 shows the contour plot for shear stress generated at the bottom of the tank from the commercially available nozzle. It generated 3105 Pa of maximum shear stress and 55.64 Pa for the average shear stress.

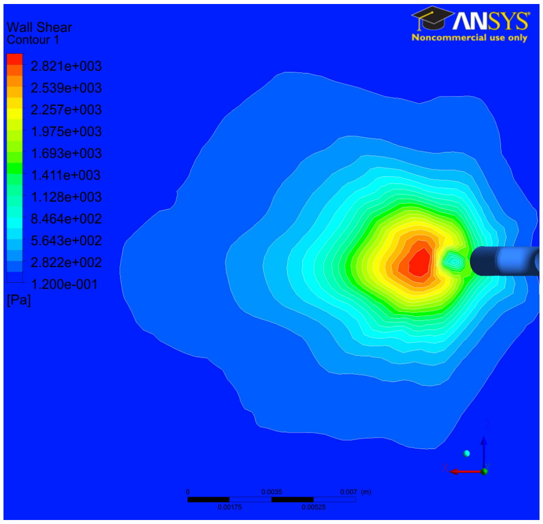
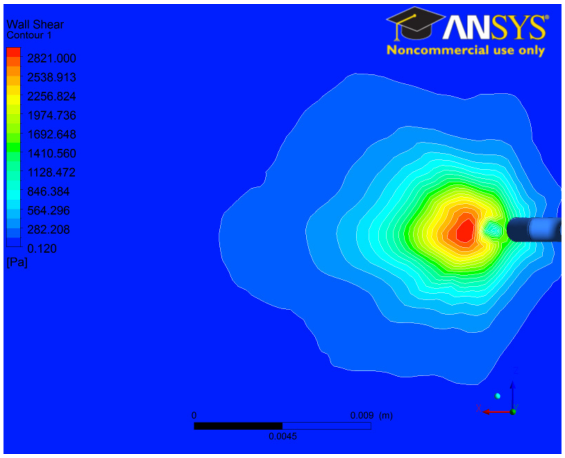


Figure 18 NuVision Engineering design performance. Generating 3105 (Pa) of maximum shear stress and 55.6 (Pa) of AVG area shear stress

The four optimum designs on the Pareto frontier shown in Table 9 were used to conduct a side-by-side comparison against the NuVision Engineering design. The following contour plots, Figures 19 – 22, provide a visual representation of the distinction between the generated shear stress.



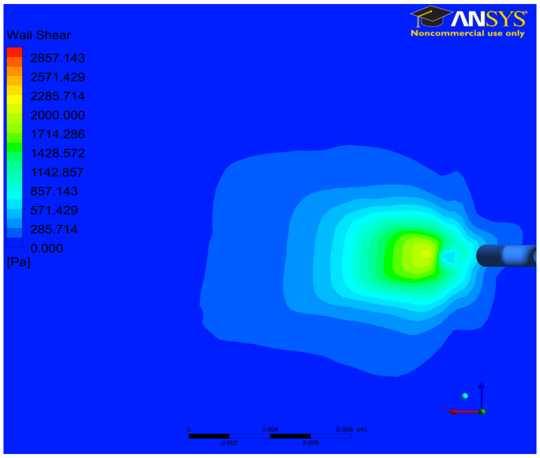


Figure 18 NuVision Engineering Nozzle
Maximum Shear Stress: 3105 (Pa)
Avg. Shear Stress: 55.6 (Pa)

Figure 19 Pareto Design: ID 5749
Maximum Shear Stress: 2083 (Pa)
Avg. Shear Stress: 43.8 (Pa)

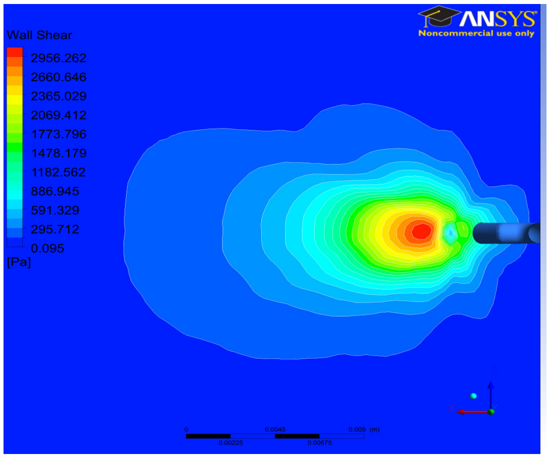
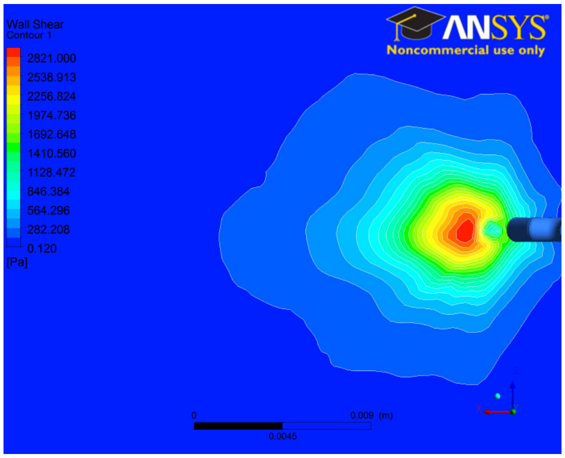


Figure 20 Pareto Design: ID 40277
Maximum Shear Stress: 3150 (Pa)
Avg. Shear Stress: 66.7 (Pa)



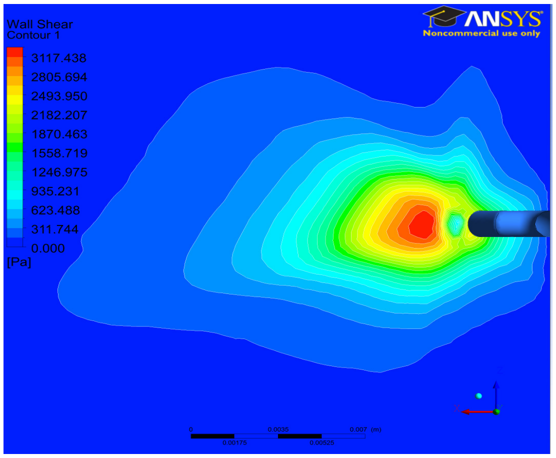


Figure 18 NuVision Engineering Nozzle
Maximum Shear Stress: 3105 (Pa)
Avg. Shear Stress: 55.6 (Pa)

Figure 21 Pareto Design: ID 40286
Maximum Shear Stress: 3382 (Pa)
Avg. Shear Stress: 64.5 (Pa)

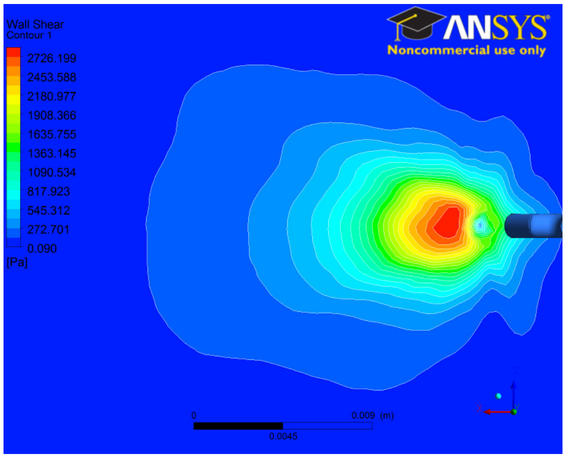


Figure 22 Pareto Design: ID 58308
Maximum Shear Stress: 2925 (Pa)
Avg. Shear Stress: 67.0 (Pa)

The performance of the Pareto designs was evaluated by determining the percentage difference in shear stress values against the NuVision Engineering design, as shown in Table 13. It can be seen from the table that design ID 5749 underperformed against NuVision's nozzle design by 33% and 21% for maximum shear stress and average shear, respectively. However, design ID 40277 and ID 40286, exceeded both the objectives of optimization against the NuVision Engineering design.

Table 13 Performance evaluation of NuVision Engineering nozzle and Pareto design nozzle

	ANSYS FLUENT		Performance Evaluation (%)	
mF ID	Max Shear Stress (Pa)	Avg. Shear Stress (Pa)	Max Shear Stress	Avg. Shear Stress
5749	2083	44	-33	-21
40277	3150	67	1.3	20
40286	3382	65	8.8	16
58308	2925	67	-5.8	21

Examining the Pareto designs produced by modeFrontier, Table 14 demonstrates the best design achieved through the optimization process, initiated with 100 initial real designs.

Table 14 The best design of the Pareto frontier

Pareto (mF) ID	Angle(deg.) Below the Horizontal	Left Radius (cm)	Right Radius (cm)	Nozzle Height (cm)	Distance Between Radii (cm)	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)

49887	54.98	1.0	1.0	12.7	1.1	3475	59

Figure 23 is the contour plot of the shear stress generated by the best design. Design ID 49887 nozzle performed 11.88% higher in maximum shear stress and 6.68% higher in average shear stress when compared with the NuVision Engineering design.

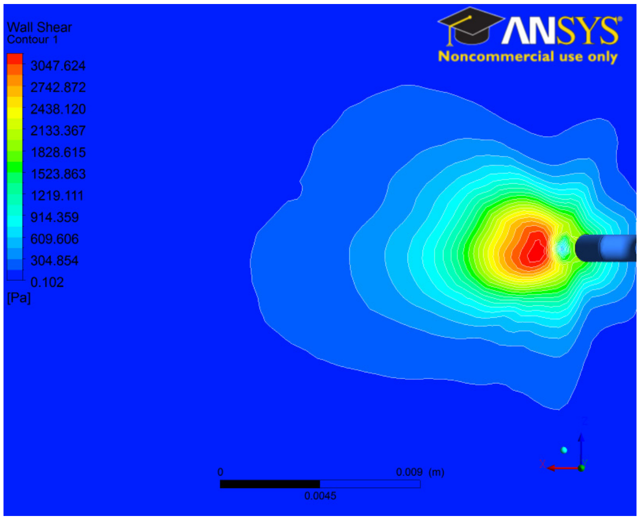


Figure 23 Best Design: ID 49887

4.7 Comparative Analysis of Two Optimization Algorithms: Use of Another Evolutionary Method for Design Optimization

Using modeFrontier, an alternate algorithm was incorporated to optimize the nozzle design, the particle swarm algorithm. Table 15 shows the virtual designs that were selected after the response surface was created with 112 real designs optimized in an earlier section of this chapter. The 112 real designs were subjected to the same process optimization procedure; the only distinction was the use of the multi-objective particle swarm algorithm. The performance between the designs generated from these two algorithms is compared here.

Table 15 Design randomly selected to verify the optimization with the used of particle swarm

ID	Angle(deg.) Below the Horizontal	Left Radius (cm)	Right Radius (cm)	Nozzle Height (cm)	Distance Between Radii (cm)	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)
12298	54.98	1.0	1.0	12.7	1.1	3105	53
24496	49.32	1.0	1.0	13.1	1.4	2889	58
25995	51.68	1.0	1.0	13.2	1.4	2947	57
49600	50.96	1.0	1.0	13.1	1.4	2938	57

For convenience purposes Table 8 is presented here again and is labeled Table 16.

Table 17 contains the analysis between virtual designs and real designs using the particle swarm algorithm. Evaluating only the data in Table 17, the virtual design values under predicted all the values of the objectives against real design values.

Table 16 Calculation of the error between real designs and virtual designs after adding Test Case #102 to Test Case # 110 in developing the response surface using the genetic algorithm

	modeFrontier		**ANSYS FLUENT**		**% Error**	
ID	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)	Maximum Shear Stress	Average Shear Stress
284	1123	55	1022	46	**9.8**	**0.2**

4109	2372	69	2348	45	**1.0**	**55**
4682	3040	52	3241	56	**6.2**	**7.0**
6897	2645	52	2780	55	**4.9**	**5.3**

Table 16 Calculation of the error between real designs and virtual designs using the particle swarm algorithm and a response surface

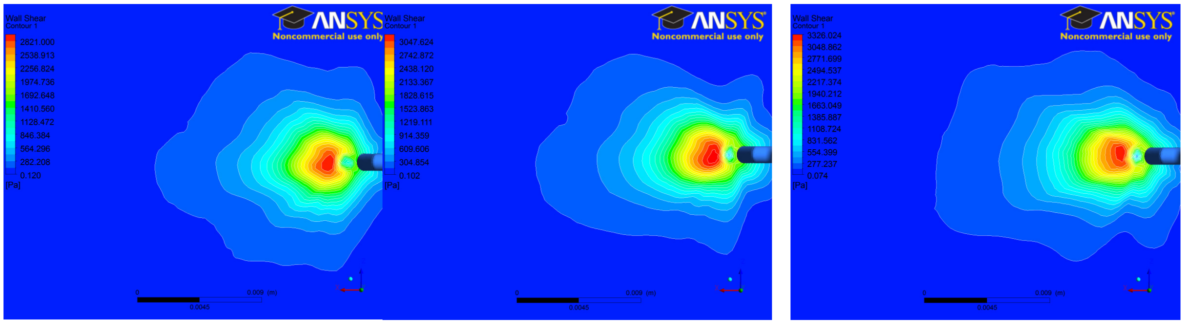
ID	**modeFrontier**		**ANSYS**		**% Error**	
	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)	Maximum Shear Stress	Average Shear Stress
12298	3105	53	3451	61	**-10**	**-13**
24496	2889	58	3407	61	**-15**	**-4.7**
25995	2947	57	3235	66	**-8.9**	**-14**
49600	2938	57	3240	67	**-9.3**	**-14**

A comparison between Table 16 and Table 17 reveals the designs in Table 17 to have larger shear stress values obtained from the real ANSYS FLUENT simulations. This confirms that particle swarm evolutionary optimization algorithm is capable of converging further than a genetic algorithm. Specifically,

- The best Pareto-optimal design obtained using the particle swarm optimizer was 23% better than the current NuVision Engineering design in achieving maximum shear stress at the tank bottom (as opposed to 11.88% improvement achieved when using genetic algorithm) as confirmed by the ANSYS FLUENT flow-field analysis software.
- The best Pareto-optimal design obtained using the particle swarm optimizer was 18% better than the current NuVision Engineering design in

achieving maximum average shear stress at the tank bottom (as opposed to 6.68% improvement achieved when using genetic algorithm) as confirmed by the ANSYS FLUENT flow-field analysis software.

The iterative process of the particle swarm optimization algorithm allows it to produce larger values for the objectives. When an "ideal" trait is identified within a population, the algorithm stores this trait i.e. parameter. Further generations are produced, and similarly, the "ideal" trait is identified. The trait carried on to subsequent generations is the additive result of the best of the population and the best from the genealogy. With genetic algorithm, on the contrary, there is a possibility that the "ideal" trait may be lost in subsequent generations due to mutation phase. However, influential characteristics from the "ideal" trait in particle swarm are always carried to subsequent generations. This yields better solutions in achieving the objectives. Presented below is Figure 24, comparing shear stress distributions on the bottom of a waste tank calculated for the NuVision Engineering nozzle, a Pareto optimal nozzle designed using genetic algorithms (ID 49887), and a Pareto optimal nozzle designed using particle swarm algorithm (.PS_ID 60480). Both optimized nozzles produce larger shear stress values than the currently used commercially available nozzle (NuVision Engineering). However, between ID 49887 and PS_ID 60480 (derived by a genetic algorithm and a particle swarm algorithm, respectively) the optimized nozzle designed by particle swarm generated larger values of shear stress at the bottom of the waste tank (Table 18).



NuVision Engineering **ID 49887** **PS_ID 60480**

Figure 24 Contour plots of shear stress produce by the NuVision Engineering , ID 49887, and PS_ID 60480 nozzle design.

Table 18 Comparison of calculated performances of the commercially available nozzle and two nozzles designed using different optimization algorithms.

	Commercially available design (NuVision Eng.)	Best design obtained using genetic algorithm	Best design obtained using particle swarm algorithm
Maximum Shear Stress (Pa)	100%	111.88%	123.00%
Average Shear Stress (Pa)	100%	106.68%	118.00%

CHAPTER V

CONCLUSIONS AND FUTURE WORK

The optimum design has been identified and its design parameters are listed in Table 14. This research examines the evolution of 112 designs. The best parameters listed in Table 14 were derived through the optimization process of these 112 designs. The steps that were followed are outlined below:

1. Perform ANSYS FLUENT runs at Sobol points.
2. Compile a table of real data.
3. Create a response surface ONLY real data.
4. Run the optimization.
5. Pick a few optimized virtual designs and verify manually by plugging in the same input parameters into ANSYS FLUENT and running the simulation.
6. Compare the optimized virtual points with the real ANSYS FLUENT runs to determine if the error is acceptable.

If the errors are not acceptable after the verification:

7. Perform more real ANSYS FLUENT runs at new Sobol points.
8. Add the new real data to the table created in Step 2, that has ONLY real data.
9. Create a new response surface with the updated table that has ONLY real data.
10. Run a new optimization.

11. Pick a few optimized virtual points and verify manually by plugging in the same input parameters into ANSYS FLUENT and running the simulation.
12. Compare the optimized virtual points with the real ANSYS FLUENT runs to determine if error is acceptable.

If error is unsatisfactory, repeat steps 7-12.

Improvement of the response surface error cannot occur without adding more real data. To refine areas of interest, i.e. region of optimum points, real data must be added to regions where there is a concentration of optimized points. In the research one step of refinement was conducted. Eight additional designs were added to the original 100 designs. The percentage errors were minimal. It was of interest to examine the area of the response surface where the optimum designs were clustered. Refinement of this region increases the accuracy of the resulting maximum shear stress value and highest value of average shear stress. Four more designs were added to achieve this.

Thus, after implementing 112 designs of experiment to create the response surface, discrepancies between the real design values for maximum shear stress and average shear stress and virtual design were present.

Increasing the real initial designs from 100 to 150 may reduce the error between real and virtual designs, thus, improving the response surface. Future work should include refining the response surface globally and locally in the region were optimum designs are clustered.

Another approach to refine the response surface would be to add real designs after every 50th generation. The response surface is created after the 50th generation. Then real designs are included to refine the response surface. A second response surface is created after the 100th generation. Additional real designs further refine the subsequent response surfaces. This process is continued until the 500th generation is derived. This process should yield an improvement in the accuracy of the response surface.

Unique nozzle designs which could be investigated include those having three and four orifices. The three orifices on the nozzle are aligned at 90°, 210º, and 320°. Another arrangement which can be investigated is a nozzle with orifices aligned at 0º, 90º, 180º, and 270º. The addition of swirl effects to the three and four-orifice nozzles is also of interest. Furthermore, the effects of three and four-orifice nozzles with opposing velocities should be considered. The aforementioned topics should be addressed in future work.

REFERENCES

1. NuVision Engineering. (2009). *Demonstration of Power Fluidic Mixing Technology to Enhance Chemical Cleaning Operations* in *High Level Waste Tanks*. Mooresville, North Carolina

2. McKeel, C.A. (2008). Type I Tank Liner Integrity under oxalic acid induced corrosion. T-CLC-F-00383.

3. Wiersma, B.J. (2010). Alternative and Enhanced Chemical Cleaning: Corrosion Studies Results FY2010. Savannah River National Laboratory, SRNL-STI-2010-00555.

4. Hay, M.S., and Koopman, D.C. (2009). Review of Alternative Enhanced Chemical Cleaning Options for SRS Waste Tanks. Savannah River National Laboratory, SRNLSTI-2009-00500.

5. McKendrick, D., Biggs, S.R., Fairweather, M., & Rhodes, D. (2008). Physical Modeling of Axisymmetric Turbulent Impinging Jets used within the Nuclear Industry for Mobilization of Sludge.

6. Patwardhan, A., & Gaikwad, S. (2003). Mixing in tanks agitated by jets. *Institution of Chemical Engineers* , 211-220.

7. Forster, D., Nelson, D.R., & Stephen, M.J. (1977). Large distance and long-time properties of a randomly stirred fluid. *Physical Review,* A-16, 732-749.

8. Dannevik, W.P., Yakhot, V. & SA Orszag. (1987). Analytical theories of turbulence and the epsilon-expansion. *Physical Fluids*, 30, 2010-2029

9. Paravareh, A., Rahimi, M., Yarmohammadi, M., & Alsairafi, A. (2009). Experimental and CFD study on the effect of jet position on reactant dispersion performance. *International Communications in Heat and Mass Transfer*, 1096-1102.

10. Ranade, V. (1996). Towards Better Mixing Protocols by Designing Spatially Periodic Flows: The case of a Jet Mixer. *Chemical Engineering Science*, 2637-2642.

11. Yakhot, V. & Orszag. SA,. (1986). Renormalization group analysis of turbulence: I. Basic theory. *Journal of Scientific Computing*, Vol 1 No 1, 3-51.

12. Forster, D., Nelson, D.R., & Stephen, M.J. (1977). Large distance and long-time properties of a randomly stirred fluid. *Physical Review,* A-16, 732-749.

13. Dannevik, W.P., Yakhot, V. & SA Orszag. (1987). Analytical theories of turbulence and the epsilon-expansion. *Physical Fluids*, 30, 2010-2020.

14. Tan, J. (2010). *A study of solving Navier-Stokes equations with a finite volume method based on polygonal unstructured grids and the computational analysis of ground vehicle aerodynamics*. (Doctoral dissertation).

15. ANSYS FLUENT Theory Guide 12.0

16. Koziel, S., & Yang, X. (2011). Computational Optimization, Methods and Algorithms. Belin, Springer-Velag.

17. Colaco, M., Orlande, Helcio R.B., &, Dulikravich G.S., (2006). Inverse and Optimization Problems in Heat Transfers. *Journal of the Brazililian Society of Mechanic Science & Engineering* , 28, 1-15.

18. Naka, S., Yura, T.G. & Fukuyama, Y. (2001). Practical Distribution State Estimations Using Hybrid Particle Swarm Optimization. *Proceedings IEEE Power Engineering Society, Winter Meeting*, 815-820.

19. Kennedy, J. & Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of the 1995 IEEE International Conference on Neural Networks,* 1942-1948.

20. Jin, R., Chen, W., & Simpson, T. (2000). Comparative Studies of Metamodeling Techniques under Multiple Modeling Criteria. *American Institute of Aeronautics and Astronautics,* 1-11.

21. Jameson, A. & L. Martinelli. (1998). Mesh Refinement and Modeling Errors in Flow Simulations*. AIAA Journal*, 36, 676-686.

22. Roache, P.J. (1998). *Verification and Validation in Computational Science and Engineering.* Albuquerque, NM: Hermosa Publishers.

23. Logan, R. & Nitta, C. (2005). Comparing 10 methods for Solution Verification and Linking to Model Validation.

APPENDICES

Appendix A: Use of Another Algorithm for the Response Surface Generation

The following data show how the use response surface generation algorithm has an effect on the accuracy and reliability of optimization results. The values present in Table 18 are gain by the use of the kriging algorithm in modeFrontier optimization software to create the response surface. Large errors exist between the values for the real designs and virtual designs.

Table 17 Use of the Kriging for the response surface generation

	modeFrontier		**ANSYS FLUENT**		**Error %**	
ID	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)	Maximum Shear Stress	Average Shear Stress
65827	2573	61	2233	45	15	35
65856	2147	46	2494	48	14	3.3
66064	2871	46	3039	52	5.5	11
66075	3132	48	2540	48	23.3	0.52
66144	2529	63	2308	45	9.6	39
66185	2462	66	2198	45	12	48
66240	2795	58	2604	58	7.3	0.69
66370	2566	61	2718	50	5.6	23

Appendix B: Tables of Values

Test Case	Angle (deg.) Belowthe Horizon	Left Radius (cm)	Right Radius (cm)	Nozzle Height (cm)	Distance Between Radii (cm)	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)
1	40.39	0.6	0.4	16.5	0.7	1214	23
3	47.89	0.7	0.6	15.2	0.9	1364	19
5	51.64	0.9	0.5	14.6	1.3	1395	223
6	36.64	0.6	0.8	17.1	1.0	453	19
7	44.14	0.5	0.6	15.8	1.2	674	15
8	29.14	0.8	0.9	13.3	0.8	706	25
9	44.20	0.6	0.5	14.1	1.1	1424	35
10	29.20	0.9	0.9	16.6	0.7	954	44
11	51.70	0.7	0.7	15.4	1.2	2697	44
12	36.70	0.4	1.0	12.8	0.9	2530	43
13	47.95	1.0	0.5	16.0	1.0	1923	43
14	32.95	0.6	0.8	13.5	1.3	1414	49
15	40.45	0.5	0.6	14.7	0.8	1133	33
16	25.45	0.8	0.9	17.3	1.2	2530	43
17	53.58	0.5	0.7	16.9	1.2	1511	37
18	38.58	0.8	0.4	14.4	0.9	1244	37
19	46.08	0.9	0.9	13.1	1.4	2747	57
20	31.08	0.6	0.6	15.7	1.0	810	34
21	42.33	0.8	0.8	13.8	0.9	2139	47
22	27.33	0.5	0.5	16.3	1.3	697	38
23	49.83	0.7	1.0	17.6	0.8	1959	42
24	34.83	1.0	0.7	15.0	1.1	1661	56
25	50.76	0.7	0.6	13.0	1.3	2117	44
26	35.76	0.4	0.9	15.5	0.9	1185	41

27	43.26	0.6	0.4	16.8	1.1	1066	33
28	28.26	0.9	0.8	14.2	0.7	1050	45
29	47.01	0.5	0.7	17.4	0.8	1259	33
30	32.01	0.8	1.0	14.9	1.2	1419	55
31	54.51	1.0	0.5	13.6	1.0	2899	45
32	39.51	0.7	0.8	16.2	1.3	1646	48
33	41.39	0.9	1.0	15.8	1.3	2026	56
34	26.39	0.5	0.6	13.3	1.0	862	39
35	48.89	0.4	0.8	14.6	1.1	1894	39
36	33.89	0.7	0.5	17.1	0.8	662	39
37	52.64	0.6	0.9	15.2	0.7	2078	38
38	37.64	0.9	0.6	17.7	1.0	1102	41
39	45.14	0.8	0.7	16.5	0.9	1686	41
40	30.14	0.5	0.4	13.9	1.2	772	36
41	52.17	0.9	0.8	17.5	0.8	1891	58
42	37.17	0.6	0.5	15.0	1.2	1117	39
43	44.67	0.4	1.0	13.7	1.0	2208	39
44	29.67	0.8	0.6	16.2	1.3	1082	58
45	40.92	0.7	0.7	13.1	1.2	2111	47
46	25.92	1.0	0.4	15.6	0.9	787	42
47	48.42	0.8	0.9	16.9	1.1	2137	51
48	33.42	0.5	0.6	14.3	0.7	932	35
49	46.55	0.7	0.5	14.7	0.8	1488	35
50	31.55	0.4	0.8	17.2	1.1	763	38
51	54.05	0.6	0.6	15.9	0.9	1702	37
52	39.05	0.9	0.9	13.4	1.3	2427	58
53	50.30	0.5	0.5	15.3	1.4	1560	36
54	35.30	0.8	0.8	12.8	1.0	1836	51
55	42.80	0.9	0.7	14.0	1.2	2146	51

56	27.80	0.6	1.0	16.6	0.9	909	47
57	43.73	0.4	0.9	16.4	1.0	1518	39
58	28.73	0.7	0.6	13.8	1.4	1166	46
59	51.23	0.9	0.7	15.1	0.8	2077	42
60	36.23	0.6	0.4	17.7	1.2	743	34
61	54.98	0.8	0.9	14.5	1.1	2876	51
62	39.98	0.5	0.6	17.0	0.8	930	32
63	47.48	0.7	0.8	15.7	1.3	2088	45
64	32.48	1.0	0.5	13.2	0.9	1341	45
65	49.36	0.6	0.7	13.5	0.9	1801	37
66	34.36	0.9	1.0	16.1	1.2	1668	58
67	41.86	0.8	0.5	17.3	0.7	1081	34
68	26.86	0.5	0.8	14.8	1.1	774	44
69	45.61	1.0	0.6	16.7	1.1	1363	39
70	30.61	0.7	0.9	14.2	0.8	1109	44
71	53.11	0.5	0.4	12.9	1.3	2251	35
72	38.11	0.9	0.7	15.4	1.0	870	36
73	48.19	0.5	1.0	14.9	0.9	2291	73
74	33.19	0.8	0.7	17.5	1.3	1127	46
75	40.69	1.0	0.8	16.2	0.8	1704	47
76	25.69	0.7	0.5	13.7	1.1	813	41
77	44.44	0.8	0.9	15.6	1.2	2265.	51
78	29.44	0.4	0.6	13.0	0.8	1229	35
79	51.94	0.6	0.7	14.3	1.4	2265	45
80	36.94	0.9	0.4	16.8	1.0	1063	40
81	42.56	0.6	0.5	17.1	1.0	1135	35
82	27.56	1.0	0.9	14.6	1.3	1417	60
83	50.06	0.8	0.4	13.3	0.8	1996	34
84	35.06	0.5	0.7	15.9	1.1	1022	39

85	53.81	0.9	0.6	12.7	1.1	2826	45
86	38.81	0.6	0.9	15.2	0.7	1428	40
87	46.31	0.4	0.5	16.5	1.2	1045	31
88	31.31	0.7	0.8	14.0	0.9	1214	47
89	47.25	0.9	0.8	13.8	0.7	2564	45
90	32.25	0.6	0.4	16.4	1.1	747	36
91	54.75	0.5	0.9	17.6	0.9	1833	38
92	39.75	0.8	0.6	15.1	1.3	1656	45
93	51.00	0.5	0.8	17.0	1.3	1827	44
94	36.00	0.9	0.5	14.5	1.0	1347	43
95	43.50	0.7	1.0	13.2	1.2	2428	53
96	28.50	0.4	0.7	15.7	0.8	605	36
97	52.87	0.8	0.5	16.0	0.9	1813	35
98	37.87	0.5	0.8	13.5	1.2	1734	30
99	45.37	0.7	0.6	14.8	1.0	1996	52
100	30.37	0.7	1.0	17.3	1.4	1108	51

***DATA ENTER AFTER 100 ORIGINAL DOE, TO REFINE RSM**

Test Case	Angle(deg.) Below the Horizon	Left Radius (cm)	Right Radius (cm)	Nozzle Height (cm)	Distance Between Radii (cm)	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)
102	29.47	0.4	0.9	14.7	0.7	782	39
103	37.20	0.8	0.5	15.4	0.7	1039	36
104	48.51	0.8	0.6	17.3	0.8	1595	37
105	50.85	0.6	0.5	16.4	1.4	1688	37
106	39.60	0.6	0.7	16.8	0.9	1130	37
107	30.23	0.4	1.0	14.9	1.1	1741	45
108	27.42	0.8	0.9	16.3	0.8	1008	47
109	38.67	0.6	0.7	12.8	0.9	1715	39
110	48.04	1.0	0.5	14.7	1.3	2532	50

****PARETO DESIGN INCORPORATED INTO RESPONSE SURFACE**

Test Case	Angle(deg.) Below the Horizon	Left Radius (cm)	Right Radius (cm)	Nozzle Height (cm)	Distance Between Radii (cm)	Maximum Shear Stress (Pa)	Average Shear Stress (Pa)
284	28.60	0.8	0.7	16.2	1.3	1122	55
4682	54.92	0.9	0.9	12.7	1.1	3040	52
4721	49.93	0.9	1.0	14.8	1.3	2093	40
6897	52.15	0.9	1.0	14.6	1.0	2645	52
4109	49.63	0.5	1.0	14.9	0.9	2372	69

Appendix C: Sobol's Algorithm Code Used to Create a Matrix of Random Numbers

```
#include <stdlib.h>
#include <iostream.h>
#include <time.h>

#include "sobol.h"

//****************************************************************************

int bit_hi1_base_2 ( int n )

//****************************************************************************
//
//  Purpose:
//
//    BIT_HI1_BASE_2 returns the position of the high 1 bit base 2 in an integer.
//
//  Example:
//
//       N    Binary    Hi 1
//    ----    --------  ----
//       0           0     0
//       1           1     1
//       2          10     2
//       3          11     2
//       4         100     3
//       5         101     3
//       6         110     3
//       7         111     3
//       8        1000     4
//       9        1001     4
//      10        1010     4
//      11        1011     4
//      12        1100     4
//      13        1101     4
//      14        1110     4
//      15        1111     4
```

```
//      16       10000     5
//      17       10001     5
//    1023  1111111111    10
//    1024 10000000000    11
//    1025 10000000001    11
//
//  Modified:
//
//    13 March 2003
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, int N, the integer to be measured.
//    N should be nonnegative.  If N is nonpositive, BIT_HI1_BASE_2
//    will always be 0.
//
//    Output, int BIT_HI1_BASE_2, the number of bits base 2.
//
{
  int bit;

  bit = 0;

  while ( 0 < n )
  {
    bit = bit + 1;
    n = n / 2;
  }

  return bit;
}
//******************************************************************************

int bit_lo0_base_2 ( int n )

//******************************************************************************
//
//  Purpose:
//
//    BIT_LO0_BASE_2 returns the position of the low 0 bit base 2 in an integer.
//
```

```
//  Example:
//
//       N    Binary    Lo 0
//    ----    --------  ----
//       0           0     1
//       1           1     2
//       2          10     1
//       3          11     3
//       4         100     1
//       5         101     2
//       6         110     1
//       7         111     4
//       8        1000     1
//       9        1001     2
//      10        1010     1
//      11        1011     3
//      12        1100     1
//      13        1101     2
//      14        1110     1
//      15        1111     5
//      16       10000     1
//      17       10001     2
//    1023  1111111111     1
//    1024 10000000000     1
//    1025 10000000001     1
//
//  Modified:
//
//    13 March 2003
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, int N, the integer to be measured.
//    N should be nonnegative.
//
//    Output, int BIT_LO0_BASE_2, the position of the low 1 bit.
//
{
  int bit;
  int n2;
```

```
  bit = 0;

  while ( true )
  {
    bit = bit + 1;
    n2 = n / 2;

    if ( n == 2 * n2 )
    {
      break;
    }

    n = n2;

  }

  return bit;
}
//****************************************************************************

void sobol ( int dim_num, int *seed, double quasi[ ] )

//****************************************************************************
//
//  Purpose:
//
//    SOBOL generates a new quasirandom Sobol vector with each call.
//
//  Discussion:
//
//    The routine adapts the ideas of Antonov and Saleev.
//
//  Reference:
//
//    Antonov and Saleev,
//    USSR Computational Mathematics and Mathematical Physics,
//    Volume 19, 1980, pages 252 - 256.
//
//    Paul Bratley and Bennett Fox,
//    Algorithm 659:
//    Implementing Sobol's Quasirandom Sequence Generator,
//    ACM Transactions on Mathematical Software,
//    Volume 14, Number 1, pages 88-100, 1988.
//
//    Bennett Fox,
```

```
//    Algorithm 647:
//    Implementation and Relative Efficiency of Quasirandom
//    Sequence Generators,
//    ACM Transactions on Mathematical Software,
//    Volume 12, Number 4, pages 362-376, 1986.
//
//    I Sobol,
//    USSR Computational Mathematics and Mathematical Physics,
//    Volume 16, pages 236-242, 1977.
//
//    I Sobol and Levitan,
//    The Production of Points Uniformly Distributed in a Multidimensional
//    Cube (in Russian),
//    Preprint IPM Akad. Nauk SSSR,
//    Number 40, Moscow 1976.
//
//  Parameters:
//
//    Input, int DIM_NUM, the number of spatial dimensions.
//    DIM_NUM must satisfy 2 <= DIM_NUM <= 40.
//
//    Input/output, int *SEED, the "seed" for the sequence.
//    This is essentially the index in the sequence of the quasirandom
//    value to be generated.  On output, SEED has been set to the
//    appropriate next value, usually simply SEED+1.
//    If SEED is less than 0 on input, it is treated as though it were 0.
//    An input value of 0 requests the first (0-th) element of the sequence.
//
//    Output, double QUASI(DIM_NUM), the next quasirandom vector.
//
{
# define DIM_MAX 40

  static int atmost = 1073741823;
  static int dim_num_save = 0;
  int i;
//  int i2;
  bool includ[8];
  static bool initialized = false;
  int j;
  int j2;
  int k;
  int l;
  static int lastq[DIM_MAX];
  int m;
```

```
  static int maxcol;
  int newv;
  static int poly[DIM_MAX] =
  {
        1,   3,   7,  11,  13,  19,  25,  37,  59,  47,
       61,  55,  41,  67,  97,  91, 109, 103, 115, 131,
      193, 137, 145, 143, 241, 157, 185, 167, 229, 171,
      213, 191, 253, 203, 211, 239, 247, 285, 369, 299
  };
  static double recipd;
  static int seed_save = 0;
  int seed_temp;
  static int v[DIM_MAX][30];
//
  if ( !initialized || dim_num != dim_num_save )
  {
    initialized = true;
//
//  Initialize (part of) V.
//
    v[ 0][0] = 1;
    v[ 1][0] = 1;
    v[ 2][0] = 1;
    v[ 3][0] = 1;
    v[ 4][0] = 1;
    v[ 5][0] = 1;
    v[ 6][0] = 1;
    v[ 7][0] = 1;
    v[ 8][0] = 1;
    v[ 9][0] = 1;
    v[10][0] = 1;
    v[11][0] = 1;
    v[12][0] = 1;
    v[13][0] = 1;
    v[14][0] = 1;
    v[15][0] = 1;
    v[16][0] = 1;
    v[17][0] = 1;
    v[18][0] = 1;
    v[19][0] = 1;
    v[20][0] = 1;
    v[21][0] = 1;
    v[22][0] = 1;
    v[23][0] = 1;
    v[24][0] = 1;
```

```
v[25][0] = 1;
v[26][0] = 1;
v[27][0] = 1;
v[28][0] = 1;
v[29][0] = 1;
v[30][0] = 1;
v[31][0] = 1;
v[32][0] = 1;
v[33][0] = 1;
v[34][0] = 1;
v[35][0] = 1;
v[36][0] = 1;
v[37][0] = 1;
v[38][0] = 1;
v[39][0] = 1;

v[ 2][1] = 1;
v[ 3][1] = 3;
v[ 4][1] = 1;
v[ 5][1] = 3;
v[ 6][1] = 1;
v[ 7][1] = 3;
v[ 8][1] = 3;
v[ 9][1] = 1;
v[10][1] = 3;
v[11][1] = 1;
v[12][1] = 3;
v[13][1] = 1;
v[14][1] = 3;
v[15][1] = 1;
v[16][1] = 1;
v[17][1] = 3;
v[18][1] = 1;
v[19][1] = 3;
v[20][1] = 1;
v[21][1] = 3;
v[22][1] = 1;
v[23][1] = 3;
v[24][1] = 3;
v[25][1] = 1;
v[26][1] = 3;
v[27][1] = 1;
v[28][1] = 3;
v[29][1] = 1;
v[30][1] = 3;
```

```
v[31][1] = 1;
v[32][1] = 1;
v[33][1] = 3;
v[34][1] = 1;
v[35][1] = 3;
v[36][1] = 1;
v[37][1] = 3;
v[38][1] = 1;
v[39][1] = 3;

v[ 3][2] = 7;
v[ 4][2] = 5;
v[ 5][2] = 1;
v[ 6][2] = 3;
v[ 7][2] = 3;
v[ 8][2] = 7;
v[ 9][2] = 5;
v[10][2] = 5;
v[11][2] = 7;
v[12][2] = 7;
v[13][2] = 1;
v[14][2] = 3;
v[15][2] = 3;
v[16][2] = 7;
v[17][2] = 5;
v[18][2] = 1;
v[19][2] = 1;
v[20][2] = 5;
v[21][2] = 3;
v[22][2] = 3;
v[23][2] = 1;
v[24][2] = 7;
v[25][2] = 5;
v[26][2] = 1;
v[27][2] = 3;
v[28][2] = 3;
v[29][2] = 7;
v[30][2] = 5;
v[31][2] = 1;
v[32][2] = 1;
v[33][2] = 5;
v[34][2] = 7;
v[35][2] = 7;
v[36][2] = 5;
v[37][2] = 1;
```

```
v[38][2] = 3;
v[39][2] = 3;

v[ 5][3] =  1;
v[ 6][3] =  7;
v[ 7][3] =  9;
v[ 8][3] = 13;
v[ 9][3] = 11;
v[10][3] =  1;
v[11][3] =  3;
v[12][3] =  7;
v[13][3] =  9;
v[14][3] =  5;
v[15][3] = 13;
v[16][3] = 13;
v[17][3] = 11;
v[18][3] =  3;
v[19][3] = 15;
v[20][3] =  5;
v[21][3] =  3;
v[22][3] = 15;
v[23][3] =  7;
v[24][3] =  9;
v[25][3] = 13;
v[26][3] =  9;
v[27][3] =  1;
v[28][3] = 11;
v[29][3] =  7;
v[30][3] =  5;
v[31][3] = 15;
v[32][3] =  1;
v[33][3] = 15;
v[34][3] = 11;
v[35][3] =  5;
v[36][3] =  3;
v[37][3] =  1;
v[38][3] =  7;
v[39][3] =  9;

v[ 7][4] =  9;
v[ 8][4] =  3;
v[ 9][4] = 27;
v[10][4] = 15;
v[11][4] = 29;
v[12][4] = 21;
```

```
v[13][4] = 23;
v[14][4] = 19;
v[15][4] = 11;
v[16][4] = 25;
v[17][4] =  7;
v[18][4] = 13;
v[19][4] = 17;
v[20][4] =  1;
v[21][4] = 25;
v[22][4] = 29;
v[23][4] =  3;
v[24][4] = 31;
v[25][4] = 11;
v[26][4] =  5;
v[27][4] = 23;
v[28][4] = 27;
v[29][4] = 19;
v[30][4] = 21;
v[31][4] =  5;
v[32][4] =  1;
v[33][4] = 17;
v[34][4] = 13;
v[35][4] =  7;
v[36][4] = 15;
v[37][4] =  9;
v[38][4] = 31;
v[39][4] =  9;

v[13][5] = 37;
v[14][5] = 33;
v[15][5] =  7;
v[16][5] =  5;
v[17][5] = 11;
v[18][5] = 39;
v[19][5] = 63;
v[20][5] = 27;
v[21][5] = 17;
v[22][5] = 15;
v[23][5] = 23;
v[24][5] = 29;
v[25][5] =  3;
v[26][5] = 21;
v[27][5] = 13;
v[28][5] = 31;
v[29][5] = 25;
```

```
    v[30][5] =  9;
    v[31][5] = 49;
    v[32][5] = 33;
    v[33][5] = 19;
    v[34][5] = 29;
    v[35][5] = 11;
    v[36][5] = 19;
    v[37][5] = 27;
    v[38][5] = 15;
    v[39][5] = 25;

    v[19][6] =  13;
    v[20][6] =  35;
    v[21][6] = 115;
    v[22][6] =  41;
    v[23][6] =  79;
    v[24][6] =  17;
    v[25][6] =  29;
    v[26][6] = 119;
    v[27][6] =  75;
    v[28][6] =  73;
    v[29][6] = 105;
    v[30][6] =   7;
    v[31][6] =  59;
    v[32][6] =  65;
    v[33][6] =  21;
    v[34][6] =   3;
    v[35][6] = 113;
    v[36][6] =  61;
    v[37][6] =  89;
    v[38][6] =  45;
    v[39][6] = 107;

    v[37][7] =  7;
    v[38][7] = 23;
    v[39][7] = 39;
//
//  Check parameters.
//
    if ( dim_num < 2 || DIM_MAX < dim_num )
    {
      cout << "\n";
      cout << "SOBOL - Fatal error!\n";
      cout << "  The spatial dimension DIM_NUM should satisfy:\n";
      cout << "    2 <= DIM_NUM <= " << DIM_MAX << "\n";
```

```
     cout << "  But this input value is DIM_NUM = " << dim_num << "\n";
     exit ( 1 );
   }

   dim_num_save = dim_num;
//
//  Find the number of bits in ATMOST.
//
   maxcol = bit_hi1_base_2 ( atmost );
//
//  Initialize row 1 of V.
//
   for ( j = 1; j <= maxcol; j++ )
   {
     v[1-1][j-1] = 1;
   }
//
//  Initialize the remaining rows of V.
//
   for ( i = 1; i < dim_num; i++ )
   {
//
//  The bit pattern of the integer POLY(I) gives the form
//  of polynomial I.
//
//  Find the degree of polynomial I from binary encoding.
//
     j = poly[i];
     m = 0;

     while ( true )
     {
       j = j / 2;
       if ( j <= 0 )
       {
         break;
       }
       m = m + 1;
     }
//
//  We expand this bit pattern to separate components
//  of the logical array INCLUD.
//
     j = poly[i];
     for ( k = m-1; k >= 0; k-- )
```

```
      {
        j2 = j / 2;
        includ[k] = ( j != ( 2 * j2 ) );
        j = j2;
      }
//
//  Calculate the remaining elements of row I as explained
//  in Bratley and Fox, section 2.
//
//  Some tricky indexing here.  Did I change it correctly?
//
      for ( j = m; j < maxcol; j++ )
      {
        newv = v[i][j-m];
        l = 1;

        for ( k = 0; k < m; k++ )
        {
          l = 2 * l;

          if ( includ[k] )
          {
            newv = ( newv ^ ( l * v[i][j-k-1] ) );
          }

        }

        v[i][j] = newv;

      }

    }
//
//  Multiply columns of V by appropriate power of 2.
//
    l = 1;
    for ( j = maxcol-2; j >= 0; j-- )
    {
      l = 2 * l;
      for ( i = 0; i < dim_num; i++ )
      {
        v[i][j] = v[i][j] * l;
      }
    }
//
```

```
//  RECIPD is 1/(common denominator of the elements in V).
//
    recipd = 1.0E+00 / ( ( double ) ( 2 * l ) );
  }

  if ( *seed < 0 )
  {
    *seed = 0;
  }

  if ( *seed == 0 )
  {
    l = 1;
    for ( i = 0; i < dim_num; i++ )
    {
      lastq[i] = 0;
    }
  }
  else if ( *seed == seed_save + 1 )
  {
    l = bit_lo0_base_2 ( *seed );
  }
  else if ( *seed <= seed_save )
  {
    seed_save = 0;
    l = 1;
    for ( i = 0; i < dim_num; i++ )
    {
      lastq[i] = 0;
    }

    for ( seed_temp = seed_save; seed_temp <= (*seed)-1; seed_temp++ )
    {

      l = bit_lo0_base_2 ( seed_temp );

      for ( i = 0; i < dim_num; i++ )
      {
        lastq[i] = ( lastq[i] ^ v[i][l-1] );
      }

    }

    l = bit_lo0_base_2 ( *seed );
  }
```

```
  else if ( seed_save+1 < *seed )
  {
    for ( seed_temp = seed_save+1; seed_temp <= (*seed)-1; seed_temp++ )
    {

      l = bit_lo0_base_2 ( seed_temp );

      for ( i = 0; i < dim_num; i++ )
      {
        lastq[i] = ( lastq[i] ^ v[i][l-1] );
      }

    }

    l = bit_lo0_base_2 ( *seed );

  }
//
//  Check that the user is not calling too many times!
//
  if ( maxcol < l )
  {
    cout << "\n";
    cout << "SOBOL - Fatal error!\n";
    cout << "  Too many calls!\n";
    cout << "  MAXCOL = " << maxcol << "\n";
    cout << "  L =      " << l << "\n";
    exit ( 2 );
  }
//
//  Calculate the new components of QUASI.
//  The caret indicates the bitwise exclusive OR.
//
  for ( i = 0; i < dim_num; i++ )
  {
    quasi[i] = ( ( double ) lastq[i] ) * recipd;

    lastq[i] = ( lastq[i] ^ v[i][l-1] );
  }

  seed_save = *seed;
  *seed = *seed + 1;

  return;
# undef MAX_DIM
```

```
}
//****************************************************************************

void timestamp ( void )

//****************************************************************************
//
//  Purpose:
//
//    TIMESTAMP prints the current YMDHMS date as a time stamp.
//
//  Example:
//
//    May 31 2001 09:45:54 AM
//
//  Modified:
//
//    04 October 2003
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    None
//
{
#define TIME_SIZE 40

  static char time_buffer[TIME_SIZE];
  const struct tm *tm;
  size_t len;
  time_t now;

  now = time ( NULL );
  tm = localtime ( &now );

  len = strftime ( time_buffer, TIME_SIZE, "%d %B %Y %I:%M:%S %p", tm );

  cout << time_buffer << "\n";

  return;
#undef TIME_SIZE
```

```
}
```