2006

An interactive design environment for coal piping system

Gengxun Huang
Iowa State University

An interactive design environment for coal piping system

by

Gengxun Huang

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Mechanical Engineering

Iowa State University

Ames, Iowa

2006

Graduate College
Iowa State University

This is to certify that the doctoral dissertation of

Gengxun Huang

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

Table of Contents

List of Figures	vi
List of Tables	x
Abstract	xi
Chapter 1 Introduction	1
1.1 Problem Statement	1
1.2 Summary of the thesis	8
Chapter 2 Literature Review Part I: Gas-solid Flow	11
2.1 Understanding Behavior and Properties of Dilute Gas-solid Pipe Flow	12
2.1.1 Time Scales and Particle Turbulence Interaction	12
2.1.2 Forces on Particles	15
2.1.3 Wall-particle Interaction	16
2.1.4 Particle-particle Collision	17
2.2 Governing Equations for Gas-solid Flows	18
2.2.1 Governing Equations for the Carries Phase	18
2.2.2 Governing Equations for the Particle Phase	19
2.3 Numerical Models for Gas-solid Flows	19
2.3.1 Carrier Phase Models	20
2.3.2 Particle Dispersion Models	22
2.4 Conveying Characteristics of Pipeline Components	23
2.4.1 Horizontal Pipes	27
2.4.2 Vertical Pipes	27
2.4.3 Inclined Pipes	27
2.4.4 Bends	29
Chapter 3 Literature Review Part II: Optimum Engineering Design	33
3.1 Optimum Engineering Design	33
3.1.1 Design Procedure	33
3.1.2 Formulation of Direct Optimization Model	34
3.1.3 Traditional Engineering Design	36
3.1.4 Numerical Engineering Design	38
3.2 Optimization Algorithms	38
3.2.1 Classification	39
3.2.2 Evolutionary Algorithms: A Brief Introduction	41
3.3 Previous Works on CFD-based Shape Optimization	43
3.4 Interactive/Steering Function in the Design Process	49
3.5 Virtual Engineering	51

Chapter 4 Computational Model and Preliminary Results 57
4.1 Choosing an Analysis Package 57
4.2 Governing Equations and Turbulence Modeling 58
4.3 Validation 62
4.3.1 Comparison of CFD Results with Experimental Data 63
4.3.2 Comparison of Baseline Simulation with Plant Measurements 74
4.3.2.1 Bifurcator 75
4.3.2.2 Baseline Pipe validation 79
4.4 Baseline Simulation Results and Discussion 82

Chapter 5 Implementation of CFD Optimization Design System 92
5.1 Traditional CFD-based optimization process 92
5.2 Modified CFD-based Optimization Process 95
5.3 Software Integration 98
5.4 Performance 108
5.4.1 Geometry 108
5.4.2 Object Function 109
5.4.3 Results 111

Chapter 6 VE-Suite 116
6.1 Basic Requirements for an Interactive Design Environment 117
6.2 Why use VE-Suite 118
6.3 VE-Suite Structure 119
6.4 Introduction to the VE-Suite API 122
6.4.1 System Configuration 122
6.4.2 User Interface 123
6.4.3 Computational Unit 125
6.4.4 Graphical Plugin 126

Chapter 7 Implementation of Interactive Design Environment 132
7.1 System Structure 133
7.2 Implementation of General Interface for Problem Setup 134
7.2.1 EAs Control Parameters Setup 137
7.2.2 Design Vectors Setup 139
7.2.3 Analysis Solver Setup 141
7.2.4 Population Initialization 141
7.2.5 Fast Setup 142
7.3 Implementation of EAs Components into VE-Suite 144
7.4 Implementation of Exacting Information 150
7.4.1 Extract Basic EAs Information 151
7.4.2 Extract Product Performance Information 152
7.4.3 Extract Design Space Information 155
7.5 Performance 161
7.6 Conclusions 166

Chapter 8 Coal Piping System Design 168
8.1 Problem Analysis 168
8.2 Problem Setup 170
8.3 Convergence History 175
8.4 Comparison between Optimum and Baseline Designs 183
8.5 Conclusions 187

Chapter 9 Conclusions and Future Works 189
9.1 Conclusions 189
9.2 Future Works 192

References 193

Acknowledgements 203

List of Figures

Figure 1.1 Arrangement of coal piping system	2
Figure 1.2 "Riffle" type bifurcator	4
Figure 1.3 Example of "roping" flow around a bifurcator	4
Figure 1.4 Streamline in simple furnace with uniform inlet condition	5
Figure 1.5 Streamline in simple furnace with real inlet condition	5
Figure 2.1 Secondary flow patterns	31
Figure 3.1 Traditional engineering design process	37
Figure 3.2 Optimum design process	39
Figure 3.3 Basic flowchart of EAs	42
Figure 3.4 Gradual fitness improvement during the run	43
Figure 3.5 Framework of the virtual engineering system	55
Figure 4.1 Sketch of the testing flow geometry	64
Figure 4.2 Grid layout for the testing model	65
Figure 4.3 Final grid for pipe cross-section	67
Figure 4.4 Compared grid for pipe cross-section	67
Figure 4.5 Comparison of CFD results with two different grids	68-70
Figure 4.6 Comparison of CFD predictions and experimental data	71-73
Figure 4.7 Secondary flow pattern in two phase flow	74
Figure 4.8 Bifurcator structure	77
Figure 4.9 Fluid velocity (single phase calculation)	78-79
Figure 4.10 Schematic of the baseline pipe (top view)	80
Figure 4.11 Schematic of the baseline pipe (front view)	81
Figure 4.12 Grid layout on the cross-section of the baseline pipe	81
Figure 4.13 Comparison between erosion holes and simulation data	82
Figure 4.14 Particle distribution in the whole baseline model	85
Figure 4.15 Particle distribution along the pipe wall	86
Figure 4.16 Particle concentration on the cross-sections	87
Figure 4.17 Particle path near the distributor	88

Figure 4.18 Velocity contours of bifurcator 89
Figure 4.19 Particle distributions downstream of the bifurcator 90
Figure 4.20 Particle concentrations on the cross-sections 91
Figure 5.1 Traditional two-cycle optimization procedure 94
Figure 5.2 Fitness values with CFD iterations 96
Figure 5.3 Velocity contours 96
Figure 5.4 Modified two-cycle optimization procedure 97
Figure 5.5 Flow chart of the CFD-based optimization process 99
Figure 5.6 Main classes for initialization of the gene chain 105
Figure 5.7 Implementation of the case 106
Figure 5.8 CreatureManager Class 107
Figure 5.9 Sketch of testing pipe geometry 109
Figure 5.10 Fitness variations in the evolution processes 113
Figure 5.11 Coal distribution contours (a) baseline (b) best design 113
Figure 5.12 EA optimization results for orifice radius 114
Figure 5.13 EA optimization results for orifice angle 114
Figure 5.14 EA optimization results for orifice location 115
Figure 6.1 Architecture of VE-Suite 121
Figure 6.2 Sample definition file "example.def" 123
Figure 6.3 PluginBase class 125
Figure 6.4 Example of TransferDataFromWindow 125
Figure 6.5 Body_Unit_i class 128
Figure 6.6 Template for SetParams and StartCalc 129
Figure 6.7 VE_Xplorer GUI 130
Figure 6.8 Streamline representation 130
Figure 6.9 Contour representation 131
Figure 7.1 Diagram of the system architecture 134
Figure 7.2 User-defined parameters during design set up 136
Figure 7.3 Entrance frame 137
Figure 7.4 Main menu window of the design interface 137

Figure 7.5 EA control parameters setup dialog 139
Figure 7.6 Design parameters setup dialog 140
Figure 7.7 Active design parameters setup dialog 141
Figure 7.8 Example of initial population table 143
Figure 7.9 Example of setup file 144
Figure 7.10 A sequence diagram of the initial setup of the process 146
Figure 7.11 SetParams and StartCalc implementation 147
Figure 7.12 Computational unit scheme 149
Figure 7.13 Connections between classes 150
Figure 7.14 Example of result parameter 152
Figure 7.15 Example of a result plot 152
Figure 7.16 Connection between computation unit and VE-Xplorer 155
Figure 7.17 Visualization of SOM of Iris data 159
Figure 7.18 Davies-Boulding Index and SOM cluster 160
Figure 7.19 Example of labels in the system 161
Figure 7.20 Fitness variations in the evolution processes 163
Figure 7.21 Screen shot from virtual environment 164
Figure 7.22 EA optimization results for orifice radius 164
Figure 7.23 EA optimization results for orifice angle 165
Figure 7.24 EA optimization results for orifice location 165
Figure 8.1 Design variables 170
Figure 8.2 Sketch of the design pipe geometry 170
Figure 8.3 Constraints for design variables 171
Figure 8.4 A used bifurcator 174-175
Figure 8.5 U-matrix and Lables for first 32 cases 176
Figure 8.6 Davies-Boulding Index and SOM cluster for first 32 cases 177
Figure 8.7 Component planes for first 32 cases 177
Figure 8.8 U-matrix and Lables for 120 cases 178
Figure 8.9 Davies-Boulding Index and SOM cluster for 120 cases 178
Figure 8.10 Component planes for 120 cases 179

Figure 8.11 Correlations between design variables with fitness 179-182
Figure 8.12 Iteration history for fitness 183
Figure 8.13 Summary of design variables 184
Figure 8.14 Geometry of the optimal pipe 184
Figure 8.15 Particle distributions in the optimum pipe 185
Figure 8.16 Summary of design variables 185
Figure 8.17 Summary of design variables 186
Figure 8.18 Particle distributions 186
Figure 8.19 Contours of particle distribution 188

List of Tables

Table 2.1 Summary of experimental studies on gas-solid channel and pipe flows 25

Table 2.2 Summary of numerical studies on gas-solid channel and pipe flows 26

Table 4.1 Geometry sizes of the testing facility 64

Table 4.2 Particle size distribution of pulverized coal 66

Table 4.3 Particle Injection Properties 67

Table 4.4 Particle injection properties 75

Table 4.5 Flow parameters of the simulation 75

Table 4.6 Numerical grid resolution for bifurcator 78

Table 4.7 Comparison among four different grids 78

Table 5.1 Geometry sizes of the testing facility 108

Table 5.2 Comparison between traditional and modified design system 112

Table 7.1 EA parameters 138

Table 7.2 Example of user-defined file 140

Table 7.3 Other pre-defined parameters 140

Table 7.4 Parameters for analysis package 141

Table 7.5 Comparison between traditional design and proposed interactive design 166

Abstract

Optimization of coal pipe design is necessary to address the increasing demands of high performance and reduced emissions for pulverized coal-fired power plants. The design of coal piping is a complex and time-consuming engineering task that involves meeting of several design objectives and constraints. The distribution of coal particles in a pneumatic pipeline can be highly inhomogeneous, depending upon the pipeline geometry, phase loading, conveying air velocity and properties of the solid material including particle size, moisture content. A particularly difficult type of inhomogeneous distribution is the roping flow regime, in which most of the moving particles are concentrated into a small portion of the cross sectional area of the pipe which is mainly caused by bends in the pipe. Current coal piping design technology relies on empirical model and does not consider particle distribution characteristics in the pipe. In this thesis, a design tool which couples a validated detailed pipe model and an interactive optimization algorithm is developed. This new design tool uses evolutionary algorithms (EAs) as the optimization algorithm, and computational fluid dynamics (CFD) as the evaluation mechanism. The process uses an iterative approach that allows design to be evaluated using CFD analysis automatically to optimize several criteria. The proposed design change is then re-meshed and displayed. Three fundamentally different techniques from traditional optimization methods were considered in order to reduce computation time. Firstly, the tool has been implemented in a virtual engineering environment using VE-Suite. Secondly, the system is integrated with a general interface to allow users to set up the design procedure and interact or guide the searching path as the design evolves. Thirdly, a fast calculation approach is used to reduce the time for single CFD case. The proposed interactive design tool is analyzed and enhanced so that it is usable by the general engineering community. A real coal pipe application was carried out using this design tool. The main objective is to distribute coal flow to its two branches as uniform as possible. The results of this work suggested that the optimum coal pipe can be found relatively fast even when using high-fidelity CFD solver as the analysis method, and the optimum pipe can greatly reduce the coal flow unbalance. This indicates that the tool presented in this thesis can be used as a new and efficient design environment for coal pipe.

Chapter 1 Introduction

1.1 Problem statement

Conveying materials in a gas stream from their storage locations to the locations where they are needed is referred to as pneumatic conveying. It is characterized by ease of installation of the conveying pipes. The most suitable characterization of pneumatic conveying systems is based on the average particle concentration and transport velocity in the pipeline. There are two modes of conveying according to this definition [62]: dilute phase and dense phase conveying. In general, dilute phase pneumatic conveying systems employ large volumes of gas at high velocities and are most widely used in pneumatic conveying systems. The materials are carried in a gas stream as discrete particles by means of drag and lift forces acting on each individual particle. A dense phase flow, on the other hand, is one in which the particle motion is controlled by particle collisions.

The present study focuses on dilute phase pneumatic conveying usually applied in a power plant for the transport of pulverized coal with particle sizes ranging from $10\,\mu m$ to $200\,\mu m$. The air-to-solids mass loading ratios are typically in the range of one to three and the average air velocity is between 15 m/s and 30 m/s. As a result, the coal particles in the distribution network are normally very dilute with a typical volumetric concentration in the order of 1% or less.

In pulverized coal plant, raw coal is ground to fine powders in the milling system and conveyed by preheated air and fuel gases through the pipe network. Each mill supplies an entire elevation of nozzles. By distributing the fuel in this fashion, a balanced fire is maintained regardless of which mill is out of order. Usually, the mills are all located at the same elevation, but are often located at different corners of the furnace. Therefore, the paths followed by the individual pipes which include the length of pipes and the number of bends used are very different from each other. These pipes typically have a diameter that ranges from 0.3 to 0.5 meters; they are composed of both horizontal and vertical pipes connected by different bends, with a single pipe supplying a single burner. The main mixture of gas and fuel is bifurcated in two main streams to different burners. A typical example of the complexity of such a coal pipe system for a pulverized coal-fired plant is illustrated in Figure 1.1.

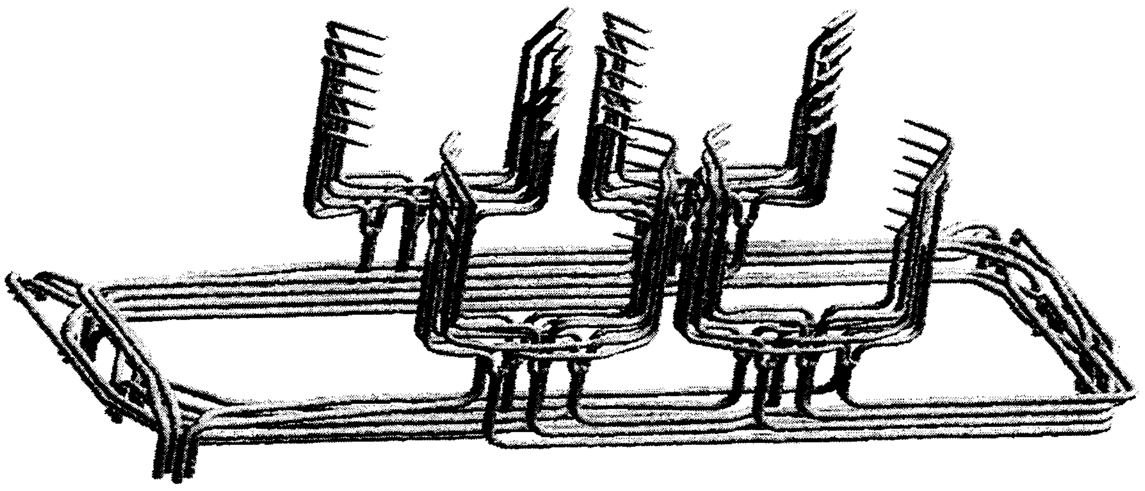


Figure 1.1 Arrangement of coal piping system of a coal-fired power plant

Today, coal-fired power plants are under increasing pressure to improve combustion performance and reduce emissions in order to minimize operation cost and meet the requirements of environmental regulations. The combustion efficiency of the burner is dependent on an even distribution of air and coal to the coal nozzles. A poor distribution of pulverized coal to the burner nozzles can adversely affect combustion, leading to premature failure of burners, local slagging, high levels of CO, and high unburned carbon in fly ash (loss of ignition). At the same time, unbalanced burners can make it necessary to operate with higher average O_2 levels in the furnace, limiting the NOx reduction that can be achieved through combustion optimization. Therefore, the coal transport piping system should be designed such as to ensure uniform rate and continuous feed of the pulverized fuel to the individual burners.

In a power plant, the paths of the individual coal pipes are complex with many bends, so uneven coal distribution is inevitable mainly because of the complex phenomena of "coal roping" (discussed in chapter 2). There are several options for balancing the coal flow from coal millers to individual burners. Devices known as the riffle type bifurcators (Figure 1.2) are commonly used to control the coal distribution of the pipe system. Schneider [9] has reported that without a coal rope in front of a distributor, the particle distribution in the inlet cross section of a distributor is uniform; the mass flow is divided relatively uniformly into both outlets. However, if coal roping occurs in front of the distributor, particle distribution is not uniform over the pipe cross-section and the coal will no longer be evenly divided between the two outlets. As a result, the fuel inputs to the burner are not balanced; combustion efficiency will be greatly decreased, and the emissions will increase. Figure 1.3 shows an example of "roping" flow around a bifurcator. Coal roping also affects the accuracy of traditional coal sampling probes that rely on a homogeneous flow. The failure of these probes to obtain accurate measurements adds difficulty to the flow control system. Therefore, despite the use of matched outlet pipes and flow splitting devices, uneven distribution of pulverized coal inevitably occurs.

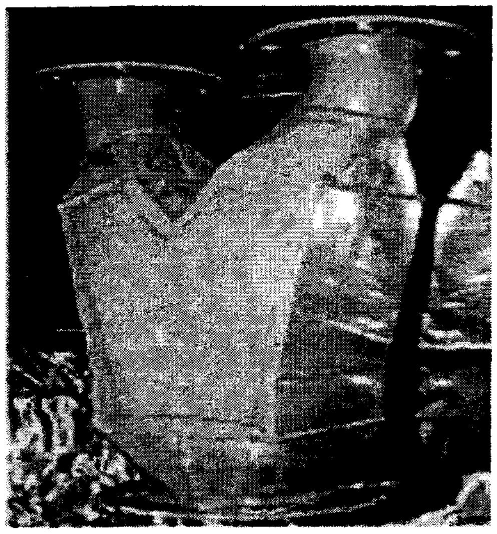


Figure 1.2 “Riffle” type bifurcator [2]

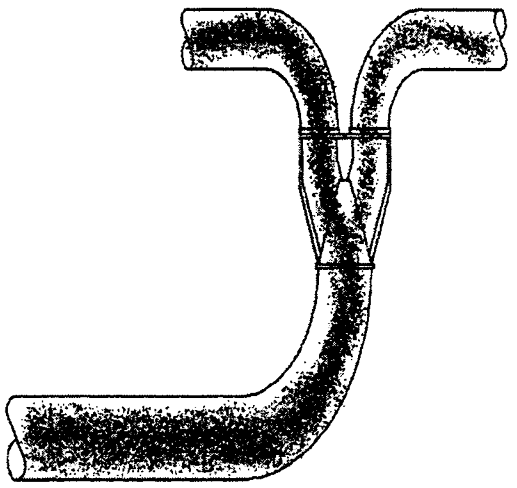


Figure 1.3 Example of “roping” flow around a bifurcator [68]

A great deal of research effort has been directed towards establishing more sophisticated online continuous measurement so that the subsequent controlling the air/coal flow rate in each pipe can be more accurate. However, there is one fundamental question that the power industry has forgotten to ask. Will or does better pipe geometry help to solve or minimize this problem? Since the uneven distribution situation is primarily caused by the complex coal distribution network. The answer to this question is “yes”. Then the next question is how to design such a network?

Current technology for designing coal piping systems is based on empirical methods which mainly focus on balancing the air flow. Empirical methods have been adequate when dealing with the size of the pipe, but they frequently result in sub-optimal designs for coal distribution. Figure 1.4 shows the coal flow streamlines in a simple furnace with the uniform inlet condition at four corners. Figure 1.5 shows the coal flow streamline in a simple furnace with two real coal pipes connecting to its four corners. As shown, when adding the real piping system to the simulation model, the unequal air distribution has a significant impact on the performance of the furnace. This indicates that an understanding of the cross-sectional distribution is essential to the coal transporting system design. Therefore, the traditional pipe design technique is not suitable for such a task.

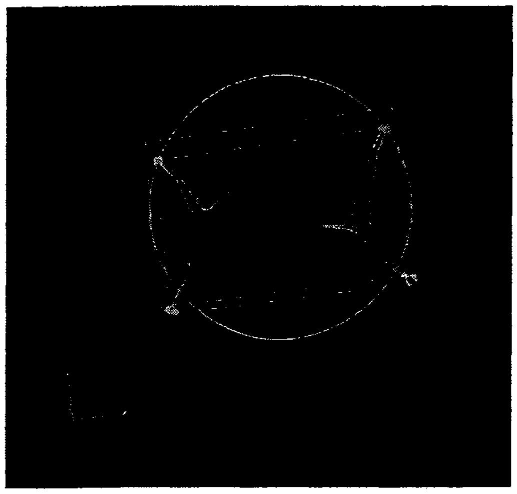


Figure1.4 Streamline in simple furnace with uniform inlet condition

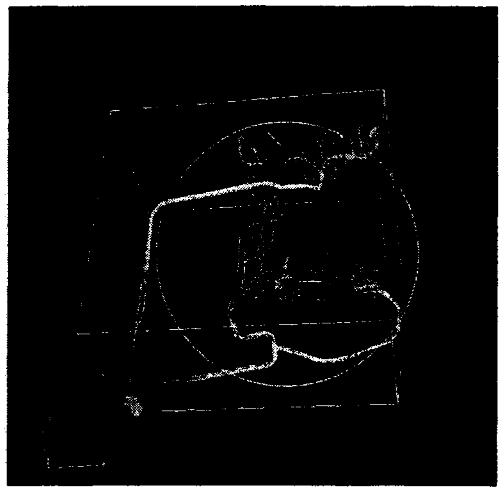


Figure 1.5 Streamline in simple furnace with real inlet condition

Due to the advances in computational fluid dynamics (CFD) and computer hardware, CFD is often used as a design tool for single phase thermal fluid systems. In contrast, CFD models of gas-solid flow are significantly less developed than single phase models. However, they can still provide a greater level of detailed and improved solutions compared to simple engineering relationships. For example, CFD simulation can provide useful information on coal flow non-uniformity in pipe cross-sections in a coal pipe network, thus allowing the impact of proposed piping changes to be studied. CFD simulation becomes especially useful when paired with numerical optimization methods. Thus, using high fidelity CFD simulation instead of experiments or simple empirical methods is a natural choice for the design and construction of efficient coal conveying systems.

The basic numerical design cycle when combined with CFD simulation is no different from the ordinary design cycle: fluid flow properties like pressure distribution, temperature, velocity, and heat flux are calculated for a baseline geometric configuration. The outcome of the CFD simulation is used in defining an objective function to be minimized or maximized. The objective function must relate geometric shape changes to comparable improvements in fluid flow characteristics of the optimum design. An optimization routine is then employed to automate evaluation of system solutions and to generate new system design. The system optimization continues until the optimization converges and an optimal system is obtained. However, applying numerical optimization with high fidelity CFD analysis is still a formidable challenge because of the following difficulties:

1) In fluid dynamics, the continuity equation, the Navier-Stokes equations, the energy equation, and the equation of state govern the flow behavior. Calculating the solution of this system of tightly coupled, nonlinear partial differential equations requires a large amount of computing power. Very often, because of the nonlinear coupled nature of the fundamental equations, the objective function landscapes are nonlinear with many local optima, plateaus, or ridges even in a simple case.

2) The nature of optimization design is iterative. Product design is a decision making process involving a lot of interactions between the designer and the designed product. Designers may change the design variables, constraints, or even design requirements and expect to see optimized performance of the designed product in these conditions. It is not uncommon for one function evaluation using a CFD analysis, especially a three-dimensional Navier-Stokes equation, to take weeks to finish. The long computing time

caused by the high fidelity models makes the aforementioned interactive design process very difficult.

Thus, a research question is raised: how can design optimization using high fidelity models, such as CFD models, be accomplished as quickly as possible, without sacrificing the accuracy of the analysis results.

The optimization design tool must include a geometry modification tool for the handling of three-dimensional designs, as well as mesh generation tools and a multiphase CFD solver. A user-centric interface is also needed. Currently, there is no known general searching method that is suitable for every kind of flow field, so the design tool must be made significantly generic to allow for a multitude of domain specific searching methods. As mentioned before, a serious concern of all applications in the area of coupling optimization design with CFD relates to the computational expense. In the case of a single phase shape optimization, it may take hours, days or even weeks to get the optimal solution using a complete CFD model. Generally, the CFD model for two-phase flow is much more complex than the single phase model. This is mainly due to the complex nature of the numerical models required to model two-phase flow. Most of the numerical models for two-phase flow require the calculation of the continuous flow field first. After this continuous flow field converges, the particle phase is injected into the flow field and the solution is iterated until the interaction between the two phases is resolved. Because of this process of quadratic iteration, the computational time required is significantly longer than the time required for a single phase flow calculation. For this reason, integrating a two-phase CFD solver, which is required for coal piping system design, into an optimization searching process becomes extremely time-consuming. Thus, an urgent and obvious need of reducing computational cost exists for a new design system.

1.2 Summary of the thesis

Yilmaz [4] points out that the flow pattern of the gas-particle flow is very sensitive to the pipe geometry such as the number of bends, the orientation of the pipe, and the orifice opening. In other words, the pipeline design has significant influence on the coal particle distribution. This thesis emphasizes how to improve design and development of complex engineering systems by employing CFD simulation and numerical optimization techniques. A special interest is how to perform shape design of power plant coal transport piping to improve coal particle distribution by utilizing the improved understanding of coal roping and new interactive simulation and optimization techniques. The goal of the proposed research can be described as:

1) Develop an understanding of the mechanisms by which coal roping arises;
2) Build a new, interactive engineering tool for design and modification of the coal transport piping;
3) Apply this understanding to the design of the coal transport system in an operational coal-fired plant;

A basic understanding of complex phenomena within coal piping system is discussed in Chapter 2. Chapter 3 focuses on the literature review of engineering design which includes shape optimization, evolutionary algorithms and interactive engineering design. A new concept called virtual engineering is also introduced in Chapter 3.

One of the first steps to use CFD modeling is to ensure that an acceptable computational model can be built. The process of building coal conveying pipelines includes choosing an analysis package, validating the results using known results, and extending the model to the case of interest. In Chapter 4, qualitative comparisons are made between the CFD simulation and the available data from literature.

In this thesis CFD simulation and optimization algorithms are integrated together into the system design process within a virtual engineering environment. Thus, it provides engineers with tools and methods which support systems engineering and analysis. Considering the nature of optimization design discussed above, there are three ways to reduce the computational cost.

1) ***Reducing the number of calls to the solver of high fidelity CFD models.*** This requires the optimization method to be computed efficiently so that better designs can be found with fewer calls to the CFD solver. Currently, this is the primary focus of increasing the optimization's performance amongst the design community and a substantial amount of research has already been carried out on methods for dealing with this issue. Although obviously very important, it is outside of the scope of this thesis.
2) ***Speeding up the convergence rate when calculating the CFD model.*** As in any design process, the faster the CFD code the better, since the more design iterations that can be performed within a given time-scale the more likely the designer is to achieve the design target within real world time constraints. This method, which focuses on reducing the computational cost by speeding up the convergence rate of the CFD model, is commonly neglected by the design community. This is mainly because of the long history of

separation between the design community and the CFD community. In this thesis, a new method of a fast calculation algorithm is presented in Chapter 5.

3) ***Introducing human experience into the optimization process to narrow the search space to a manageable size.*** On the whole, the basis of the design process is trial and error, and the success of the final design largely depends on the knowledge and intuition of the designer. In this thesis a product design system is developed to give designers the opportunity to guide the design and optimization process using known engineering rules and their experience. By placing the human interactions in the design procedure, high fidelity models like CFD will be able to display its ability to the fullest extent when it is coupled with numerical optimization methods. Chapter 6 and Chapter 7 describe this new interactive design tool.

The application of this new interactive design system to the design of a coal pipe system is presented in Chapter 8. The computing time and results are documented, and the efficacy of this design tool is demonstrated by comparing it with the traditional design optimization process. Conclusions and directions for future work are discussed in Chapter 9.

Chapter 2 Literature Review Part I: Gas-solid Flow

The pneumatic conveying of solid particles is an important method for the transport of bulk materials. Due to its many advantages such as simplicity and flexibility in operation, it has been widely used in diverse industries such as chemical processing, food processing, and also in transporting pulverized coal in thermal power plants. This wide application has led to extensive research on pneumatic conveying of solids. Pneumatic conveying may be classified into several regimes such as dilute phase conveying and dense phase conveying based on particle loading and transport velocity. The main concern of the present study is dilute phase conveying where the particles are fully dispersed in the flow, and no deposition occurs, particularly in the case of pneumatic conveying of pulverized fuel in coal-fired power plant.

Theoretically, pneumatic transport takes place under gas-solid pipe flow conditions. Although dilute gas-solid pipe flows are modeled using the basic laws of fluid mechanics, a thorough understanding of multiphase flow has yet to be obtained. The reason for this is that gas-solid flows are extremely complex. As one of the major topics in multiphase flow science, researchers in particle technology and fluid dynamics are trying to present more accurate numerical models to describe this complex phenomenon. Essential for a reliable CFD model is the appropriate modeling of the relevant physical mechanisms affecting the particle motion, for example, turbulent transport of particles, wall interactions of particles, collisions between particles and agglomeration. However, in some cases the physical phenomena are too complicated to allow for a derivation of the model from basic principles of physics for realistic devices. Therefore, numerical two-phase computations may be performed on different levels of complexity related to the resolution of the interface between the phases and the turbulence modeling. On the other hand, engineers working on pneumatic systems need to answer the following questions: how will the system respond to a change in the process conditions? Will it change if the particle size, particle hardness, temperature, or in the case of two phase flow, the superficial fluid velocity change? For many dilute phase systems, it is also important to be able to predict the particle concentration in a turbulent flow. Due to the different focus between the micro scale and macro scale issues, the research on gas-solid flow and pneumatic conveying system are generally treated as two different topics.

2.1 Behavior and Properties of Dilute Gas-solid Pipe Flow

Gas-solid pipe flow takes place under turbulent pipe flow conditions. It is necessary to understand and be able to identify certain fluid and particle properties that control the mechanisms that influence the flow field. The primary control properties of the gas-solid pipe include: time scales, particle turbulence interaction, forces on particles, and wall-particle interaction. It is worth mentioning that there have been other surveys of gas-solid flow techniques in the literature. For example, review paper [70] focuses on explaining the mathematical and conceptual issues involved in dilute phase modeling.

2.1.1 Time Scales and Particle Turbulence Interaction

Consider an incompressible turbulent flow, where the flow and the particles are subject to some boundary and initial conditions. Generally speaking, the flow turbulence gives rise to the diffusion of particles, and the particles exert an opposite force on the flow, producing a modification in the flow known as "turbulence modulation". Turbulent flows contain eddies of various sizes, each having different amounts of rotational kinetic energy. Also each eddy has an associated length and time scale, referred to as *eddy characteristic size* and *lifetime of the eddy*, respectively. In other words, eddy size is equivalent to the physical dimension of a particular rotating structure, and eddy lifetime is the time interval for which that structure maintains its original size before it completely dissipates or breaks down into small structures or eddies [70]. In a turbulent flow, the maximum eddy size possible is known *as integral length scale*; the smallest eddy size is called the *Kolmogorov length scale* (η); and the time associated with this eddy is known as *Kolmogorov time scale* (τ). The expressions for Kolmogorov length scale and time scale are given by:

$$\eta = \left(\frac{\upsilon^3}{\varepsilon}\right)^{1/4} \tag{2.1}$$

$$\tau = \left(\frac{\upsilon}{\varepsilon}\right)^{1/2} \tag{2.2}$$

where υ is the turbulent kinematic viscosity, and ε is the turbulent kinetic energy dissipation rate.

The size of the particle with respect to the eddy is an important parameter in determining the outcome of the eddy-particle interaction. It is common to assume that if a small particle (Note: a particle is referred to as small if its diameter is smaller than the Kolmogorov scale) is trapped inside an eddy, it will see a uniform velocity field during its residence within that eddy. If the particle is dense ($\rho_p > \rho_f$), the inertial force at the fluid-particle interface will dampen the fluctuations in its velocity compared to the fluctuations observed for the surrounding fluid. This reduction is characterized by a time scale called the *particle relaxation time*. The particle relaxation time is defined as the rate of response of particle acceleration to the relative velocity between the particle and the carrier fluid. The mathematical expression for this relaxation time is:

$$\tau_p = \frac{4\rho_p d_p^2}{3\mu_f C_D \mathrm{Re}_p} \tag{2.3}$$

where ρ_p, d_p, and Re_p are the particle density, diameter, and Reynolds number, respectively, μ_f is the fluid viscosity, and C_D is the drag coefficient. The expression for particle Reynolds number which is widely accepted for dilute flows is given by:

$$\mathrm{Re}_p = \frac{\rho_f \left|\vec{V}_p - \vec{V}_f\right| d_p}{\mu_f} \tag{2.4}$$

Where $\vec{V}_p$ and $\vec{V}_f$ are the particles velocity and fluid velocity vectors, respectively. A frequently used correlation for the drag coefficient is the Schiller-Naumann model, which fits the data up to Re_p=1000 reasonably well

$$C_D = \left(\frac{24}{\mathrm{Re}_p}\right)\left(1 + 0.15\mathrm{Re}_p^{0.687}\right), \text{ if } \mathrm{Re}_p < 10^3 \tag{2.5}$$

It is generally agreed that turbulent eddies, which have random velocities relative to the mean flow, carry along with them fluid properties and particulate contaminate. Particle inertia will cause the particles to respond sluggishly to fluid turbulence, with particles not following the higher frequencies of turbulent fluctuations. As the particle size and inertia increase, the particle motion becomes less coupled to that of the gas phase.

Stoke number (*St*) is defined as the ratio of the particle relaxation time τ_p and a turbulent time scale. The Stokes number represents the effect of particle inertia in the interaction with the turbulence. The choice of the appropriate turbulent time scale has been a matter of controversy in the past, but most literature agrees that the *Kolmogorov time scale* is a proper choice. One may summarize the particle's ability to follow the flow by listing the criteria associated with Stokes Number:

- St >>1 particle motion is governed by gas phase turbulence
- St <<1 particle motion is only slightly affected by gas phase turbulence, in a random way
- St ≈ 1 particle motion is affected by gas phase turbulence and its inertia at the same time

It is worth mentioning that most of the numerical work on the particle-turbulence interaction has been on homogeneous isotropic turbulence. However, the structure and dynamics of near-wall turbulence is very different from homogeneous isotropic turbulence, resulting in a different behavior for both particles and turbulence. The interaction between the particles and the turbulence leads to a non-uniform particle concentration and the formation of particle clusters, which are quite different from the "preferential concentration of particles" found in homogeneous isotropic turbulence [11].

2.1.2 Forces on Particles

Newton's Second Law of Motion can be used to describe the motion of particles which are immersed in a turbulent flow. This requires the consideration of all relevant forces acting on the particle. Considering spherical particles and neglecting heat and mass transfer phenomena, Fan [1] pointed out that these forces can be placed in four categories: (1) forces that act on a particle due to the motion of the particle such as Basset (i.e. resistant to acceleration), virtual mass (describing the force that is required to accelerate the fluid entrained by the particles); (2) forces that act on a particle due to the motion of the surrounding fluid; (3) forces that act on a particle irrespective of the fact that the particle is immersed in fluid or not; and (4) forces that act on any object immersed in fluid irrespective of either particle or fluid motion.

2.1.3 Wall-particle Interaction

Since flows in pneumatic conveying systems belong to confined flows, particle-wall collisions become relevant. The momentum loss of a particle caused by an inelastic wall impact is associated with a re-acceleration of the particle after rebound. Hence, momentum is extracted from the fluid phase for this acceleration, causing an additional pressure loss. A first estimate of the impacts of particle-wall collisions may be based on the ratio of the *particle response distance* λ_p to the dimension of the confinement, e.g. the diameter of the pipe D. The particle response distance can be estimated from the following equation:

$$\lambda_p = \tau_p u_t \tag{2.6}$$

where u_t is the terminal velocity of the particle. In gravitational settling, the particle quickly reaches a constant velocity which is the maximum velocity attainable under the circumstances. This maximum settling velocity is called terminal velocity, which is given as:

$$u_t = \sqrt{\frac{2g(\rho_p - \rho_f)m}{A_p \rho_p C_D \rho_f}} \tag{2.7}$$

where A_p is the projected area of the particle in the plane perpendicular to the flow direction, g is the gravity acceleration, and m is the particle mass. For the case λ_p is larger than the dimension of the confinement D, the particles' motion is dominated by particle-wall collision since they are not able to respond to the carrier flow before they collide with the opposite wall.

2.1.4 Particle-particle Collision

Particle-particle collision is another factor that is important in determining particle motion. It is common to consider that particle-particle collision can be safely ignored in dilute pipe flows. However, even in the case where average particle concentration in the cross section of the pipe is low enough that inter-particle collision frequency is negligible, as the gas-solid mixture exits the elbow, the coal particles are concentrated together in a small region of the pipe cross-sections. This high particle concentration region in a pipe cross-section is often referred to as the rope region (this will be discussed further in Section 2.3). Since particles of high velocities flow into this region space, complex inter-particle collisions occur very frequently. In other words, it is particle-particle and wall-particle interactions rather than interaction between fluid and particles that determines behavior of the solid-flow in such places. This leads to another important time scale, the *particle-particle collision time* τ_c, which can be defined as:

$$\tau_c = \frac{d_p}{24\alpha g_0}\left(\frac{\pi}{T_2}\right)^{1/2} \tag{2.8}$$

where T_2 is the granular temperature and g_0 is a distribution function. τ_c represents the time experienced by a given particle between two consecutive binary collisions.

A classification of particle-laden flows in terms of importance of particle-particle collisions may be based on the particle response time τ_p to the averaged time between collisions τ_c. If $\frac{\tau_p}{\tau_c} < 1$, the time between successive inter-particle collisions is larger than the particle response time, whereby the fluid dynamic transport of the particles is the dominant transport effect. If $\frac{\tau_p}{\tau_c} > 1$, the time between particle-particle collisions is smaller than the particle response time, the particles are not able to completely respond to the fluid between successive collisions.

In this section, an overview of the particle response to turbulent eddies under simple conditions has been presented. Although the discussion was limited to basic concepts, several factors are important when predicting turbulent particle dispersion in pipe flows. Some other basic principles about particle motion in a confined pipe line such as wall-particle and particle-particle collision have been discussed extensively in the literature such as review [85].

2.2 Governing Equations for Gas-solid Flows

Since in the dilute phase the flow carrier phase plays a major role in determining the flow pattern, understanding turbulence phenomena is necessary to develop an understanding of the particle dispersion.

2.2.1 Governing Equations for the Carrier Phase

In general, the carrier phase flows under the influence of external forces, e.g. pressure force, and is governed by the Navier-Stokes equation and conservation of mass equation in the Eulerian framework. In the cases of incompressible flow, low particle Reynolds number and dilute concentration, the carrier phase plays a major role in determining the dynamics of the system; the effect of the particles on the continuity equation can be neglected, and the interaction between the particles and the carrier phase occurs through an exchange of momentum. The continuity and Navier-Stokes equations for the continuous phase become

$$\nabla \cdot U = 0 \tag{2.9}$$

$$\rho_f \left\{ \frac{\partial U}{\partial t} + (\nabla U) \cdot U \right\} = -\nabla P + \mu_f \nabla^2 U + \mathcal{F} \tag{2.10}$$

Where U is the velocity of the fluid (continuous phase), P is the pressure, and $\mathcal{F}$ is the force per unit of volume due to the particles.

2.2.2 Governing Equations for the Particle Phase

In addition to solving the transport equations for the carrier phase, the trajectories of particles must also be computed. As discussed before, the motion of the dispersed phase occurs due to forces generated by the moving fluid and acting through the interface such as fluid drag force. The particles can also be acted upon by other forces, e.g. gravity, thermophoresis, electrostatic (for charged particles); all these forces paired with Newton's second law govern the Lagrangian trajectory of motion of each particle in the fluid flow domain.

For each particle we have

$$m_p \frac{dU_p}{dt} = F + F_g + F_{pg} + F_A \tag{2.11}$$

Where U_p is the velocity and m_p the mass of the particle. F is the force exerted by the fluid on the particle. And subscripts g, pg, and A denote force components arising from gravity, flow pressure gradient, and added mass effect, respectively.

2.3 Numerical Models for Gas-Solid Flows

As discussed before, gas-solid flows involve many different phenomena such as turbulence-particle interaction, particle-particle interaction, wall-particle interaction and the effect of existing force fields. The ideal numerical solutions of the above equations for a fluid-particle flow would provide a "fully resolved simulation" with the detailed flow around every particle. Thus given global initial and boundary conditions together with the boundary conditions imposed at the surface of each particle, in theory it is possible to obtain the motion of each particle by integrating the particle equations using the local velocity, temperature, and density of the carrier flow and accounting for all particle-particle collisions. However, this is impractical due to the limitations of computer resources available today. There are several reviews (e.g. [10] and [11]) on two-phase flow modeling in the literature. The following section only provides a broad overview of the numerical simulation of two phase turbulence flow and some of the available choices and results that follow from them.

2.3.1 Carrier Phase Models

Accurate prediction of particle transport is strongly dependent upon providing a realistic description of the time-dependent, three dimensional velocity field encountered along particle trajectories. To study the behavior of particles in a turbulent flow field numerically, one needs a proper representation of turbulence itself first. Therefore, the way to model governing equations for the carrier phase forms the starting point for various numerical approaches which are categorized as direct numerical simulation (DNS), large-eddy simulation (LES) and Reynolds average Navier-Stokes modeling. All these turbulence models are cases in an Eulerian reference frame.

Among these three approaches, the most sophisticated way to simulate turbulence is DNS. With DNS studies, the Navier-Stokes equations and continuity equations are capable of generating all the details of real turbulence fluctuations. Unlike other approaches, these complex equations are solved without approximation and are sufficient to cover the small dissipative structures. Because of this, DNS methods require a large number of grid points that are sufficiently fine so they can resolve all flow eddies down to the smallest scale. DNS also requires a large number of time steps to reach a statistically steady state [20]. However, turbulence flow consists of a range of scales that increases rapidly with Reynolds number. In order to capture all of the length scales DNS needs a computational mesh with spatial grids in order of $O(\mathrm{Re}^{9/4})$ and time steps in order of $O(\mathrm{Re}^{3/4})$. This restricts DNS to low Reynolds numbers. Therefore in industrial applications, where Reynolds numbers are usually very high, DNS modeling of turbulence is not feasible due to real-world time constraints.

The equations, used most commonly for carrier phase engineering problems, are based on Reynolds-averaged Navier-Stokes (RANS) equation with several modeling assumptions (i.e., Boussinesq's hypothesis). A primary shortcoming of RANS methods for the prediction of two-phase flows is related to deficiencies associated with the model used to predict properties of the Eulerian turbulence field. Deficiencies in the prediction of turbulence properties in RANS calculations adversely impact prediction of dispersed phase transport. One of the most popular turbulence models in use today is an eddy-viscosity model known as $k-\varepsilon$ model. The $k-\varepsilon$ model was solved simultaneously with the average Navier-Stokes equations to obtain a prediction of the mean flow process. However, it is known in single-phase flow modeling that the Boussinesq approximation yields an unsatisfactory description for certain types of flows, such as flows over curved surfaces, three-dimensional flows, flows with boundary layer separation and flows with secondary motions. When the mathematical and physical complexity increases, alternatives such as Reynolds stress models (RSMs) and Algebraic stress models (ASMs) are possible.

The LES method is an approach which is not as severely restricted in the range of Reynolds numbers as DNS. The LES method involves both direct simulation and Reynolds-averaging approaches. With the LES method, the large scale eddies, which control turbulent diffusion of momentum or heat, are computed directly and only small eddies are modeled. A significant advantage of LES over RANS methods is that it permits a much more accurate accounting of particle-turbulence interactions. However, with today's computer capacity, LES are feasible for turbulent flows in simple geometries and become computationally expensive in cases of inhomogeneous turbulent flow of practical interests.

2.3.2 Particle Dispersion Models

The particle dispersion models for dilute gas-solid flows can be divided into the "trajectory" approach and the "two-fluid" approach based on the type of reference frame used for formulation. The trajectory approach is referred to as the Lagrangian method since it treats the particles as discrete entities interacting with turbulent eddies in a Lagrangian coordinate frame. For two-fluid models, the reference frame is stationary, and the particles pass through fixed differential control volumes; thus, it is also known as the Eulerian model. The Eulerian approach treats the particulate phase as a continuum having conservation equations similar to those of a continuous gas phase. In general, the Lagrangian method is a more natural way to treat particles in dilute flows; hence, these models are very popular in applications such as spray and pulverized coal combustion systems. Eulerian models are popular when the particle loading is high, as in the case of the fluidized-bed combustion systems. However, they can also be used in modeling dilute particle-laden flow. The comparative studies of the two modeling approaches have been summarized in [22]. Since the research in this work is limited to dilute flows, the following section focuses on discussion on Lagrangian models.

The Lagrangian particle models can be further classified into two major trajectory models. In the first type, the particle trajectories are generated directly using a stochastic model (random walk model) for Lagrangian velocities. The models are based on Taylor's approach for fluid particle dispersions. In the second type of model, the particle trajectories of representative samples are obtained by solving the particle momentum equation through the Eulerian fluid velocity field. The main problem of this approach is determining how to estimate the instantaneous fluid velocity that appears in the particle momentum equation. Although theoretically possible to track each individual particle using the Lagrangian approach, it is not always practical to do so. Normally, particles are divided into representative samples, each having the same dimension and initial conditions. The trajections of these representative samples are then determined in order to reduce the total number of trajectory computations to a reasonable figure. However, the number of trajectories should be high enough to provide statistics that characterize the actual particle behavior.

2.4 Conveying Characteristics of Pipeline Components

Due to the complex nature of the gas-solid flows, most of the modeling techniques discussed above have been restricted to very simple cases. When information on pressure drop, solids concentration, or particle velocities is required for pipe design in practical situations with large scale complex piping, the piping designer has no alternative but to utilize one of the empirical correlations available in the literature. Therefore, in the past the majority of research on the dilute phase pneumatic system has been focused on finding better empirical equations. There are numerous experimental and theoretical/numerical studies on finding empirical equations [89, 90, 91].

Most recently, with the help of improved gas-solid flow numerical models and the increase of computer power, many physical and numerical experimental studies of gas-solid flows [72-79] have been conducted. In general, this kind of research starts with a comparison between the prediction that uses a specific numerical model and actual experiments in order to validate the accuracy of the numerical model. Once it proves that the numerical model can yield reliable predictions, conclusions are made based on the numerical results. Usually, the conclusions involved either prove that the numerical model can yield better predictions than other models or some predictions on the conveying characterization of gas-solid flow in order to help engineers better understand the gas-solid flow phenomena. However, it is important to keep in mind that such conclusions have a limited value since particle dispersion models are limited by incomplete knowledge of the instantaneous turbulent flow field. Therefore, it is generally accepted that the most reliable method is to base the predictions on data obtained from conveying trials in an instrumented test plane. The tables from Summerfel's review paper [90] summarize the work done in the modeling of lean phase pneumatic conveying. Table 2.1 is for the more experiment oriented examinations, and Table 2.2 is for those mainly concerned with modeling, numerical experiment oriented examinations.

Many factors influence the process of pneumatic conveying. The dominant ones used from recommendations in the literature for gas-solid flows in complete pipes are as follows [4]:

- Gravitational settling in horizontal pipes;
- Inertial behavior in pipe bends and branches;
- Turbulent dispersion;
- Transverse lift forces induced by particle rotation;
- Lift forces due to shear flow;
- The collision of the particles with the rough walls of the pipe;
- Wall collision process for non-spherical particles;

Table 2.1 Summary of experimental studies on gas-solid channel and pipe flows [90]

Reference	Flow configuration	Dimension	Gas velocity [m/s]	Type of particles	Particle density [kg/m^3]	Particle diameter [mm]	Mass loading [kg/kg]	Instrumentation
Matsumoto and Saito (1970)	Horizontal channel	$L = 6$ m, $H = 25$ mm	7 and 10	Polystyrene Glass Copper	1040 2500 8700	0.94 0.5 and 0.95 0.51	0.5	Visualisation
Lee and Durst (1982)	Vertical pipe	$L = D = 41.6$ mm	5.7	Glass beads	2500	0.1, 0.2, 0.4, and 0.8 mm	Up to 1.0	LDA
Lourenco et al. (1983)	Horizontal channel	$L =$ unknown 60×30 mm^2	6–13	Glass beads	2400	0.25 and 0.5	Up to 3.0	LDA
Tsuji et al. (1985)	Horizontal pipe	$L = 3.56$ m, $H = 30.5$ mm	7–20	Polystyrene	1000	0.2 and 3.4	Up to 5.0	LDA
Tsuji et al. (1984)	Vertical pipe	$L = 5.11$ m, $H = 30.5$ mm	8–20	Polystyrene	1000	0.243, 0.5, 1.42, and 2.8	Up to 5.0	LDA
Sommerfeld (1992)	Vertical channel	$H = 25$ mm	8.6	Glass beads	2500	0.45 and 0.11	Very low	LDA
Kulick et al. (1994)	Downward channel	$L = 5.0$ m, $H = 40$ mm	10.5	Lycopodium Glass beads Copper	300 2500 8700	0.1 and 0.5	Up to 0.8	LDA LDA
Huber and Sommerfeld (1994)	Different pipe elements	Different length $D = 80$ mm	10–30	Glass beads	2500	0.042 and 0.110	Up to 2.0	PDA, Imaging for concentration
Varaksin et al. (1998, 1999)	Vertical pipe	$L = 1.38$ m, $D = 46$ mm	5.2 and 6.4	Glass beads Aluminia	2550 3950	0.05 and 0.1 0.05	Up to 1.2	LDA
Sommerfeld and Huber (1999)	Horizontal channel	$L = 3.0$ m, $H = 30$ mm	5–15	Glass beads Quartz	2500 2400	0.1 and 0.5 0.2	Very low	Particle tracking (streak technique)
Kussin and Sommerfeld (2002)	Horizontal channel	$L = 6$ m, $H = 35$ mm	10–25	Glass beads Quartz	2500	0.06–1.0	Up to 2.0	PDA PIV

Table 2.2 Summary of numerical studies on gas-solid channel and pipe flows [90]

Reference	Numerical method	Flow configuration, dimensions	Gas velocity [m/s]	Type of particles	Particle density [kg/m^3]	Lift forces	Wall collision	Inter-particle collision model
Ottjes (1978)	Predefined flow, particle tracking, no coupling	Horizontal pipe (two-dimensional) $D = 12$ mm	20	Spherical particles, 3 mm	820	Slip rotation	Inelastic, rotation, no roughness	No collisions
Tsuji and Morikawa (1978)	Predefined flow, particle tracking no coupling	Pipe bend ($D = 27$ and 50 mm)	10–20	Polystyrene 1.6 and 2.7 mm	1000	Slip rotation	Inelastic, rotation, no roughness	No collisions
Tsuji et al. (1987)	Euler/Lagrange, two-way coupling, no turbulence	Horizontal channel ($H = 25$ mm)	7 and 15	Polystyrene 1.0 mm	1000	Slip shear, slip rotation	Inelastic, rotation, virtual wall	No collisions
Tanaka and Tsuji (1991)	Predefined flow, particle tracking, no coupling	Vertical pipe, periodic domain ($D = 40$ mm)	16	Polystyrene 0.4 and 1.5 mm	1040	Slip shear, slip rotation	Inelastic, non-sphericalparticles	Deterministic, rotation
Tsuji et al. (1991)	Euler/Lagrange, two-way coupling, no turbulence	Horizontal pipe ($D = 52$ mm)	10	0.41–1.0 mm	1038	Slip shear, slip rotation	Inelastic, non-spherical particles	No collisions
Oesterlé (1991)	Predefined flow, particle tracking, no coupling	Pipe bend ($D = 80$ mm)	20 and 40	Spherical particles, 0.05 and 0.1 mm	2620	Slip shear, slip rotation	Inelastic	Stochastic, uncorrelated velocities
Sommerfeld and Zivkovic (1992)	Euler/Lagrange, two-way coupling	Horizontal channel ($H = 26$ mm) and pipe ($D = 80$ mm)	10.7 and 15	Glass beads 0.1 and 0.04 mm	2500	Slip shear, slip rotation	Inelastic, rotation, roughness	Stochastic, uncorrelated velocities
Oesterlé and Petitjean (1993)	Euler/Lagrange, no coupling	Horizontal pipe ($D = 30$ mm)	25.5	Spherical particles 0.1 mm	2620	Slip shear, slip rotation	Inelastic, rotation	Stochastic, uncorrelated velocities
Sommerfeld (1995)	Euler/Lagrange, no coupling	Horizontal channel ($H = 30$ mm)	20	Glass beads 0.045 and 0.1 mm	2500	Slip shear, slip rotation	Inelastic, rotation,wall roughness	Stochastic, uncorrelated velocities
Cao and Ahmadi (1995)	Euler/Euler, two-way coupling	Vertical pipe ($D = 30.5$ mm)	10–20	Polystyrene 0.2 and 0.5 mm	1040	Neglected	Slip boundary condition	Collisional stresses
Tu and Fletcher (1995)	Euler/Euler, no coupling	Channel bend	52	Spherical particles, 0.05 mm	2990	Neglected	No slip, generalised Eulerian	No collisions
Lun and Liu (1997)	Euler/Lagrange two-way coupling	Horizontal channel ($H = 25$ and 30 mm)	7–15	Glass beads 0.25 and 1.0 mm	2500	Slip shear, slip rotation	Inelastic, rotation	Deterministic
Huber and Sommerfeld (1998)	Euler/Lagrange four-way coupling	Horizontal pipe, pipe bend, vertical pipe ($D = 80$ and 150 mm)	24 and 27	Glass beads 0.04 mm	2500	Slip shear, slip rotation	Inelastic, rotation, wall roughness	Stochastic, uncorrelated velocities

As mentioned in Chapter 1, flexibility of routing is a feature particular to pneumatic conveying systems. The majority of pipelines, however, probably have a major proportion of horizontal pipe, vertical up sections are common, bends are prominent, and some have vertical down sections. In some cases, the systems also include some inclined pipes. The performance characteristics of the various components contribute to the total pipeline system performance characteristic. For example, conveying characteristics for straight pipelines are very different from those for pipeline bends. If the proportion of straight sections and bends varies between two pipelines, the conveying characteristics for the total pipeline systems can also be very different. To illustrate the influence of individual pipeline sections on the conveying performance for a total pipeline system, the discussion on different characterizations of components is presented in the following sections.

2.4.1 Horizontal Pipes

Wen and Simons studied the flow characteristics of glass beads and coal of various sizes through horizontal glass pipes and discussed the flow patterns on the basis of visual observation. In horizontal pneumatic transport of granular solids, the flow features are very complicated because of the action of gravitational forces perpendicular to the flow direction. The solids distribution over the cross section of the conveying pipe is substantially influenced by gravitational forces, gas-phase turbulence, collision between particles, and collision between particles and the wall when particles are transported in a horizontal tube by a turbulent gas stream.

2.4.2 Vertical Pipes

The most important aspect in designing a vertical pneumatic conveying system is the selection of the working velocity of the handling gas. Using a high gas velocity leads to unnecessary energy losses and increased erosion whereas a low velocity produces a choking hazard characterized by particles leaving the stream, followed by cessation of conveying and finally clogging in the pipes. From an economical point of view, the practical gas velocity should be slightly higher than the choking velocity.

2.4.3 Inclined Pipes

Apart from horizontal and vertical pneumatic conveying systems, inclined pneumatic conveying lines are quite common in the industry and thus attract the attention of researchers. A particular interest has been displayed in the transportation of particulate solids along inclined pipelines in lieu of horizontal and vertical pipelines due to its potential advantages over vertical and horizontal configurations. When compared with the numerous studies conducted for vertical and horizontal pneumatic transport, fundamental studies of inclined transport are still very limited, mostly to experiments on dilute-phase suspended upflow. Currently, most of these investigations were focused on pressure drop information. Zenz [1] and others [2] pioneered work investigating the pressure gradient in inclined gas-solids upflow. For a given solids flux and gas velocity, they found that the pressure gradient increases with the inclination angle from the horizontal to the vertical and that it differs only slightly at high angles of inclination between 67.5 and 90 degrees. The experiments performed by Morikawa et al [128] showed that the pressure gradient along a pipeline with inclination angles of 30, 45, and 60 degrees is higher that that along a vertical one under the same operating parameters. In addition, the work of Klinzing et al [129] confirmed a steady growth in the pressure gradient with increase in the inclination angle at relatively low angles ranging from 0 to 45 degrees at constant solids flux and gas velocity. Ginestet [127] also reported that, especially in the dense-phase region, the pressure gradient at high inclinations, above 72 degrees is significantly and invariantly greater than that at an inclination angle of 90 degrees for the same solids flux and gas velocity. All of these previous experimental studies indicate that a maximum pressure gradient exists that correspond to a high angle of inclination to the vertical. Ginestet attributed this phenomenon to the fact that, for angles greater than this angle, the reduction of the pressure gradient due to particle reflux along the pipe wall with increasing inclination exceeds the increase in gravitational force. For angles of inclination below this angle, the reduction in gravitational force with decreasing inclination exceeds the potential increase in pressure gradient due to particle reflux along the wall. Levy [126] investigated the pressure drop in inclined pneumatic conveying systems by both experiments and numerical simulation. Hirota [83] experimentally studied the influence of mechanical properties of powders on pressure drops in inclined gas-solid flows.

2.4.4 Bends

Many two-phase flow phenomena can occur in a bend of the piping system. Assume that particles enter the tubing with uniform distribution. If this gas-solid mixture flows along a straight pipe, the solid phase may separate from the air because of the gravity. As the gas and particle mixture flows through an elbow, the momentum of the particles carries them to the outer radius of the elbow and this volume with high particle density tends to stay together. As the mixtures exits the elbow, the particles are concentrated together in a small region of the pipe cross-sections. This high particle concentration region in a pipe cross-section is often referred to as a rope region. This rope region has a much higher particle concentration than the remainder of the pipe cross-section. In addition, the particles are decelerated in the elbow due to particle-wall and particle-particle interactions. Thus, an acceleration region is required in the pipe downstream of the elbow to reaccelerate the particles to the conveying gas velocity. Downstream from the elbow, the dust rope begins to disperse due to turbulent flow mixing and the accentuated double vortex flow structure created by the elbow [6].

Secondary flow

It is well known that a bend causes a secondary, transverse flow pattern in the tubing, consisting of two counter rotating vortices. This flow can be understood in the following manner. In a curved pipe flow, the pressure closer to the outer wall is higher, thus a net pressure force acts on fluid elements toward the inner wall. For fluid elements in the main flow, this pressure gradient is balanced by centrifugal force acting outward and proportional to the square of the velocity. In the boundary layer however, there is no such balance: the bulk pressure profile permeates the layer, but the velocity is low there. Thus the pressure difference in the boundary layer is a driving force for a flow in that layer from the outer wall to the inner wall. This is the flow driving the two counter rotating vortices, the so-called Dean vortices. The turbulent flow field through the bends is much more complicated, and up to now only few numerical simulations of this highly intricate flow have been carried out. The numerical predications of turbulent gas-particle flow through an elbow show the single-phase solutions are modified by the momentum transfer between the gas phase and the particulate phase. The secondary flow patterns are also modified, resulting in four vortices in the pipe cross-section instead of the two large vortices present in single-phase flow. Figure 2.1 compares these two different secondary flow patterns. The secondary flow patterns exhibit a flow current away from the outer wall towards the pipe center. This flow pattern provides the mechanism for removal of particles from within the rope to particle-free regions in the pipe cross-section [6].

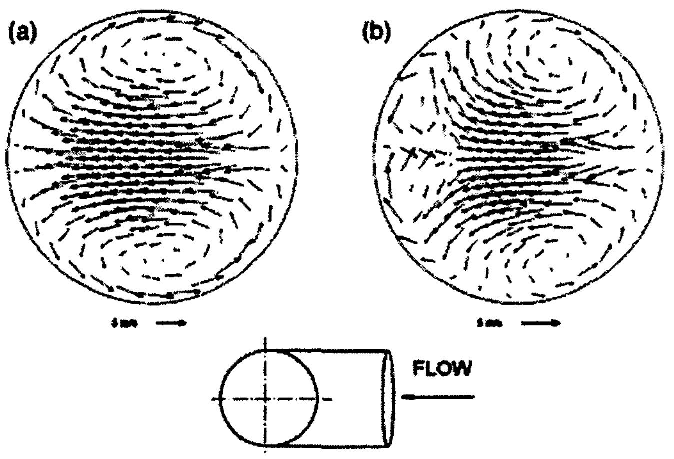


Figure 2.1 Secondary flow patterns (a) single-phase flow solution (b) gas-particle solution [6]

Roping

References to the formation of ropes date back to the late 1950's. Due to the extreme difficulty of the rope problem, numerical studies of turbulent gas-solid flows through pipe bends are very rare. Previous research [5-8] has shown that rope characteristics depend on many parameters, such as centrifugal forces, secondary flows, conveying gas velocity, elbow radius of curvature-to-pipe diameter ratio, solids-to-gas mass flow rate ratio, pipe orientation, particle diameter and particle density and the roughness of pipe wall. Although a very large amount of work has been done in this field over the past 50 years, because of the complexity of the phenomenon, most of the experimental and numerical research is still focused on the fundamental models such as vertical-to-horizontal pipe with 90^{o} elbow, or the horizontal-to-vertical pipe found today. A full understanding of the physics even in a simple pipe has not yet been achieved. Among the available research, an important study on gas and particle flow through bends is that of Huber and Sommerfeld [10]. They studied the cross-sectional distribution of fine powders in dilute phase pneumatic conveying through different pipe elements. They recommended the rope to be viewed as a third "phase", wherein the particle velocity is considerably lower than the conveying gas velocity. The rope disintegrated after about one meter in a vertical pipe section behind the bend. Yilmaz [6] combined experimental and numerical results to describe the mechanisms involved in rope formation and dispersion in vertical pneumatic conveying line following horizontal-to-vertical bends. They observed that both turbulence and secondary flows disperse the particle ropes. For vertical-to-horizontal bends, McCluskey et al [124] showed that the particles in the rope at the bend exit have a velocity one third of that of the average gas velocity. They observed the formation of particle deposits near the elbow exit and concluded the deposit was created as the rope, traveling along the bottom of the pipe, was slowed by frictional forces to a velocity of zero. Levy and Mason [40] observed that the maximum concentration occurs just downstream of a bend for small particles and further downstream for larger particles after the elbow. Also the flow became fully developed sooner for larger particles. Because real particulate phases have a particle size distribution, segregation of particles in the pipe downstream of a bend is expected.

Chapter 3 Literature Review Part II: Optimum Engineering Design and Virtual Engineering Technology

This research addresses several areas, including gas-solid flow, optimum engineering design, and virtual engineering. Two-phase flow of coal within a typical coal transport system in a power plant is discussed in Chapter 2. In this chapter, an overview of optimum engineering design and virtual engineering is provided.

3.1 Optimum Engineering Design

Engineering design is a special form of problem solving where a set of objectives has to be optimized while satisfying some constraints. When the design process is specifically applied to find optimal shapes, it is called shape design. Shape design is one of the most common design problems for fluid machinery such as thermal process equipments and aircrafts where fluid flow plays an essential role. The work presented in this thesis concerns shape optimization coal transport piping.

3.1.1 Design Procedure

It is worth noting that methods for shape design can be categorized into two groups: direct and inverse optimization methods. The distinction between the two is in how the design problem is formulated. In fluid mechanics, the inverse method deals with pressure or velocity distributions rather than the geometry. The task is to find the shape or geometry that will give rise to the desired pressure or velocity distribution. In some cases, the inverse method can be highly efficient compared to the direct method, but it has two drawbacks.

First, the distribution imposed on the formulation may not be physically realizable, so a solution may be impossible. Second, even if the distribution imposed is physically realizable, it may not be an optimal distribution [36].

If the design problem is posed as a minimization problem of an objective function subject to constraints on geometry or flow conditions, the direct optimization method is formed. The direct optimization method becomes time-consuming as a result of a number of numerical simulations needed to design profiles. However, with the development of computational speed and computer memory, the direct method is becoming a more promising optimization method, especially for multiobjective problems.

3.1.2 Formulation of Direct Optimization Model

It has been shown that design optimization problems from different fields of engineering can be formulated into the form of a standard model. The objectives and constraints have to be formulated as a function of the design evaluation and optimization can then be formulated as the classical mathematical problem of finding the extreme values of functions. Therefore, optimization techniques are quite general and the concepts and methods are applicable to many fields. The standard design optimization model is defined as follows:

Find n-vector

$$\vec{X} = (x_1, x_2, \ldots, x_n) \tag{3.1}$$

to minimize the function

$$f(\vec{X}) = \min f(x_1, x_2, \ldots, x_n) \tag{3.2}$$

Subject to the p equality constraints

$$h_j(\vec{X}) \equiv h_j(x_1, x_2, \ldots, x_n) = 0; \; j = 1 \text{ to } p \tag{3.3}$$

And the *m* inequality constraints

$$g_i(\vec{X}) \equiv g_i(x_1, x_2, \ldots, x_n) \leq 0; \; i = 1 \text{ to } m \tag{3.4}$$

Where *p* is the total number of equality constraints and *m* is the total number of inequality constraints.

Using this model, the general procedure of the shape design consists of four components:

Design variables

An important step in the formulation of any optimum design problem is for the designer to explicitly identify an appropriate set of design variables. There are a lot of parameters that can be defined as design variables according to different design requirements. For shape design, the most interesting class of design variables is the geometrical one which can be used to define particular product shapes.

Objectives and constraints

Basically, objectives describe what one hopes to achieve through the optimization. As in all optimization problems the formulations of the objective function is crucial to the outcome of the optimization. Therefore, the objective function has to reflect all relevant system characteristics. Constraints determine whether the design is feasible or not.

Simulation

System simulation is employed in order to predict the performance of the system. Therefore, the system optimization is based on the simulation results.

Optimization algorithm

A particular optimization algorithm is employed in order to find the minimum or maximum of the objective function.

3.1.3 Traditional Engineering Design

Figure 3.1 shows a flow chart for a traditional engineering design process. Traditional engineering design processes begin with formulating design requirements such as low initial cost, long life, and good performance. Then an initial design is evaluated. This can involve building a prototype and developing a model. When the design is found to be deficient in a requirement, it is changed. The design search process is typically a trial-and-error one that relies heavily on previous experience. The new design is repeatedly subjected to the test phase. The process iterates until the requirements are either met or changed to fit the performance. Often, the process is extremely time-consuming. In the past, people would solve simple problems (with a few design values) by making intelligent guesses about the values of the parameters and using trial-and-error. In many cases, this would yield an acceptable but not optimum result. Furthermore, manual design often limits the scope of the search process to what the human expert is trained to consider as a good solution.

In the fluid machinery field, although experiments still play a crucial role in verification of fluid flow models and in validation of final products, ideally the search for a good design can be based on CFD models. CFD is a practical tool which can be used to predict the performance of fluid machinery components or to propose modifications to the original design in rehabilitation or upgrading projects. In general, traditional design using CFD can be classified into two categories: design-by-analysis and design-by-sensitivity

[121]. Design-by-analysis is a sequential process; the designer makes changes to the geometry and then runs CFD models to predict the flow field. After this step, the designer will depend on his or her knowledge and experience to decide if the test case is good. For most real applications in the industry, design-by-analysis is the only choice. In the design-by-sensitivity approach, the CFD is used to produce a sensitivity matrix which gives the influence that a change in geometry will have on either the flow field or the design objective function. This approach requires more expertise from designers in order to make optimal choices for the design matrix. This can be a parallel process since it can be done concurrently on a pool of computers.

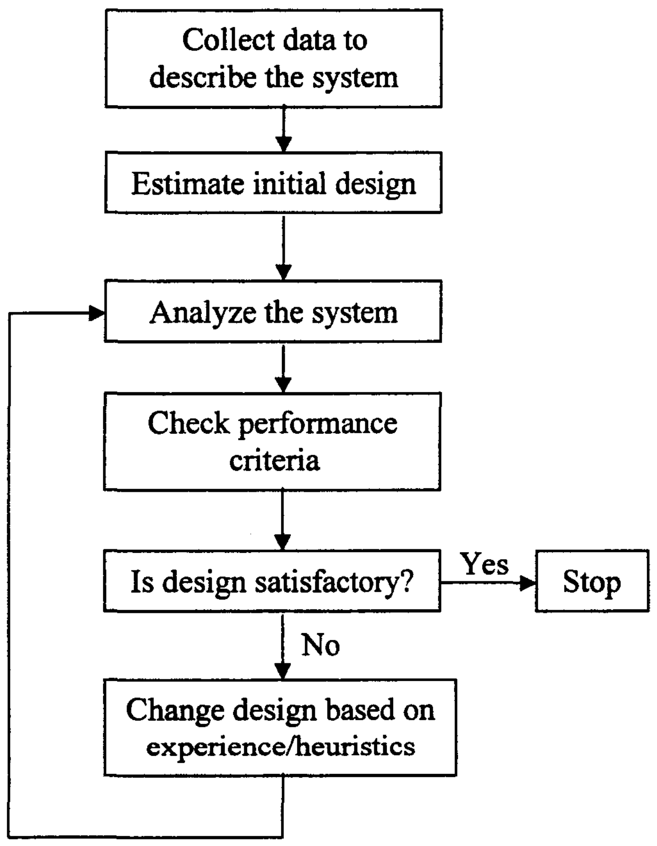

Collect data to describe the system
Estimate initial design
Analyze the system
Check performance criteria
Is design satisfactory?
Yes
Stop
No
Change design based on experience/heuristics

Figure 3.1 Traditional engineering design process [120]

3.1.4 Numerical Engineering Design

During the last two decades, numerical optimization algorithms have been actively studied and have become increasingly more important. The objective of any numerical optimization technique is an automation of the conventional design process so that the best design, in terms of a presupposed criterion, is obtained [19]. Figure 3.2 shows a standard numerical design process. This numerical design process differs from the traditional process in that the iteration loop is computerized. An optimizing algorithm, which serves as the design modifier, is coupled with an appropriate engineering analysis code. The analysis code performs the test phase in the iteration loop. When used in the numerical optimization systems, the CFD code provides a series of function evaluations that are used by the optimizer to search for the optimum value of the given objectives' functions. Very often, the significant computational cost associated with CFD modeling makes numerical optimization of these systems too lengthy and computationally expensive. As a consequence, numerical optimization systems are routinely performed using lower fidelity models. Today, CFD is primarily used to provide insight into a limited number of specific design issues rather than as a design and optimization tool. One of the goals in this work is to use CFD as an analysis component for a design and optimization tool.

3.2 Optimization Algorithm

One of the most important issues in the shape optimization is how to search for an optimal solution to the design problem. With the development of powerful computers and modern computational techniques, numerical optimization algorithms are more predominant in many areas such as air foil design.

3.2.1 Classification

In general, the numerical optimization algorithms could be divided into derivative/deterministic methods and stochastic methods. Deterministic algorithms follow a fixed search pattern, while stochastic algorithm involves some random processes in generating new designs.

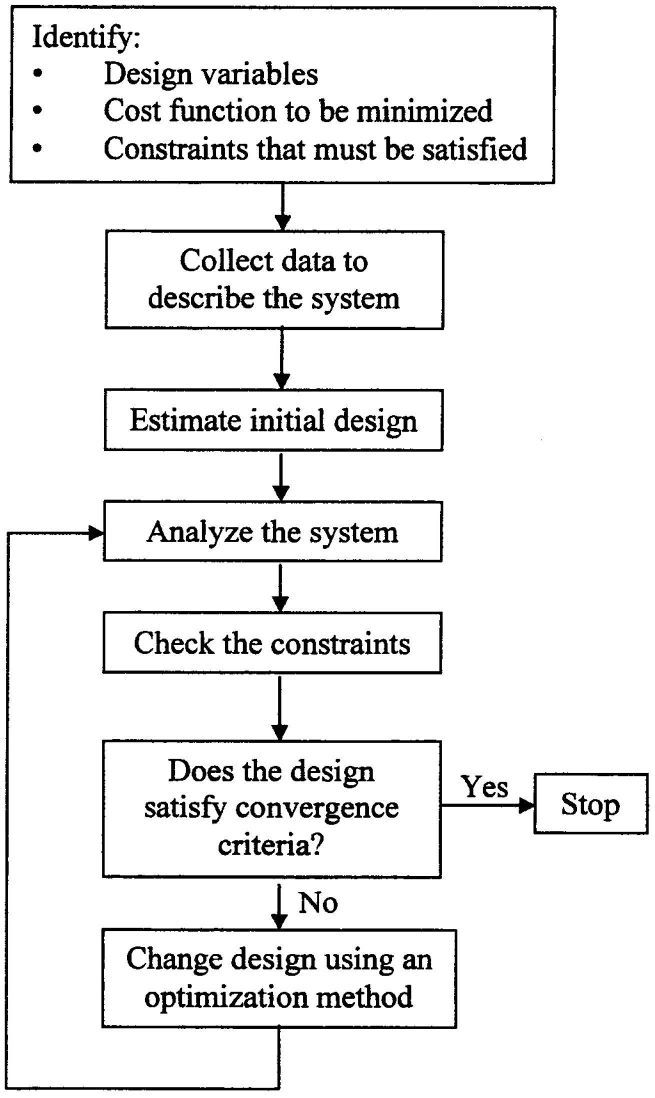

Identify:
• Design variables
• Cost function to be minimized
• Constraints that must be satisfied
Collect data to describe the system
Estimate initial design
Analyze the system
Check the constraints
Does the design satisfy convergence criteria?
Yes
Stop
No
Change design using an optimization method

Figure 3.2 Optimum design process [120]

Gradient-based methods and sensitivity analysis are two of the most well-known deterministic methods. The gradient-based method is very efficient in finding the local minima of continuously differentiable problems with simple fitness landscapes at a reasonable cost. The main drawback of this method is that the optimum obtained may not be a global one. In addition, if the optimization involves a large number of design variables, it can be very difficult to obtain the descent direction and the step size needed to carry out the optimization process. The object of sensitivity analysis, in principle, is to analyze the behavior of the system response by calculating the sensitivities around a chosen parameter or state variable. For many optimization problems, once these sensitivities are available, the computational cost can be reduced significantly.

On the other hand, stochastic optimization methods that are robust in searching for the global optimum are more suitable for general engineering design problems. EAs and other closely related evolutionary strategies are one class of non-gradient methods that have recently grown in popularity. When EAs are applied to optimize design problems, the major advantage over other optimization procedures is mainly related to the robustness of the procedure. EAs do not require any derivatives of the objective function in order to calculate the optimum and can use discrete rather than the continuous design inputs required by traditional optimization schemes. They are also known as black box methods and easy to implement. Another advantage is that they are more likely to find the global optima as compared to the conventional methods that search from a point to another point such as the gradient-based methods. During the optimization, EAs explore the entire feasible space and search from different design points in one run; thus, the probability of finding a local peak instead of the global is reduced significantly. EAs are also an attractive method for multi-objective design applications being offering "pareto optimal sets" instead of a limited single design point traditionally provided by other methods. EAs have become a widely accepted technique in shape optimization to better search the design space. Because of this, EAs are becoming more predominant in shape optimization and engineering design. EAs also have some disadvantages. They are computationally demanding, since many candidate solutions have to be evaluated in the optimization process. In addition, EAs typically have more algorithmic parameters to tune compared with simpler techniques. These parameters are unfortunately problem dependent and few quantified rules are available to set them.

3.2.2 Evolutionary Algorithms: A Brief Introduction

In 1975, John Holland [37] proposed an optimization technique that exploited an analogy between function optimization and the biological process of evolutionary adaptation. This technique has gained popularity as a robust optimization tool. Regarding the implementation of EAs, there is a great variety both in population structures and evolutionary operators. However, all EAs have an initialization phase followed by an iteration phase that evolves the initial population to a better set of solutions to the problem. Roughly speaking, evolutionary algorithms attempt to find the optimal solution of the problem at hand by manipulating a population of candidate solutions [28].

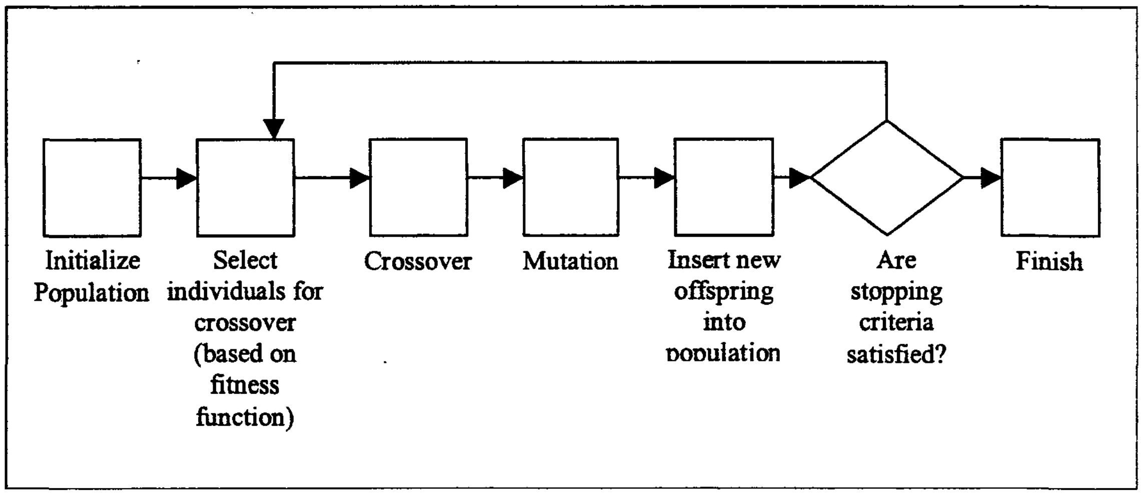

Initialize Population
Select individuals for crossover (based on fitness function)
Crossover
Mutation
Insert new offspring into population
Are stopping criteria satisfied?
Finish

Figure 3.3 Basic flowchart of EAs [28]

A general flowchart for all EAs is shown in Figure 3.3. To implement an evolutionary algorithm, a number of candidate solutions are created throughout the search space. Every individual candidate consists of a genome and a fitness. The genome consists of a number of genes that altogether encode a solution to the optimization problem. The fitness represents the performance of the candidate solution encoded in the individual's genome, and it is usually measured using a so-called fitness function. The reproduction is achieved using crossover and mutation operators. The children (new candidate solutions) are inserted into the population and the procedure starts over again until the population has converged or the maximum number of generations has been reached. During the run, the fitness of the best individual improves over time and typically tends to stagnate towards the end of the run (see Figure 3.4).

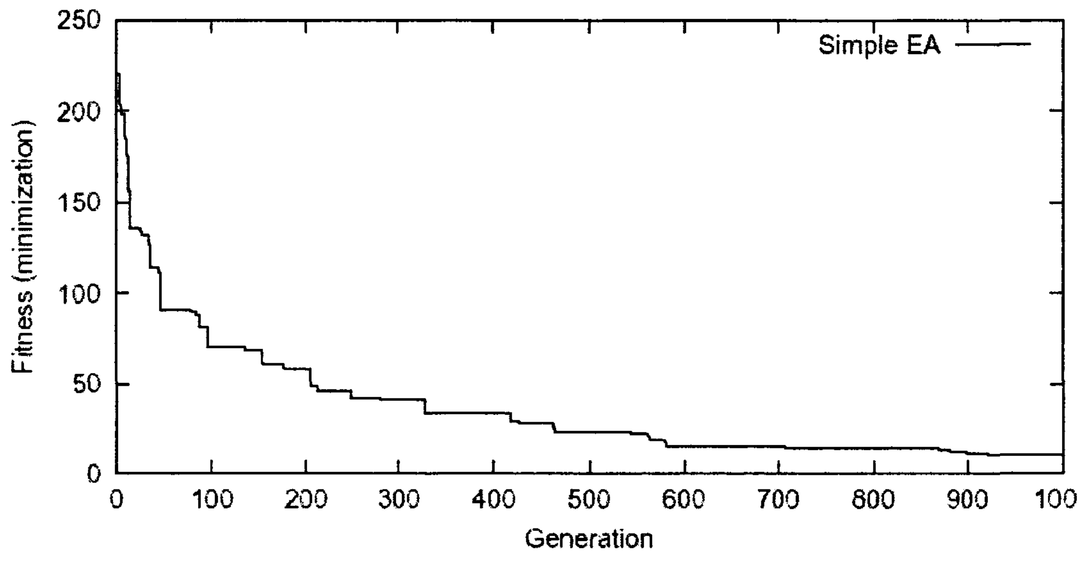


Figure 3.4 Gradual fitness improvement during the run [51]

There are four main operators that define how designs evolve within a population.

1) the method of selection of the parents from the population,
2) the crossover operators,
3) the mutation operators, and
4) the basis for placing the evolved children into the population.

3.3 Previous Work on CFD-based Shape Optimization

In many cases, the design and engineering of thermal fluid systems requires high-fidelity models (e.g., CFD) to understand the details of the fluid flow, heat transfer, or other phenomena of interest. Because of this, CFD has the potential to become an integral part of engineering design and analysis due to its ability to predict the performance of new designs or processes before they are ever manufactured or implemented. As discussed above, the selection of the optimization algorithm is highly problem dependent. For fluid machinery, the continuity equation, the Navier-Stokes equations, the energy equation, and the needed equation of state govern the flow behavior. Finding the solution of such a system with tightly coupled nonlinear partial differential equations requires large amounts of computing power. Very often, because of the nonlinear coupled nature of the fundamental equations, the objective function landscapes are nonlinear with many local optima, plateaus or ridges even in simple problems. Thus, the sensitivity information cannot be easily extracted from CFD code. The main reason for this is because the sensitivity equations are obtained by differentiating the non-linear governing equations with respect to the design variables. As a result, the computational effort is significantly increased in evaluating the responses of the flow to geometry perturbations. This creates a significant obstacle for using gradient-based methods when design optimization is combined with CFD. Also, when using a commercial CFD package, the gradient information is not available directly. Options such as adjoint gradient calculation and the automatic differentiation method must be eliminated because of the unavailability of the source code of the CFD solver. In such cases, gradient-free optimization methods such as EAs are the preferred optimization algorithm.

There have been a number of studies of utilizing EAs in conjunction with high fidelity CFD models to optimize a wide variety of shape optimization projects. These projects have included airfoils [39, 40, 41, 55], heat exchangers [42, 43], two-dimensional blade profiles [44], missile nozzle inlets for high-speed flow [46, 47], three-dimensional shape optimization [48], sailing yacht fin keel [49], and stoves [50, 51]. Graph based evolutionary algorithms (GBEAs), which are a form of EAs, have also been used to optimize the rate or spread of information through the evolving population relative to the complexity of the search [51]. This provides a means to preserve population diversity and speed up the optimization process.

Increasing EAs performance has been studied extensively and several methods have been suggested to overcome the computational cost of using high fidelity models to perform the fitness evaluation. One of the most common approaches has been to first use a low detail representation of the thermal-fluids design problem to evolve the designs and then to utilize a high detail model to validate and refine the solution [47]. Other attempts to reduce computation time are reported in the following section.

Shigeru Obayashi [21] has applied the single-objective EA and multiple-objective EA to optimization problems of supersonic wings. The aerodynamic optimization problem searched for an optimal supersonic wing shape using the Euler equations. The multidisciplinary optimization problem looked for an optimal supersonic wing platform shape using linearized aerodynamics and wing weight algebraic estimation. To couple the evolutionary approach with CFD code, the unstructured grid approach was employed. The unstructured grid generation was performed for each design candidate by using the dynamic mesh technique; the flow calculation was also accelerated by using the space marching technique. The designed wing was not practical since the design space is very limited, but the result indicates that such evolutionary computations could be used in airfoil design.

Jones [55] used the Message Passing Interface (MPI) to implement a manager/worker parallel evolutionary algorithm into the design of helicopter rotor airfoils. In this case, the GA operations included an advanced n-branch tournament selection, uniform crossover, and low probability mutation. The communication between manager and workers was via a dynamic load-balancing model. This model allowed the workers that had completed their evaluation to begin new tasks without waiting for slower workers to finish. Finally, the generated Pareto-optimal airfoil set was compared to the performance of a typical rotorcraft airfoil under identical flight conditions. The results showed that these designs appeared to be promising, non-traditional shapes for improved aerodynamic and aeroacoustic performance.

Makinen et al. [39] used a modified nondominated sorting genetic algorithm (NSGA) for two-dimensional airfoil design. The flow was modeled by thin-layer Navier-Stokes equations and time-harmonic Maxwell equations, and the convergence was accelerated using a multigrid algorithm. In his modified NSGA, the diversity of the population was preserved using the so-called tournament slot sharing method. The probability for the individual i to enter a tournament was given by

$$Sh(d_{ij}) = \begin{cases} 1-(\frac{d_{ij}}{\sigma_{share}})^2, d_{ij} < \sigma_{share} \\ 0 \end{cases} \tag{3.5}$$

$$P_i = \frac{1 \Big/ \sum_{j=1}^{n} Sh(d_{ij})}{\sum_{k=1}^{n} \left(1 \Big/ \sum_{j=1}^{n} sh(d_{kj}) \right)} \tag{3.6}$$

where the parameter n is the size of population, Sh() is the share function, and dij is the genotypic distance between the individual i and j. σ_{share} is the maximum sharing distance for a tournament slot.

Li and Satofuka [27] presented a new method for the multiobjective aerodynamic design of a compressor cascade. In their study, Bezier curves were used to represent the geometry of the cascade shape. A two-dimensional Navier-Stokes solver was used to evaluate the aerodynamic performance of design candidates. The idea of the two-branch tournament selection method for a two-objective optimization problem, which was presented by Crossley [56], was adopted to develop the two-branch Boltzman selection. The two-branch Boltzmann selection approach was organized so that designs compete on one of two objectives. One branch of the Boltzmann selection measures design on the first objective, and the second branch evaluated individuals on the second objective. This two-branch Boltzmann selection can select excellent individuals. The basic Boltzman selection is essentially a selection operator for controlling the selective pressure in the GA approach. The probability of individual selection in the Boltzman selection is obtained by

$$P_i = \frac{e^{f_j(x)/T}}{\sum_i e^{f_i(x)/T}} \tag{3.7}$$

where $f_i(x)$ is the fitness of the individual, T is the temperature, the numerator contains the Boltzmann weighting term, and the denominator is a normalization factor. The formulation of T is described by

$$T = \max\left[\left(1 - e^{-\left(1-k/(K+1)\right)^{\lambda}}\right)T_0, \delta\right] \tag{3.8}$$

where T_0 is the initial temperature, set to be 10 times the maximum fitness of the first generation.

Other approaches suggest implementing a neural network (NN) and/or a response surface method (RSM) to fit curves through the design space [45, 52, 53, 54]. This enables an approximate answer to be obtained for a design rather than having to call a CFD solver for a computationally expensive fitness evaluation. The combination of NNs and RSMs in these applications has been shown to significantly reduce the time for the optimization process. Three approaches have utilized neural networks. In the first, design of a sailing yacht, fin keel was optimized by combining a traditional EA with a recursive recall neural network [49]. In this technique, neural networks were used with a conjugate gradient optimization routine following the optimization with an EA. By using the information available from the EA process in a neural net to construct a global approximation of the fitness, the number of CFD calls by the conjugate gradient solver was minimized. In addition, the conjugate gradient solver reduced the number of calls to the EA by allowing the EA to establish the general location of the optima and then seek the answer more directly. In another approach, Fan [45] used a neural network to construct a model of the search space based on the potential flow around a turbine blade. This is similar to constructing a response surface. This model was then used in an EA to optimize the velocity profile on a turbine blade. In both of these cases, the neural network is used to predict the solution directly from the design inputs. Another approach utilizes an artificial neural network (ANN) with a feature weighted general regression neural network (GRNN) to develop a real-time estimate of the final fitness and error bounds for a thermal fluids system during each iteration of the CFD solver [50]. During each fitness evaluation, the CFD solver iteratively solves the fluid flow and heat transfer characteristics of the proposed design. Normally the stopping point for this process is based on the traditional convergence criteria for CFD analysis. By developing a real-time estimate of the fitness and error bounds at each iteration, the algorithm can determine when the fitness of the design is known with sufficient accuracy for the evolutionary process. This significantly reduces the number of iterations required.

3.4 Interactive/Steering Function in the Design Process

Most current optimization tools do not support interaction between designers and the design problem. Currently, numerical design optimization typically takes place in batch mode. The stereotypical role of the designer in these systems is to specify the problem including predefined constraints and control parameters, and then initiate a computer search to find an optimal solution. Using the CFD-based evolutionary algorithm as an example, a typical scenario might be as follows:

- The user provides EAs parameters sets (crossover rate, mutation rate, etc.) and object functions according to the specific problem at hand.
- The searching algorithm is then executed, often taking hours if not days or weeks to finish; during the execution there is little if any interaction with the system.
- Upon completion of the searching algorithm, the solutions are assessed, and any data output by the algorithm is graphed.
- Using this information, parameters may be altered, or more significant changes may be made to the algorithm, and then the algorithm is run again, hopefully with improved performance and results.

This kind of machine-based numerical optimization has significant utility within well-defined routines and detailed design domains. However, in practical applications, it is the solution suggested by the optimizer but not the actual details of the design that are most interesting. Anderson [99] pointed out that in many contexts interaction is more important than efficiency since the optimization algorithm is working with an impoverished objective function and the ability to successfully implement the solution depends on how well people understand and trust it. Users must understand and trust the generated solutions to make effective use of them. The real design process can also be quite complex; often the problem cannot be stated in a precise form for complete analysis, and there are uncertainties in the design data. In many instances, the formulations of the problem must be developed as part of the design process. Therefore, it is neither desirable nor useful to optimize an inexact problem from start to finish in a batch environment. Indeed, it would be a waste of time to find out at the end that wrong data was used or a constraint was inadvertently omitted. Obviously, a combination of batch processing and simple post-processing is inadequate in the real world.

In recent years research related to interactive optimization has increased. More and more researchers agree that users are more likely to understand a solution that they helped to create than one that is simply presented to them. Research shows that including humans "in-the-loop" during the design process can enhance optimization performance. Interactive evolutionary algorithms is an optimization approach where the EA optimizes target systems based on human evaluation. Simply speaking, this approach is an EA whose fitness function is a human user. Interactive evolutionary algorithms have been applied to several tasks in artistic, speech and image processing, hearing aid fitting, data mining, virtual reality and so on. However, most literature in this field focuses on extreme cases; for example, some interactive optimization algorithms require user guidance at every step. Without user instruction, such algorithms halt. Therefore, they are typically only considered in cases where it is hard or impossible to use numerical models to represent the problem such as works of art. Understandably, given the demands on user time imposed by algorithms of this kind, they are seldom used in the engineering design process.

Other researchers used computational steering technology to interactively control a computational process such as CFD simulation and optimization during execution. With computational steering, users are continuously provided with visual feedback about the state of their simulation and can change parameters on the fly. Therefore, designers can vary parameters to optimize their product. Computational steering was recognized as an important issue twenty years ago. There are several general computational steering environments that have been developed over the years such as VASE from the University of Illinois [130], SCIRun from the University of Uath [103], and CUMULVS from Oak Ridge National Laboratory [131]. Although the concept of computational steering is a good idea, the implementation of computational steering is very difficult in practice. It requires the user to have a high level of knowledge of the simulation, visualization, user interface, and data communication. Therefore, while these packages offer significant promise, they are generally time consuming to use and hard to adapt to meet actual project requirements. For example, to integrate an existing application into these systems, the application source code has to be manually annotated with program statements by the application developer.

3.5 Virtual Engineering

One of the reasons why human feedback is not integrated into the design optimization process is the difficulty of showing the designed product and its performance in an understandable and intuitive way. That is, designers cannot decide which design is superior by simply looking at the parameters. However, designers can observe the analysis result of a design and identify whether it is the result they wanted, given that the result is shown in a realistic and intuitive manner. By creating a realistic experience from a computed simulation and maintaining the visitor's focus within that experience, the full potential of human thought and skill can be brought to bear on the problem. This enables the user to focus entirely on the engineering problem, ensuring that complexities can be understood and that the full range of engineering solutions can be explored.

Virtual engineering is an emerging technology that has recently grown in popularity and is defined as a technology that integrates geometric models and related engineering tools such as analysis and simulation, optimization and decision making tools, etc. within a computer generated environment that facilitates multidisciplinary collaborative product realization [110]. Figure 3.5 illustrates the basic functionality of a virtual engineering system. Obviously, each component in the figure represents a very broad research topic and involves many difficult questions. Several research groups have worked to couple engineering analysis and virtual environments, enabling the user to perform engineering tasks from the virtual environment [57, 58, 59, 60]. One of the earliest applications was Boilermaker [57]. Boilermaker coupled together a computational model of an industrial furnace with a virtual environment to enable the user to make changes to the inlet nozzle configuration and then see the resulting furnace performance on the fly. This enabled the engineer to explore various furnace configurations on the fly and consider a number of variables to create the optimum design for a particular furnace. Another example of coupling engineering analysis and virtual environments is DN-Edit [59]. DN-Edit allows NURBS-based (Non-Uniform Rational B-Splines) surface geometry to be altered interactively. Virtual cursors allowed interaction with geometry surfaces, enabled surface points to be displaced, material to be added or removed and provided exact control of surface normals at specific points. The resulting surfaces are NURBS-based and can be exported to various CAD or analysis programs. The shaping of three-dimensional geometry in a virtual environment creates an intuitive process that bypasses the obstacles and limitations of a two-dimensional human-computer interface. In addition to those mentioned above, applications coupling virtual environments and engineering have been developed for assembly [58], hose routing, and design of three-dimensional mechanisms [59].

The common thread amongst all the current applications that couple virtual environments and engineering analysis is the need for rapid engineering analysis enabling interactive manipulation and interrogation of the engineering component or system. Generally, this rapid analysis is achieved through simplified analysis or low fidelity models. In many cases, the surrounding environment is shown as a three-dimensional graphic environment. A virtual engineering environment is one that provides a user-centered, first-person perspective enabling the user to interact with the engineered system in a natural way and providing the user with a wide range of accessible tools. This requires an engineering model that includes the geometry, physics, and any quantitative or qualitative data from the real system. The user should be able to walk through the operating system and observe how it works and how it responds to changes in design, operation, or any other engineering modification. Interaction within the virtual environment should provide an easily understood interface, appropriate to the user's technical background and expertise, that enables the user to explore and discover unexpected but critical details about the behavior of the system. Similarly, engineering tools and software should fit naturally into the environment and allow the user to maintain her or his focus on the engineering problem at hand. A key aim of virtual engineering is to engage the human capacity for complex evaluation. The key components of this environment include:

- User centered virtual reality visualization techniques—when presented in a familiar and natural interface, complex three-dimensional data becomes more understandable and usable, enhancing the understanding of the user [61]. Coupled with an appropriate expert (e.g., a design engineer, a plant engineer, or a construction manager), virtual reality can reduce design time for better solutions.
- Interactive analysis and engineering—today nearly all aspects of power plant simulation require extensive off-line setup, calculation, and iteration. The time required for each iteration can range from a day to several weeks. Tools for interactive collaborative engineering in which the engineer can establish a dynamic thinking process are needed to permit real-time exploration of "what-if" questions essential to the engineering process. In nearly all circumstances, an engineering answer now has much greater value than an answer a day, a week, or a month from now. Because of this, although many excellent engineering analysis techniques have been developed, they are not routinely used as a fundamental part of engineering design, operations, control, and maintenance. This is because the time required to set up, compute, understand the result, and repeat the process until an adequate answer is obtained significantly exceeds the time available. This includes techniques such as CFD, finite elements analysis (FEA), and optimization of complex systems. Instead, these engineering tools are used to provide limited insight to the problem, to sharpen an answer, or to understand what went wrong after a bad design and how to improve the results next time. This is particularly true of CFD analysis.
- Integration of real processes into the virtual environment—as noted earlier, engineering is more than analysis and design. A methodology for storage and rapid access to

engineering analyses, plant data, geometry, and all other qualitative and quantitative engineering data related to plant operation needs to be developed.

- Engineering decision support tools—optimization, cost analysis, scheduling, and knowledge based tools need to be integrated into the engineering processes.

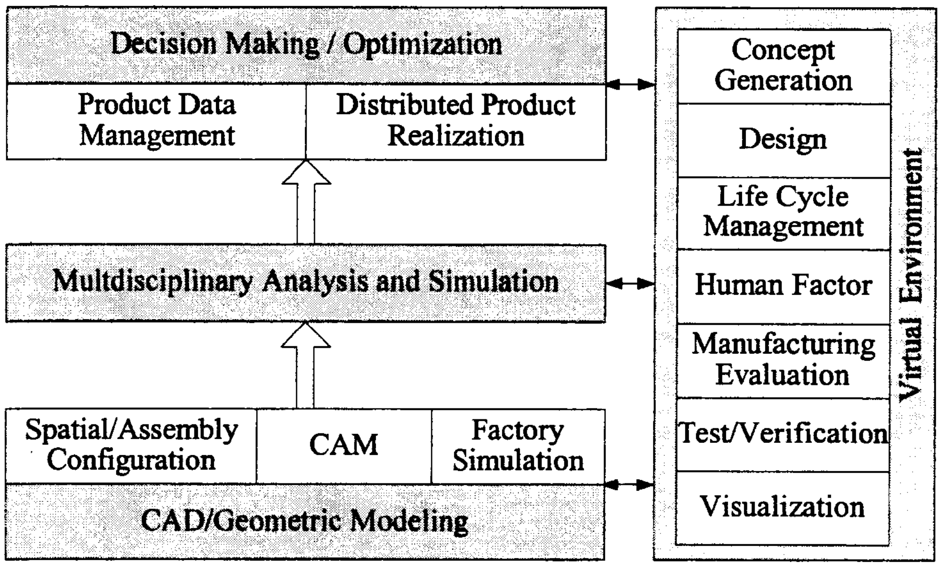


Figure 3.5 Framework of the virtual engineering system [110]

As previously noted, evidences from existing literature show that virtual engineering can be a powerful tool that could greatly enhance productivity. This begs the question, why isn't it routinely used? Simply put, today the implementation of virtual engineering is extremely challenging. Each of the four components has to be considered when developing a virtual engineering application. In other words, the implementation of virtual engineering requires knowledge of the application and visualization including virtual reality technology, user interfaces, data communication and his/or own application. In addition, end-users (e.g. design engineers), are generally not programming experts.

Virtual engineering applications can be constructed from scratch, but as with any construction task, applications can be constructed more easily and efficiently by integrating an existing application; the existing application could be automated, or at least a higher lever user interface to assist in the annotation of the application can be provided. The general virtual engineering environments that contain software tools should be able to provide higher level functionality on aspects common to arbitrary virtual engineering applications. The existing research on virtual engineering discussed above shows that the construction of the virtual engineering application is an underdeveloped aspect; existing systems can only support specific applications. In addition, existing systems are not tailored to the specific requirements of an effective virtual engineering application. These deficiencies led to the development of VE-Suite for a general purpose virtual engineering environment that fills the gap in existing systems. The details of VE-Suite are discussed in Chapter 6.

Chapter 4 Computational Model and Preliminary Results

As mentioned in Chapter 1, in order to achieve better coal pipe design, high fidelity solver is needed. CFD is attractive to the design community since it is more cost-effective than physical testing. Also, it is generally much better at predicting trends which is what design optimization requires. However, one must note that complex flow simulations are challenging and error-prone, and it takes significant engineering expertise to obtain valid solutions. In fact, the importance of good validation can not be overstated. Validation does not mean that the code has to give identical correct solutions on every possible geometry variation, but, the solver should behave consistently across a range of similar geometries and provide sufficiently accurate solutions to enable decision making. Therefore, to use CFD to design a coal transport system, one of the first steps is to ensure that an acceptable computational model can be built. The key steps in the process of building this model include choosing an analysis package, validating the results using known results, and extending the model to the case of interest.

4.1 Choosing an Analysis Package

There are several choices available for CFD analysis packages ranging from open source projects to proven commercial products. Open source CFD products have several advantages. They can provide a base platform to which an analyst can add special features that may be needed. Also, open source CFD products can often be tuned to provide faster answers than general purpose commercial CFD products. The disadvantage of open source products is that they usually lack commercial technical support and generally only support simple geometries that can be modeled using structured grids. Because of this, the industry generally chooses to use commercial CFD products. For our sample case, coal roping has successfully been modeled using CFX^{TM}, and the company sponsoring this research uses Fluent^{TM} as their corporate CFD analysis package. Based on this, Fluent^{TM} was chosen as the CFD analysis package used in this research.

Fluent^{TM} is a comprehensive CFD analysis tool. It can be used to model turbulence, combustion, and multiphase applications. Models and meshes are created using the preprocessor Gambit^{TM}. Fluent^{TM} also has post-processing capabilities for organizing and interpreting data and images. In the case of a dilute gas-solid flow behavior in complicated geometries, Fluent^{TM} allows the user to simulate a discrete second phase in a Lagrangian frame of reference. It uses an iterative technique to account for the coupling between the motion of the gas and particle phase based on the particle-source-in-cell method of Crowe et al. [63]. First, the gas flow field is calculated assuming no particles are present. This flow field is used to calculate particle trajectories; after this, momentum source terms for each cell throughout the flow field are determined. The gas flow field is solved again, incorporating the new particle trajectories that constitute the effect of the gas phase on the particles. The new gas flow field is used to establish new particle trajectories that constitute the effect of the gas phase on the particles. After calculating new source terms and incorporating them into the gas flow field, this iteration is continued until convergence is obtained.

4.2 Governing Equations and Turbulence Modeling

The conservation equation for mass and momentum are solved for the continuous phase of the flow in FLUENT^{TM}. For the gas phase, the continuity and momentum equations are given by

$$\frac{\partial \rho}{\partial t}+\frac{\partial}{\partial x_i}\left(\rho u_{gi}\right)=0 \tag{4.1}$$

$$\frac{\partial}{\partial t}\left(\rho u_{gi}\right)+\frac{\partial}{\partial x_i}\left(\rho u_{gi}u_{gj}\right)=\frac{\partial p}{\partial x_i}+\frac{\partial \tau_{ij}}{\partial x_i}+\rho g_i+F_i \tag{4.2}$$

Where u_{gi} is the local gas-phase velocity in the i_{th} direction in an inertial reference frame, p is the static pressure, τ_{ij} is the stress tensor, g_i is the gravitational acceleration, and F_i is the interface force between gas and solid phases (per unit volume in the i_{th} direction). The stress τ_{ij} is given by

$$\tau_{ij}=\left[\mu_g\left(\frac{\partial u_{gi}}{\partial x_j}+\frac{\partial u_{gj}}{\partial x_i}\right)\right]-\frac{2}{3}u_g\frac{\partial u_{gl}}{\partial x_l}\delta_{ij} \tag{4.3}$$

Where u_g is the gas-phase viscosity and the second term on the right side is the effect of volume dilation.

For the fluid turbulence model, the renormalization group (RNG) $k-\varepsilon$ turbulence model was used. Use of the RNG $k-\varepsilon$ model can yield significant improvements over the standard $k-\varepsilon$ model for recirculatory flows because of its better performance over the standard $k-\varepsilon$ model in predicting the streamline, radial velocity components and Reynolds shear stresses within a 90° circular pipe bend. The transport equations for turbulent kinetic energy κ and its dissipation rate ε in RNG $k-\varepsilon$ model have the same forms as in the standard $k-\varepsilon$ model with the exception of the additional quantities of the inverse effective Prandtl and numbers α_k and α_ε, and the R term in the ε equation.

$$\frac{\partial(\rho k)}{\partial t}+\frac{\partial}{\partial x_i}(\rho u_i k)=\frac{\partial}{\partial x_i}\left(\alpha_k \mu_{eff}\frac{\partial k}{\partial x_j}\right)+G_k+G_b-\rho\varepsilon-Y_M+S_k \tag{4.4}$$

$$\frac{\partial}{\partial t}(\rho\varepsilon)+\frac{\partial}{\partial x_i}(\rho\varepsilon u_i)=\frac{\partial}{\partial x_j}\left(\alpha_\varepsilon \mu_{eff}\frac{\partial \varepsilon}{\partial x_j}\right)+C_{1\varepsilon}\frac{\varepsilon}{k}(G_k+C_{3\varepsilon}G_b)-C_{2\varepsilon}\rho\frac{\varepsilon^2}{k}-R_\varepsilon+S_\varepsilon \tag{4.5}$$

In these equations, G_k represents the generation of turbulence kinetic energy due to the mean velocity gradients. When using the Boussinesq hypothesis, it can be calculated as

$$G_k=\mu_t S^2 \tag{4.6}$$

G_b is the generation of turbulence kinetic energy due to buoyancy. When dealing with the ideal gas, it can be calculated as

$$G_b=-g_i\frac{\mu_t}{\rho \Pr_t}\frac{\partial\rho}{\partial x_i} \tag{4.7}$$

Y_M represents the contribution of the fluctuating dilatation in compressible turbulence to the overall dissipation rate. It can be calculated as

$$Y_M=2\rho\varepsilon M_t^2 \tag{4.8}$$

Where M^t is the turbulent Mach number, defined as

$$M_t=\sqrt{\frac{k}{a^2}} \tag{4.9}$$

The quantities α_k and α_ε are the inverse effective Prandtl numbers for k and ε, respectively. They are computed using the following formula derived analytically by the RNG theory:

$$\left|\frac{\alpha-1.3929}{\alpha_0-1.3929}\right|^{0.6321}\left|\frac{\alpha+2.3929}{\alpha_0+2.3929}\right|^{0.3679}=\frac{\mu_{mol}}{\mu_{eff}} \tag{4.10}$$

Where $\alpha_0 = 1.0$. In the high-Reynolds-number limit $(\mu_{mol}/\mu_{eff} \langle\langle 1)$, $\alpha_k = \alpha_\varepsilon \approx 1.393$

The main difference between the RNG and standard $k-\varepsilon$ models lies in the additional term in the ε equation given by

$$R_\varepsilon = \frac{C_\mu \rho \eta^3 (1-\eta/\eta_0)}{1+\beta\eta^3}\frac{\varepsilon^2}{k} \tag{4.11}$$

Where $\eta \equiv Sk/\varepsilon$, $\eta_0 = 4.38, \beta = 0.012$

Among the above equations, S is the modulus of the mean rate-of-strain tensor, defined as

$$S \equiv \sqrt{2S_{ij}S_{ij}} \tag{4.12}$$

μ_t is the turbulent viscosity which is computed by combining k and ε as

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon} \tag{4.13}$$

with $C_\mu = 0.085$

The model constants $C_{1\varepsilon} = 1.44, C_{2\varepsilon} = 1.92, S_k$ and S_ε are user-defined source terms.

In addition to solving the transport equations for the continuous phase, the trajectories of particles in a discrete second phase (dispersed phase) are computed in a Lagrangian reference frame. Coupling of the two phases is included, where the particle trajectories and the continuous phase flow affect each other. The trajectory of a discrete-phase particle is predicted by integrating the force balance on the particle, equating the particle inertia with the forces acting on the particle

$$\frac{du_{pi}}{dt}=\frac{18\mu_g}{\rho_p D_p^2}\frac{C_D \mathrm{Re}}{24}\left(u_{gi}-u_{pi}\right)+g_i\frac{\left(\rho_p-\rho\right)}{\rho_p}+F_{xi} \tag{4.14}$$

Here, u_{gi} is the gas-phase velocity, u_{pi} is the particle velocity, u_g is the viscosity of the gas, ρ is the gas density, ρ_p is the density of the particle, and D_p is the particle diameter. Re is the relative particle. Eq.4.14 describes the balance of forces acting on the particle. The term on the left hand side is the inertia force acting on the particle due to its acceleration, and the right-hand side terms are the external forces acting on the particle. $18u_g / \rho_p D_p^2 \left(C_D \mathrm{Re}\right)/24\left(u_{gi}-u_{pi}\right)$ is the drag force (in the ith direction) per unit particle mass and the second term $g_i(\rho_p-\rho)/\rho_p$ is the buoyancy force due to gravitational acceleration, and F_x represents the other external forces, for example, "virtual mass" force, Basset force, which can play important roles in calculating under special circumstances.

Despite the readily available code, the selection of appropriate boundary conditions is critical in obtaining meaningful information in the case of the particle flow. In particular, Huber and Sommerfeld [10] demonstrated the importance of simulating particle-wall and particle-particle interactions and the dominant effect that this has on the particle stream. These effects are beyond the scope of this work. In Fluent, no particle-particle interaction is available.

4.3 Validation

The overall objective of validation is to demonstrate the accuracy of CFD code so that it can be used with confidence for fluid simulation and so that the simulation results can be considered credible for decision making in design. Validation is defined as "the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model" [35]. A good tutorial on CFD code validation is presented on the NASA Verification and Validation website. The most common way to validate the accuracy of a CFD simulation is to compare CFD results to available experimental data. In this study, the tests of the validity of the Fluent™ DPM (discrete particle model) model include:

- compare simulation results with the experimental results from existing literature;
- compare baseline simulation results with the plant's measurements.

4.3.1 Comparison of CFD Results with Experimental Data

Yilmaz and Levy [16] carried out a series of experimental studies to understand the formation and dispersion characteristics of ropes in a pneumatic conveying system. The majority of the experiments were performed in the flow geometry, which consisted of a horizontal pipe with a length of 5 pipe diameters, the elbow section, and a vertical pipe with a length of 20 pipe diameters (See Figure 4.1). The elbow is a 90° elbow and the radius of the elbow is three times the pipe diameter. Table 4.1 shows the dimensions of this facility. The experiment facility was operated with gas velocities from 15 to 30m/s. The ratio of solids to air mass flow rates were from 0.33 to 1.0. Velocity was determined from two closely streamwise positioned fiber optic probes using continuous wave laser light and the concentration was determined by the obscuration of laser light over small distance by comparison with calibration data. The data obtained shows the overall pattern of the rope. The time-varying nature of the rope was illustrated by the concentration measurements at various radial locations;

Table 4.1 Geometry sizes of the testing facility

Pipe Diameter (m)	Horizontal pipe length (m)	Vertical pipe length (m)	Elbow Radius (m)
0.154	0.77	3.08	0.462

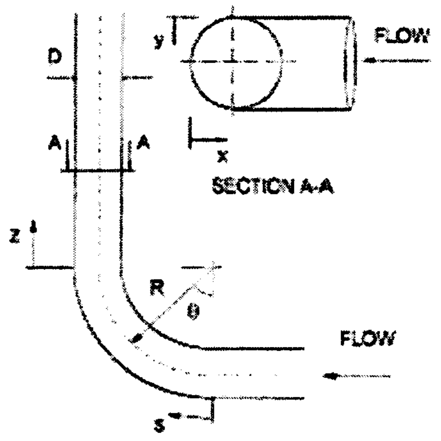


Figure 4.1 Sketch of the testing flow geometry [6]

Validation of the CFD model to ensure the accuracy of its application in gas-solid flows is first completed using experimental results of Yilmaz and Levy [6], by comparing both quantitative and qualitative trends. In the CFD simulation, the gas and particle flow through this experimental geometry was modeled numerically using the grid shown in Figure 4.2. The model is three-dimensional with 108,216 computational cells. The mesh was generated using Gambit™, the preprocessing software companion of Fluent™.

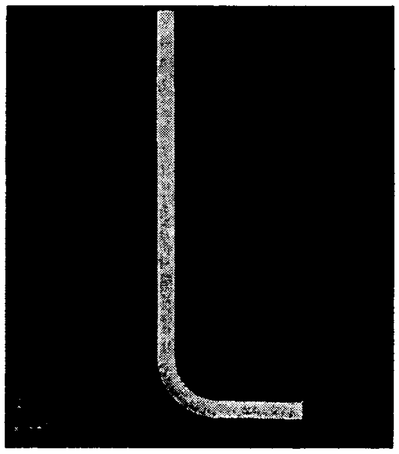


Figure 4.2 Grid layout for the testing model

Turbulence quantities were calculated using the following relations [6]:

$$K_{inlet} = \frac{3}{2}\left(\frac{U_{inlet}}{10}\right)^2 \tag{4.15}$$

$$\varepsilon_{inlet} = \frac{K_{inlet}^{3/2}}{0.3D} \tag{4.16}$$

The velocity at the inlet was defined using a "velocity-inlet" boundary condition. Therefore, the stagnation pressure of the flow was not fixed at the inlet, and the code changed it according to the value needed to obtain the prescribed velocity distribution. An "outflow" boundary condition was used to model the flow at the outlet since the details of the flow velocity and pressure were not known before the flow problem was solved.

The RNG theory cannot be extended to viscosity dominated flow regions like the viscous sublayer of a turbulent boundary layer. Fluent™ implements the wall function approach to bridge the viscous sublayer; such a technique permits skipping the detailed solution of the flow field inside the boundary layer. It connects the potential field (outside the boundary layer) to the wall with statistical laws coming from the turbulent boundary layer analysis. In this case, the boundary layer was treated using the standard wall function approach.

The pressure-velocity-coupling algorithm SIMPLEC introduces pressure into the continuity Eq.4.1 and can achieve quick convergence in problems where pressure-velocity coupling is the main deterrent to obtaining a solution. The second-order upwind discretization scheme was used for the momentum, turbulent kinetic energy, and dissipation rate equations.

The Rosin-Rammler Diameter Distribution method was used to calculate the particle size distribution in the simulation. The distribution function can be written as

$$M_D = \exp\left(\frac{-D}{\overline{D}}\right)^n \tag{4.17}$$

Where M_D is the mass fraction of the particles with diameter greater than D and $\overline{D}$ is the mean diameter, with n representing the spread of the data from the mean. Table 4.2 shows the particle size distribution of pulverized coal in the experiment. Table 4.3 shows how this was implemented in the CFD codes.

Table 4.2 Particle size distribution of pulverized coal

Diameter (μm)	Weight (%)
>125	1.5
106-125	11.0
90-106	17.9
75-90	16.7
63-75	13.1
45-63	20.1
<45	19.7

The required grid resolution was established by increasing the grid until the details of interest are independent of the grid. In this case, the amount of grid in the piping cross-section is the critical determinant to obtaining accuracy. Figure 4.3 shows the final grid (with 108216 cells) that was chosen. Figure 4.4 shows the grid with the next highest resolution (with 121576 cells). Figure 4.5 compares the CFD results of these two different grids, showing the gas velocity profiles in the x direction at different z/D distances. The absolute values of velocity magnitudes at the different distances did not vary significantly for different grid sizes. The comparison shows that the two grids have a very close velocity profile, indicating the acceptability of the grid.

Table 4.3 Particle Injection Properties

Particle mean diameter (μm)	75
Min diameter (μm)	25
Max diameter (μm)	150
Spare number	2.33
Spread number	7
Density (kg/m^3)	1680
Mass flow rate (m/s)	0.22056

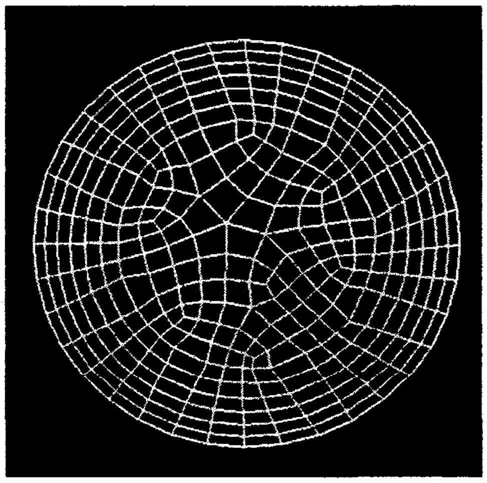


Figure 4.3 Final grid for pipe cross-section (with 108216 cells)

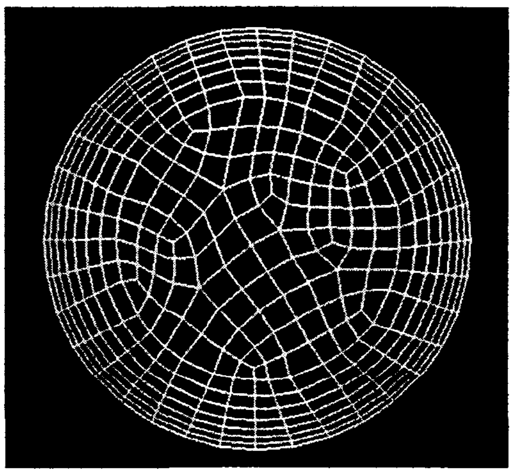
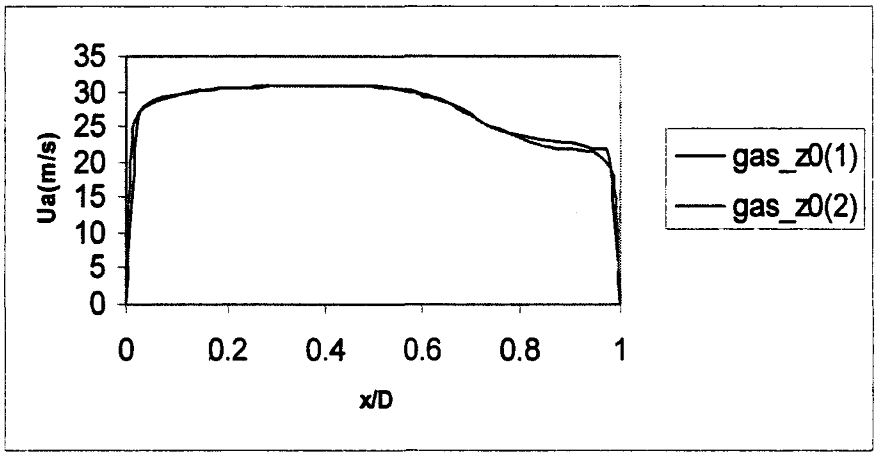
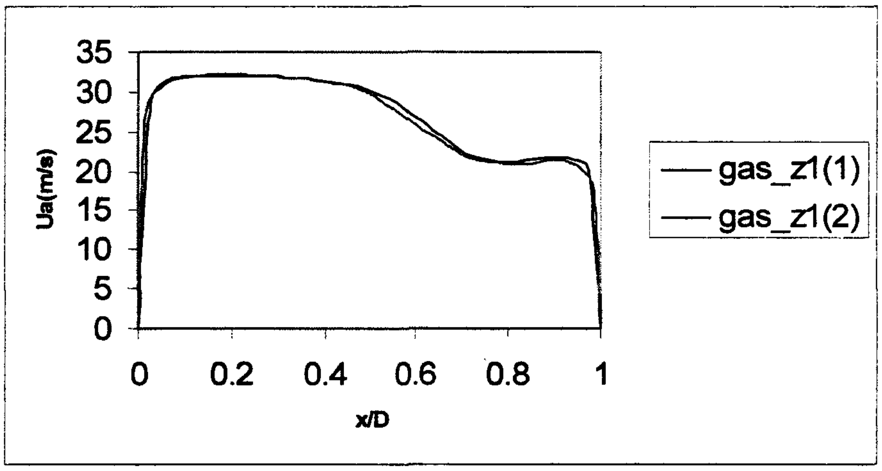
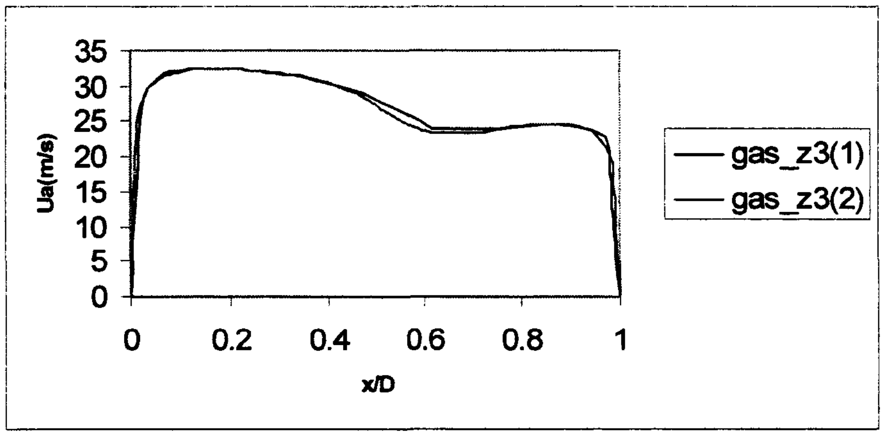
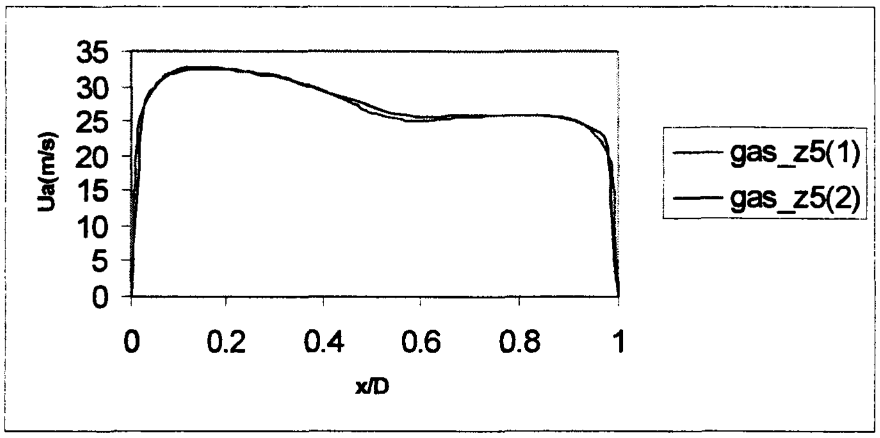


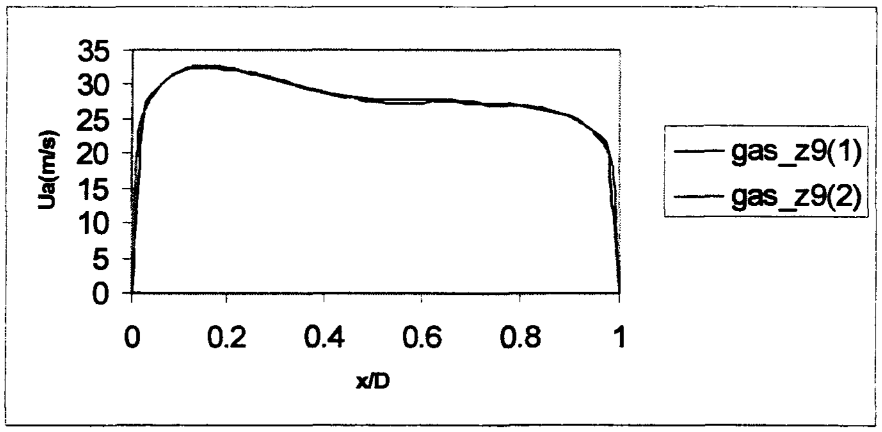
Figure 4.4 The compared grid for pipe cross-section (with 121576 cells)

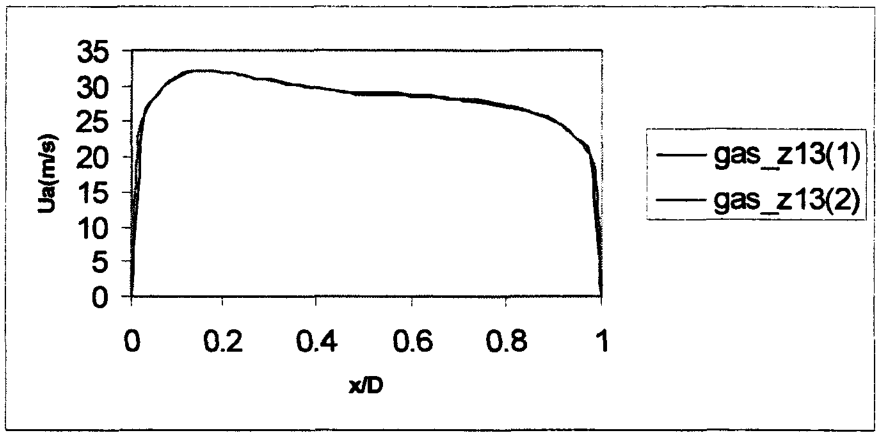

Ua(m/s)
35
30
25
20
15
10
5
0
0
0.2
0.4
0.6
0.8
1
x/D
gas_z0(1)
gas_z0(2)


Ua(m/s)
35
30
25
20
15
10
5
0
0
0.2
0.4
0.6
0.8
1
x/D
gas_z1(1)
gas_z1(2)


Ua(m/s)
35
30
25
20
15
10
5
0
0
0.2
0.4
0.6
0.8
1
x/D
gas_z3(1)
gas_z3(2)


Ua(m/s)
35
30
25
20
15
10
5
0
0
0.2
0.4
0.6
0.8
1
x/D
gas_z5(1)
gas_z5(2)


Ua(m/s)
35
30
25
20
15
10
5
0
0
0.2
0.4
0.6
0.8
1
x/D
gas_z9(1)
gas_z9(2)


Ua(m/s)
35
30
25
20
15
10
5
0
0
0.2
0.4
0.6
0.8
1
x/D
gas_z13(1)
gas_z13(2)

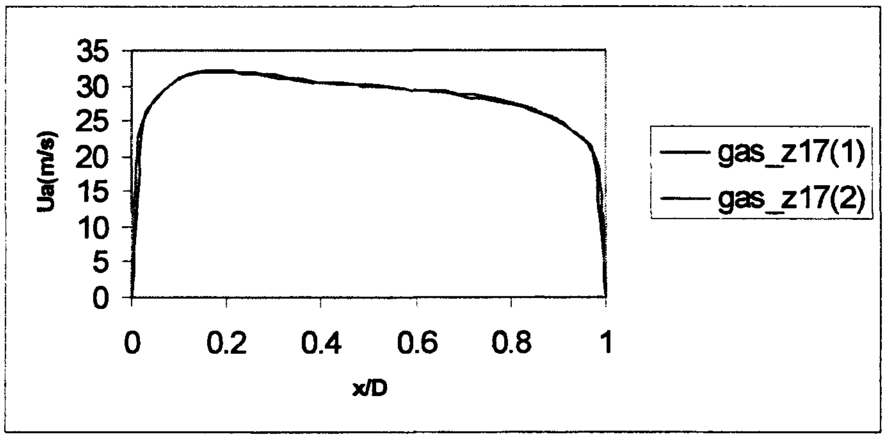
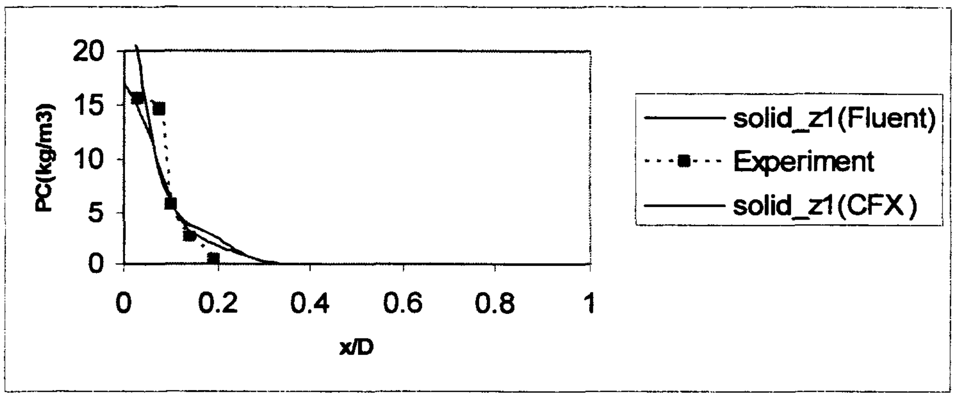


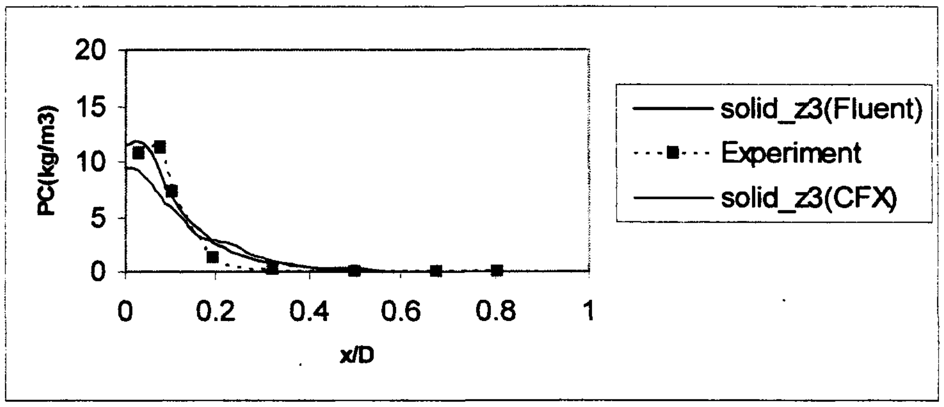
Figure 4.5 Comparison of CFD results with two different grids

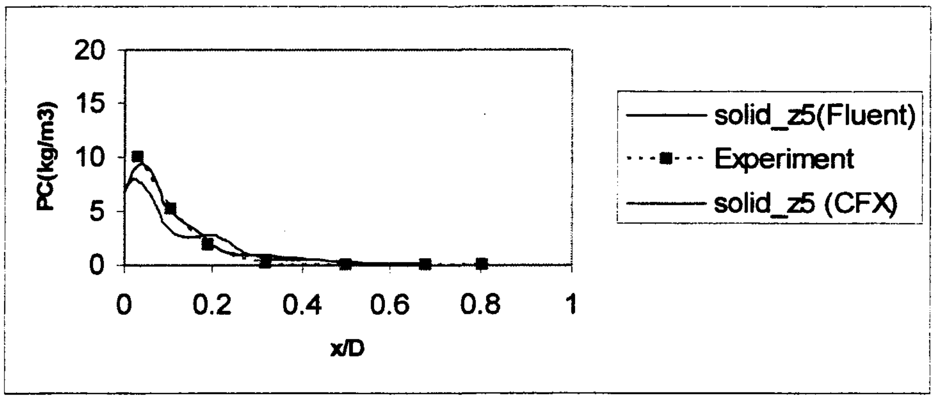
(Note: In the above graphs, gas_z represents the gas phase velocity at z/D=*, for example, gas_z17 represents the data that was picked from the pipe cross-section at z/D=17)*

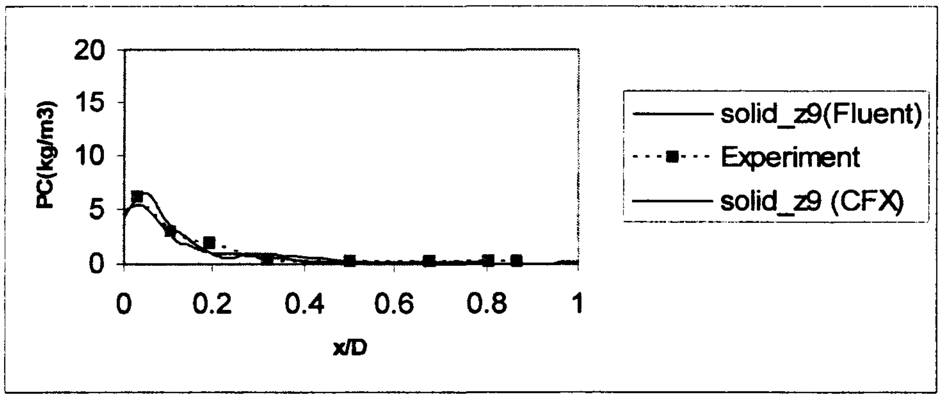
After a fully converged continuous phase flow field was attained, particles were added to the main stream. After several tests, it was decided to use surface injection in this project. Using this injection method, one particle is released for each cell in the injection surface. Every particle carries the same solid mass flow rate. The initial particle velocities were set to the average conveying gas velocity. Five stochastic attempts were used to simulate turbulent dispersion of the particles, imposing random effects on the turbulence quantities calculated in the fluid medium. Figure 4.6 shows the particle concentration profiles along the pipe diameter in the x direction at different z/D distances. In order to compare with other people's work, Yilmaz's [16] simulation results using CFX^{TM} are also listed in Figure 4.6 as a green line. In most cases, Yilmaz's simulated particle concentrations show a greater departure from the experimental data than the computational results obtained in this project, especially near the wall. The values of particle concentration between the simulations and experiments are in good agreement at a short distance from the bend (z/D=1, z/D=3) and the values at distances further from the bend exit (z/D =9 and z/D=13) compared very well. As Yilmaz pointed out the dispersion of ropes is caused by the secondary flow. As shown in Figure 4.7, two main vortices are present; this correlates well to the two main vortices calculated by CFX™ from Yilmaz and Levy. Low fluid velocities near the outer wall reveal two minor vortices distinct from the two main vortices due to the presence of a solid phase.

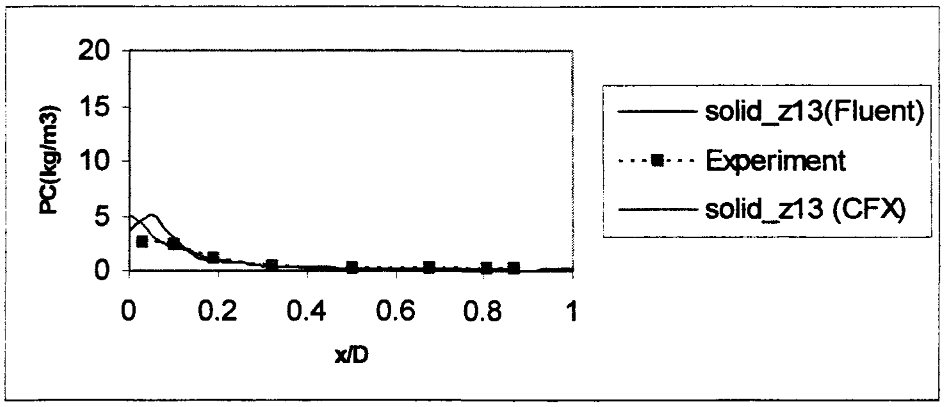
The comparison between the simulation results and experimental data shows that we are able to capture sufficient flow detail to provide reasonable results for the case examined here. The simulation data predicts slower dispersion of the coal rope than was observed in the laboratory. Considering the complexities of the flow phenomena, the comparison showed good agreement between the experimental data and simulation. The primary impact of the faster predicted dispersion of the rope is that actual coal roping is likely to be better than predicted coal roping. This likely occurs because the simulation ignores particle-particle interactions and the particle-wall collision is too simple. Based on this, we can use Fluent™ for our engineering model provided we exercise good engineering judgment when using the simulation results. Subsequently, CFD simulation was applied to analyze the real flow field in a real coal transporting pipe.




PC(kg/m3)
x/D
0
5
10
15
20
0
0.2
0.4
0.6
0.8
1
solid_z3(Fluent)
Experiment
solid_z3(CFX)


PC(kg/m3)
x/D
0
5
10
15
20
0
0.2
0.4
0.6
0.8
1
solid_z5(Fluent)
Experiment
solid_z5 (CFX)


PC(kg/m3)
x/D
0
5
10
15
20
0
0.2
0.4
0.6
0.8
1
solid_z9(Fluent)
Experiment
solid_z9 (CFX)



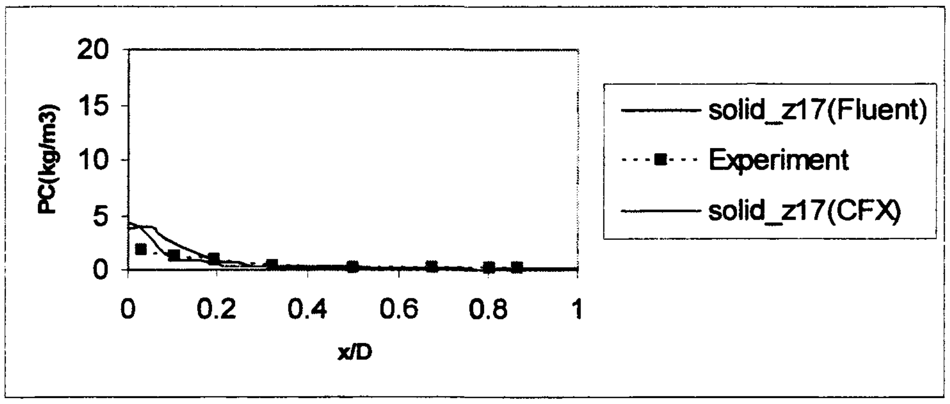


Figure 4.6 Comparison of CFD predictions and experimental data

(*Note: In the above graphs, solid_z* represents the solid distribution concentration at z/D=*, for example, solid_z17 represents the data that was picked from the pipe cross-section at z/D=17.*)

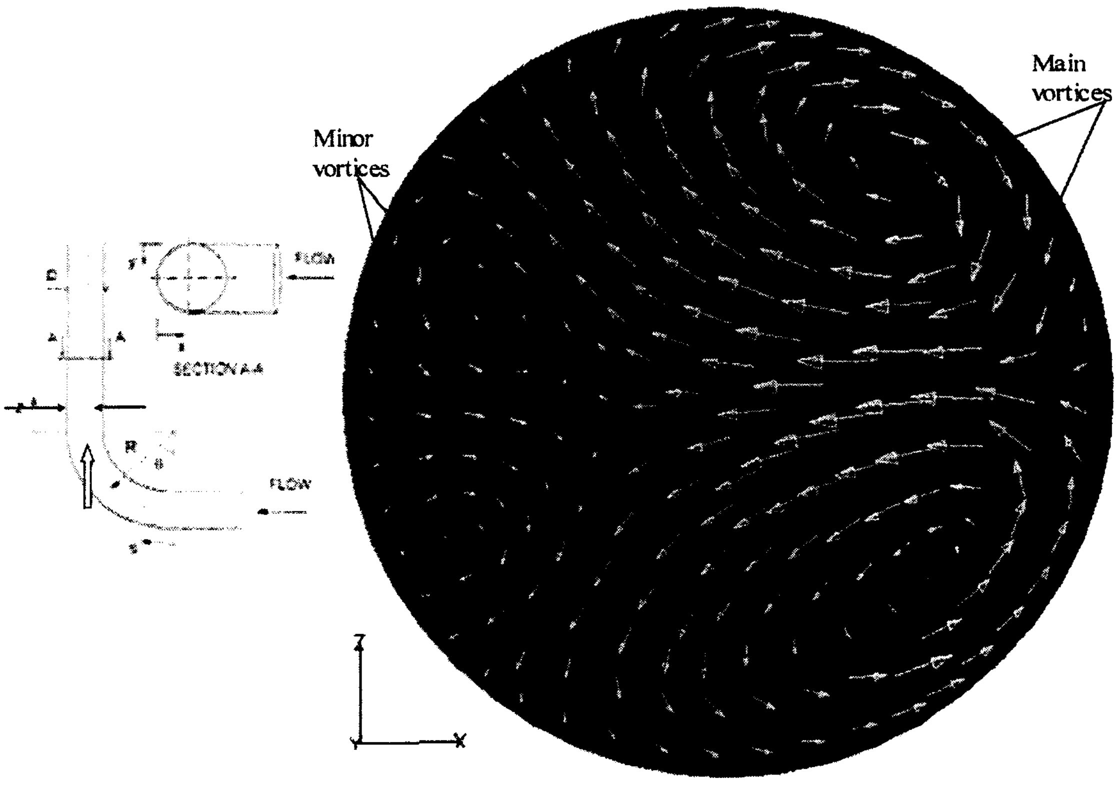


Figure 4.7 Secondary flow pattern in two phase flow for R/D=1.5, Y/D=0.5

4.3.2 Comparison Baseline Simulation Results with Plant Measurements

Although the coal rope phenomenon has been experimentally measured in simplified geometries, no detailed experimental data for the complex piping system accumulated in an actual power plant is available. Based on this, an initial baseline simulation was also performed to model the current operating condition and to verify the accuracy of the model. The model uses air velocities and solids loading ratios (the ratio of the solid and gas mass flow rate) typical of those in utility power plants. The main variables for simulation are chosen as described in Section 4.2.1. The RNG $k-\varepsilon$, along with Lagrangian particle tracking, is used to model the gas-particle flows through the testing pipe. Turbulence quantities were calculated using Eqs. 4.5 and 4.6. The diameters of the computational particles at each starting location were stochastically sampled from a Rosin Rammler distribution function. Table 4.4 shows the particle properties and Table 4.5 shows the flow parameters used in the simulation.

Table 4.4 Particle injection properties

Particle mean diameter (μm)	20
Min diameter (μm)	1.7
Max diameter (μm)	230
Spare number	1.4064
Spread number	10
Density (kg/m^3)	1680
Mass flow rate (m/s)	25.94

Table 4.5 Flow parameters of the simulation

Inside diameter (in)	26
Area of coal pipe (ft^2)	3.687
Temperature at exit of coal mill (F)	125
Density of primary air (lb/ft^3)	0.067986
Viscosity (lbm/ft-hr)	1.324444
Coal flow rate (lbm/s)	10.05787
Primary air to coal ratio	2
Primary air flow per pipe (lbm/s)	20.11574
Primary air velocity (ft/s)	85.09

4.3.2.1 Bifurcator

In order to build this baseline model, a model of a rope splitter, which is used in the baseline pipe, has to be built first. The device of interest is a bifurcator which is used in large coal-fired power plants. This bifurcator is used in the very complex pipework between coal mills and burners that is necessary to ensure a uniformly distributed supply of the burners with pulverized fuel from the coal mills. For the purpose of dispelling coal ropes the bifurcator model contains a very complex structure; it consists of a system of 78 different inclined channels that are directly attached to a system of vanes that lead the flux alternative to both legs of the bifurcator. Figure 4.8 shows the structure of the bifurcator in a pipe. If a rope meets the riffle box, it is dispelled by the checkerboard like system of channels in several parts that are then distributed to both legs because adjacent channels always lead to different legs. The construction of a numerical grid is challenging because of the complex structure of the fixtures. First, all the 78 channels inclined against each other are a serious problem especially for a structured grid that consists of hexahedrons, so the grid in this region has to split into different branches that pass either a left inclined or a right inclined channel. To achieve this, a thin layer between the sets of left and right inclined channels that does not belong to the gridded area forms a thin wall of finite thickness. After passing the channels the grid unites again and must map to the guiding vanes that alternately lead to the left and the right leg of the bifurcator and form a triangular structure, which is also difficult to grid with hexahedrons. The position of the guiding vanes marks the block boundaries in the interior of the triangle which causes another difficulty.

Grid-independence tests were conducted in this bifurcator first. Some of the results are shown in Table 4.6 and Table 4.7. The bottom part of Figure 4.9 shows the gas velocity magnitude on the testing line. From these comparisons, grid3 and grid4 show a very close result. The results did not vary by more than 3% when the grid size was varied by as much as 50%.

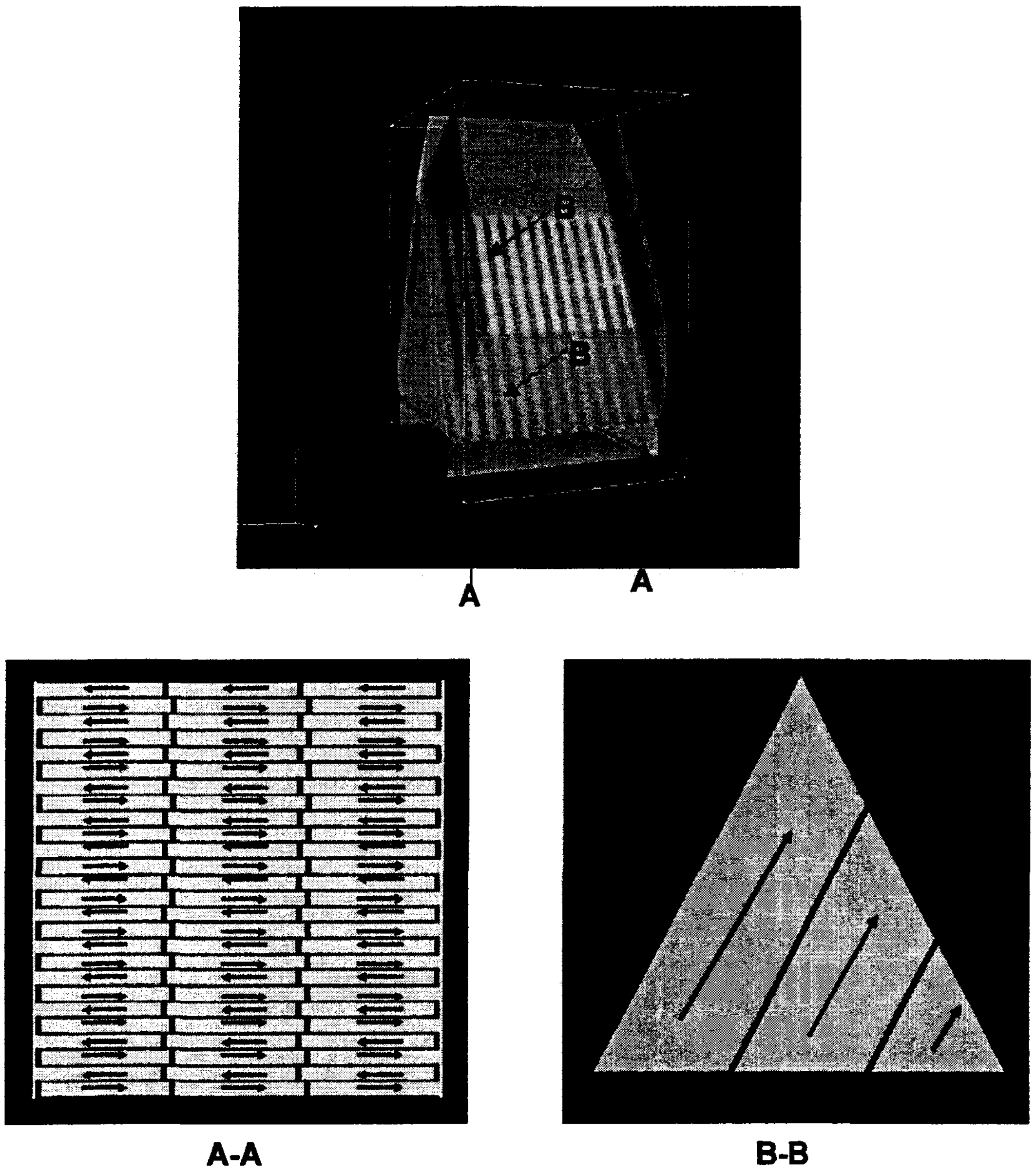


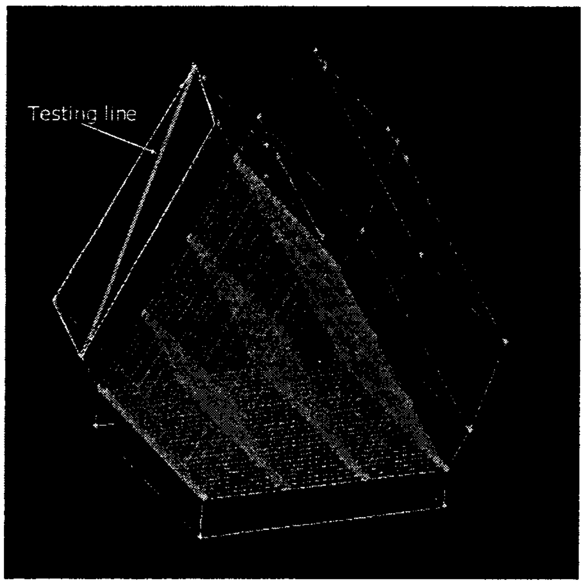
Figure 4.8 Bifurcator structure

Table 4.6 Numerical grid resolution for bifurcator

	Cell Number	Node number
Grid 1	284,298	326,051
Grid 2	413,750	476,422
Grid 3	569,647	656,337
Grid 4	767,423	882,495

Table 4.7 Comparison among four different grids

	Leftout (lbm/s)	Rightout (lbm/s)	LeftAverage (ft/s)	RightAverage(ft/s)
Grid 1	10.015812	10.094857	35.313	34.547
Grid 2	9.981	10.132823	34.542	33.600
Grid 3	10.0207	10.09333	34.694	33.772
Grid 4	10.011785	10.102689	34.389	33.558



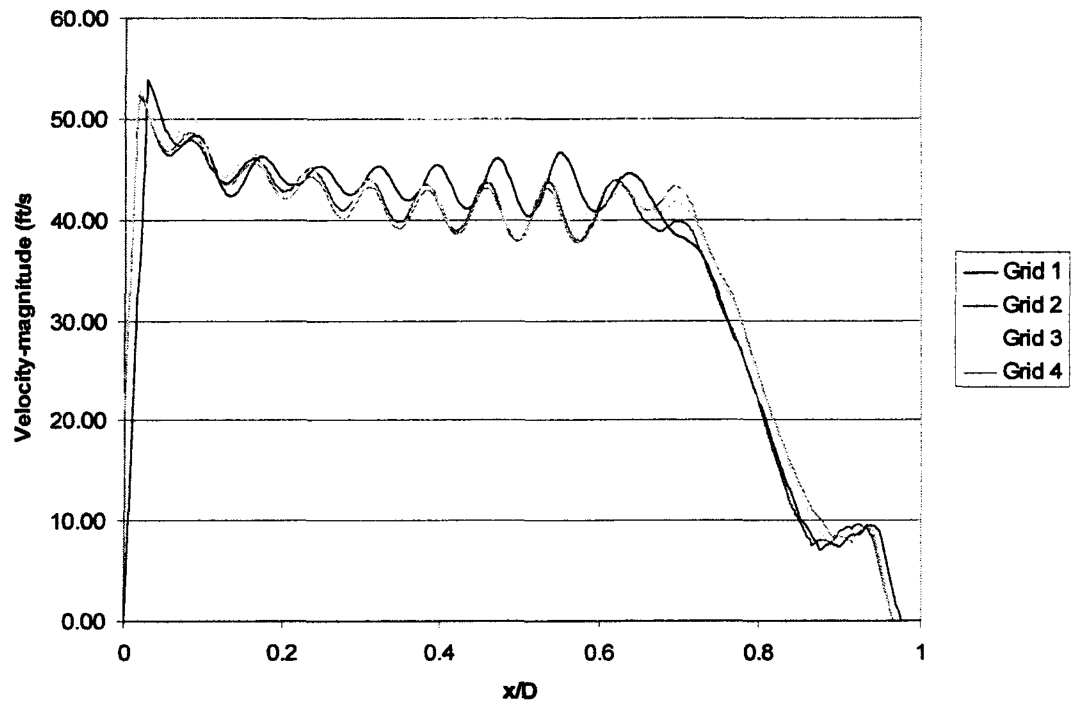


Figure 4.9 Fluid velocity (single phase calculation) in the diagonal line at the leftout

4.3.2.2 Baseline Pipe Validation

The baseline pipe chosen in this study is an actual pipe used in a power plant to transport pulverized coal from the mills to the burner. Figure 4.10 and 4.11 shows the operating pipework of the baseline model. Pipe material is steel walled pipe; the total pipe height is 37 ft, with nominal inlet pipe diameter 26 inches. The bend radius to pipe diameter ratio is 7.85 for two bends before the bifurcator. Immediately downstream of the splitter the two branches are 20 inches in diameter and the bend radius to this pipe diameter is 7.2. Between the mill and the splitter, there is an orifice line to control the airflow rate in order to balance the flow rate among the four pipelines that come out of the same mill. The left branch after the splitter has an extra orifice in order to balance the airflow rate between the two branches. Bend 1 is fitted to the flow direction of vertical upward to a horizontal straight. Bend 2 is for the flow direction of horizontal-to-vertical downward, and Bends 3 and 4 are vertical-to-horizontal after the splitter. Bend 5 features a direction of flow from horizontal to horizontal, and bends 6 and 7 are horizontal-to-vertical upward. Bends 8 and 9 are vertical-to-horizontal. The pipe and bend portions were modeled numerically using the grid shown in Figure 4.12, which is similar to the grid used for comparison with the experiment data (Section 4.2.1).

Figure 4.13 gives an impression of the erosion in the entrance region of the bifurcator. The results from the pipe simulation were compared to the areas of high damage from coal particle erosion in the plant. Here, red regions represent a maximum of particle concentration. The particle concentration of an area increases as the color of the area approaches red. The circled areas are those with higher particle concentrations. As shown, the areas with significant erosion match those areas identified by the simulation as having high particle concentrations. Thus, the numerical simulation is in very strong agreement with field-observed zones of high erosion. Based on this, the computational model is capable of simulating the real coal pipe in the field condition and can be used for the design or modification of a coal transport piping system with confidence.

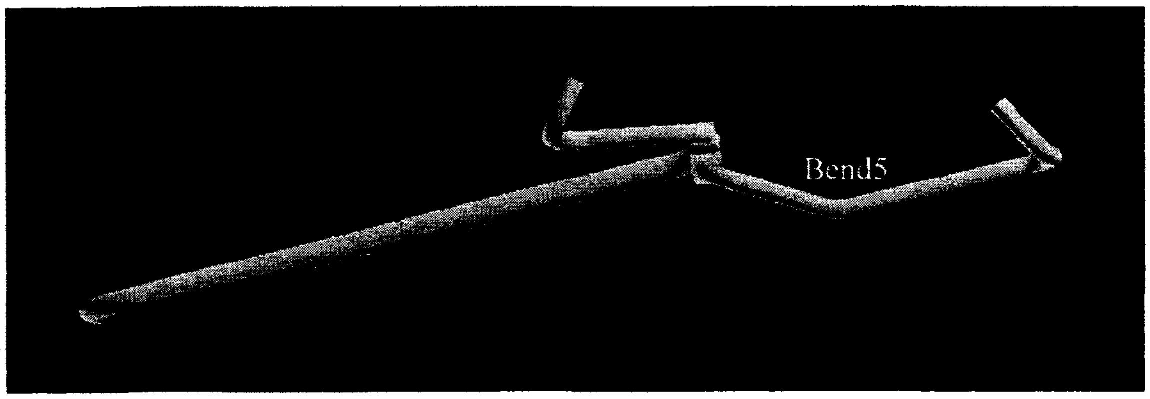


Figure 4.10 The schematic of the baseline pipe (top view)

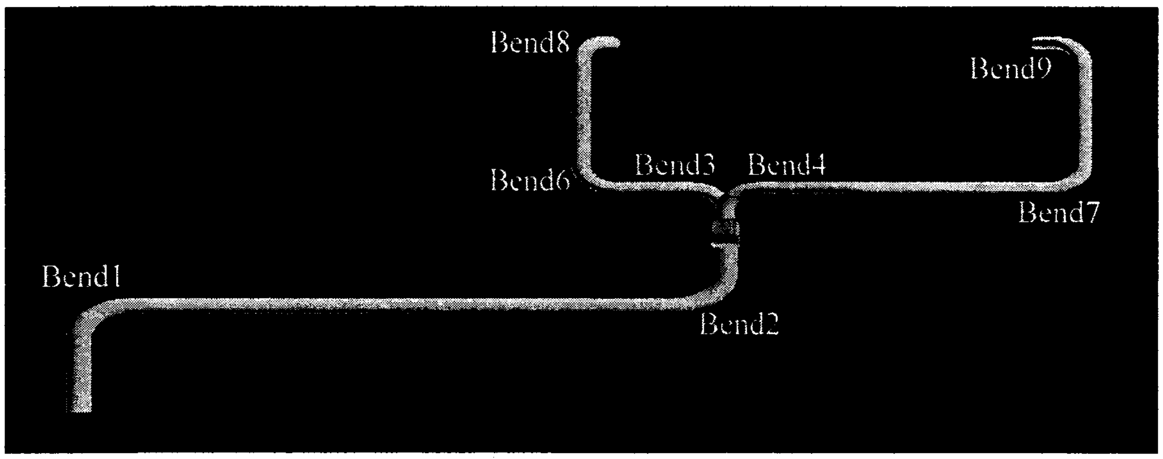


Figure 4.11 The schematic of the baseline pipe (front view)

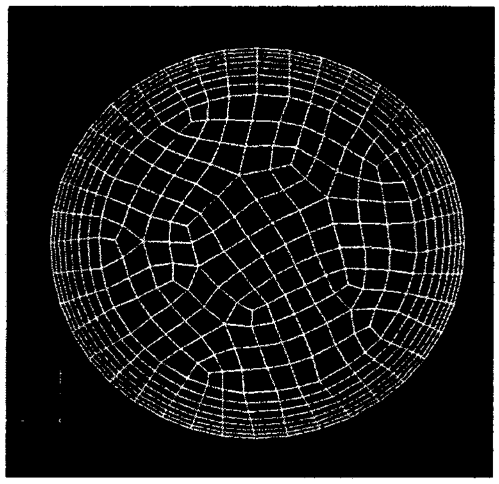


Figure 4.12 Grid layout on the cross-section of the baseline pipe

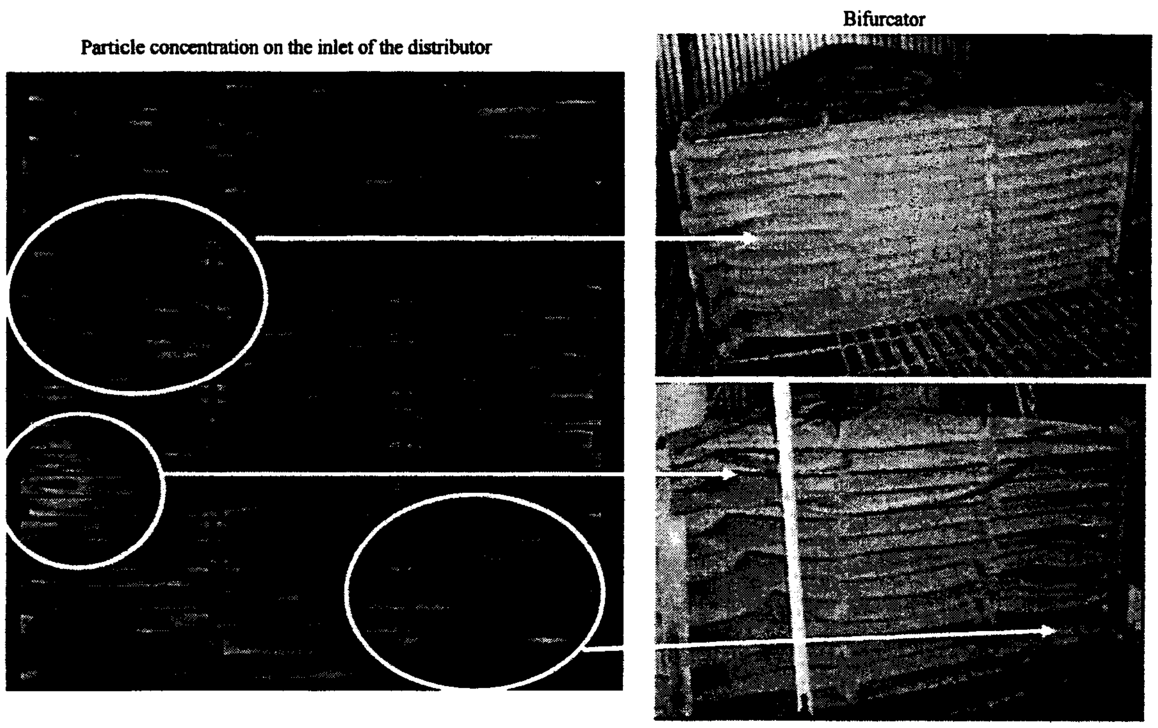


Figure 4.13 Comparison between actual erosion holes and simulation data

4.3 Baseline Simulation Results and Discussion

The computations were performed on a PC with dual Intel Xeon™ processors running Red Hat Enterprise Linux Workstation 3. The CFD calculation revealed the air and particle flow patterns in the pipe. The predictions start with a uniformly distributed inlet velocity of 24.5m/s.

The corresponding particle distribution in the entire pipeline is shown in Figure 4.14. As shown in these figures, the slow moving stream of particles against the wall contains most of the mass of the powder delivered. Each bend in the circuit clearly affects the particulate; subsequent to each bend, secondary flow structures cause uneven distribution of particulate concentration in the cross-section. In significantly long horizontal sections which is downstream from a vertical-to-horizontal bend, the rope was symmetrically positioned and was located at the wall downstream of the outer radius of the elbow as Yilamz [16] pointed out. The single-elbow configuration produces a double vortex which is symmetrically positioned within the pipe cross-section and is aligned with the axis of the inlet pipe. Secondary motion and flow turbulence created by the bend causes the rope to move towards the center of the pipe as it flowed downstream from the bend exit. Gravity also causes the concentrated particulate to fall. Therefore, before the rope hits the orifice, most of the particles move towards the bottom of the pipe. Due to the bigger mixing mechanism caused by the orifice installed in the horizontal pipe, particles are carried into the main stream and most particles are concentrated in the center part of the pipe cross-section. When the two-phase flow enters the second bend, the similar coal roping structure is created again. However, the typical spiraling of the particle stream near the wall is not observed after the second bend because of the orifice. Figure 4.15 and Figure 4.16 show the particle distributions in the section upstream of the bifurcator.

As previously discussed in the above section, the splitting of the particle-stream due to the bifurcator can be observed especially on the entry region where three areas of higher erosion are to be observed. Uneven particle distribution in the bifurcator leads to different impact flux of particles, which is a strong influence factor on erosion. Due to the fact that the bifurcator is designed for the destruction of particle ropes, it is extremely susceptible to particle erosion. One observation from Figure 4.13 is that the erosion is focused mainly in the lower part of the riffle box, which is where the most particles are accumulated in this area due to the rope effect. In this entry region of the riffle box the main gas-solid flow has not yet adjusted its direction to the inclination of the guiding vanes. Therefore, the coal particles hit the guiding vanes in this inlet part of the bifurcator under higher incidence angles causing higher erosion rates at these locations. Although the particle were mostly on the back of the cross-section in front of the bifurcator, the riffle box makes sure that the particle mass flow is subdivided nearly uniform(around 15% imbalance) to both legs of the bifurcator. The particles concentrate near these vanes due to inertial effects and form certain ropes that will be redistributed upstream over the entire cross-section by the particles interaction with the fluid turbulence. Further downstream the coal particles adjust quickly to the new flow conditions due to their small particle diameter and relatively small particle density. The small inertia of particles results in very small impact angles for particle-wall collisions on most parts located downstream of the bifurcator and pipe walls. Therefore, the erosion in this part of the bifurcator is less significant. Only those regions where the main particle flow is not yet oriented in the direction of the guiding vanes that lead to the front and the back leg are hit with a remarkable quantity of particles under effective angles of attack leading to higher erosion rates at this location. The particles' travel path is shown in Figure 4.17. As indicated in Figure 4.17, not only was the coal flow not distributed evenly, but most large particles (red particles) mainly went to the front branch. Therefore, it is likely that the front part of the bifurcator will suffer more severe erosion. Results of the computations can be seen in Figure 4.18; they give an impression of the velocity field in the bifurcating region.

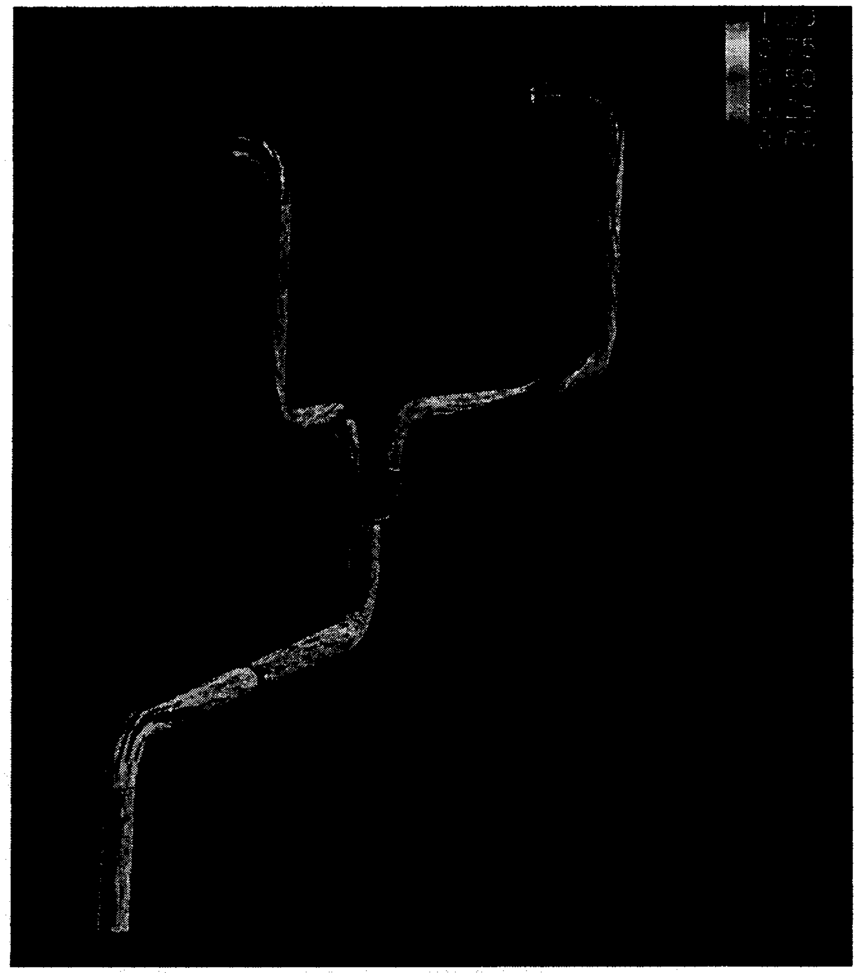


Figure 4.14 Particle distribution (kg/m3) in the whole baseline model

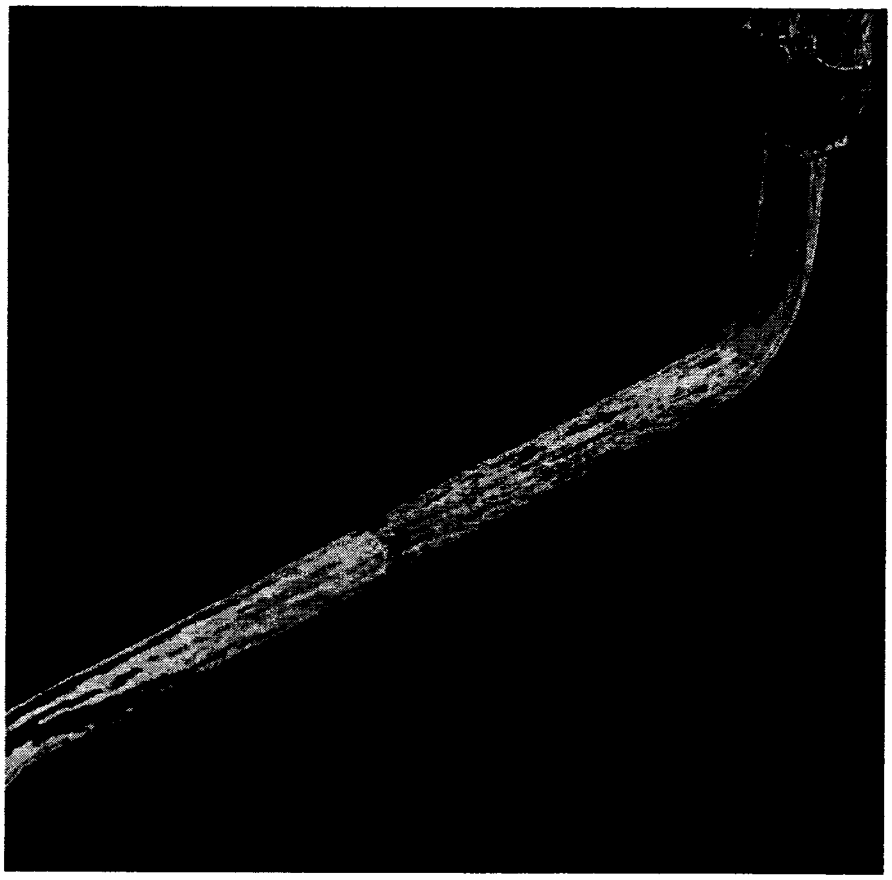


Figure 4.15 Particle distribution in the pipe upstream of the bifurcator

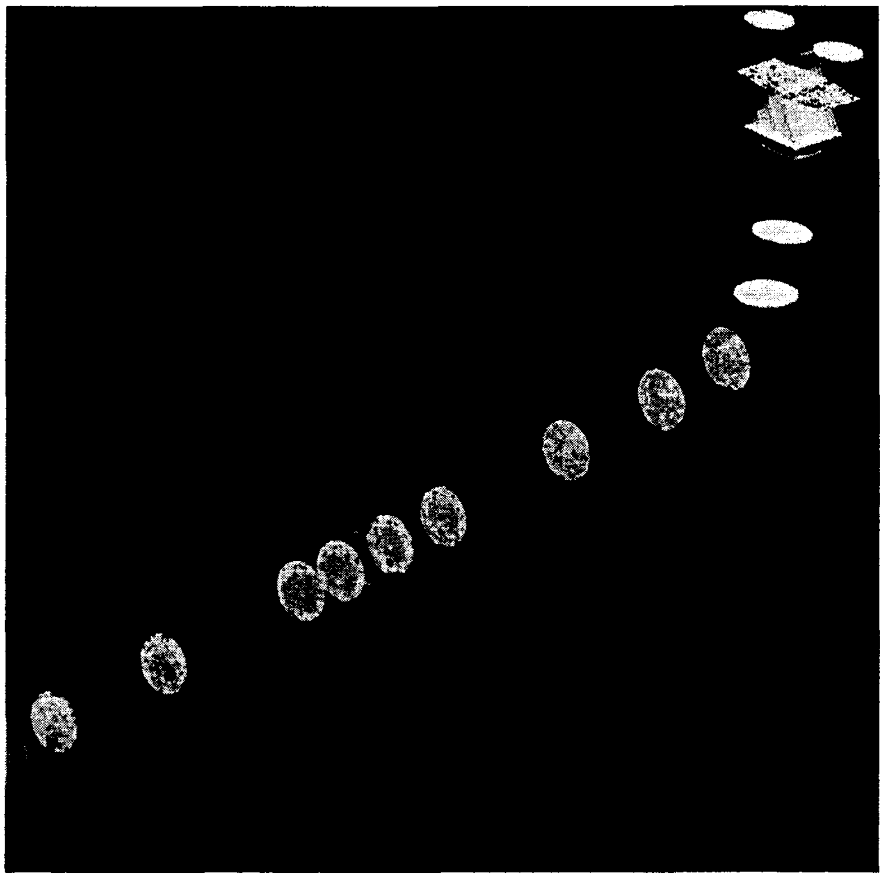


Figure 4.16 Particle concentration on the cross-sections in the pipe upstream of the bifurcator

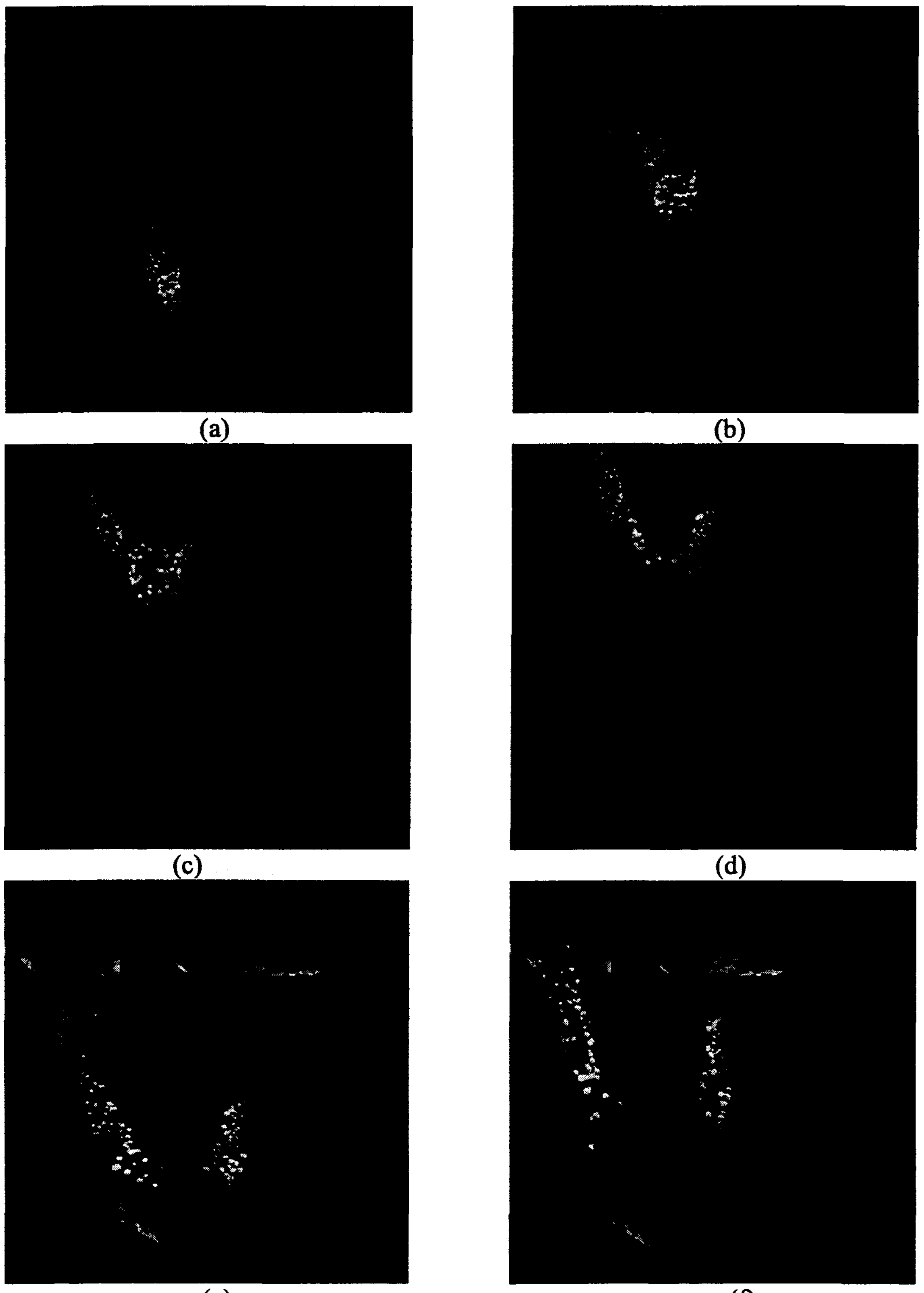

(a) (b) (c) (d) (e) (f)

Figure 4.17 Particle path near the distributor

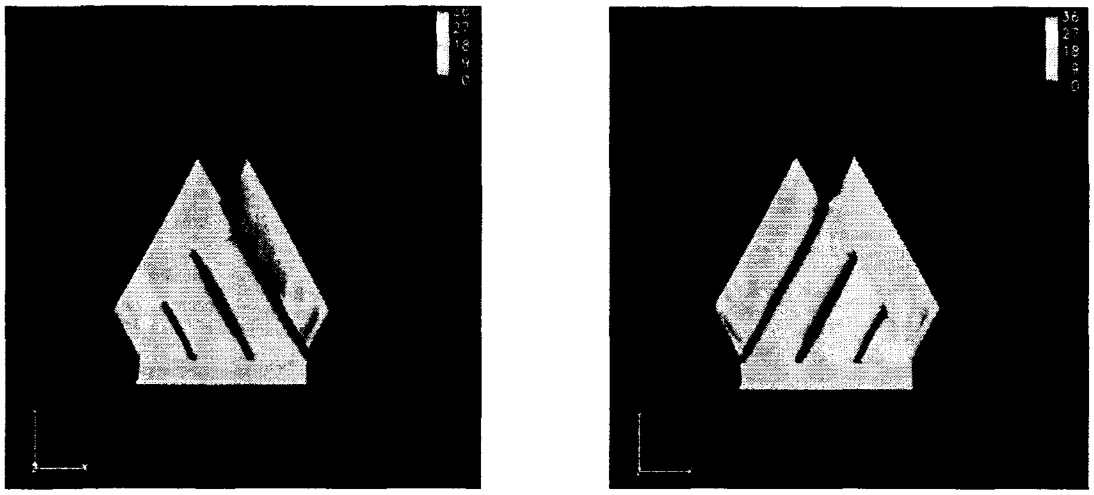

(a) (b)

Figure 4.18 Velocity contours of bifurcator

Figure 4.19 shows the corresponding particle concentration profiles in the pipe downstream of the bifurcator. The rope was stationary, but it spiraled around the inside of the vertical pipe adjacent to the pipe wall. The angular position of the center of the rope angle can be seen in Figure 4.20 which shows the concentration of powder on several sections across the pipe. The spiral is dampened within only a few pipe diameter lengths downstream of the bend. In contrast to the single bend effect, the secondary flow field set up by two elbows prevented motion of the rope away from the pipe wall. Once the rope is formed, the destination of the particles is determined by the wall conditions due to the low air velocity in the boundary layer and the effect of gravity in the vertical sections. Bilergan [17] shows that angular position of the rope, peak particle concentration, and particle velocity within the rope depends strongly on the length of the pipe connecting the two elbows. The numerical simulations show that the gas phase secondary flow fields generated in the elbows are responsible for the spiraling motion observed in the double-elbow case. The double-elbow geometry produces a secondary flow field dominated by a single vortex which causes large-scale rotation over the pipe cross-section.

The shift of the particles around the cross-section prior to a split can be observed, and the split ratio is determined by the distribution of particulate concentration within the dense rope. The work suggests that in order to control the split, the position of the rope must be controlled, either by getting it away from the wall and then re-diverting the dispersed rope, or by forcing the wall-bound rope to a predetermined position on the wall.

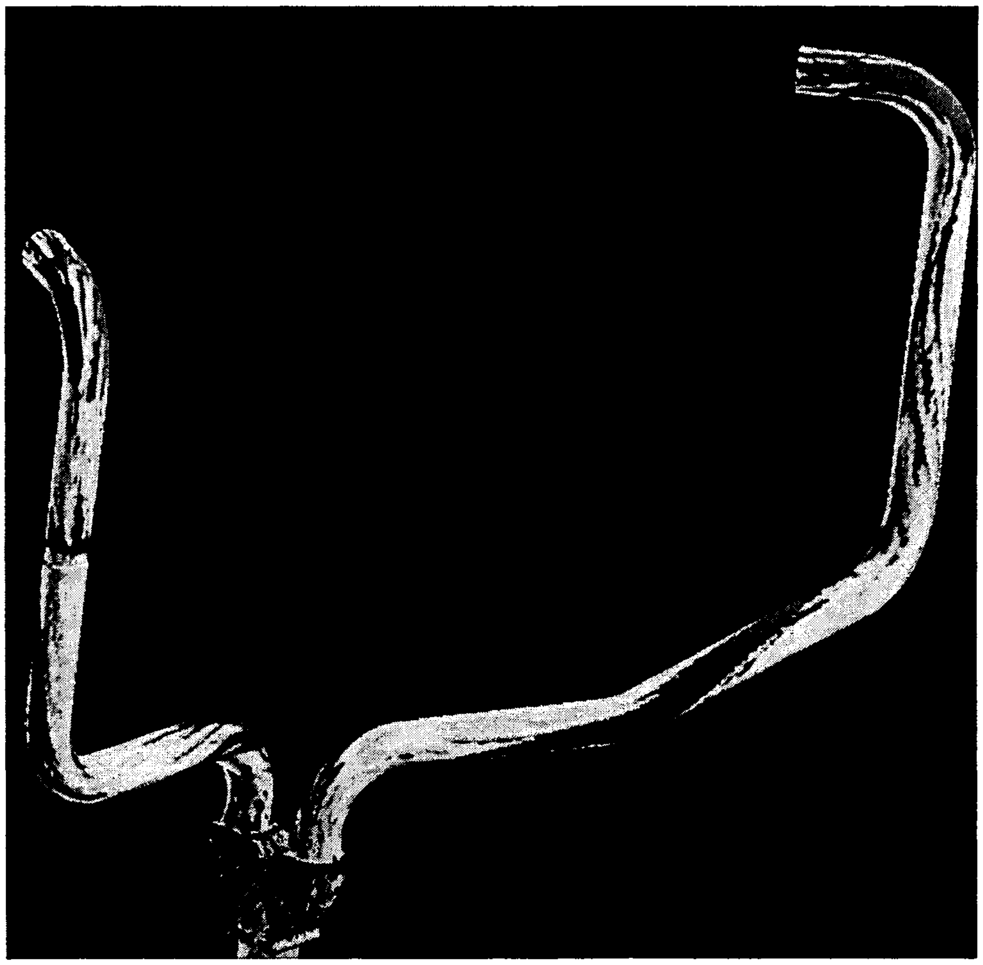


Figure 4.19 Particle distribution in the pipe downstream of the bifurcator

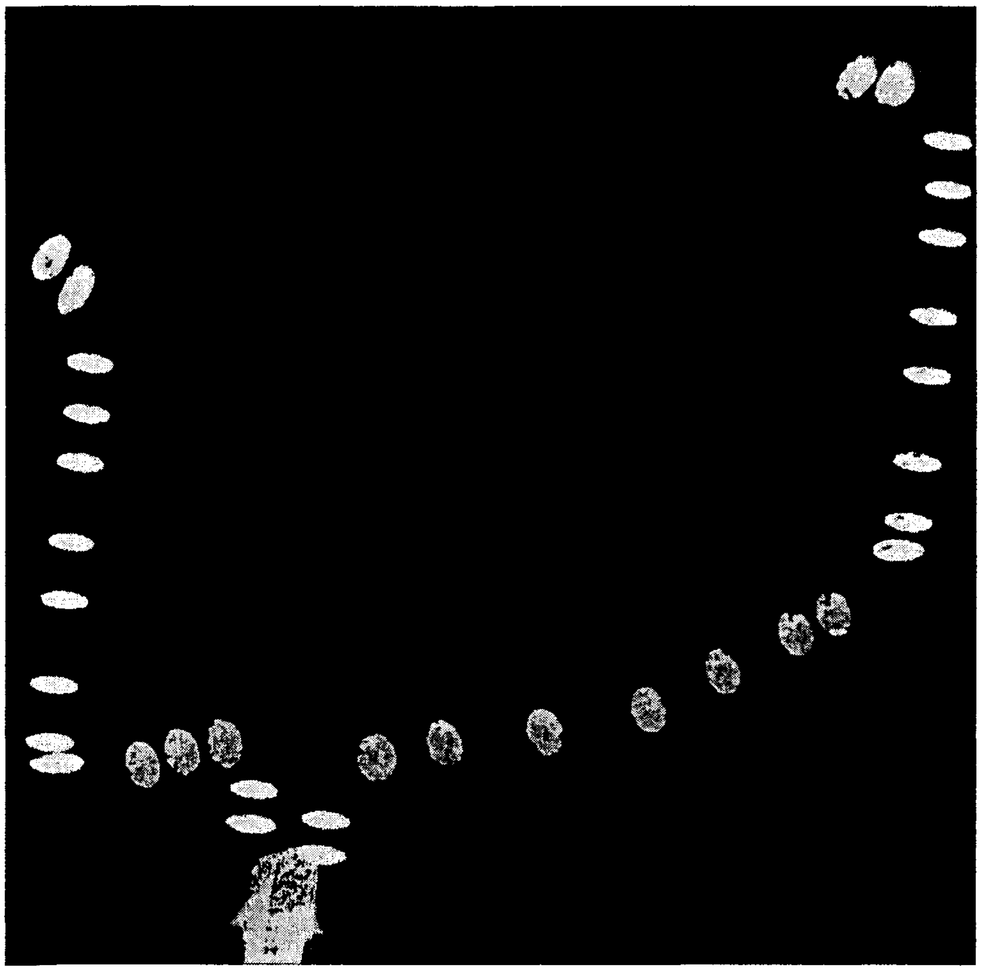


Figure 4.20 Particle concentration on the cross-sections in the pipe downstream of the bifurcator

Chapter 5 Implementation of CFD Optimization Design System

To develop a coal transport pipe system design tool that allows efficient exploration and visualization of design alternatives, the coal transport system simulation and evolutionary optimization algorithm need to be linked together so that proposed design changes are automatically evaluated. In this chapter, the CFD-based optimization in this work is introduced with a special focus on a fast calculation algorithm in order to reduce computational time and increase search performance. After the description of this optimization design environment, its performance is demonstrated by a simple pipe design example.

5.1 Traditional CFD-based Optimization Process

Combining high-fidelity simulation models such as CFD into population-based optimization tools is a significant challenge. The fast calculation algorithm we propose here is based on the converging characteristics of CFD computing and the fitness function evaluation in an EA process. Before presenting the modified design engine, a brief description of the traditional CFD-based optimization problem will be presented. Essentially, a CFD-based design optimization problem is no different from other design optimization problems. Assume, without loss of generality, that a minimization problem can be formulated as following:

$$\begin{array}{ll} \textbf{Find} & \vec{p}^{*} \\ \textbf{Minimize} & D(\vec{P}), \vec{P} \in \phi \end{array} \qquad (5.1)$$

Where $S(\vec{p},\vec{f}) \leq 0$

in which $\vec{p}$ represents the design variables (for simplicity, we focus on shape optimization in fluid engineering; hence, $\vec{p}$ contains geometry shape parameters), $\vec{p}^*$ denotes the optimum solution, ϕ is the feasible space for design variables $\vec{p}$, $D = \{D_1, D_2, \cdots, D_m\}$ represents an objective vector with *m* objective functions to be minimized, and *S* represents constraint functions. In general, these functions are given explicitly in terms of dependent flow variables $\vec{f}$ and in terms of geometric quantities. $\vec{f}$ is a vector of unknowns determined by the governing equations, containing velocities, static pressure and possibly turbulence modeling quantities such as κ and ε. In other words, the flow field variables are decided by solving the flow governing equation F which is subject to the boundary condition *B*. Clearly, the flow is implicitly controlled by the design parameters through surface parameterization, mesh generation, and flow analysis. The flow governing equation *F* can be Navier-Stokes, Euler, potential flow equations or an approximation of any of these equations.

$$\begin{aligned} B(\vec{p},\vec{f}) &= 0 \\ F(\vec{p},\vec{f}) &= 0 \end{aligned} \tag{5.2}$$

The traditional CFD-based design optimization process, a two-cycle procedure, can be seen in Figure 5.1. The optimization process starts by giving a proposed design vector $\vec{p}$. The inner iterative cycle solves the analysis problem for $\vec{f}$ given a proposed design $\vec{p}$, while the outer iterative cycle determines the optimum $\vec{p}^*$. Since the values of objective and constraint functions depend on the flow field solutions, the flow governing equation F must be solved every time D is evaluated. The system optimization continues until the optimization converges and an optimal system is obtained.

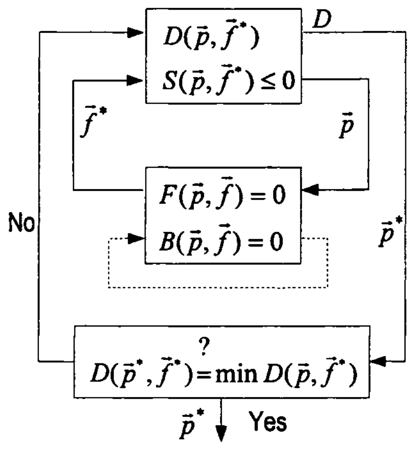


Figure 5.1 Traditional two-cycle optimization procedure

When using CFD to solve a flow field (the inner cycle), the governing partial differential equations F and the boundary condition equation B are approximated by a large set of coupled algebraic equations of discrete variables. These algebraic equations are often nonlinear and hence are solved by an iterative technique. A solution is guessed, the equations are linearlized about that solution and a correction is found by solving the linear equations. The process is repeated until a converged result is obtained. Such iteration procedures are generally driven to minimize some residual of the system. A common and effective strategy used in CFD code is to solve the unsteady form of governing equations (F) and march the solutions in time until they converge to a steady value. The design vector $\vec{p}$ is held fixed while the flow solution evolves in time. It is updated only after the flow solution is converged to a steady state. Usually, some selected measure of the difference between $\vec{f}_{t-1}$

and $\vec{f}_t$ (t-time step) is referred as the residual. Although different CFD codes have different definitions for residual, the most common definition is shown in Eq. (5.3):

$$R = \frac{\sqrt{\sum_{j=1}^{N}\left(\vec{f}_{t,j} - \vec{f}_{t-1,j}\right)^2}}{\sum_{j=1}^{N}\left|\vec{f}_{t,j}\right|} \tag{5.3}$$

in which $\vec{f}_{t-1}$ and $\vec{f}_t$ represent the values of flow field variables in two continuous steps, and N is the number of cells in the flow field, which means this definition keeps every cell in the flow domain in consideration. For most CFD analysis models, the result is considered converged when R falls into the level of 10^{-6}. When dealing with CFD simulation itself, this convergence criterion is necessary since the goal of the CFD solver is to get accurate flow data. However, in the CFD-based design optimization process, only the objective function values are used to drive the search process. Usually, all basic operators, such as crossover and mutation, are only dependent on the fitness value when using EAs as the search method. By defining a suitable objective function, engineers can choose which information they need before the search procedure starts. Therefore, it is not necessary to spend time waiting for the CFD solver to calculate something that will not be used in the design process if the fitness function reaches the value that is no longer being affected by the precision of the flow field variables. In other words, the design decision can be made even though the flow field variables are not yet completely converged from the CFD analysis point of view.

5.2 Modified CFD-based Optimization Process

As mentioned previously, traditional optimization iteration is typically a nested two-cycle process. It consists of two convergence criteria for the inner and outer cycles: the convergence of flow field variables for the inner cycle and the convergence of design parameters for the outer cycle, respectively. By observing many practical engineering cases, we found that the fitness function generally converges much faster than the flow field variables. Figure 5.2 shows how fitness values typically change with CFD iterations. In both cases, the convergence of flow field variables need about 2500 CFD iterations, while the fitness values come to stable values at around 250 iterations. Figure 5.3 shows the velocity contours at different CFD iterations. As shown in the figure, the contour for iteration 230 and 1250 didn't change much. For this case, if the fitness is based on the velocity on the surface, the inner cycle CFD iteration can be terminated at around 250 iterations saving 90% of the CFD computing time without dramatically affecting the design.

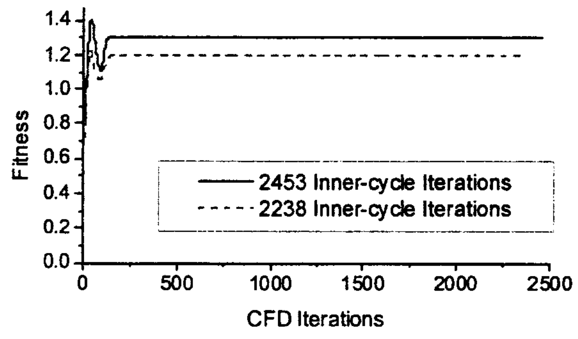
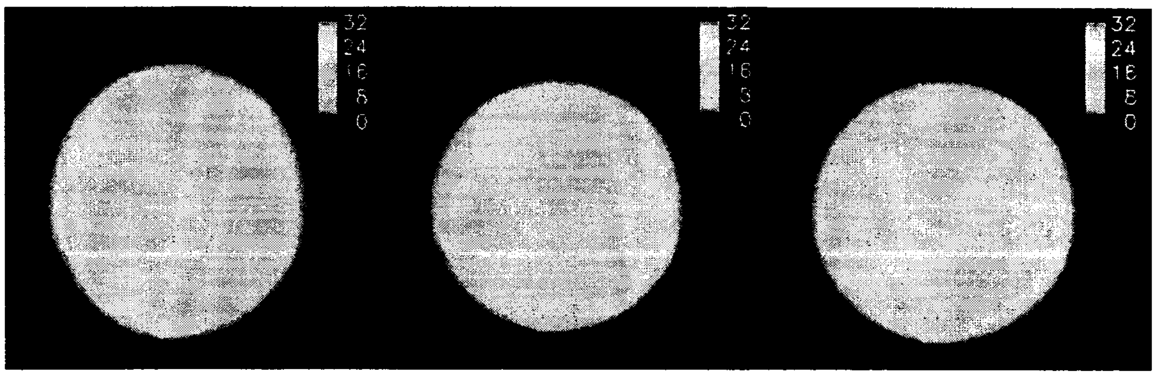


Figure 5.2 Fitness values with CFD iterations



(a) (b) (c)

Figure 5.3 Velocity contours on (a) iteration 147 (b) iteration 230 (c) iteration 1250

Based on this observation, a modified two-cycle CFD design optimization process was implemented. As shown in Figure 5.4, the objective function D and constraint function S were moved into the CFD iteration cycle, and the fitness value is calculated and used as the second convergence criterion in the inner cycle. Hence, at the end of each step of the CFD iteration, the objective function, or the fitness function in this case, is calculated, and the CFD iteration stops when either the flow field residual or fitness value converges. In other words, in the modified CFD-based optimization algorithm, the CFD solver monitors the convergence dynamically not only by checking traditional residuals but also fitness residual in order to reduce the number of iterations needed. Adding an objective function evaluation into the CFD iteration will increase the computing burden slightly. However, this extra time is small compared to the time saved by reducing the number of iterations needed. For instance, in our example case, which will be discussed later, by using the modified two-cycle optimization procedure, the reduction in time can be as high as 95%.

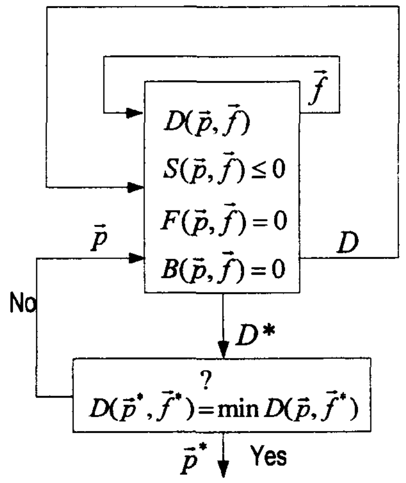
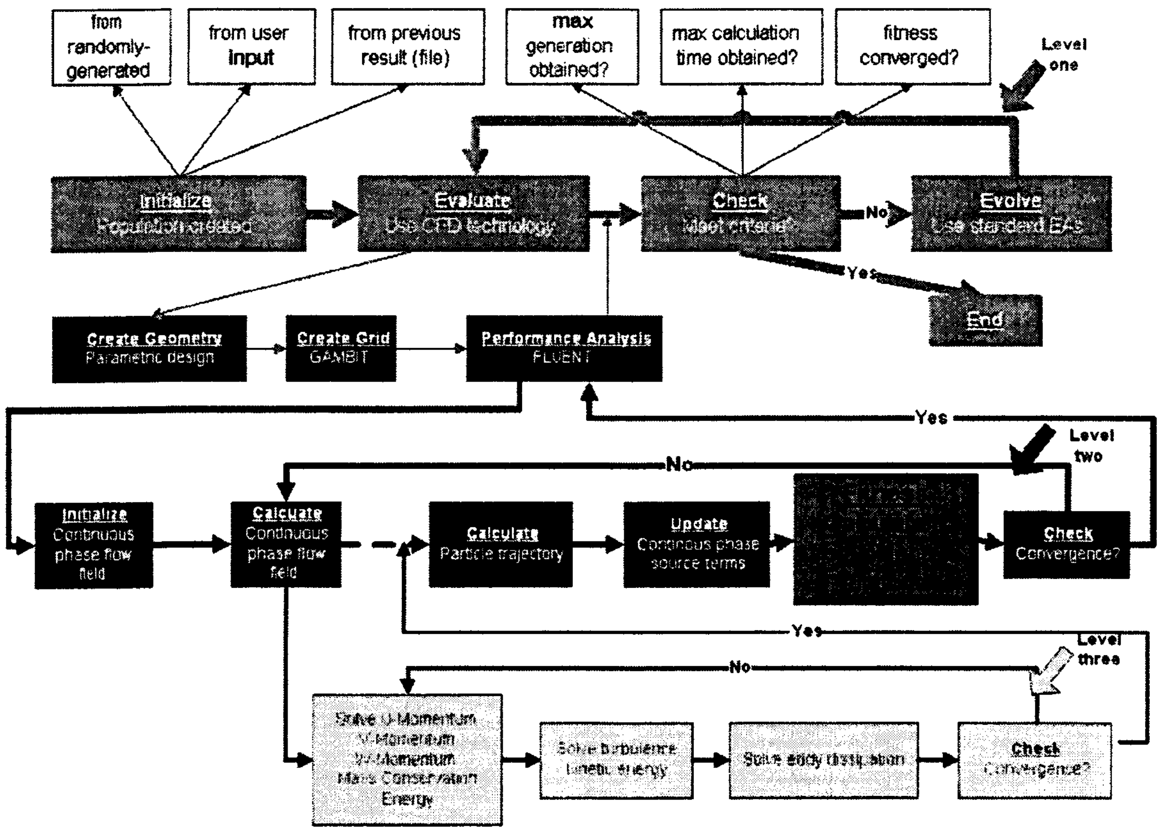


Figure 5.4 Modified two-cycle optimization procedure

Adding fitness residual into the CFD iteration cycle when using Fluent™ is simple. It can be accomplished by an external C user-defined function (UDF). The UDF can be dynamically loaded with the Fluent™ solver to enhance the standard features of the package; by doing this, the original model does not need to be changed. A suitable numerical routine has been developed as a part of the UDF with the purpose of interpolating input and output data from the fitness calculation to the fluid dynamic one and vice versa. The UDF and the Fluent™ solver interact in the following manner: at each iteration, the UDF receives input values needed by the fitness function from Fluent™ and returns output as the fitness residual. The calculated fitness residual is then used by the Fluent™ solver for checking for convergence. If convergence is being monitored, the CFD solution will stop automatically when each variable meets its specified convergence criteria.

5.3 Software Integration

The flow chart in Figure 5.5 gives an overview of the CFD-based optimization process. As indicated, the process proceeds in an iterative manner, for each feasible design change, a CFD analysis is automatically executed in the background to evaluate fitness (e.g., the fluid and particle concentration of the design). The CFD analysis is followed by another design change along with the computational re-meshing of the new design. The design is again evaluated, and through this evolutionary optimization process, the best design is found.



from randomly-generated
from user input
from previous result (file)
max generation obtained?
max calculation time obtained?
fitness converged?
Level one
Initialize Population created
Evaluate Use CFD technology
Check Meet criteria?
No
Evolve Use standard EAs
Yes
End
Create Geometry Parametric design
Create Grid GAMBIT
Performance Analysis FLUENT
Yes
Level two
No
Initialize Continuous phase flow field
Calculate Continuous phase flow field
Calculate Particle trajectory
Update Continous phase source terms
Check Convergence?
Yes
Level three
No
Solve U-Momentum V-Momentum W-Momentum Mass Conservation Energy
Solve turbulence kinetic energy
Solve eddy dissipation
Check Convergence?

Figure 5.5 Flow chart of the CFD-based optimization process

The practical integration of optimum design tools into engineering design environments is a multi faceted endeavour, which encompasses diverse disciplines such as geometric modelling, mesh generation, analysis, optimization and software engineering. The optimization design tool must include a special geometry modification tool to handle three-dimensional designs; additionally, mesh generation tools and a multiphase CFD solver must be included. A user centered interface, which will be discussed later, is also needed to drive the system. In general, this process includes setting up a CFD-problem, solving it, and extracting data for the EA optimization process. A description of some of the major components in the process follows:

(a) *Grid Generation* — In order to carry out the flow simulation, automatic grid generation is necessary; for fast grid generation the grid topology is kept unchanged. Therefore, the modification of the geometry is restricted. A Gambit™ journal file was used to automate grid generation; Gambit™ journal files are text files that contain Gambit™ program commands and parameters. Gambit™ parameters constitute numeric or string constants that one can use in any modeling or meshing operation in lieu of actual numerical or string input. For example, if one defines a numeric scalar parameter named "r" with a value of 1.0, one can execute the following command from within a journal file to create a sphere of radius 6.25 units. "$r=1.0" "Volume create sphere radius $r". A C++ program was written to automatically generate the journal. The main function of the file is to update the parameters which define the geometry. This journal file takes the number of grid points and the clustering parameters as its input. The wall and boundary conditions are also specified in the journal files and stored in mesh files. The C++ program creates a different journal file for each design case; this allows the complete process of grid generation to be invoked in a batch mode of operation using a single Gambit™ journal file.

(b) *Analysis Using Fluent™* — The process of grid generation is integrated with the solver through Fluent™ journal files. A Fluent™ journal file contains a sequence of Fluent™ commands, arranged as they would be typed interactively into the program or entered through the Fluent™ GUI. The purpose of a journal file is to automate a series of commands instead of entering them repeatedly on the command line. The mesh files contain the grid and boundary conditions; when one is ready to perform a large number of iterative calculations, he or she can run Fluent™ in batch or background mode. The batch mode allows the computer resources to be prioritized and enables users to control the process from a journal file. The Fluent™ journal files specify the input file format, flow model specifications and parameters such as the underrelaxation factor, boundary conditions, simulation details and the output format. Therefore, executing the CFD solver can be reduced to preparing a single script.

(c) *Fitness Evaluation* — The most difficult part of an automatic optimization is the definition of a suitable quality function, which has to be minimized or maximized. The quality function depends very much on the specific situation and requires a lot of experience. Often it contains not only fluid dynamical parameters but also economical aspects. For example, in the coal pipe design case adding an orifice may increase the disperse mechanism which is good for rope dispersion, but it also raises the pressure drop in the pipe which in turn causes more power consumption. In order to simplify the problem, only fluid dynamic parameters are applied as a cost function in this project. Thus, the objective function depends on the solution of the CFD model, which depends on the shape of the flow domain. A program was developed using the Fluent™ API to access the flow variables, calculate the fitness, and store the result in an output file which may be accessed by the optimizer. The fitness values are also used to stop the CFD iterations when the convergence criterion is satisfied.

(d) *Post-Processing at the End of Inner-Cycle* — This involves the interactive design feature which will be discussed in chapter 7.

(e) *EA Optimization engine* — The EA optimizer utilizes simple Evolutionary Algorithm optimization software currently. It can be understood as a stand-alone method which includes several components: selector, crossover, mutation, and replacement. In classical EAs, a binary coding is used for genes. Instead, we use floating point coding. This is more natural for this project, since the genes here are real-valued design variables. At the beginning the initial population is created and the fitness function is calculated for each individual. After assigning the fitness to each pipe configuration, by default, selection for mating is performed by a tournament selection method. Moreover, elitism, that is the best individual replicated into the next generation, is invoked. This ensures the survival of the best individual from each generation, and a pipe configuration with high fitness has a higher probability of contributing in the next generation. Crossover proceeds in two steps. First, selected pipes are coupled at random. Second, each pair of pipes undergoes a partial exchange of their design vectors at a random crossing site with a probability. This results in a pair of individuals of new generation. After that, Gaussian mutation is used to generate two children. Then, two random creatures from the last generation, excluding the two creatures with the highest fitness values, were replaced with the newly generated creatures. In this way, creatures with better fitness values are likely to generate more offspring in the following generation.

Several challenges were overcome to develop this design tool. The optimization engine has to be a general-purpose utility. Otherwise, users will have to spend considerable effort to rewrite the software every time a new optimization task arises. Our design environment consists of a main program and several subroutines written in C++ that do not need to be changed by the user. In order to solve a design problem, the user must prepare additional C++ subroutines for the program. Customizing these subroutines for specific applications allows for maximum application performance. What follows is a description of the program and its subroutines; the reader is assumed to possess a basic familiarity with C++. In the main program, several important classes are defined to treat general EA optimization problems:

CreatureFactoryBase defines a base application interface that is inherited from to create customized initialization methods to start the EAs. It holds information about the first generation and encapsulates all the information about the EA's settings. This includes information such as population size, mutation rate, crossover rate, and the other control parameters being used. *CreatureFactoryBase* is also responsible for accepting all design information such as the design parameters' name, and upper and lower limits of the design parameters. Figure 5.6 shows the class information. *CreatureBase* holds information such as number of parameters, the fitness value of this creature, the vector of the current design parameters, and if this creature needs to be revaluated. *DesignParamBase* holds the basic information for the design vector, such as its boundary and name.

In our coal pipe design tool, every pipe has a different geometry. Therefore, the design parameters will be different for different pipes. However, once the design information is set up, the rest of the performance analysis is very similar. The *CalcCase* class in Figure 5.7 executes performance analysis that is needed by the optimizer. It defines a base interface that the user can inherit from to define a method to calculate fitness. In other words, this is the main function that users need to customize. This abstraction allows users to manage all details specific to their fitness evaluation. Another advantage of this abstraction is that users can focus on their own fitness evaluations without worrying about connecting with the optimizer. A user creates his or her own class which inherits from the *CalcCase* class, and registers itself in the classes constructor. For example, *RegisterApp("Pipe_AAD", this).* A very significant benefit of this approach is that it allows integration of diverse EAs and models with little to no modifications of the simulation code and models. Therefore, it is very easy to switch application cases.

CreatureManagerBase handles the details of the EA search process. It includes several components: selection, crossover, mutation and replacement as discussed before. Different methods for these components can be selected according to users' preferences. Although at this stage only simple EA routines are implemented, in order to use different EA algorithms (such as GBEA), users can inherit from this base class to implement new algorithms. In order to add a different algorithm, all that users need to do is to give it a name and register in it the constructor.

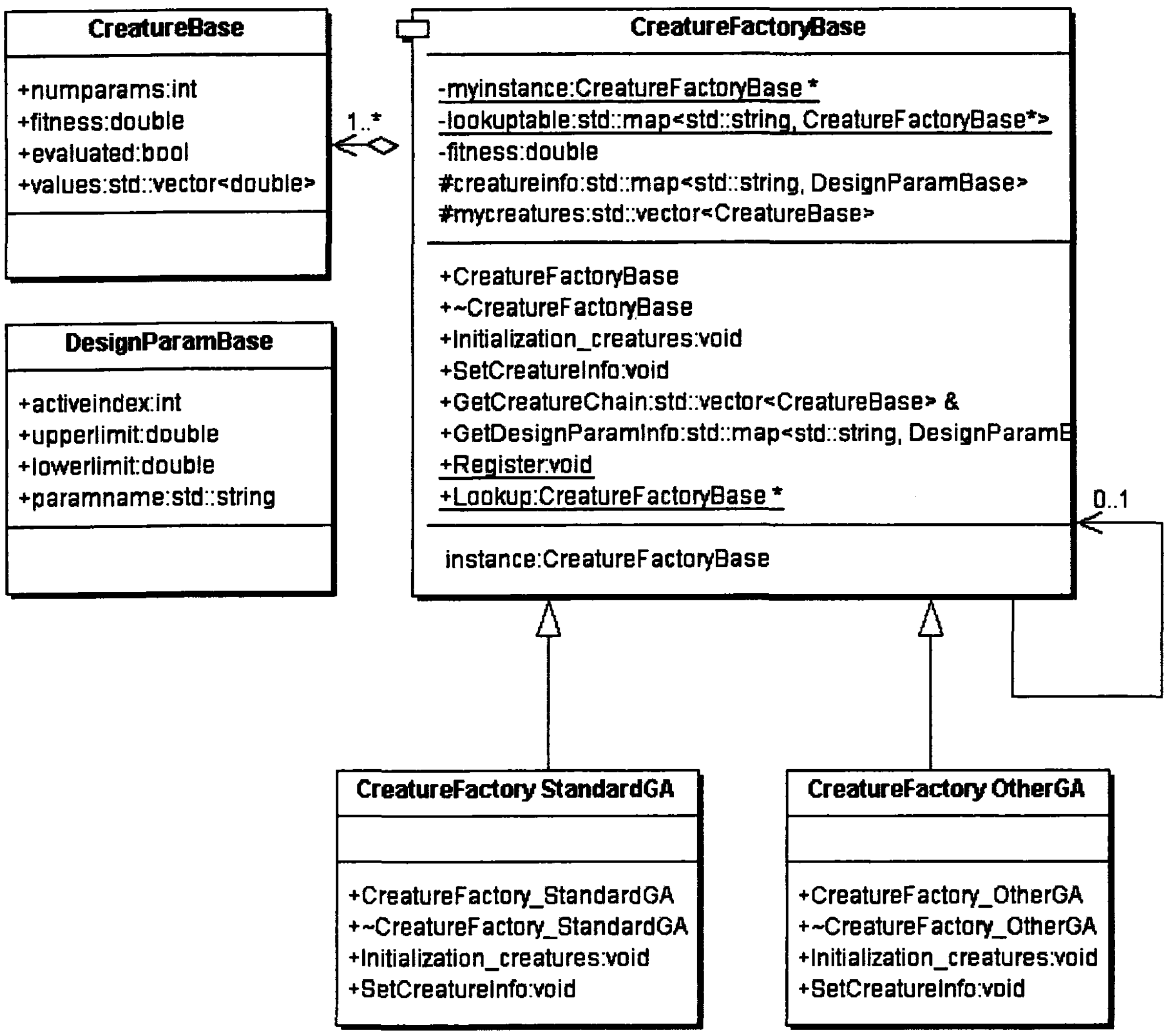

CreatureBase
+numparams:int
+fitness:double
+evaluated:bool
+values:std::vector<double>
1..*
DesignParamBase
+activeindex:int
+upperlimit:double
+lowerlimit:double
+paramname:std::string
CreatureFactoryBase
-myinstance:CreatureFactoryBase *
-lookuptable:std::map<std::string, CreatureFactoryBase*>
-fitness:double
#creatureinfo:std::map<std::string, DesignParamBase>
#mycreatures:std::vector<CreatureBase>
+CreatureFactoryBase
+~CreatureFactoryBase
+Initialization_creatures:void
+SetCreatureInfo:void
+GetCreatureChain:std::vector<CreatureBase> &
+GetDesignParamInfo:std::map<std::string, DesignParamB
+Register:void
+Lookup:CreatureFactoryBase *
instance:CreatureFactoryBase
0..1
CreatureFactory StandardGA
+CreatureFactory_StandardGA
+~CreatureFactory_StandardGA
+Initialization_creatures:void
+SetCreatureInfo:void
CreatureFactory OtherGA
+CreatureFactory_OtherGA
+~CreatureFactory_OtherGA
+Initialization_creatures:void
+SetCreatureInfo:void

Figure 5.6 Main classes for initialization of the gene chain

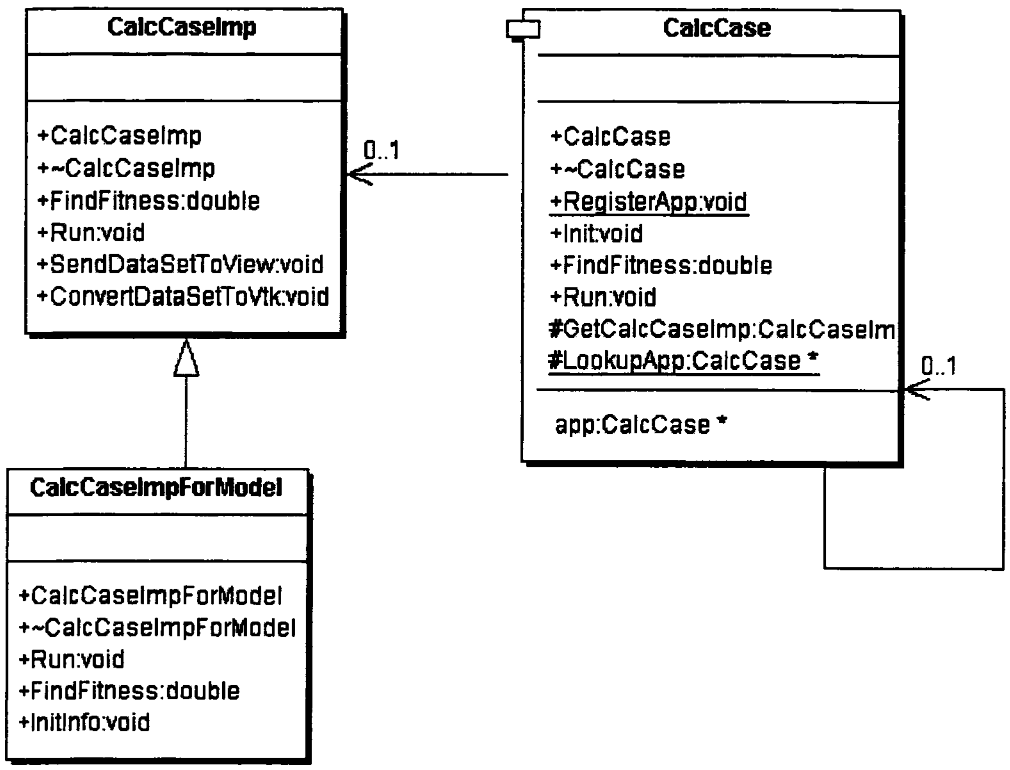


Figure 5.7 Implementation of the Case

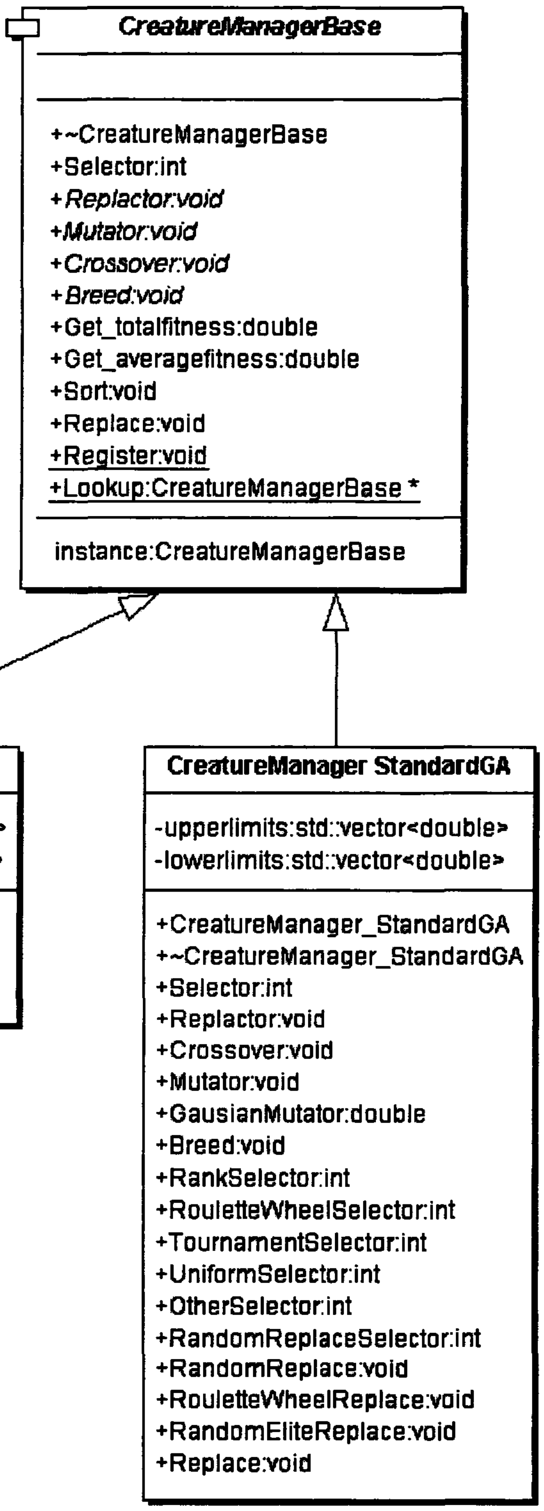


Figure 5.8 *CreatureManager* Class

5.4 Performance

This CFD-based optimization design environment is applied to a simplified pipe design case in this section. Applying this tool to this simplified model provides a proof of concept and an essential test bed; it also enables developers to focus on the methodology and keep the effort tractable. The computing time and results are documented and the efficacy of this design tool is demonstrated by comparing with the traditional design optimization process.

5.4.1 Geometry

The basic pipe geometry consists of a vertical pipe with a length of 5 pipe diameters, the elbow section, an horizontal pipe with a length of 15 pipe diameters followed by another elbow and a vertical pipe with a length of 3 pipe diameters. Both elbows are 90° elbows and the radius of the elbows is 3 times the pipe diameter. The computational model was operated with gas velocities from 15 to 30m/s. Table 5.1 shows the geometric dimensions of this pipe and Figure 5.9 shows the sketch of the testing pipe geometry.

Table 5.1 Geometry sizes of the testing facility

Pipe Diameter, d (m)	First Vertical Pipe Length	Horizontal Pipe Length	Second Vertical Pipe Length	Elbow Radius
d=0.154	$5 \times d$	$15 \times d$	$3 \times d$	$3 \times d$

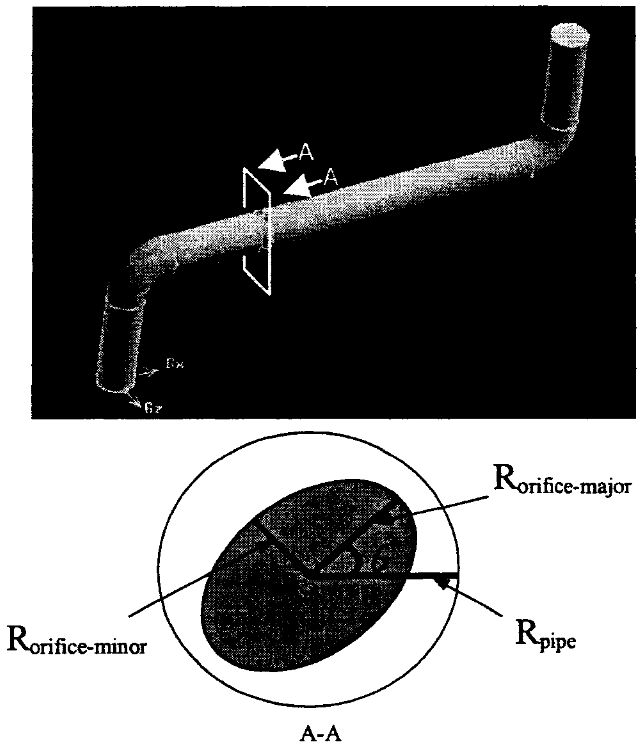


Figure 5.9 Sketch of testing pipe geometry

5.4.2 Object Function

Orificing the coal pipes has been done in the past with fixed orifices to improve the air distribution. By using adjustable orifices with actuators the air distribution can be influenced and set to the target values. This method has a serious disadvantage, however; an orifice creates additional pressure drop and mainly influences the flow of the transport air in the pipe. Due to changes in pressure drop the particle distribution is only changed indirectly. This can result in operation with insufficient transport airflow and consequently endanger the coal layout in the horizontal pipe. However, because it is easy to install or replace with relatively low cost orifices are widely used in pipelines today. Since we want to keep the pressure drop, the question becomes how can we maintain the pressure drop level and at the same time increase the speed of rope dispersion. Therefore, in our test, to break the coal rope, one elliptical orifice was installed somewhere in the long horizontal pipe with fixed opening to preserve the pressure drop. It is well known that in a single-phase flow, fluid that enters through a noncircular inlet entrains more fluid than does comparable fluid entering through a circular inlet. If this is also the case for gas-solid flow, the stronger mixing mechanism causes the dispersion rate of the coal rope to increase. Although there is no literature that discusses using elliptical orifices in the coal pipe, in this case study, this approach has shown that elliptical orifices can be used to increase dispersion rate for coal roping. The lower part of Figure 5.9 shows the parameters that determine the shape of the orifice. The design parameters in this case include the location of the orifice, L, the orientation of the orifice, θ, the ratio of the radius of the orifice ($R_{\text{orific-major}}$), and the pipe, $\tilde{r}$. The goal of the pipe design optimization is to minimize the coal distribution difference at the exit surface of the pipe. To facilitate a quantitative comparison between the cases with different orifices, a mixing index (*MI*) defined by Bilirgen [68] is used as the index to the coal distribution. The mixing index is computed using:

$$MI = \frac{1}{C_p}\left[\frac{1}{n-1}\sum_{k=1}^{n}\left(C_m(k) - C_p\right)^2\right]^{1/2} \tag{5.4}$$

Where C_p is the mean particle concentration in the pipe's cross-section, $C_m(k)$ is the local particle concentration measured at different locations on the surface, and n is the total number of cells on the surface. Eq. 5.4 gives the degree of variation in concentration as a standard deviation. The flow is assumed to be well mixed when mixing index tends to zero at the inlet. The optimization problem is then formulated as shown

***Minimize*:**

$$MI$$

where:

$$0^o < \theta < 90^o$$
$$2 \cdot d < L < 14 \cdot d$$
$$0.7 < \tilde{r} < 0.9$$
$$\frac{S_{orifice}}{S_{pipe}} = 0.8$$

Where $S_{orifice}$ and S_{pipe} represent the area of the cross section of orifice and pipe, respectively

5.4.3 Results

Two runs were carried out in this study; each has an initial population of 32 and stopped after 770 generations. Both runs were performed on a PC with dual Intel Xeon™ processors running Red Hat Enterprise Linux Workstation 3. The members of the initial population in both runs were selected arbitrarily from the design space by the computer. The computer randomly generated the crossover operator probability p_c and the mutation operator probability p_m. The technique successfully converged; the optimum parameters such as orifice radius and orifice angles, converged to very close values with different initial design candidates. The variations of the best individual's fitness during the evolution process in the run are shown in Figure 5.10. Table 5.2 shows that the computational time was reduced to 35 hours from the original 658 hours by using the modified design platform.

The contours of the particle concentration at the exit in a baseline pipe and the EA generated optimal result are shown in Figure 5.11 (a) and (b), respectively. The technique converged successfully. The optimum parameters such as orifice radius and orifice angles, converged to very close values with different initial design candidates. The best design required the orifice to have an elliptical cross-section (Figure 5.12). The orifice should be installed with less than 10° degree angle (Figure 5.13). This might be connected with the secondary flow created by the pipe elbow; however, for the orifice location (Figure5.14), EA optimization results differ. This might indicate low sensitivity of the orifice location in the region of optimum, anywhere between 2.5 to 10 times the pipe diameter will not make a significant difference, as shown in the figure.

Table 5.2 Comparison between traditional and modified design system

	Traditional Design System	Modified Design System
Randomly-Generated Initial Popsize	32	32
Mating Events	650	650
Time Required to Complete One Case by Computer (min.)	15	0.8
Total Calls to CFD solver	2632	2632
Total Time (hrs.)	658	35

As demonstrated in the test case, we were able to reduce the total computational time for the test case by more than 90%. Hence, it is safe to say, for a simple CFD related design optimization problem, this design environment can help designers solve the problem within a much shorter time frame.

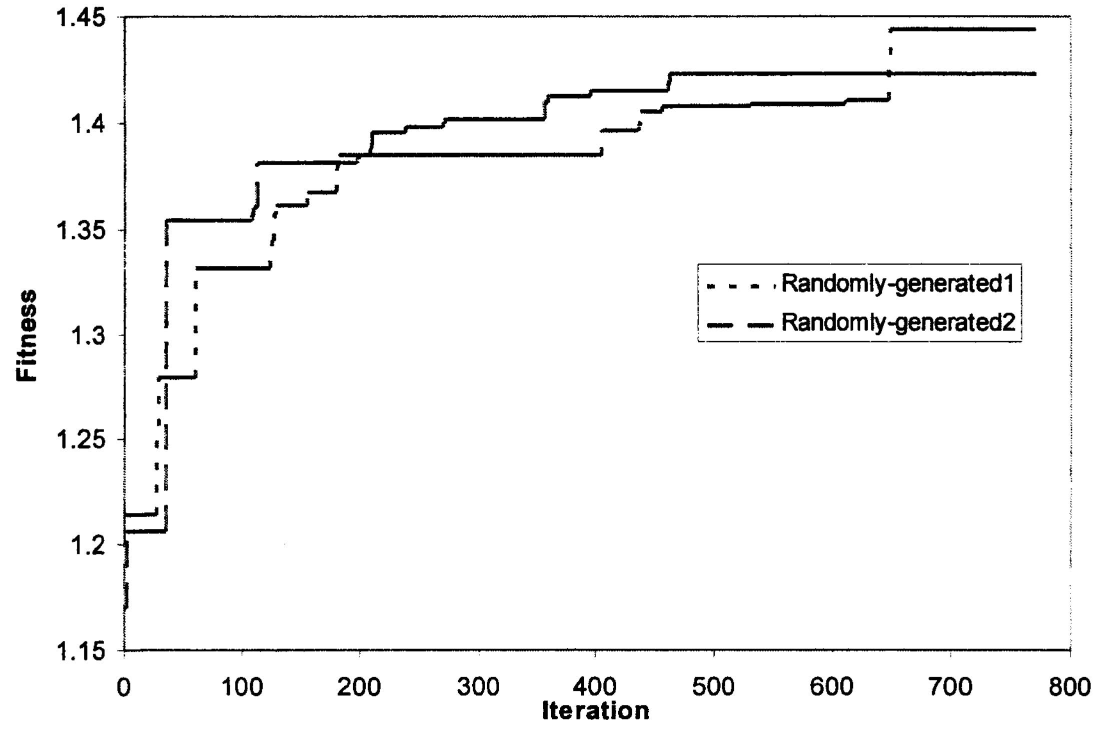


Figure 5.10 Fitness variations in the evolution processes

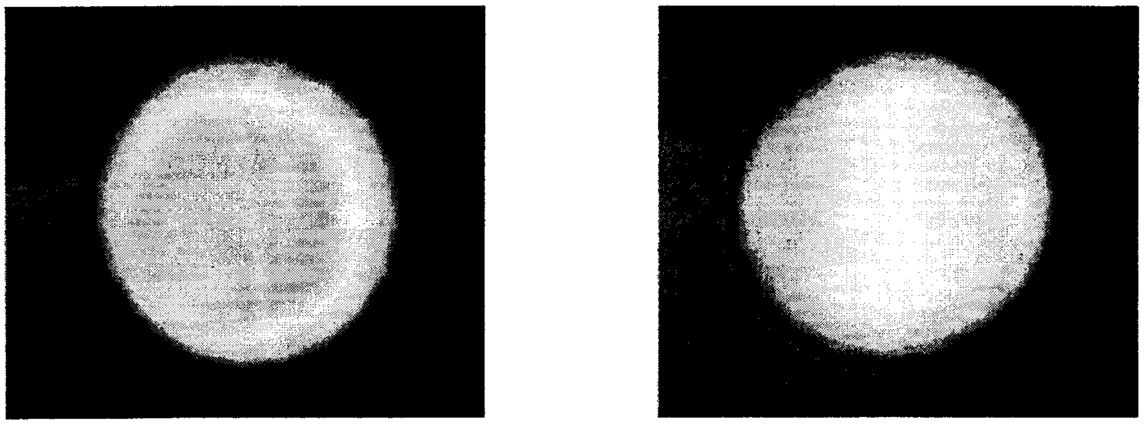


Figure 5.11 Coal distribution contours (a) baseline (b) best design

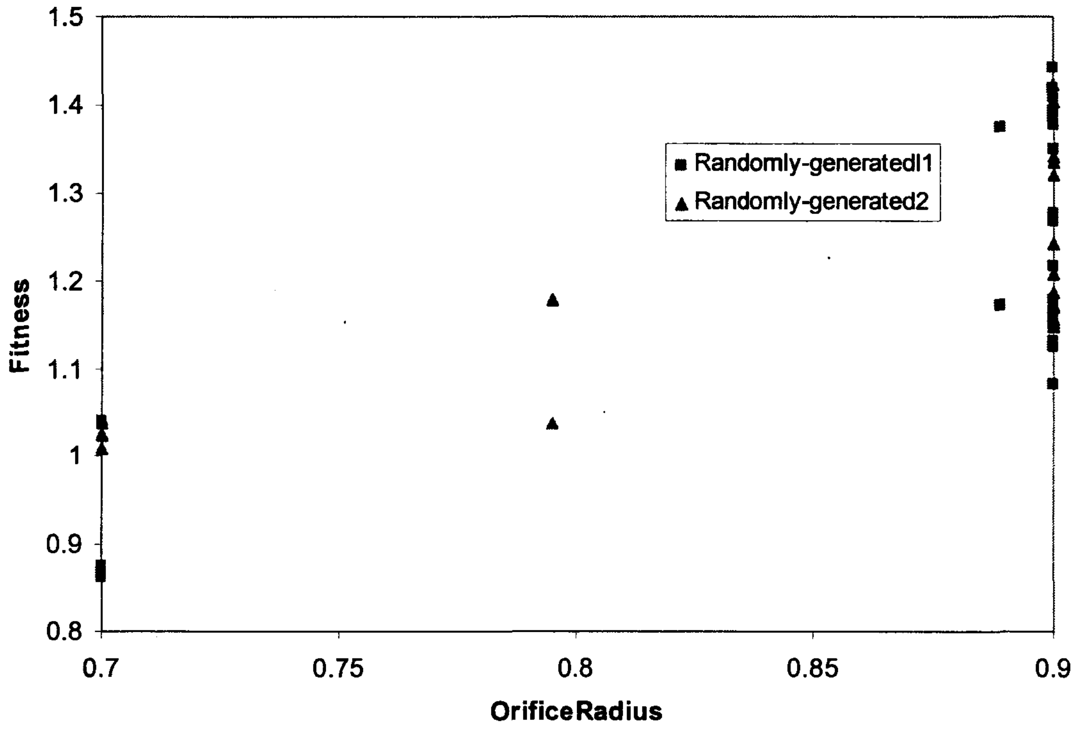


Figure 5.12 EA optimization results for orifice radius

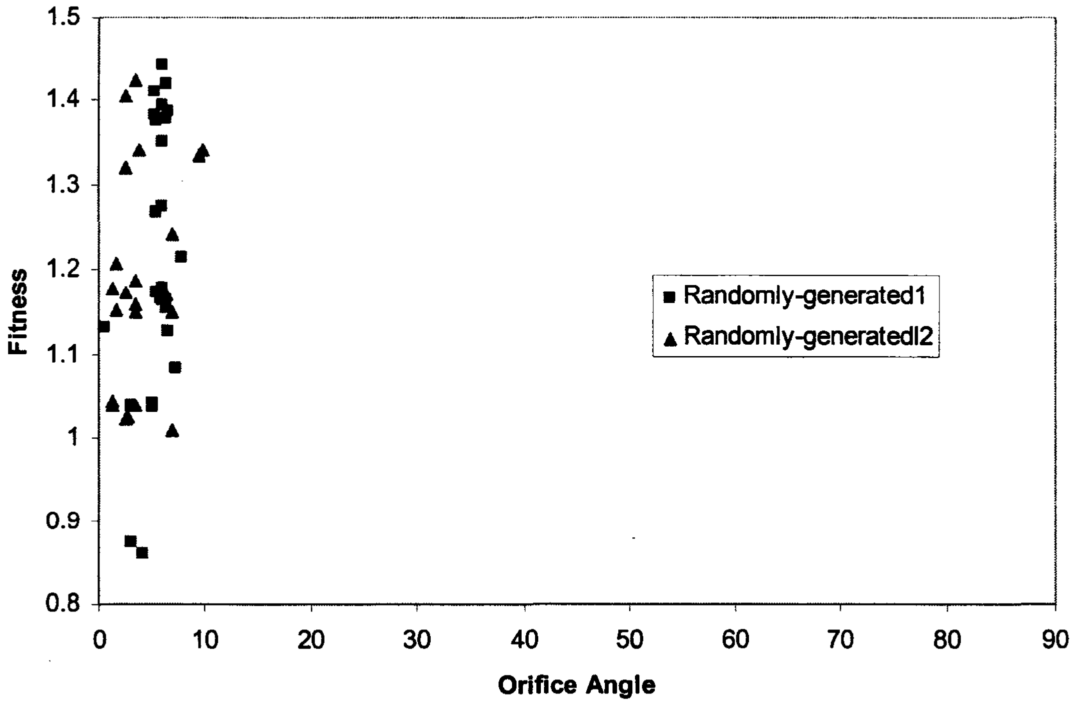


Figure 5.13 EA optimization results for orifice angle

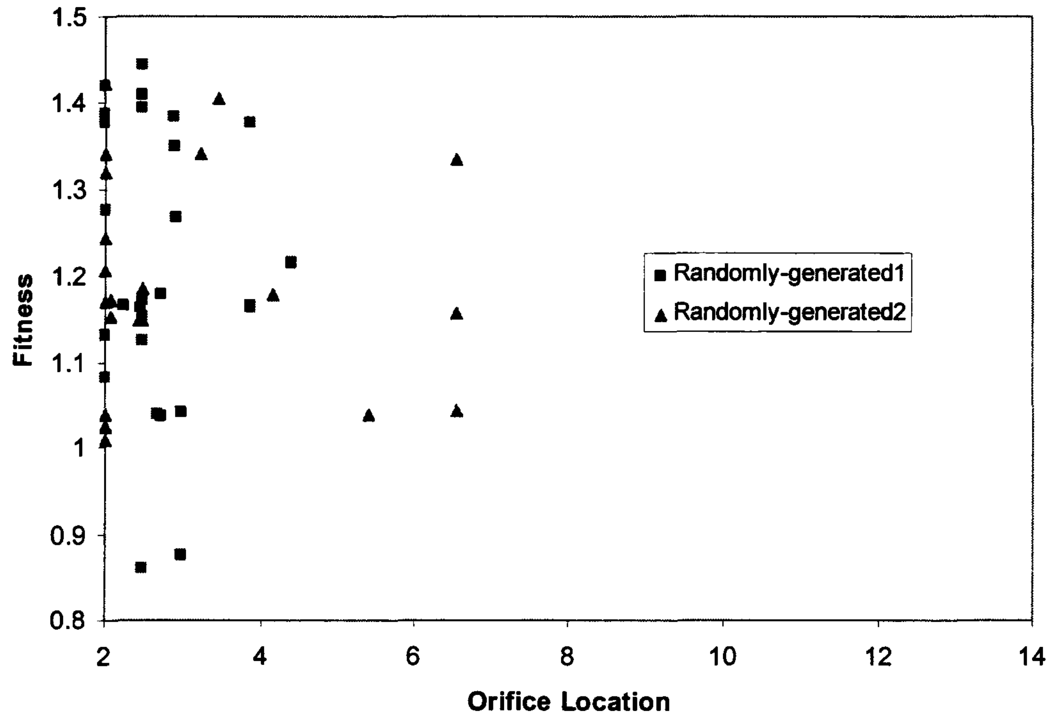


Figure 5.14 EA optimization results for orifice location

Chapter 6 VE-Suite

Knowing whether one idea out-performs another is one of the most important features of doing design analysis. Reliable approximations of the altered CFD flow can provide qualitative results that determine whether the change of an individual component was effective. Enhancements in simulation and design software allow engineers' questions to be answered to a suitable degree and provide the necessary atmosphere for new ideas and creative progress to take place. Also, in order to narrow the search space to a manageable size, having the engineer guide the process by utilizing a set of known guidelines as well as intuition developed from experience can be very useful. More importantly, the approach in this thesis expands on a concept known well in optimization literature—the designer must generally see some physical results in order to understand design trade-offs. This idea is expanded in this thesis, allowing the designer to see physical results of the entire design in real-time during optimization and to use this information interactively to change aspects of the problem statement or redirect the search. In the next two chapters, we describe an interactive simulation and analysis tool that includes three components: the display device and graphics software, the computational software, and the communication layer that passes information between the two. The goal is to enable an engineer to control the design process and to inject his or her creativity into it.

Before deciding which development system is best suited to a particular application, the developer needs to know what questions to ask. This includes knowing what capabilities are required to implement the application and how to tell which systems meet those requirements. As shown in the previous chapters, it is clear that flow fields of different pipes in the coal piping system will be different mainly because these pipes have different geometry configurations. The coal pipe design system should be general enough so that it can be used for individual pipe systems without requiring users to write new routines for a new pipe system. This section gives an overview of the basic requirements of a general interactive design environment.

6.1 Basic Requirements for an Interactive Design Environment

To monitor progress of the optimal design process, users need appropriate hardware and software. The software must have proper interactive facilities for the designer to change the course of the design process when necessary. The design information must be displayed in a comprehensible manner; proper help facilities should also be available. A graphical display of various data and information can facilitate an interactive decision making process, so it should also be available.

From our experience, we observe that there are two key issues in designing interactive engineering systems. First, an appropriate division of labor between humans and computers is needed so that the humans' superior abstract thinking and the computers' superior computational speed can work together to produce a synergistic effect. The extent to which human interaction should influence the optimization process is still unclear. Fortunately, end-users usually have more experience handling this issue if the system allows the user to guide the design process. The second issue is in order for the system to gain acceptance, it must be easy to use and provide real-world usability. If the environment cannot adapt to new applications or the learning curve is too steep, this software environment will be of limited use; a developer must not be required to rewrite optimization code for every new application. This would make the application too time consuming to provide real-world usability. Based on these two key requirements, the system should have the following characteristics:

1) Simplicity— the system should be easy to configure and to learn. The Application Programming Interfaces (APIs) and/or languages used to create applications should be cleanly designed and should hide as much of the system's underlying complexity as possible. This means users from different fields should be able to easily build applications inside the system or add new capabilities without dealing with system programming issues;
2) Extensibility— enable the system to grow by extending existing capabilities and adding new technologies. If a system does not allow easy extension, it becomes difficult for users to write applications that will still be useful in the future.
3) Flexibility — enable users to choose from a variety of solvers and other computer aided engineering tools in a platform independent manner;
4) Physically-based, runtime visualization— enable users to observe the analysis result in a realistic and intuitive manner, which is necessary for the designer to interpret the massive amount of information that the high-fidelity solver would provide.

6.2 Why Use VE-Suite

Currently, there is no interactive design system available that meets these requirements. As mentioned before, virtual engineering allows engineers to carry out geometric modeling, performance analysis, numerical analysis, and decision making in a computational environment. It is ideally suited to integrate the above three methods within one product design system. Also, virtual engineering technology can be used as a way of gaining insight into the design space. Furthermore, virtual engineering technology can be used to quantitatively and qualitatively identify innovative design options, which is exactly what the interactive design system requires. Hence, we have worked to extend the virtual design and engineering capabilities of VE-Suite to handle interactive product design. In order for the reader to gain a better understanding of the internal structure of the new interactive design toolkit, this chapter outlines the structure of VE_Suite. The next two chapters discuss the implementation details of the new design system built on top of it.

VE-Suite (www.vesuite.org) is an open source virtual engineering software package that is currently in active development by the virtual engineering research group at Iowa State University. VE-Suite is designed as a high-level support tool for engineers who want to transform their traditional applications into virtual engineering-based applications. Essentially, VE-Suite enables users to easily incorporate component models and corresponding two-dimensional and three-dimensional graphical representations to create new, plug-and-play framework components.

6.3 VE-Suite Structure

The structure of VE-Suite is shown in Figure 6.1. The core modules (the three modules inside the red circle) of VE-Suite are VE-Xplorer (the graphical engine which is used to view comprehensive two-dimensional or three-dimensional graphic results), VE-Conductor (the GUI front end to the virtual engineering framework which provides easy user interaction), and VE-CE (the computational engine). VE-Suite is general in nature and the three key components can run separately on a geographically diverse set of heterogeneous computer platforms. This separation is convenient because the VE-CE can run on the same machine as the application (computational unit), and VE-Conductor, which presents a graphical user interface to the user, can execute remotely on a separate machine. For example, the VE-CE component can run on a Linux cluster; the VE-Xplorer component can run on an SGI rendering machine; and VE-Conductor can run on a portable Tablet PC. Therefore, the framework components can be distributed across computational resources to make the most efficient use of these resources. This architecture is also advantageous because VE-CE must exist through the application's lifetime while VE-Conductor does not share this requirement. VE-Conductor is transitory and can connect to the server many times throughout the server's (application's) existence. Since the client may use visualizations for data interpretation, the end-user may choose to run the client on a high-performance graphics system. Also, the three core components of VE-Suite can function as complete stand-alone applications provided the necessary input files are prepared by the user.

The communication between the different components and user-defined modules (three green modules in Figure 6.1) is built upon the widely adapted and stable Common Object Request Broker Architecture (CORBA) standard developed by the Object Management Group over the last decade. The Executive module (one of the key modules in VE-CE) implements two of the standard CORBA services bundled with The Ace ORB (TAO) CORBA [105] distribution. The first service is the COSS Naming Service that is used as a lookup table of currently running processes which allows clients to find running process based on a given ID. The second service is the interface that houses the functional and data type definitions and provides them graphically to the user to allow him or her to define a workflow in the graphical interface. An Interface Definition Language (IDL) between VE-

CE and other components was designed to generate general datatypes in order to meet the requirements of different applications.

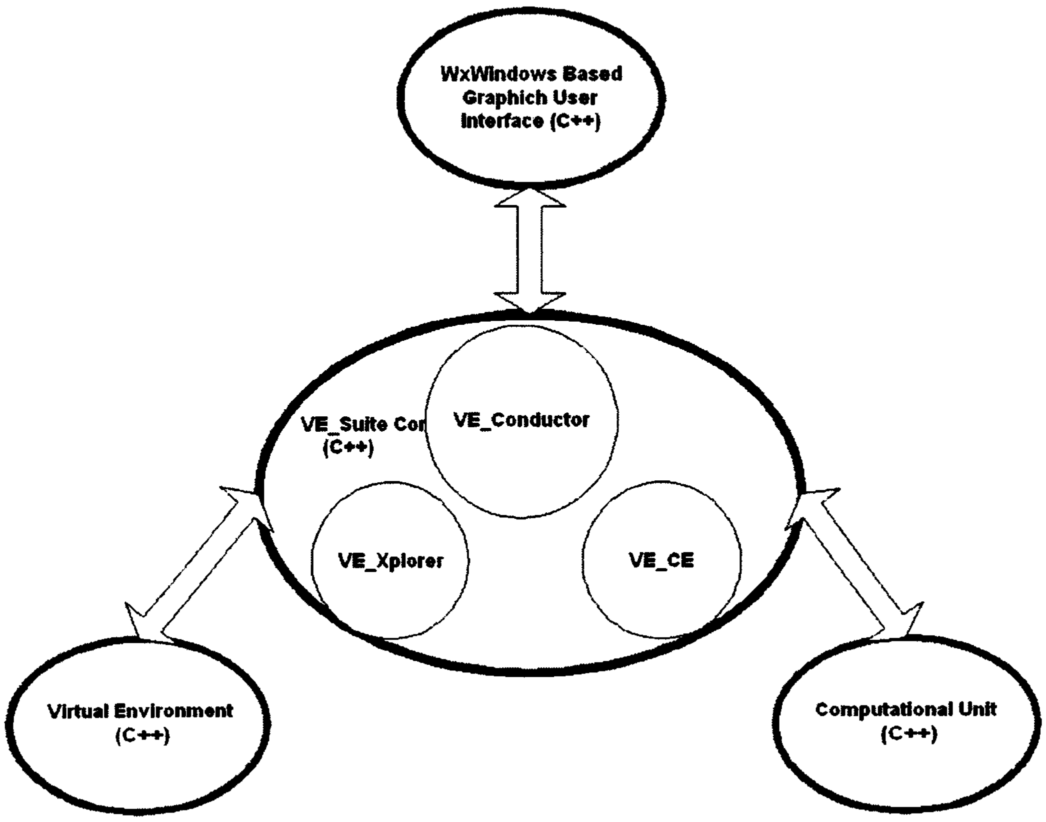

WxWindows Based Graphich User Interface (C++)
VE_Suite Co
(C++)
VE_Conductor
VE_Xplorer
VE_CE
Virtual Environment (C++)
Computational Unit (C++)

Figure 6.1 Architecture of VE-Suite

In the VE-Suite framework, the running process (usually the computational unit) can broadcast its status and analysis information to multiple GUI clients. Any given GUI can connect to the system information stream at any point in time and view the current state of the running process. The framework is designed to allow the GUI to be shutdown and restarted at will without any impact on the computational unit's execution. This attach/detach functionality gives the user the ability to easily monitor the computational process. As an example, this functionality allows a user to build and start a simulation and then detach from the computational engine. The user could then go to a different location, re-attach to the running process, and regain monitoring and control functions. The other advantage of the use of component architecture design techniques is that multiple GUIs can also be connected simultaneously from different computers and allow multiple users to monitor a simulation from different locations.

To enable portability on multiple operating systems and immersive technology platforms, VE-Xplorer is built upon VR Juggler [107], SGI OpenGL Performer [104], and Kitware's Visualization Toolkit (VTK) [106]. Additional details about internal structure of VE-Suite [67].

6.4 Introduction to the VE-Suite API

When building a virtual engineering-based application, the implementation can be broken down into four tasks: system configuration, wxWidgets based user interfaces, the computational unit (see Figure 6.1) and visualization results. These tasks comprise the VE-Suite API.

6.4.1 System Configuration

VE-Suite communicates with a new application through pre-defined interfaces that all applications must define. Users should first create a *.def file. This file should include all the variables that will be passed from the user interface (in VE-Suite it is called a GUI Plugin) to the model that the GUI is designed for. This definition file is a simple text file including two columns. A typical definition file is shown in Figure 6.2. In this case, the module name is *FuelCell*. It has four variables (as shown in the right column): *length*, *cells*, *specie* and *values*. According to the definitions on the left column, *length* is a variable of type double, *cells* is an int, *specie* is a string, and *values* is a double vector.

```
MOD_NAME FuelCell #This is the module name

Double          length
Int             cells
String          specie
Double1D        values

END
```

Figure 6.2 Sample definition file "example.def"

VE-Suite provides an executable file ModWiz; if ModWiz is running on Linux/Unix, and using the above definition file as an example, after running the "ModWiz example.def" command, five files (example.h, example.cpp, example_UI_Dialog.h, example_UI_Dialog.cpp and a Makefile) are generated. After filling up the first four generated files, users can build their own GUI plugin which is specifically targeted to their own application.

6.4.2 User Interface

The GUI is where the user is able to create the system configuration, set model inputs, start and stop execution of the simulation, and view simulation results based on the system configuration they designed in the first step. Developers can create their own GUIs and then compile the resulting code into a Dynamic Link Library (DLL) in Windows or a shared library in Linux/Unix. Thus, it is more likely to meet user specific requirements because the interface is built by the person who will use it. This user interface will be custom-tailored to the user's needs and can easily be modified if the user's focus of interest changes. VE-Suite is able to dynamically discover, identify, and load the user's GUI from these shared libraries.

VE-Suite provides a simple API built on top of wxWidgets for the user to build his own interface to fit the requirement of the actual application. WxWidgets [108] was chosen as the UI library because it is one of the most functional, stable cross-platform UI libraries. It has been actively maintained for over fourteen years and is easy to learn and use. In order to transfer data from the GUI to its control, the function TransferDataFromWindow in the generated example_UI_Dialog.cpp file must be overridden. A C++ plug-in base class (Figure 6.3) defining the basic GUI interface is provided to all module developers. Since this user-defined GUI is inherited and extended from the plug-in base class, it contains all of the functionality provided by the framework. As mentioned above, building GUI plugin is the process of filling in the blanks of the four computer-generated files. If users choose to use a simple GUI, they only need to build a GUI based on common wxWidgets functions and no base class functions need to be overriden. If users choose to customize the GUI connection with the computational unit, some important base functions such as UnPackResult, Result and others from the plug-in base class must be overridden. Figure 6.4 illustrates a simple, yet complete example of how to transfer data from a GUI component (wxTextCtrl) to its control.

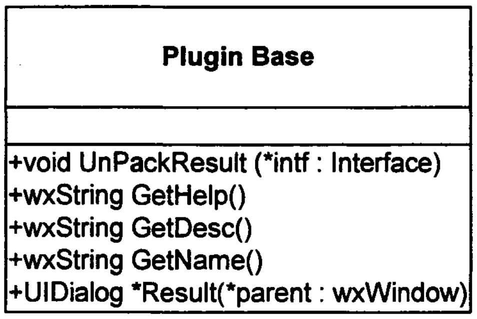
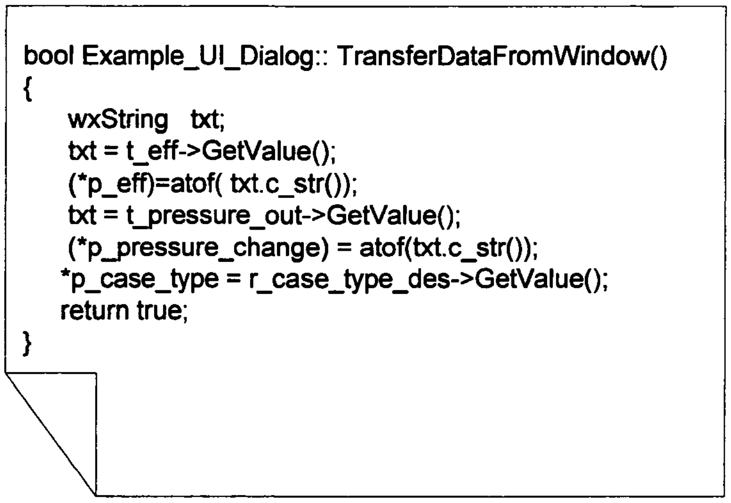

Plugin Base
+void UnPackResult (*intf : Interface)
+wxString GetHelp()
+wxString GetDesc()
+wxString GetName()
+UIDialog *Result(*parent : wxWindow)

Figure 6.3 PluginBase class



```
bool Example_UI_Dialog:: TransferDataFromWindow()
{
    wxString  txt;
    txt = t_eff->GetValue();
    (*p_eff)=atof( txt.c_str());
    txt = t_pressure_out->GetValue();
    (*p_pressure_change) = atof(txt.c_str());
   *p_case_type = r_case_type_des->GetValue();
   return true;
}
```

Figure 6.4 Example of TransferDataFromWindow

6.4.3 Computational Unit

As mentioned before, VE-CE is used to construct, coordinate, schedule, and monitor the running processes. VE-CE provides a CORBA server with which the detachable GUI and computational unit connect. It is capable of running a simulation containing a multitude of different types of models, each accepting and generating a myriad of data types. Since users from different fields have a wide variety of needs, the analysis tools and their use require a detailed understanding of the problem. Therefore, it is the user's responsible to develop their application objects (computation unit). VE-Suite uses the concept of the computational unit to allow flexibility of the system. This decreases coupling between the GUI and computation unit. Changes to GUI or unit will not affect the other unless the change involves a change of the system configuration. Once a client-server connection is made, the GUI is able to send parameters and commands to the unit, and the unit is able to send results, messages, updates, and communications back to GUI in real time. This leads to more robust and extensible code. Because of this, existing commercial, in-house, and open source analysis packages can be used almost directly with VE-Suite. The analysis packages can vary from Microsoft Excel™ spread sheets to process models to CFD models. The analyst only needs to provide a few routines to declare the communication variables.

VE-Suite provides a template for users to fill their own computation units into the VE-Suite framework. Figure 6.5 shows the basic functions in the Body_Unit_i class; Figure 6.6 shows two important functions that users need to modify in order to implement their own applications. First, SetParams needs to be implemented for the user's customized module; this call takes the GUI inputs passed by the CORBA interface (Executive Module). Using Figure 6.3 as example GUI input, every GUI variable that is needed can be retrieved from the interface by calling p.intfs[0].get***("paramname") (Note: *** represents the data type such as Double, Int, etc.). Second, StartCalc has to be implemented according to project requirements. This is the place developers put the existing application into VE-Suite framework.

6.4.4 Graphical Plugin

A key aim of virtual engineering is to fully engage the human capacity for problem solving by creating a realistic experience for the user so that he or she can focus entirely on the engineering problem. The advantage is that previously indescribable complexities can be understood and the full range of engineering solutions can be explored. The graphical engine (VE-Xplorer) provides the core visualization functionality for the virtual engineering aspect of the framework. It can load geometry files, three-dimensional simulation data and experimental data of almost every format into a scene. VR Juggler is used to handle interfacing with VR hardware and graphics rendering platforms. VE-Suite handles the creation of the virtual environment and VR Juggler allows software to run with any type of virtual environment, from a regular 2-D screen to a six-walled immersive virtual space. Due to the generality of the visualization requirements, the VE-Suite core provides a complete visualization GUI so that users can navigate and control the scene. Thus, although most users of VE-Suite are not necessarily expert software developers, they need not worry about the complexities of details of graphics and virtual reality programming and can instead spend time on their applications.

The GUI is laid out in a tabbed notebook format as shown in Figure 6.7. Typical operations can be performed on each tab of the GUI. For example, the navigation tab is the main window for users to navigate through the scene and choose the location he or she wishes to observe the data from. The visualization Tab is the main tab for displaying steady state data. Through the visualization tab, users can select different visualization methods (contour surface, vector fields, etc.) to visualize the fluid field. Figure 6.8 shows the streamline of flow through a 90 degree elbow. Figure 6.9 shows the temperature contour of a power plant furnace. The Data Set Tab allows engineers to select multiple data sets to view.

The Scalars Tab allows the engineers to select from a set of different scalar data because real systems usually have many scalars of interest. The Streamlines Tab is used for creating streamlines to show a flow field. Engineers select how many seed points they want to use and what orientation the streamlines will be. More details on how to use VE-Xplorer can be found from http://www.vesuite.org. It should be noted that VE-Suite also provides an interface that allows advanced users to add or modify the existing functions or visualization GUI.

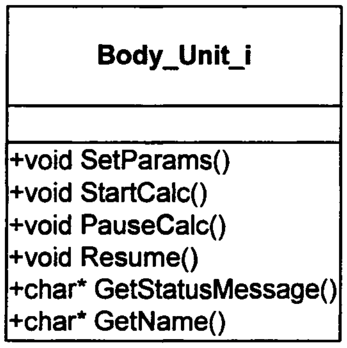


Figure 6.5 Body_Unit_i class

```
Void Body_Unit_i::SetParams (const char* param)
{
        if(string (param)= =" ")
        return;
        Package p;
        p.SetSysId ("gui.xml");
        p.Load(param, strlen(param))
        //Now make use of p.intfs to get your GUI vars out
        eff = p.intfs[0].getDouble("eff");
        pressure_out = p.intfs[0].getDouble("pressure_out");
        pressure_change = p.intfs[0].getDouble ("pressure_change");
        case_type = p.intfs[0].getInt ("case_type");
}

Void Body_Unit_i::StartCalc()
{
        bool rv;
        Package p;
        const char* result;
        //Add your implementation here
        ......
        //Fill out the output stream
        p.SetPackName ("ExportData");
        p.SetSysId ("test.xml");
        ......
        result = p.Save(rv);
        //Marks the end of the execution
        executive_->SetModuleResult(id_, result);
}
```

Figure 6.6 Template for SetParams and StartCalc

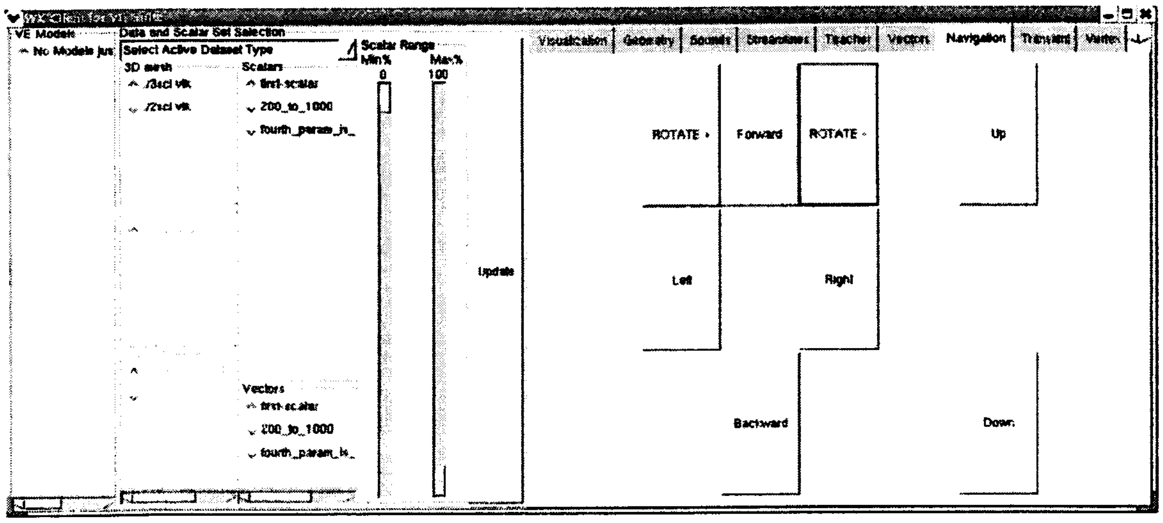


Figure 6.7 VE_Xplorer GUI

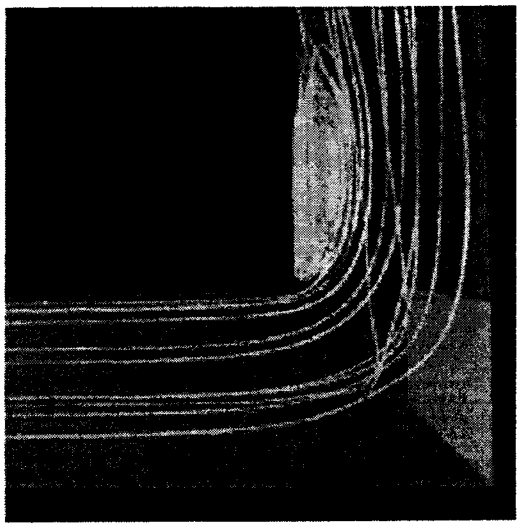


Figure 6.8 Streamline representation

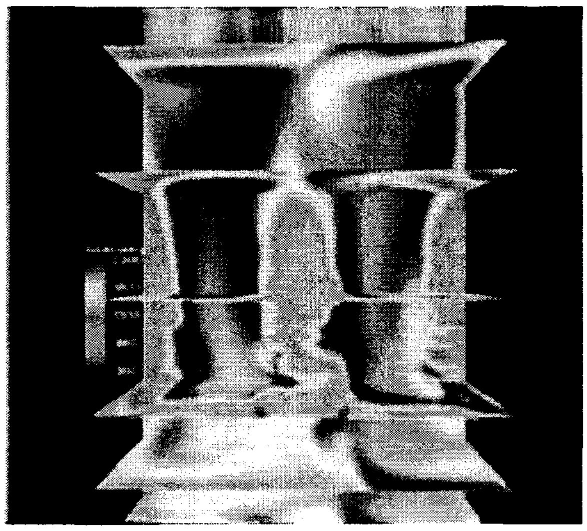


Figure 6.9 Contour representation

Chapter 7 Implementation of Interactive Design Environment

Chapter 6 shows that VE-Suite can function as a powerful foundation for interactive design systems. As discussed before, to allow effective user interaction, the design system should provide users with data and other information about the designed product and its performance in an understandable and intuitive way. The design system should also give users the ability to exert influence on the process in an intuitive manner. VE-Suite already provides some basic functionality, such as a detachable GUI, realistic graphical engine and bi-directional bindings between the GUI and the computational unit. This makes the implementation of the proposed interactive design platform much easier than developing it from scratch. Due to its generality, VE-Suite gives developers significant freedom to develop their own modules; however, sometimes it will force end-users to transfer their applications repeatedly. For example, if users need to change or add more variables into the data transfer stream, VE-Suite will force users to rebuild the def file and its generated files. This kind of repetitive work is not merely an inconvenience for the end user; it will reduce the efficiency which can be gained from using VE-Suite.

As mentioned before, once the design problem is properly formulated, numerical methods can be used to optimize the system. The methods are iterative and generate a sequence of design points before converging to the optimum solution. From this point of view, optimization problems are highly repetitive and the concepts and methods are applicable for cases from different fields. Therefore, a second development on top of VE-Suite in order to suit the requirements of the design system is needed. This indeed provides a major motivation for the work reported here.

7.1 System Structure

Based upon the requirements that the system must be extensible and flexible, the system needs to evolve as it grows, support the ability to add new functionality, and make changes to existing services without affecting the entire system. The first question that needs to be answered is how to design a suitable core that could support the needs of such a dynamic system. Figure 7.1 shows the basic information flow of this system. By using CORBA, the two sides of the network connection can be written in any language and can operate on two different computing platforms. The primary component in the system that users interact with is the plug-in GUI. Using this system, designers can not only access the traditional design information, such as tables, two-dimensional plots and colour codes whenever they want, but they can also view three-dimensional virtual images (i.e. streamline) from high fidelity datasets such as CFD data. On the computational unit side, a multi-threaded approach is used in order to minimize the effort of adapting the existing EAs optimization code into this interactive design space. Threading is done using the high-level threading API provided in the VR Juggler Portable Runtime Library (VPR). The Model thread controls the traditional EAs search component while the View thread deals with the interactive feedback from the EAs. These two tasks can be viewed as separate components that execute relatively independently of the other. The main module code is responsible for making sure that the components are synchronized when needed. A centralized Database Manager module is used for interpreting the user's request, data management and transfer. This additional Database Manager module provides a simple way to track the state of the running process and exchange information between the Model thread and the View thread. The following sections will first describe how to set up the application and then focus on the implementation of the computational unit. The View thread is a separate thread that executes in the same memory space as the main application thread. The main application thread's architecture allows the application to execute normally. In fact, it can execute entirely without interference from the view thread. The view thread has three basic tasks: interact with the main thread, gather monitoring output from the application, and steer the application with users' direction.

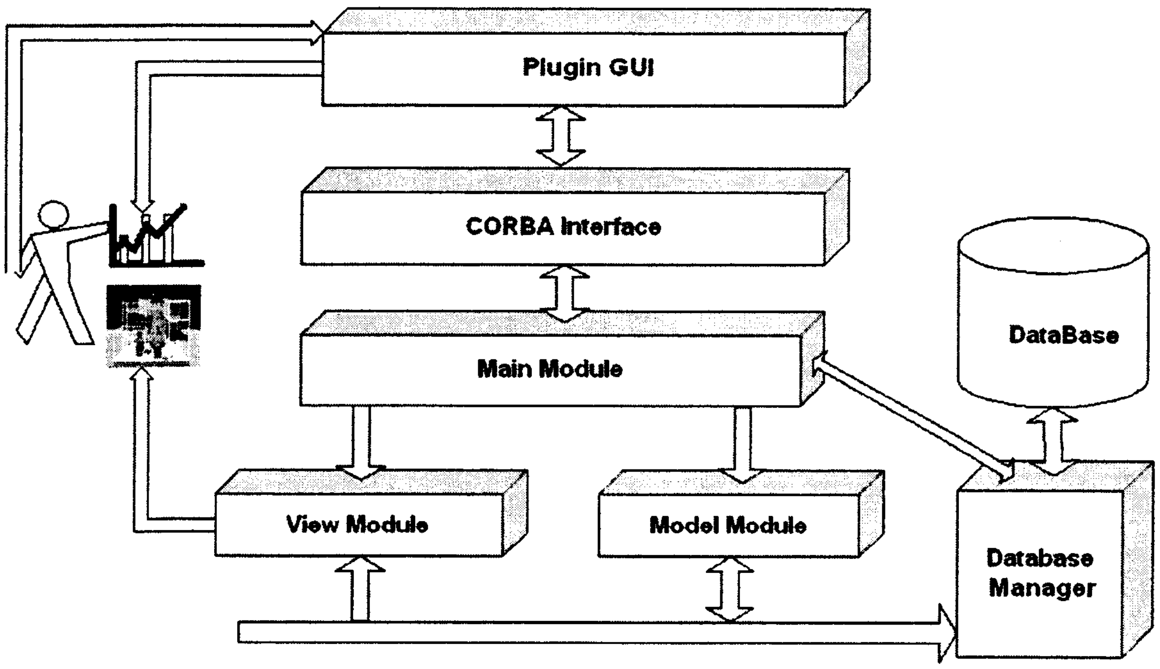


Figure 7.1 Diagram of the system architecture

7.2 Implementation of General Interface for Problem Setup

From the user's point of view, setting up the optimization of engineering problems can be divided into 8 steps as shown in Figure 7.2. To enable the numerical optimization search, the problem must be transformed into the standard VE-Suite form by choosing design variables, constraints and object functions. For performance analysis, if one chooses to use a CFD package, it is most likely that the software package would provide user-made alternative discretization and solution strategies. For optimization search methods, some control parameters need to be set before iterations begin. Current design tools will usually fragment these three tasks into three separate programs. Learning and remembering three separate user interfaces distracts the user. Especially if users want to integrate their applications into VE-Suite, very often they have to write their own PlugIn GUI. Sometimes this preparation is tedious. The novice user would find this task rather difficult and may be discouraged from using the software altogether. Thus, the need for a user interface especially for this interactive design system became apparent. In this system, these are combined into a single easier interface. This interface is oriented towards in the direction of general and usable by the general engineering community. Our ultimate goal is that the same interface can be used with all applications whether the application is a simple modelling simulation or a CFD application and the GUI presented to the user is done in a consistent fashion so that he or she can focus on other tasks such as analysis of the problem specific requirements.

Figure 7.3 shows the interface users can use to setup the necessary control parameters and other necessary initial information. As shown in this figure, the outside frame is provided by VE-Conductor; it provides the interface to CORBA. The inside frame is provided by this design tool. Figure 7.4 shows the main menu which provides many control functions. Over the course of the development of this GUI, it was found that certain features are required. For example, the ability to save the current case, reload the existing case file, basic EAs control parameters setup, etc. Although the current GUI is still in its initial stage, it has basic functionality such as selecting active design parameters, defining EA parameters, defining constraints, and selecting initial design candidates. Since one of the goals of this study is to set up an interactive design system that allows users to provide input parameters to the underlying high fidelity analysis models, observe the analysis result and optimize the product design, the graphical user interface must be easy to use.

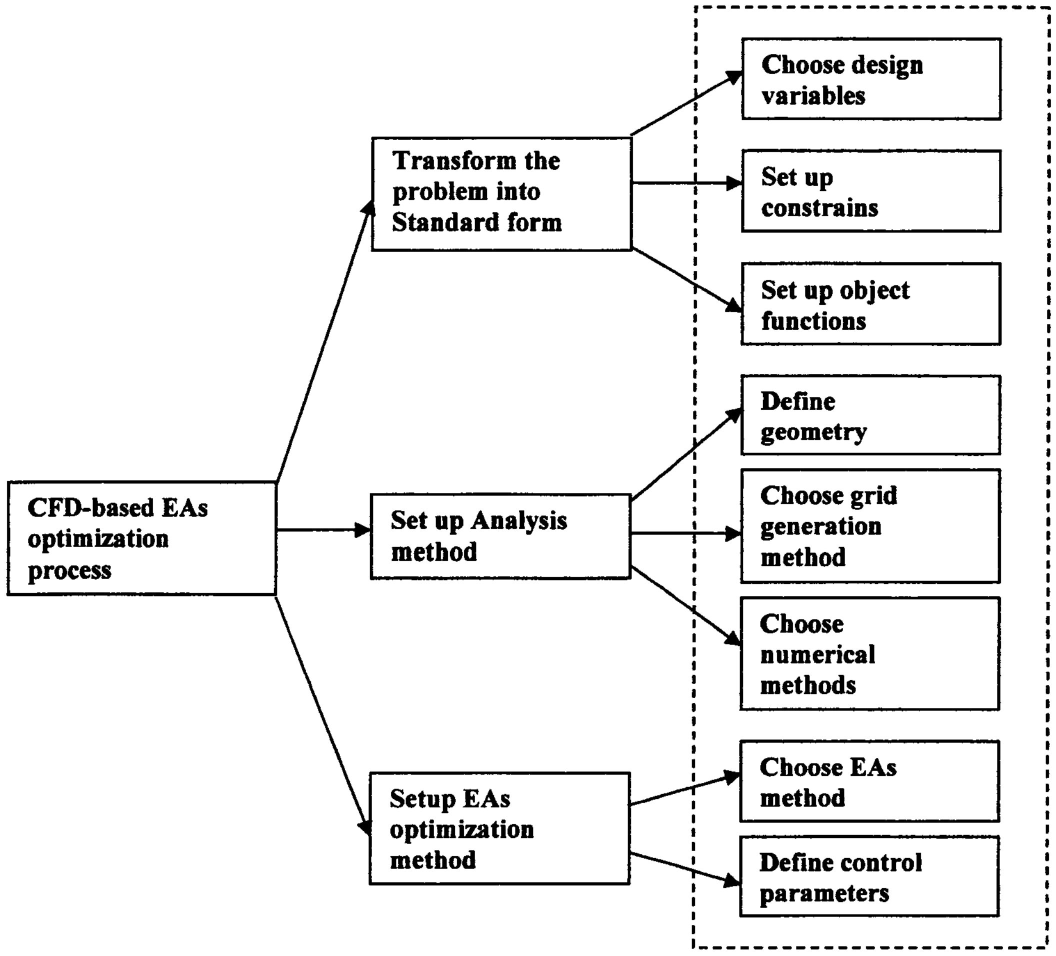
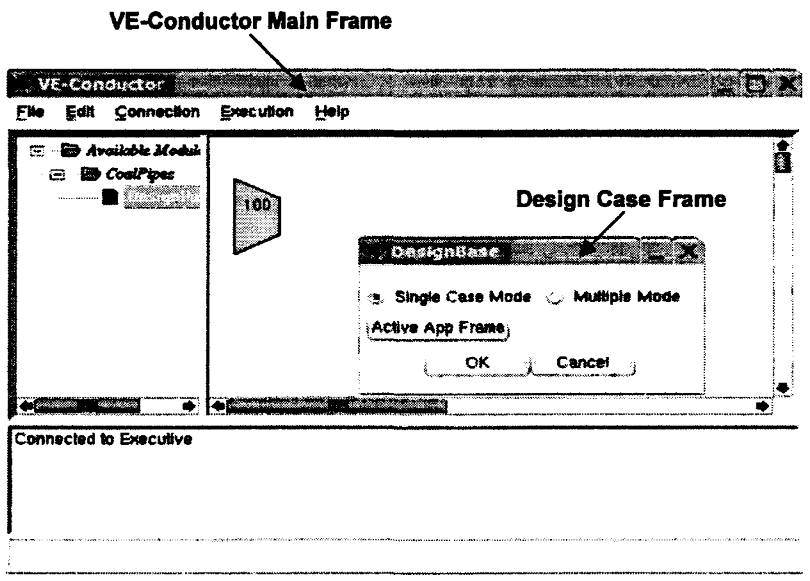
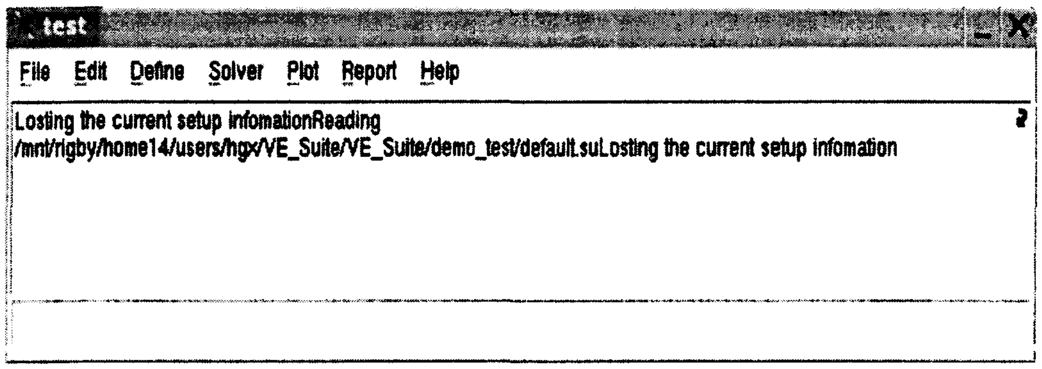

CFD-based EAs optimization process
Transform the problem into Standard form
Set up Analysis method
Setup EAs optimization method
Choose design variables
Set up constrains
Set up object functions
Define geometry
Choose grid generation method
Choose numerical methods
Choose EAs method
Define control parameters

Figure 7.2 User-defined parameters during design set up



Connected to Executive

Figure 7.3 Entrance frame



```
Losting the current setup infomationReading
/mnt/rigby/home14/users/hgx/VE_Suite/VE_Suite/demo_test/default.suLosting the current setup infomation
```

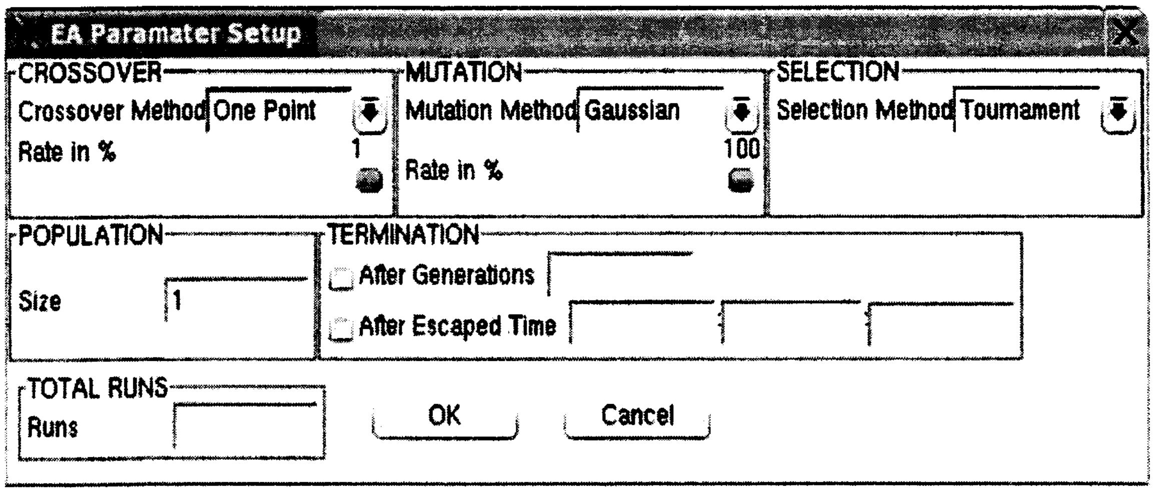
Figure 7.4 Main menu window of the design interface

7.2.1 EAs Control Parameters Setup

When using EAs to solve an optimization design problem, one has to define a vector of design parameters which can be used to define the system. In many situations, one will have tens or hundreds of parameters. In addition to these parameters, EAs typically have the population size, mutation and crossover rates and the number of generations. This poses a problem: what are the correct values to use for each individual application? Some research on this topic has been done, but it usually deals with problems too abstract to be of any use in the real world. Typically, the user tries different settings, perhaps using some kind of visualization to steer the probing (most often a generation/fitness graph). Therefore, the system should give the user the ability to change these EA control parameters. In our system, the EA parameters are pre-defined; Table 7.1 shows the default values. Users can change these values through the GUI provided from the system (Figure 7.5). The user can recall or change any control parameter for the algorithm at any time during or after the completion of the algorithm.

Table 7.1 EA parameters

Name	Data Type	Description	Default value
PopulationSize	Int		32
TerminationMode	Int	0—after generation 1—after time (mins)	1
Time	Int		120
Generations	Int		1000
TotalRuns	Int		1
MutationMode	Int	0—Gausissan 1—Uniform	0
CrossoverMode	Int	0—One point 1—Two points	0
SelectionMode	Int	0-Tournamnet 1-Rank 2-Random	0
MutationRate	Double		0.5
CrossoverRate	Double		0.5


EA Paramater Setup

CROSSOVER
Crossover Method One Point
Rate in % 1

MUTATION
Mutation Method Gaussian
Rate in % 100

SELECTION
Selection Method Tournament

POPULATION
Size 1

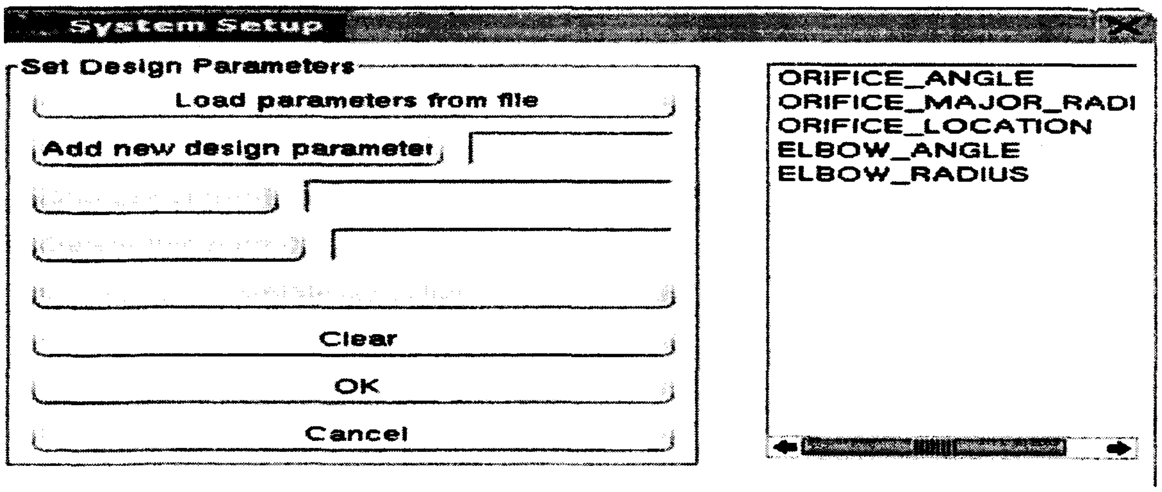
TERMINATION
After Generations
After Escaped Time

TOTAL RUNS
Runs

Figure 7.5 EA control parameters setup dialog

7.2.2 Design Vectors Setup

Users can define the design parameter vector using the GUIs shown in Figure 7.6 or load necessary information such as design parameters' names and their constraints from an existing file. The file format is quite general; for example, one may set the names of four design parameters and possible side constraints on the vector of design variables as shown in Table 7.2. Internally, the system defines four other vectors to store the design information. The advantage of this configuration is that it gives one the freedom to change or edit this pre-defined information. It also allows the system to be easily adapted to different problems without rewriting the definition file for VE-Suite. In many cases, not every parameter has the same impact on the performance of the product. The user can choose the active parameters according to their experience after the potential parameters have been loaded into the system. For instance, as shown in Figure 7.7, only ORIFICE_MAJOR_RADIUS, ORIFICE_LOCATION, and ELBOW_RADIUS have been selected to be the active design parameters in the optimization process.



ORIFICE_ANGLE
ORIFICE_MAJOR_RADI
ORIFICE_LOCATION
ELBOW_ANGLE
ELBOW_RADIUS

Figure 7.6 Design parameters setup dialog

Table 7.2 Example of user-defined file

Parameter Name	Lower Limit	Upper Limit
ORIFICE_ANGLE	0	90
ORIFICE_MAJOR_RADIUS	0.7	0.9
ORIFICE_LOCATION	4.5	9.0
ELBOW_RADIUS	0.5	1.0
ELBOW_ANGLE	60	90

Table 7.3 Other pre-defined parameters

Name	Data Type	Description	Default Value
DesignParamNames	String1D	Stores the names of the potential design parameters	No
ActiveParamNames	String1D	Stores the names of the actual design parameters	No
UpperLimits	Double1D	Stores the upper limits of all potential design parameters	$[0,0,\cdots,0]$
LowerLimits	Double1D	Stores the lower limits of all potential design parameters	$[0,0,\cdots,0]$
ActiveDesignParams Values	Double1D	Stores the values of design the parameters	No

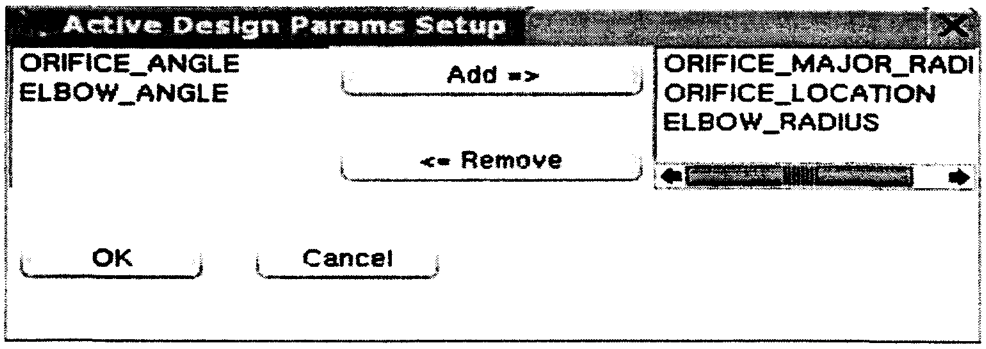


Figure 7.7 Active design parameters setup dialog

7.2.3 Analysis Solver Setup

The analysis solver setup is relatively simple compared to other setups so far since in our project Fluent™ is used as the analysis solver. However, an interface is still provided in case users want to use different packages. Table 7.4 shows the internal variables that store the user's choices.

Table 7.4 Parameters for analysis package

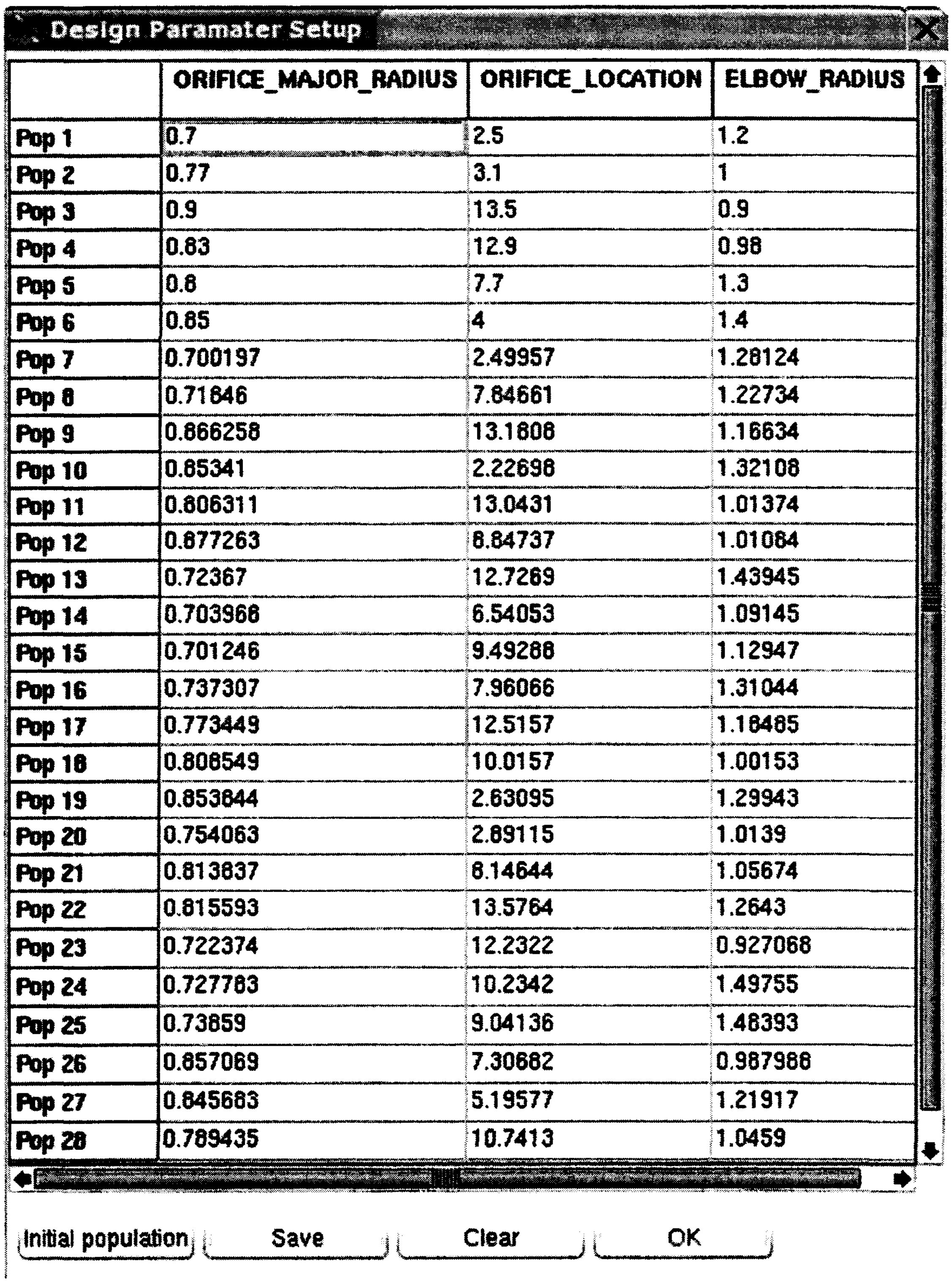
Name	Data Type	Description	Default Value
CFDPackage	String	The name of the CFD package	FLUENT
IS_3D	Int	0 — 2d model 1 — 3d model	1
IS_TurbulentFlow	Int	0 — laminar 1 — turbulent	1
Residules	Double1D	Residules criteria	$[10e-6, 10e-6, \cdots, 10e-6]$

7.2.4 Population Initialization

The initial population specifies the starting points of the search. The initial population can be created in a number of ways. Four approaches are provided by the system: random generation by the computer, loading from an existing file, manual entry and any combination of the above three methods. As an example, in Figure 7.8, 6 out of 32 creatures in the population are assigned by the user, and the other 26 are loaded from a previous run. After the initialization is finished, *ActiveDesignParamsValue* is automatically initialized according to this table.

7.2.5 Fast Setup

Another important feature of the setup process is that once users have gone through the above steps to setup a design problem, he or she can save the newest setup information into a file and reload it later. Users can also make this file themselves. Thus, the eight steps required for setting up the problem are not unnecessarily repeated, and the problem setup can be simplified into one simple step. An example of the setup file is shown in Figure 7.9.



	ORIFICE_MAJOR_RADIUS	ORIFICE_LOCATION	ELBOW_RADIUS
Pop 1	0.7	2.5	1.2
Pop 2	0.77	3.1	1
Pop 3	0.9	13.5	0.9
Pop 4	0.83	12.9	0.98
Pop 5	0.8	7.7	1.3
Pop 6	0.85	4	1.4
Pop 7	0.700197	2.49957	1.28124
Pop 8	0.71646	7.84661	1.22734
Pop 9	0.866258	13.1808	1.16634
Pop 10	0.85341	2.22698	1.32108
Pop 11	0.806311	13.0431	1.01374
Pop 12	0.877263	8.84737	1.01084
Pop 13	0.72367	12.7269	1.43945
Pop 14	0.703968	6.54053	1.09145
Pop 15	0.701246	9.49288	1.12947
Pop 16	0.737307	7.96066	1.31044
Pop 17	0.773449	12.5157	1.18485
Pop 18	0.808549	10.0157	1.00153
Pop 19	0.853844	2.63095	1.29943
Pop 20	0.754063	2.89115	1.0139
Pop 21	0.813837	8.14644	1.05674
Pop 22	0.815593	13.5764	1.2643
Pop 23	0.722374	12.2322	0.927068
Pop 24	0.727783	10.2342	1.49755
Pop 25	0.73859	9.04136	1.48393
Pop 26	0.857069	7.30682	0.987988
Pop 27	0.845683	5.19577	1.21917
Pop 28	0.789435	10.7413	1.0459

Figure 7.8 Example of initial population table

```
<TEST>
   <DesignParamInfo>
        <ORIFICE_ANGLE lowerlimit="0" upperlimit="90" />
        <ORIFICE_MAJOR_RADIUS lowerlimit="0.7" upperlimit="0.9" />
        <ORIFICE_LOCATION lowerlimit="2" upperlimit="14" />
        <ELBOW_ANGLE lowerlimit="0" upperlimit="90" />
        <ELBOW_RADIUS lowerlimit="0.9" upperlimit="1.5" />
   </DesignParamInfo>
   <ActiveDesignParamInfo>
        <ORIFICE_ANGLE ORIFICE_ANGLE="0" />
        <ORIFICE_MAJOR_RADIUS ORIFICE_MAJOR_RADIUS="1" />
        <ORIFICE_LOCATION ORIFICE_LOCATION="2" />
        <ELBOW_ANGLE ELBOW_ANGLE="3" />
        <ELBOW_RADIUS ELBOW_RADIUS="4" />
  </ActiveDesignParamInfo>
  <CreatureChainInfo>
        <Pop0>45 0.7 2.5 60 1.2</Pop0>
        <Pop1>15.9 0.77 3.1 30 1.0</Pop1>
        <Pop2>90 0.9 13.5 20 0.9</Pop2>
        <Pop3>74 0.83 12.9 50 0.98</Pop3>
        <Pop4>81.7 0.8 7.7 27 1.3</Pop4>
  </CreatureChainInfo>
 </TEST>
```

Figure 7.9 Example of setup file

7.3 Implementation of EAs Components into VE-Suite

Once the setup is complete, the task can be submitted to the computational unit. The optimization algorithm (the computational unit) proposed in Chapter 5 can be easily modified to add necessary interactive functionalities so that users are given the capability to influence the process such as altering system parameters or changing objective functions. Unlike typical EAs, our system lies somewhere between the two extremes of typical interactive EAs and traditional machine-based EAs by adding human interaction into the design process.

Using our framework, the user does not determine the behaviour of the computational process as it runs. Therefore, basically the system is still a machine-based optimization.

To better explain the computational unit, which is our design component, we will now describe how to start the system up. The first step in starting the system is to connect the unit with the CORBA Naming Service. This is done automatically through the main function of *DesignBaseUnit_client*. The next step in starting the system is to create and initialize an application; first, an application object (*App*) must be instantiated. The first thing the object does is to create a regular thread that instantiates the Model object inside the *CreateModelThread* method. Once this is done, the thread executes the optimization loop and runs the analysis routines through the *CaseCalc*, *CreatureFactoryBase*, and *CreatureMangerBase* classes. It is worth noting all three of these classes are the same as discussed earlier in Chapter 5. During this process, one can also initialize his or her interactive checking function by creating a view thread through App's *CreateViewThread* method. Figure 7.10 shows the Unified Modelling Language (UML) diagram of this process. The class *POA_Body* inherits from *DesignBaseUnit_i* and provides necessary CORBA interfaces required for framework integration and communication. Like all VE-Suite applications, the two most important methods in *DesignBeseUnit_i* are *SetParams* and *StartCalc*. The starting point of the computational unit is from the *StartCalc()* method. Inside *StartCalc*, *SetParams* is called in order to take input from the GUI that is passed in through the CORBA interface. Part of the implementation of the *SetParams* and *StartCalc* in the design system is as shown in Figure 7.11.

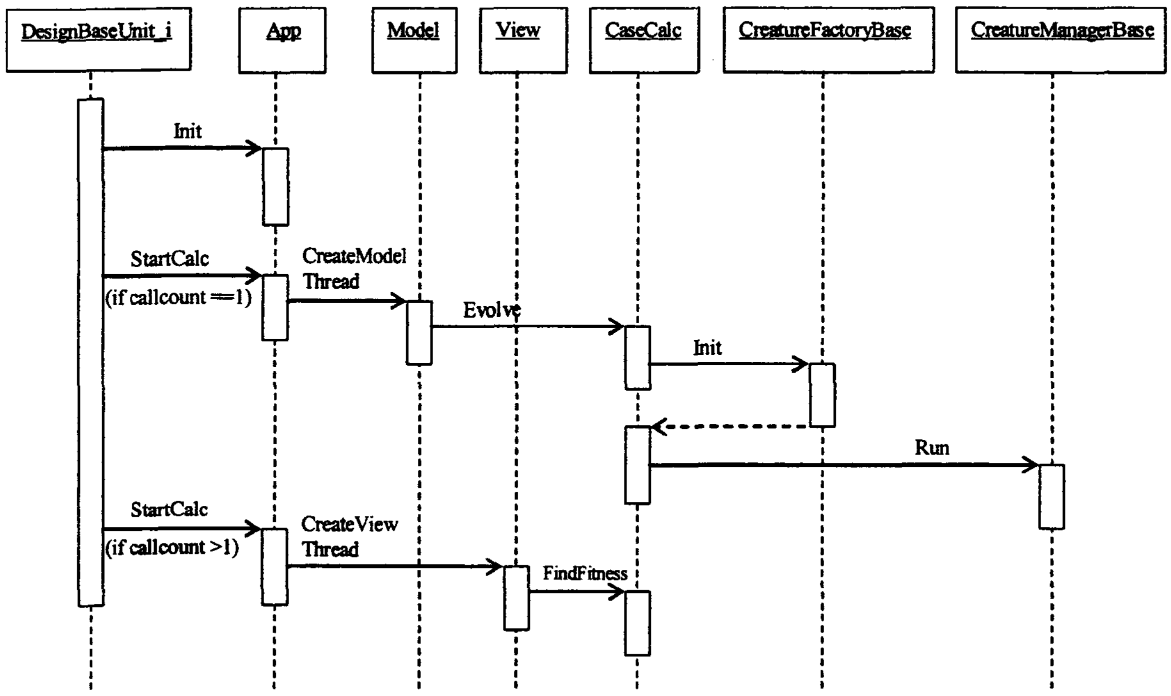


Figure.7.10 A sequence diagram of the initial setup of the process

```
void DesignBaseUnit_i::SetParams ( )
{  // Add your implementation here
   Package p;
   p.SetSysId("gui.xml");
   p.Load(param, strlen(param));
   callcount++;
   popsize =p.intfs[0].getInt("popsize");
   designparams = p.intfs[0].getDouble1D("designparams");
   designparamnames = p.intfs[0].getString1D("designparamnames");
   upperlimit = p.intfs[0].getDouble1D("upperlimits");
   lowerlimit = p.intfs[0].getDouble1D("lowerlimits");
    ......
}

void DesignBaseUnit_i::StartCalc ()
{  //Add your implementation here
   //when it is the first time, create model thread
   if(this->callcount==1)
   {
        myveapplication->CreateModelThread();
   }
  else
   {
        myveapplication->CreateViewThread();
   }
}
```

Figure 7.11 SetParams and StartCalc implementation

A simple *App* class is as an interface between the regulate optimization routine (*Model*) and the *DesignBase_Unit_i* class. A classic Subject/Observer/Mediator design pattern is used amongst these three classes. Figure 7.12 shows the class hierarchy. One of the requirements of an interactive design software tool is the capability to interrupt the iterative process and report the status of the design to the user. In our approach, the *App* class acts as a mediator between the other two classes; it is the observer of *Model* and also the subject of *DesignBaseUnit_i*. As the subject of *App*, *Model* will notify its observer, *App*, when a new result is ready for the user to pick up. As a subject of *DesignBaseUnit_i*, in *App*'s *Update* function, it notifies *DesignBaseUnit_i* class and passes information to it. This Subject/Observer/Mediator trio is commonly known as the Model View Controller (MVC) pattern. This is mainly used when test results need to be sent back to the plug-in GUI so users can assess the latest design information. One advantage of this approach is that the *Model* class requires little modification to extend. For example, since it may be of interest to the designer to visualize the path taken by the EAs optimizer, the *NotifyObservers* function can be added at the end of each mating event loop as shown in Figure 7.13. In this example, at the end of each mating event, *Model* will update its *NotifyObservers* function. Since *Model* is the subject of *App* class, its *NotifyObservers* function will activate *App*'s Update function which in turn activates its own *NotifyObservers*. *App*'s *NotifyObservers* function will then activate the update function in *DesignBaseUnit_i*. Therefore, users can always access the latest information such as the fitness and new creature's geometry whenever they check the result from the GUI and the required modification of the original code is minimized. In the current design tool, the control parameters such as the current max fitness and the current generation's information are updated at the end of the every mating event. Only several additional lines to *DesignBaseUnit_i* class are needed to check additional information. Another advantage is that the main module doesn't need to check the model thread all the time (also known as busy waiting or polling) in order to know when new information is ready; thus saving significant CPU time.

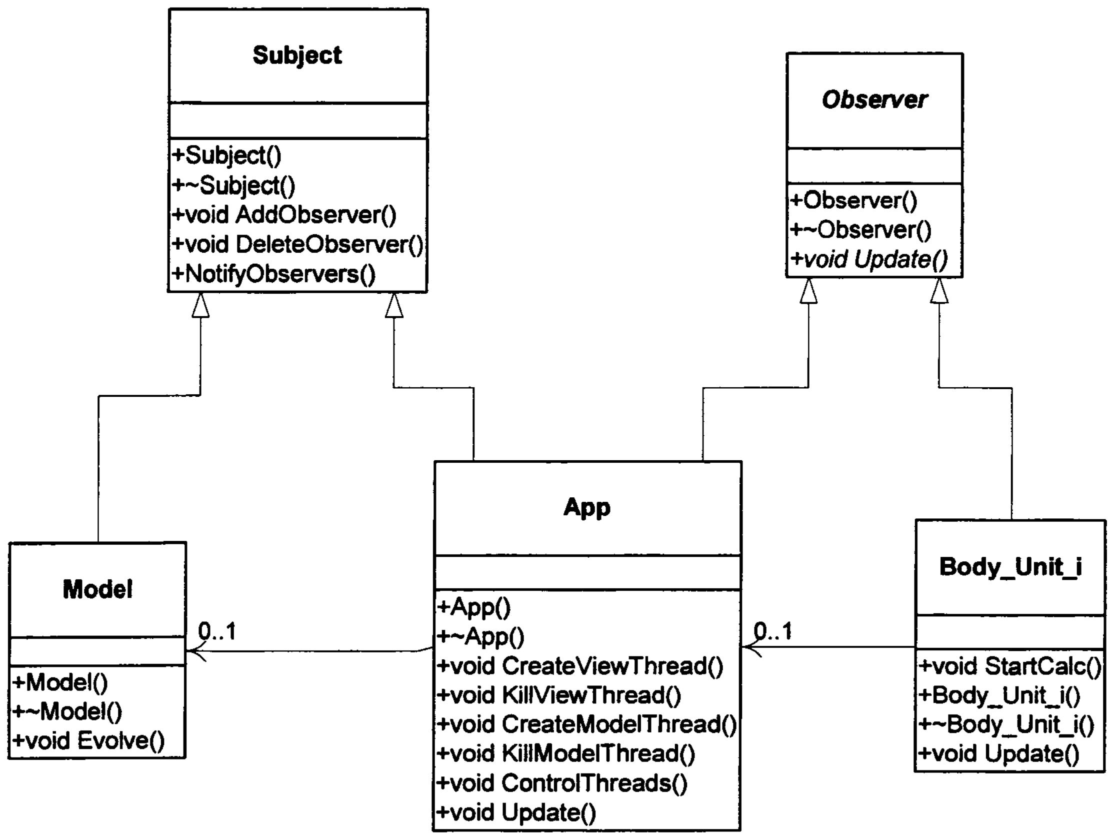


Figure 7.12 Computational unit scheme

```
void Model::Evolve(void* unused)
{ ........
  for(int mev=0; mev<totalmev; mev++)
  { .......
    NotifyObservers();
  }
}
------------------------------------------------------------
void Subject::NotifyObservers()
{
  int iter;
  for(iter=0; iter<_observers.size();iter++)
  {
    _observers[iter]->Update(this);
  }
}
------------------------------------------------------------
void App::Update(Subject* theChangedSubject)
{
  if(theChangedSubject == mModel)
    NotifyObservers();
}
------------------------------------------------------------
void DesignBaseUnit_i::Update(Subject* theChangedSubject)
{
  if(theChangedSubject == myapplication)
    this->UpdateLastModifiedResult();
}
```

Figure 7.13 Connections between the DesignBaseUnit_i, App and Model classes

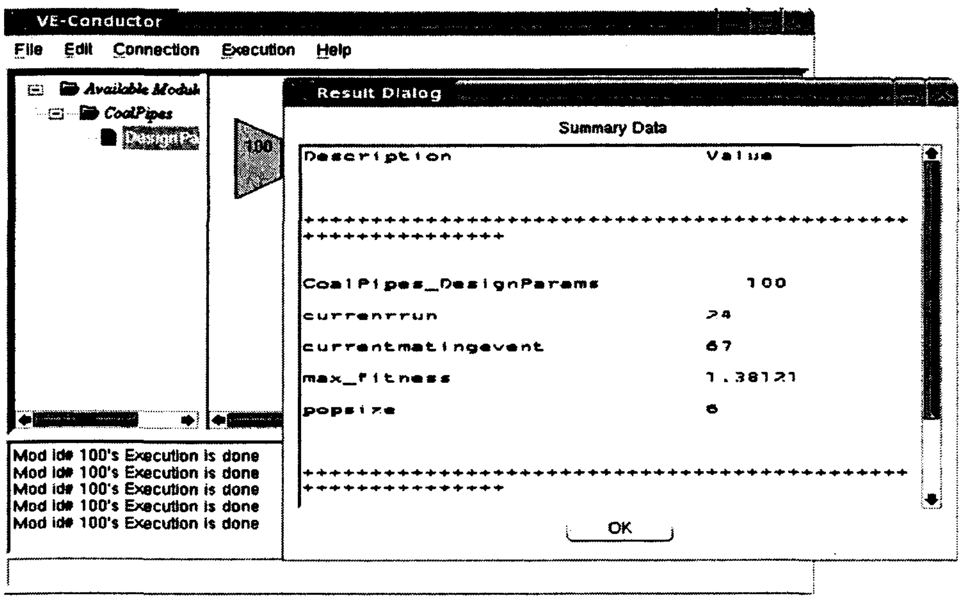
7.4 Implementation of Extracting Information

There is no doubt users will have to need as much information as possible from the system so that they will have a better idea on how to control the system. The question is how the system can best provide useful information to them so that they can interactively influence or guide the direction of evolution of the system such as introducing or removing certain unnecessary design variables, extending or reducing the range of design variation, and so on.

Graphical visualization techniques are some of the simplest and at the same time most powerful methods for analyzing and communication information [119]. In addition, the human vision system is extremely sensitive to graphical patterns, making graphical representations an extremely useful analysis tool. Well designed graphical elements should be in very concise and compact formats at the same time able to convey large amounts of information. In this thesis, users can access the following three kinds of information anytime during an optimization lifetime.

7.4.1 Extract Basic EAs Information

Visualization techniques have been used to study EAs both on-line and off-line in the EAs community for a long time. For an interactive design system, obviously on-line system is required since it allows users to closely follow and evaluate the progress of an EA. However, most of these visualization techniques are too complex for engineers because they are mainly focus on assisting EAs researchers. In this system, thus far only the most basic displays have been used. All the information shown is sent from the computational unit through a CORBA connection. Therefore, users can immediately see quantitative data about the design performance and the search progress. For example, the Result Dialog (Figure 7.14) shows the latest text information from the computational unit such as the current best fitness value and design values in the current generation. By adding the open source software GNUPLOT [135] into the GUI, any design parameter or other useful information such as individual aspects of the population (such as best or average fitness) with respect to iteration can be selected for plotting. A sample of a created plot can be seen in Figure 7.15, which displays the history of fitness variation through the evolutionary process that is summarized in Figure 7.14. Since VE-Conductor can access the database anytime during the optimization process, the user can shutdown the GUI and restart it at will without any impact on the optimization process.

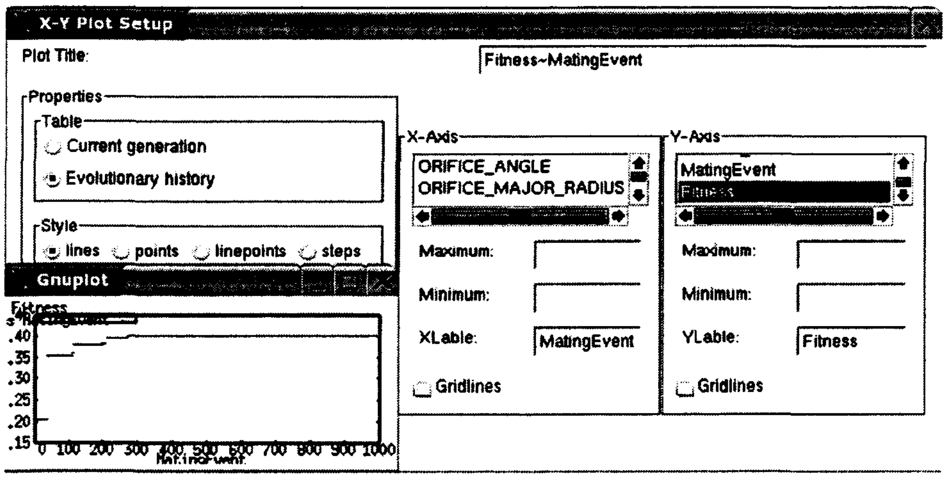


Summary Data

```
Description                                   Value


+++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++


CoalPipes_DesignParams                           100

currenrrun                                    24

currentmatingevent                            67

max_fitness                                   1.38121

popsize                                       6


+++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++
```

```
Mod id# 100's Execution is done
Mod id# 100's Execution is done
Mod id# 100's Execution is done
Mod id# 100's Execution is done
Mod id# 100's Execution is done
```

Figure 7.14 Example of result parameter



Plot Title: Fitness~MatingEvent

Table: Current generation [✓ Evolutionary history]

Style: [✓ lines] points linepoints steps

Maximum:	
Minimum:	
XLable:	MatingEvent
Gridlines	

Maximum:	
Minimum:	
YLable:	Fitness
Gridlines	

Figure 7.15 Example of a result plot

7.4.2 Extract Product Performance Information

In general, engineers not only want to know what designs are better than the other candidates; they also need to understand the physical phenomena that can explain why. Much research has shown that by creating a realistic experience from a computational simulation and maintaining the visitor's focus within that experience, the maximum potential of human thought and skill can be utilized. Therefore, in addition to receiving quick visual feedback about various configurations through traditional two-dimensional plots, the design tool also connects to VE-Xplorer which provides a connection to a three-dimensional virtual environment so that the designer can observe the analysis result of a design to catch the complex fluid physics and identify whether it is the result he or she wanted. Occasionally, those actions require extensive time since the performance data generated from the CFD package is vast which makes the visualization and interaction slow and cumbersome, especially if it needs to be carried out on a different system (say a specialized graphics workstation for visualization). Assuming this kind of task from the user will occur infrequently compared to the other tasks, in order to be able to transfer this large amount of data to the graphical engine without interrupting the overall network communication, the computation unit communicates with VE-Xplorer through VPR sockets instead of CORBA. VPR sockets were chosen for two reasons. First, we already use VPR's threading technology. Second, very often, the computational unit and graphics engine are carried on different operating systems. VPR sockets are cross-platform and can transfer data amongst windows, UNIX, and Linux systems.

Figure 7.16 shows the connection between the computational unit and VE-Xplorer. Fluent™ post-processing functions are first used to export the CFD analysis result in AVS format. VE-Suite provides a translator which can covert an AVS file into the VTK data format since VE-Xplorer is built on top of VTK. Because the amount of data is usually very large, it is natural that the data be compressed before it is sent. The binary VTK data (vtkUnstructuredGrid) is compressed utilizing the compression function VTK provides and then sent through a VPR socket. Therefore, compressed data, the length of the compressed data, and the length of the original data are ready to be sent from the computational unit to VE-Xplorer. When sending data across the network, the computational unit and VE-Xplorer can behave as client and server, respectively. When the View thread starts, it allocates a socket, publishes the sockets port, and listens on that port for incoming connections. Once VE-Xplorer connects to this socket, the cfdVeView class creates a new stream connection for that client-server pair. From there on out, the message protocol allows bi-directional messages between the client and server. The server listens on the client's socket for incoming messages. When a message is received, the server decodes the message and executes any actions requested by the client. Since there are three data components that need to be sent, the above actions will be repeated three times. Once VE-Xplorer gets all three of these data components, it will uncompress it and convert it back to a VTK dataset so VE-Xplorer can use it to visualize. After this step, he or she who is using this design environment can take full advantage of graphical benefits provided by VE-Suite.

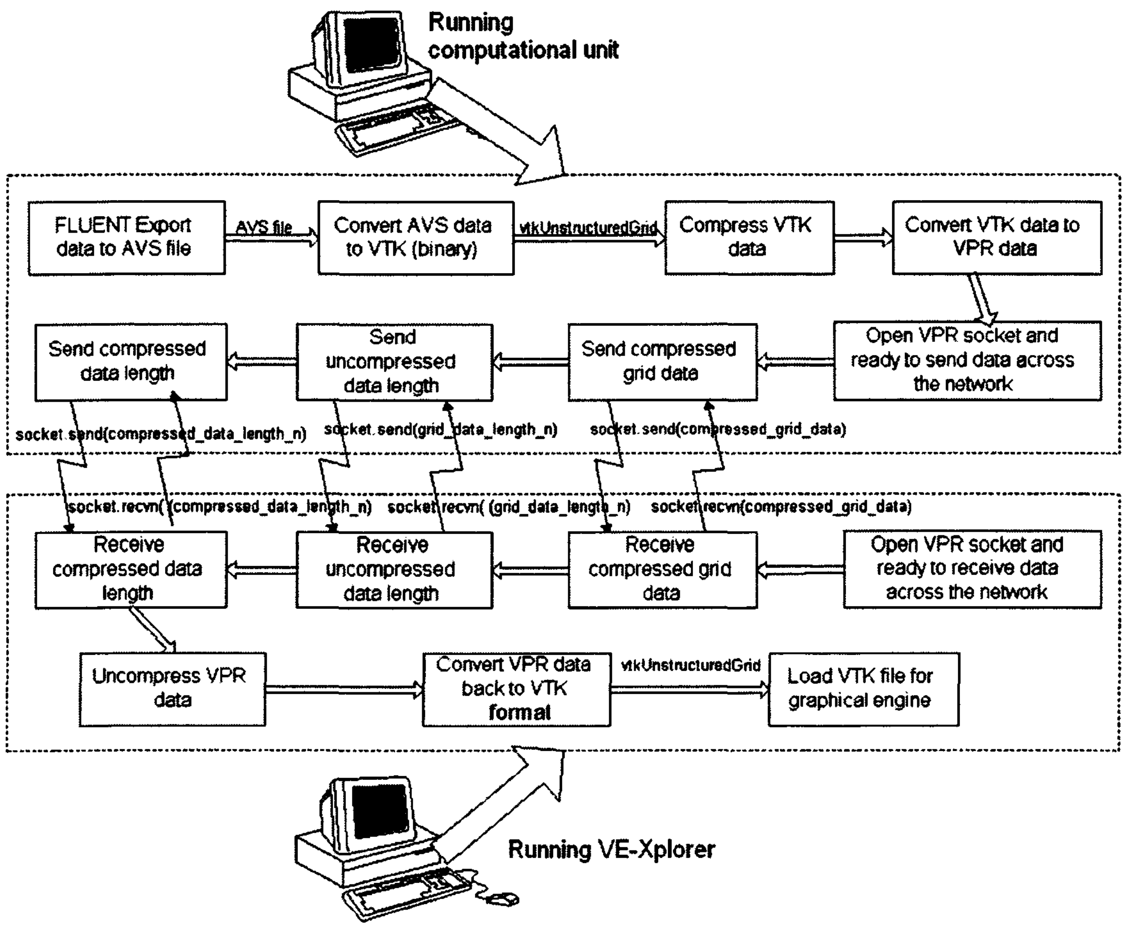

Running computational unit
FLUENT Export data to AVS file
AVS file
Convert AVS data to VTK (binary)
vtkUnstructuredGrid
Compress VTK data
Convert VTK data to VPR data
Open VPR socket and ready to send data across the network
Send compressed grid data
Send uncompressed data length
Send compressed data length
socket.send(compressed_data_length_n)
socket.send(grid_data_length_n)
socket.send(compressed_grid_data)
socket.recvn(compressed_data_length_n)
socket.recvn((grid_data_length_n)
socket.recvn(compressed_grid_data)
Receive compressed data length
Receive uncompressed data length
Receive compressed grid data
Open VPR socket and ready to receive data across the network
Uncompress VPR data
Convert VPR data back to VTK format
vtkUnstructuredGrid
Load VTK file for graphical engine
Running VE-Xplorer

Figure 7.16 Connection between computation unit and VE-Xplorer

7.4.3 Extract Design Space Information

Usually, an optimization procedure not only locates results about the solutions to a specific problem, but also provides a wealth of information about the design space. However, this kind of information is rarely communicated in an effective way to the designer during the search process. In addition to viewing the high-dimension analysis results, our system allows the user to interactively review the design space information from the large amount of design parameter vectors and their fitness values generated during the previous search. This kind of information can be used to help users gain knowledge about the design features of the product which in turn develops the experience and intuition necessary to make the most effective interaction with the search process.

Using knowledge gained from past evaluations to speed up the search process is not new in the literature. Currently, various approaches exist to reduce the number of fitness function evaluations required to reach an acceptable solution by exploiting knowledge of the history of evaluated points. For instance, information gathered from the initial search can be used to build an empirical model that approximates the fitness function to optimize. The approximation is then used to predict promising new solutions at a smaller evaluation cost than the original problem. This process is called evolutionary control [118]. The accuracy of this fitness model depends on how much information should be pulled up from the original objective functions. Therefore, the problem with this approach is that for more complex problems, virtually all individuals should be controlled. This leads to no benefits sometimes. Another example is the Cluster-oriented genetic algorithms (COGAs) proposed by Professor Parmee [111]. COGAs identifies high-performance (HP) regions through the on-line adaptive filtering according to solutions generated from previous fitness evaluations. Once HP regions are defined, the search space can be reduced. Again, the capability of this approach relies upon the size of the data set. For CFD based optimization procedure, generating large points in the design space will cause too much time.

Although the above two approaches are not suited for our goal, finding high-performance regions is undoubtedly very useful for users if they want to influence the search path. The problem usually is that the design space is more than three dimensions and humans simply cannot visualize high dimensional data. However, in two-dimensional space, humans can grasp rough topology information from the distribution of search points. If this n-dimensional space can be mapped to two-dimension space and also maintain the similarities amongst the data set, the user can visually select possible design vectors and send them to the EA as a new possible parent in the middle of the search. We expect the combination of humans' superior abstract thinking and computers' superior computational speed can work together to produce better performance than either could do alone. Advances in domain knowledge and in areas such as information technology offer the possibility of accelerating and improving human interaction in the design process.

The Self-Organizing Map (SOM) is a neural network algorithm which is especially suited for the analysis and visualization of high-dimensional data. It is a data visualization technique invented by Professor Teuvo Kohonen [132] which reduces the dimensions of data to make high dimensional data sets more understandable. Since its introduction in 1984, many variants of SOM have been created, but the basic philosophy remains the same: unlike other neural networks, the SOM tries to preserve the topology of the input data; it maps nonlinear statistical relationships between high-dimensional input data to simple geometric relationships on a two-dimensional grid. The mapping roughly preserves the most important topological and metric relationships of the original data and, thus, inherently clusters the data; in other words, the SOM groups similar items together. In clustered areas the original reference vectors will be close to each other, and in the empty space between the clusters the vectors will be sparse. Thus, the cluster structure in the data set can be visualized by displaying the distances between reference vectors of neighboring units. Thus SOMs accomplish two things; they reduce dimensions and display similarities.

Unified distance matrix (U-matrix) and component planes are the two most widely used visualization techniques used in SOM. U-matrix shows distances between neighboring map units, and helps to see the cluster structure of the map: high values of the U-matrix indicate a cluster border, uniform areas of low values indicate clusters themselves. Component plane visualizes the values of one variable in each map unit. It shows what kind of values the prototype vectors of the map units have for different vector components. The U-matrix and component planes are linked by position. Figure 7.17 shows the U-matrix and the four component planes of the map for the well-known Iris data set. This data set consists of four measurements from 150 Iris flowers: 50 Iris-sectosa, 50 Iris-versicolor and 50 Iris-virgincia. The measurements are length and width of sepal and petal leaves.

One can refer to the color bar for U-matrix to see which colors mean high values in Figure 7.17. Here the dark regions show the close connections in U-matrix plane which represent the clusters and the light areas comprises the cluster borders. A rough inspection on the U-matrix gives the idea about possible two classification regions (the top three rows and the bottom 7 rows) after training. By looking at the labels, one can immediately see that the top region corresponds to the Setosa subspecies. The two other subspecies Versicolor and Virginica form the other cluster. The U-matrix shows no clear separation between them, but from the labels it seems that they correspond to two different parts of the cluster. From the component planes it can be seen that the petal length and petal width are very closely related to each other since these two component planes are almost identical which indicates that petalL and petalW have linear correlation. Also some correlation exists between them and sepal length. The Setosa subspecies exhibits small petals and short but wide sepals. The separating factor between Versicolor and Virgincia is that the latter has bigger leaves [134].

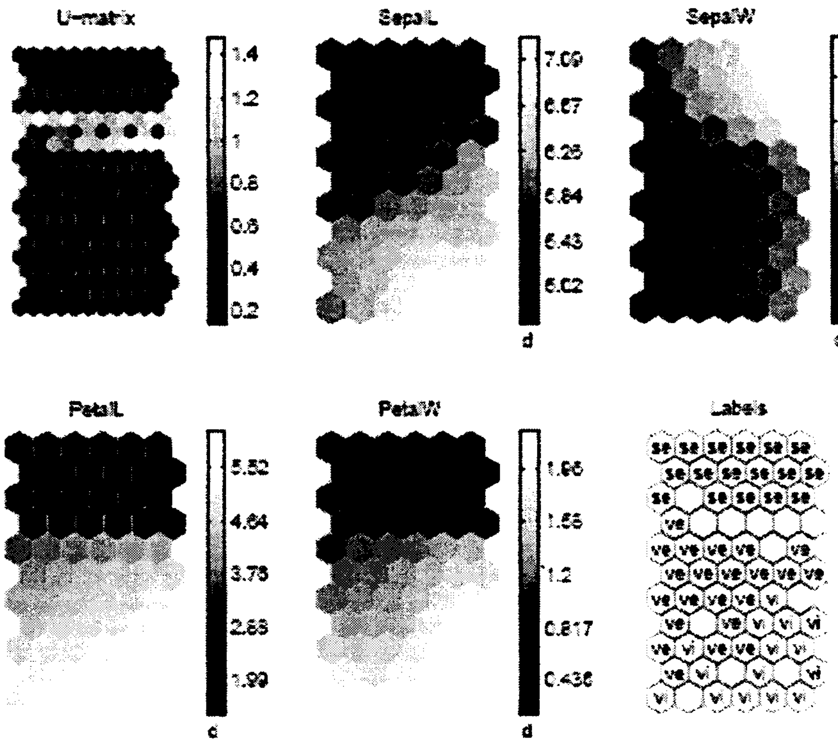


Figure 7.17 Visualization of SOM of Iris data [132]

One of the drawbacks of SOM analysis is that unlike other cluster methods, the SOM has not distinct cluster boundaries. When data sets become more complex which often happens in the design space, it is not easy for human to distinguish the cluster by pure visualization. The choice of the best cluster can be determined by the Davies-Boulding Index [133]. It is a function of the ratio of the sum of within-cluster distance and between-cluster distance. Optimal clustering is determined by V_{DB}, which can be calculated as :

$$V_{DB} = \frac{1}{N}\sum_{k=1}^{N} \max_{k \neq l} \frac{S_N(D_k)+S(D_l)}{T_N(D_k,D_l)} \tag{7.1}$$

Where N is the number of clusters, D is a matrix of the data set X. S_N is the within-cluster distance between the points in a cluster and the centroids for that cluster and T_N is the between-cluster distance from the centroid of one cluster to the other. The optimal number of clusters is the one that minimizes V_{DB}. If the clusters are well separated, then V_{DB} should decrease over time as the number of clusters increases until the clustering reaches convergence. The left figure in Figure 7.18 shows Davies-Boulding Index for the same Iris data set and the right figure shows the SOM cluster by color code which is minimized with best clustering. As shown, the data set contains two clusters and the border is clear too.

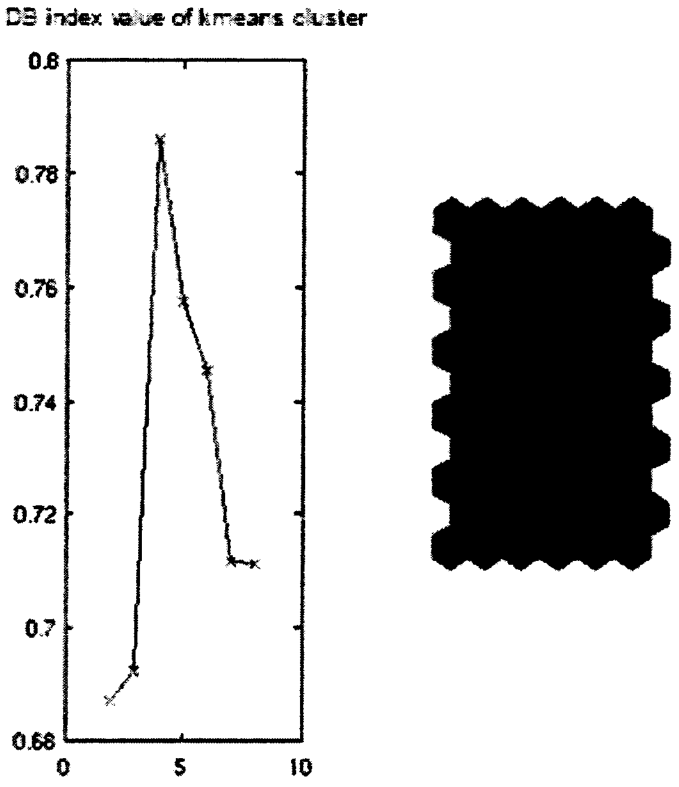


Figure 7.18 Davies-Boulding Index and SOM cluster for Iris data set

SOM ToolBox [134] is used to generate above graphs in this thesis. SOM Toolbox is an easy-to-use implementation of the SOM in Matlab^{TM}. It can be used to preprocess data, initialize and train SOMs using a range of different kinds of topologies, visualize SOMs in various ways, and analyze the properties of the SOMs and data. Because SOM Toolbox is built on top of Matlab^{TM}, it can be added into the design environment seamlessly.

Since the system would accumulate the data set from previous evaluations and generate proper graphics, the users' responsibility is to define the thresholds according to fitness values so that these prototype vectors can be labeled. Figure 7.19 shows the labels using this tool. In general, the design space can be divided into three regions: Bad, Middle, and Good. With a little training on how to analyze the data through U-matrix, components plane, by observing these graphics together, users can identify the characteristic of the design space easily and it will in turn help users to guide the search process.

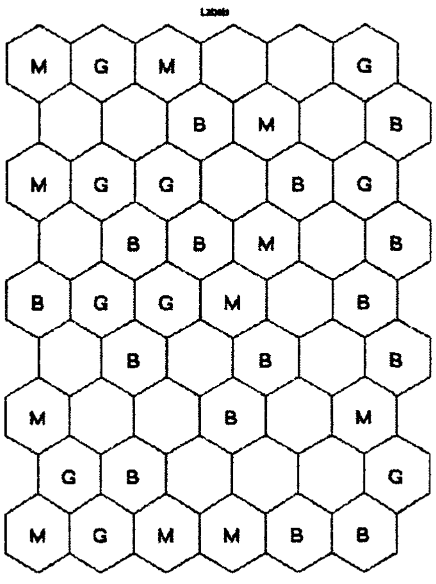


Figure 7.19 Example of labels in the system

7.4 Performance

It is known that when using EAs as optimization algorithms to search optimal design a set of healthy initial population can help reach the global optima much faster and easier. The most common setup for initial population is the random initialization where the creatures are randomly assigned, preferably using a uniform distribution. The goal is to create a population with a good coverage of the search space, and thereby have a gene pool with good potential for breeding better solutions. An alternative approach is to incorporate expert knowledge into the initialization. In some cases, it is possible to assign the initial search space positions based on specific knowledge about the objective function. Domain experts usually have an idea of what a reasonably good solution is. In other cases, especially when CFD simulations are involved, designers can not decide which design is superior by simply looking at the parameters. However, a designer can observe the analysis result of a design and identify whether it is the design he/she wanted, given that the result is shown in a realistic and intuitive manner. Since our design system provides the tool that users can check their design performance easily, we evaluate how human intervention accelerates the convergences of the EA search by establishing a test which designers can select better initial creatures in our design environment. A problem with such an initialization is that the search may be too focused on the area around the special solution. In our test case, designers defined the first 24 members (from 58 tries) based on their experience and intuition; the computer randomly generated the remaining 8 members to keep diversity amongst the population and allow the EA to discover fundamentally different solutions in comparison with what a human would have proposed.

In order to compare the convergence performance of the interactive design tool and the normal EAs, the same test case used in Chapter 5 is used here. Figure 7.20 shows the variations of the best individual's fitness during the evolution process in the run. Two randomly and user-chosen seeded optimization runs tend to converge toward each other as the number of generations increases, even though they follow different paths. Indeed, after sufficient generations this effect tends to disappear, indicating the robustness of the optimization technique. It also shows that with a good initial population, EAs can find the optimal solution much faster. In our case, the required mating events reduce from 650 to 330. Figure 7.21 shows that several designs can be loaded into the virtual environment for comparison. Figure 7.22, 23, and 24 show the optimum parameters such as orifice radius and orifice angle, are convergence to very close values with different starting initial design candidates. Table 7.5 shows that the total time reduction by using the virtual engineering design tool can be as high as 97%

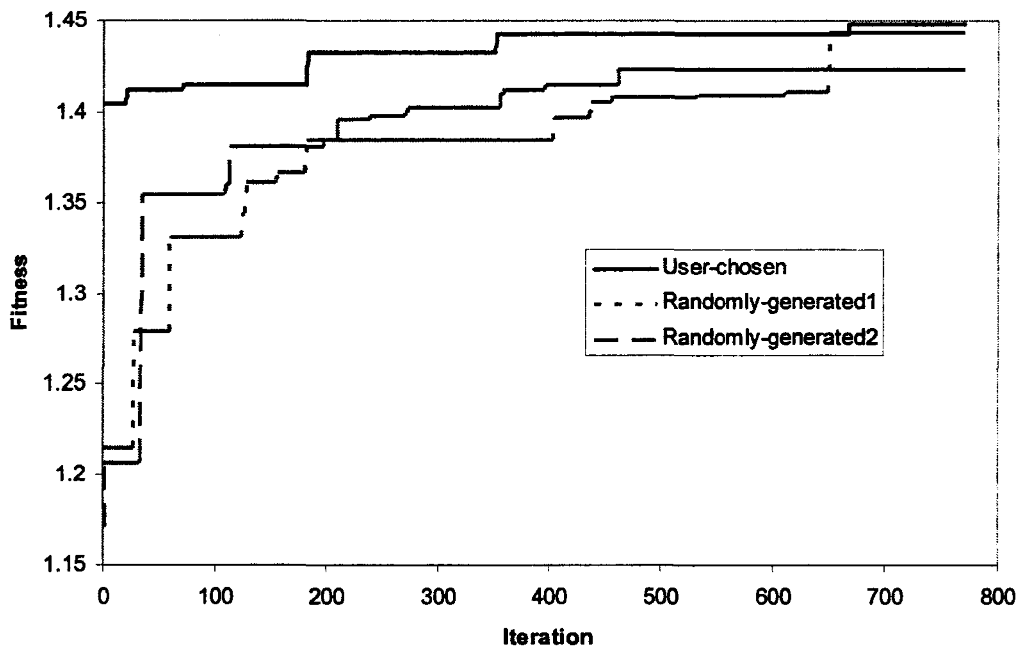


Figure 7.20 Fitness Variations in the evolution processes

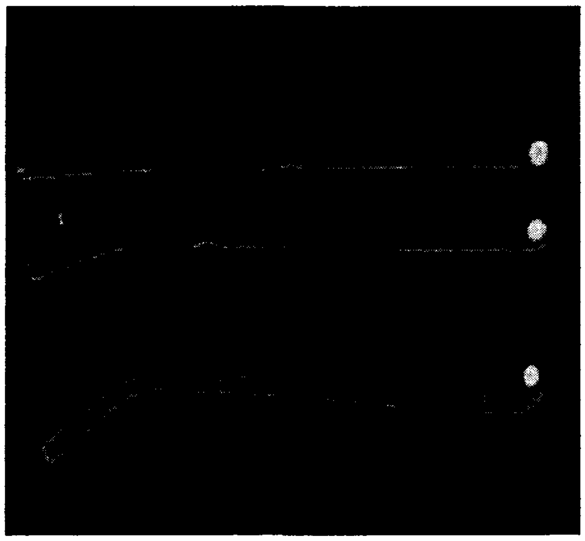


Figure 7.21 Screen Shot from virtual environment

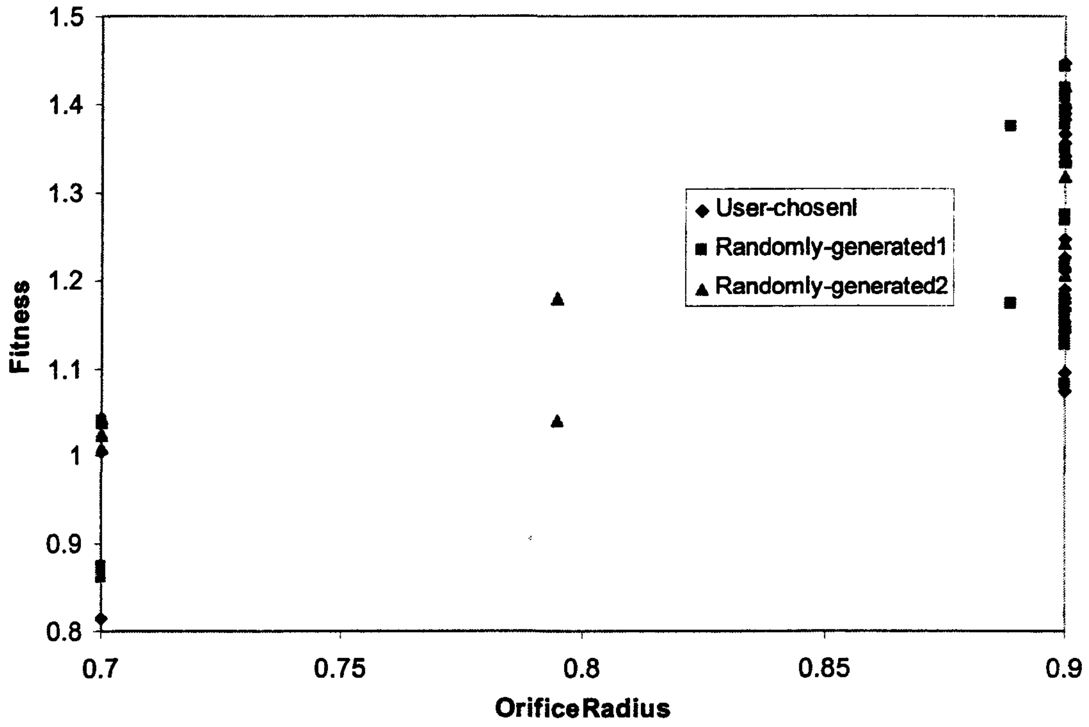


Figure 7.22 EA optimization results for orifice radius

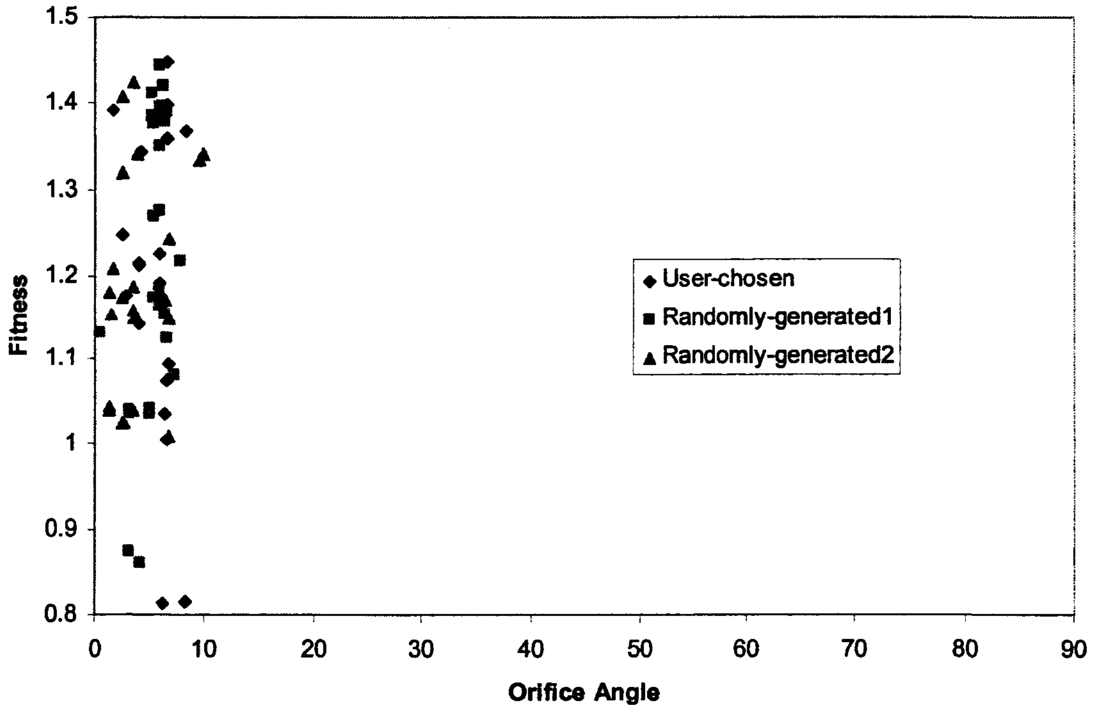


Figure 7.23 EA optimization results for orifice angle

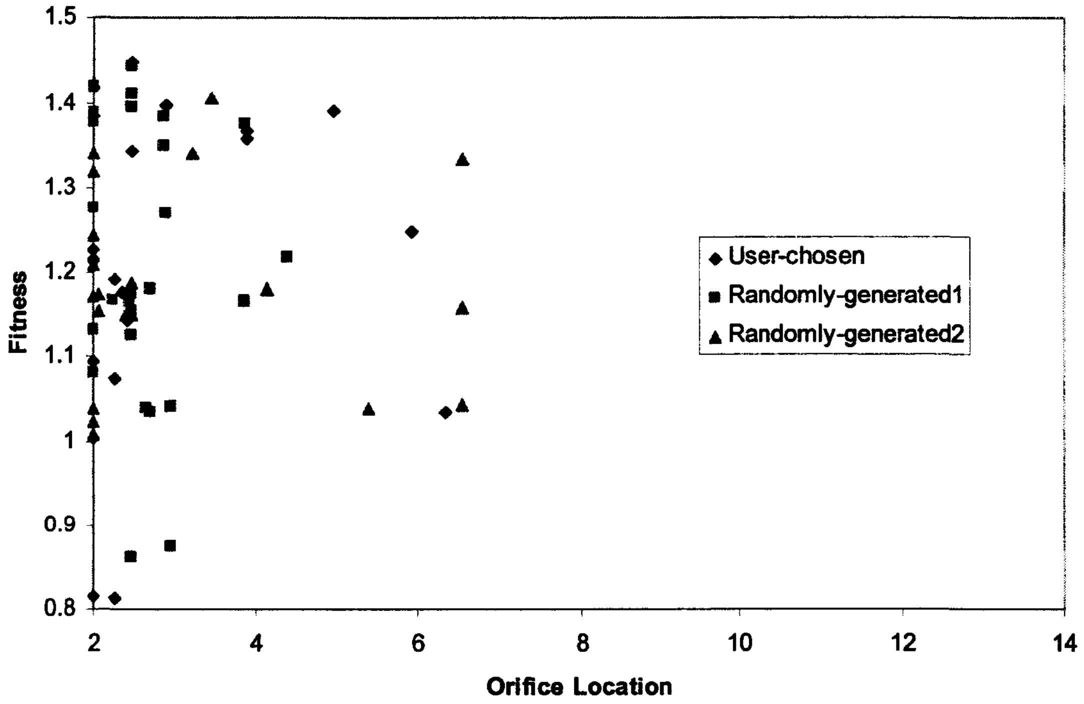


Figure 7.24 EA optimization results for orifice location

Table 7.5 Comparison between traditional design and proposed interactive design

		Initial pop size	Mating events	Total calls to CFD iterations	Time for single CFD iteration (min)	Total time (h)	Time reduction
Traditional numerical design	Original two-cycle	32	650	2632	15	658	
Virtual engineering design	Modified two-cycle	32	650	2632	0.8	35	95%
	User defined initial population	32	330	1378	0.8	18	48%
							97%

7.5 Conclusions

In this chapter, we introduced the basic software implementation framework of virtual engineering design tool. The creation of the general GUI has made the use of the VE-Suite based interactive design system much easier for engineers. By having the interface, the tedious preparation of the necessary input files or coding for PlutIn GUI has been eliminated. The end user does not even have to be familiar with the specific requirements of the VE-Suite framework.

As a VE-Suite application, the EAs model discussed in Chapter 5 can be treated as a regular computational unit. The focus here is how to add human interaction functionalities into the design environment. Thanks to the structure that VE-Suite provides, the implementation is not difficult. By adding a common data base and utilizing the multi-threaded approach, the effort of adapting the existing EAs optimization code into this interactive design space is minimized.

When optimizing expensive fitness functions, it is important to reduce the number of function evaluations required as much as possible. This can be achieved by introducing designers into the optimization process. In order to help users make better decisions, the current system allows users to access three different types of important information:

- *EAs information using traditional two-dimensional plots;*
- *Physical phenomena using three-dimensional virtual reality technology;*
- *Design space characteristic using SOM maps trained from past evaluated points.*

It is reasonable to believe these graphical elements generated by the system can provide design engineers with much relevant information which can be discussed and analyzed both subjectively and objectively by the designer.

By introducing designers into the optimization process and implementing the fast calculation algorithm, we were able to reduce the total computational time by more than 90% for this test case. Hence, it is safe to say, for a simple CFD related design optimization problem, this virtual engineering design tool can help designers solve the problem within a much shorter time. For an interactive design system, a significant problem is user fatigue. For our fairly simple test case, 50 seconds are needed for a single CFD analysis and 18 hours to find the set of a better design. Therefore, the calculation time is still somewhat disappointing with respect to our expectation for "real-time" analysis and optimization. More efforts should be focus on this issue later.

Chapter 8 Coal Piping System Design

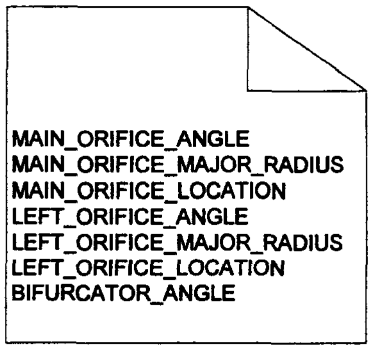
This final chapter presents the shape optimization of a real coal pipe design using the developed tools. The baseline coal pipe problem, addressed in Section 4.3, has the objectives of tailoring distribution of the coal flow rate to its two branches as equal as possible and also tailoring a particle distribution profile upstream to a bifurcator to become as uniform as possible. This pipe has been chosen because it is the only pipe among the 28 coal pipes in the coal-fired power plant that we were able to gather data from and validate the CFD simulation results with. The main objective of this chapter is to demonstrate considerations on the practical implementation of coupled analysis and optimization using the current design environment.

8.1 Problem Analysis

Like other optimization of engineering problems, analysis of the problem specific requirements is always the first step in setting up the optimization process. For this case, the problem definition is mainly on how to choose the proper design parameters.

The problem considered here is a pipe with nine bends, two orifices and one bifurcator in the system. The shape of the pipe is represented by a number of variables such as the degree of bends, the length of pipes, the geometrical definition for orifices, and so on. The set of parameters can vary according to the degrees of freedom wanted by the user. In the current design case, choosing the proper design parameters is based on the author's observation from the CFD result of the baseline model.

Originally, two circular-shaped orifices have been installed to balance the flow resistances. From our preliminary result from Section 5.4, elliptical-shaped orifice can increase the mixing mechanism. Therefore, in this section, elliptical instead of circular orifices are used again. The orifice A is placed in the main pipe upstream to the bifurcator to prevent the unbalance flow rates among the four pipes that extend from the same mill. The orifice B is placed in the left branch of the pipe after the bifurcator to balance the flow resistance between these two pipe branches downstream to the bifurcator. In current power plant design, the locations of these two orifices in the pipes are undefined. They are mainly installed in the pipe wherever it is convenient for installation and replacement. However, our preliminary results show that when the coal rope exists in the pipe, the location of orifice is important in order to maximize the dispersion effect of the orifice inserts into the flow field. Assuming the accessibility for convenience of installation and maintenance is granted wherever the location of the orifices, orifice A is put in the long horizontal pipe after the horizontal-to-vertical bend and the orifice B is put in the long vertical pipe after the horizontal-to-vertical bend. Both orifices can be moved along the length of the pipe. Also, since our goal is to balance the coal flow rate, we will keep the pressure drop consistent. Therefore the opening of the orifice which determines the pressure drop is constant through out the whole design process. Based on this, the orifice's radius can be modified as long as the opening of the orifice remains unchanged. As discussed in the previous chapters, the angular position of the rope is found to depend strongly on the pipe geometry especially the length of the pipe between two bends. Therefore, although it is omitted in practice, in our study, the orientation of the bifurcator is also taken into consideration. The assumption is that if the center lines of the bifurcator are aligned with the center of the rope, it might be able to distribute the coal flow more uniformly. Therefore, as indicated in Figure 8.1, in this application, seven design variables are applied to control the objective function. Figure 8.2 shows the sketch of the geometry of the design pipe.



MAIN_ORIFICE_ANGLE
MAIN_ORIFICE_MAJOR_RADIUS
MAIN_ORIFICE_LOCATION
LEFT_ORIFICE_ANGLE
LEFT_ORIFICE_MAJOR_RADIUS
LEFT_ORIFICE_LOCATION
BIFURCATOR_ANGLE

Figure 8.1 Design variables

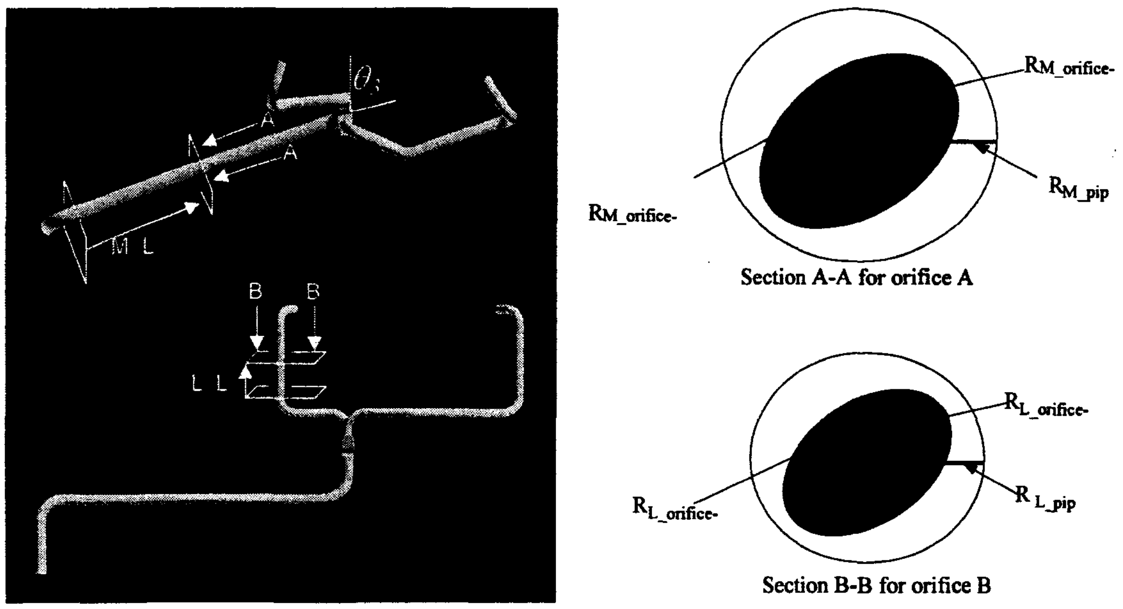


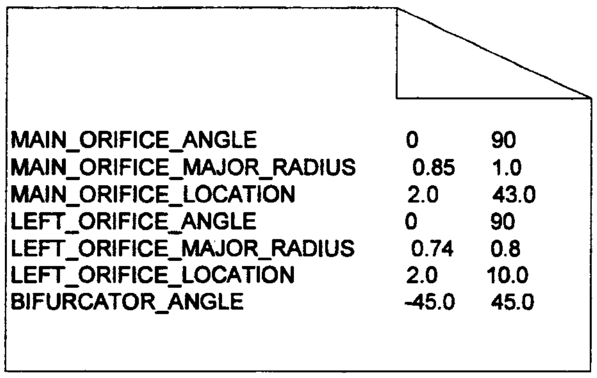
Figure 8.2 Sketch of the design pipe geometry

8.2 Problem Setup

In this section, the process setup using the developed tool is presented. The design tool provides a well-suited starting point for this work due to the various existing modules and the advantage that these have a shared organization of data. Now, based on the previous discussion, setting up an optimization problem in the developed tool starts by the following preprocessing steps.

— ***Define the design variables***. This is mainly done in the step of problem definition.

— ***Define the constrain file***. Seven inequality constraints are used in the optimization to obtain realistic configurations. Figure 8.3 shows the complete constraint file for this model.



```
MAIN_ORIFICE_ANGLE              0      90
MAIN_ORIFICE_MAJOR_RADIUS       0.85   1.0
MAIN_ORIFICE_LOCATION           2.0    43.0
LEFT_ORIFICE_ANGLE              0      90
LEFT_ORIFICE_MAJOR_RADIUS       0.74   0.8
LEFT_ORIFICE_LOCATION           2.0    10.0
BIFURCATOR_ANGLE                -45.0  45.0
```

Figure 8.3 Constraints for design variables

In addition to the inequality constrains mentioned above, two side constraints are used to ensure that the openings of the two orifices are consistent with the baseline design. Therefore only one radius of the elliptical orifice is the independent variable. Note that the opening constraints may be refined in the future when a more detailed system analysis is performed. It is worth mentioning that users can manually input this information through the GUI the system provided.

— ***Make GAMBIT and FLUENT journal files***. Making a general GAMBIT journal file so that GAMBIT can automatic generate meshes for different design cases is usually a time-consuming process. Very frequently, it will take several months to find a suitable grid for a special model. For this case, the most difficult geometry part is the bifurcator, which has been discussed in Chapter 4. Since we already have baseline model, the process of geometry parameterization and mesh generation is relative simple. However, because the geometry is very complex, it turns out two general journal files are needed in order to avoid a collapse of the computational mesh around the bifurcator. During the development process, the easy-to-use capability of the design system allows the switch between different meshes and much more convenient to test results. On the other hand, FLUENT journal is the same for different design cases.

— ***Define fitness evaluation functions***. The design of a coal transport system for a coal-fired power plant needs to address two main objectives:

First, the piping system should deliver equal mass flow rates (gas and solid) at the main outlets (left and right) since the presence of areas of rich combustion results in increased NOx emissions and a non-homogeneous temperature distribution may damage the turbine blades. The objective function measuring the coal flow balance can be expressed as:

$$F(\vec{x}) = \sqrt{\frac{1}{2}\left(\left(Q_L - \overline{Q}\right)^2 + \left(Q_R - \overline{Q}\right)^2\right)} \tag{8.1}$$

Where $\vec{x}$ is the design variable vector, $Q_L, Q_R, \overline{Q}$ denotes the coal mass flow rate associated with the left and right outlet, and the average outflow rate respectively.

Second, the particle distribution at the cross-section right before the bifurcator should be tailored as uniformly as possible. This is an important factor contributed to extend the distributor's life and meet the requirement of flow splitter. The pneumatic transport of pulverized coal can create serious erosion problems that lead to unscheduled outages, downtime, and lost revenue. As a device designed for the destruction of particle ropes, it is extremely exposed to particle erosion. Using the bifurcator shown in Figure 8.4 as example, it is obvious that the bottom half of the bifurcator was suffered more serious erosion from incoming gas-solid flow. It is not only worn out, but also twisted especially the left side. This indicates that most of the particles enter the channel through this region. Although reducing erosion is not the focus of this thesis, it is reasonable to believe that if the particles enter the channel more evenly the damage to the bifurcator will be much slower. Also, most researchers believed that if particles evenly distribute on the cross-section right before the bifurcator, it is most likely the bifurcator can do its job as flow splitter more efficiently [9]. The object function measuring the quality of the particle distribution profile is determined by the standard deviation of the particle concentration in the cross-section right before the bifurcator *MI*,

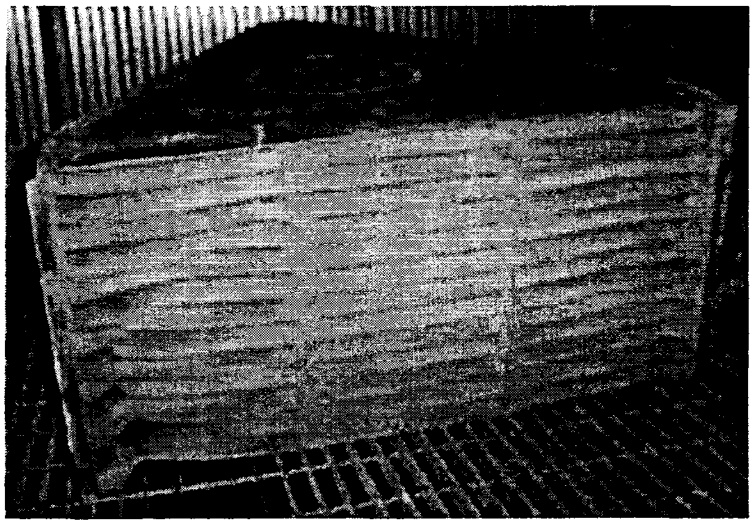
$$MI = \frac{1}{C_p}\left[\frac{1}{n-1}\sum_{k=1}^{n}\left(C_m(k)-C_p\right)^2\right]^{1/2} \tag{8.2}$$

Where C_p is the mean particle concentration in the pipe's cross-section, $C_m(k)$ is the local particle concentration measured at different locations on the surface, and n is the total number of cells on the surface.

Since the correlation between these two objectives is unclear, the weighted-sum approach is chosen to linearly combine these two objectives into one fitness evaluation function. Therefore, the optimization problem can be simplified as a single objective instead of a multi-objective optimization problem.

— ***Add fast calculation method into the flow simulation.*** This is an option choice. Since every application will have different fitness definition, it is up to the user to design whether he or she will combine the fast calculation method discussed in Chapter 5 into the optimization search process. For this case, again all computations were performed on a PC with XEON processor running under the operating system RedHat Linux. At least six hours are needed to get a fully converged CFD result. However, since the two fitnesses defined in the last step will only need data from three cross-sections (leftout, rightout, and cross-section before bifurcator) we monitored the fitness changes on these three planes. Whenever the fitnesses based on the data from these three planes have converged, the CFD simulation for the current case stops and the system automatically starts a simulation for the next case. Hence, it takes roughly one and a half hours for the flow solver simulation to run before the fitness information can be used for the optimization search.

— ***Choose suitable EAs parameters.*** This can be simply done by choosing the right EAs parameters from GUI.



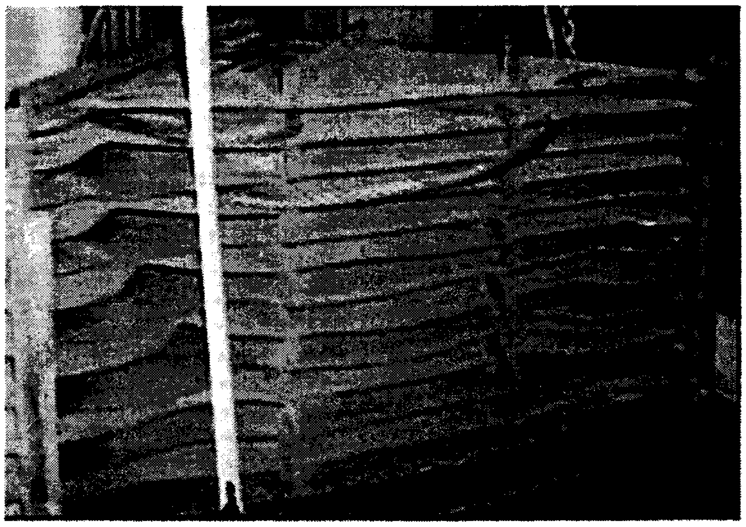


Figure 8.4 A used bifurcator

8.3 Convergence History

Once the problem setup is completed, the optimization search can start. One run was carried out in this study; it has an initial population of 32 and stopped when the best fitness from each matching event converge. The human intervention was carried along during the search process whenever the author wanted to. Since a CFD case takes six hours to fully converge or one and half hour if the fast calculation method is used, how human intervention helps to accelerate the convergence of EA search is hard to evaluate. Therefore, we did not compare the convergence performance of this design tool with the traditional EA design method.

Figure 8.5 to 8.7 show the SOM results from the first 32 cases. Corresponding to the cluster distribution, the connections between these seven design variables can be found through component planes. For example, for a good region, the main orifice should be put close to the end of the horizontal pipe, its centreline should be close to the pipe and the ration of its two radiuses should be large. On the other hand, for left orifice, it should be close to its upstream bend, and its centerline should be aligned with the pipe's and also it should be an elliptical shape instead of circular. The bifurcator's install angle should be close to 45 degrees. Bear in mind, this result was based on a small data set. With the search going, the database will increase gradually. Therefore, the SOM result will be more accurate. Figure 8.8 to 8.10 show the SOM results from a data set with 120 cases. As we can see, the good region has different requirements for design variables. In order to compare with the traditional method of visualization high dimension data set with the SOM method, Figure 8.11 shows the correlations between the single design variable with fitness using the same 120 cases result. As shown in Figure 8.11, it is very hard for users to gain knowledge from it.

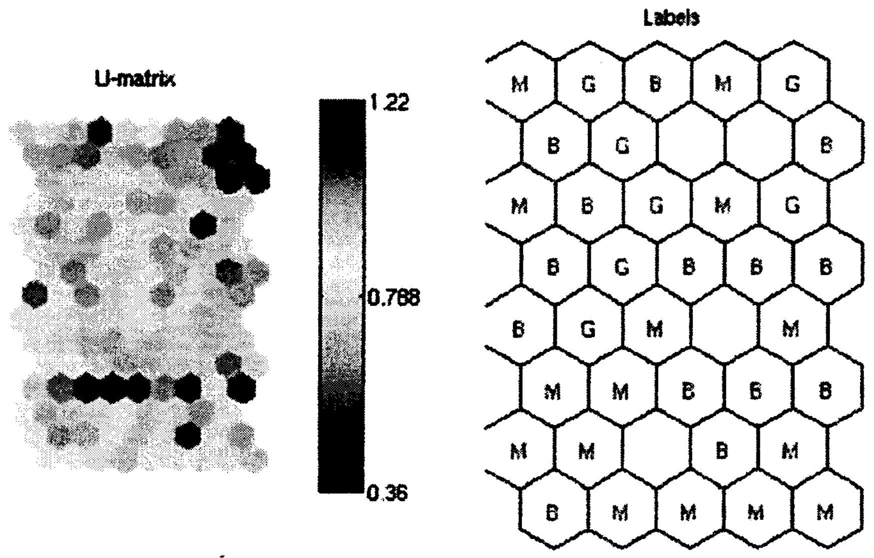


Figure 8.5 U-matrix and labels for first 32 cases from initial generation

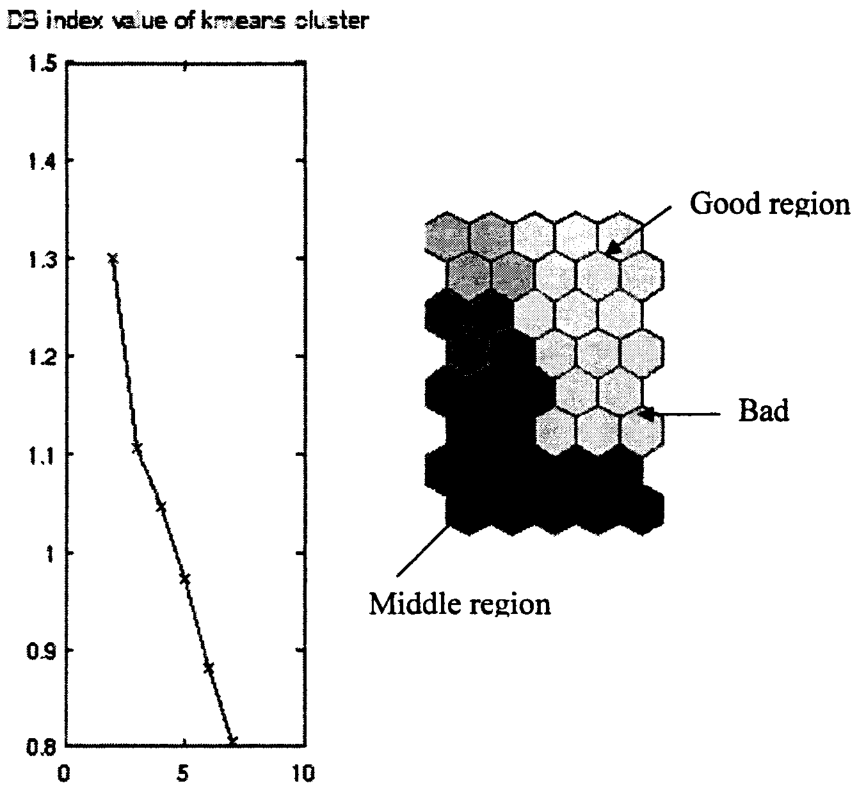


Figure 8.6 Davies-Boulding Index and SOM cluster for first 32 cases

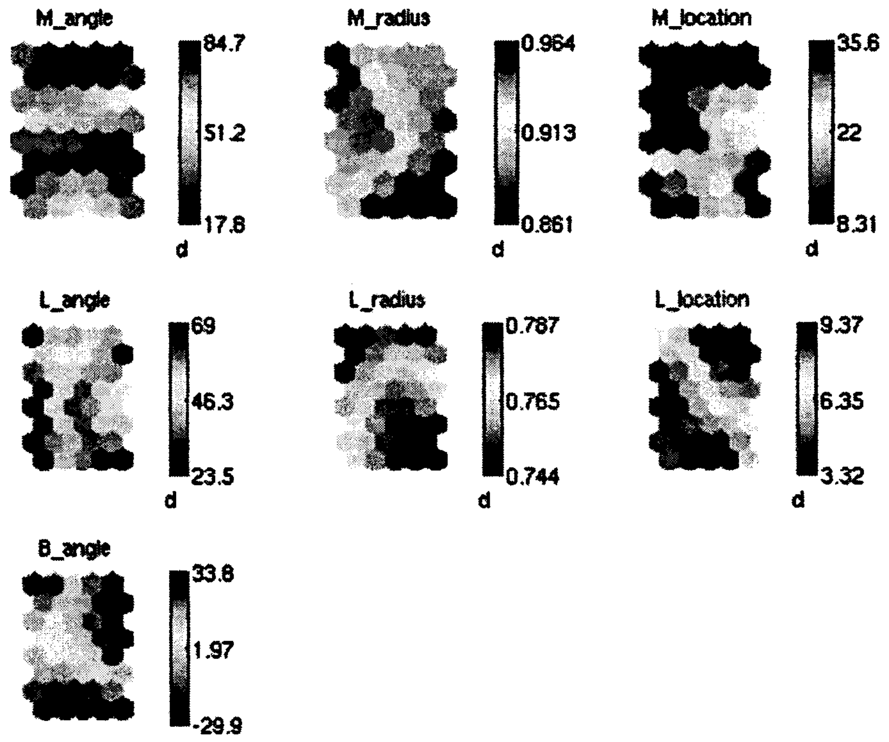


Figure 8.7 Component planes for first 32 cases

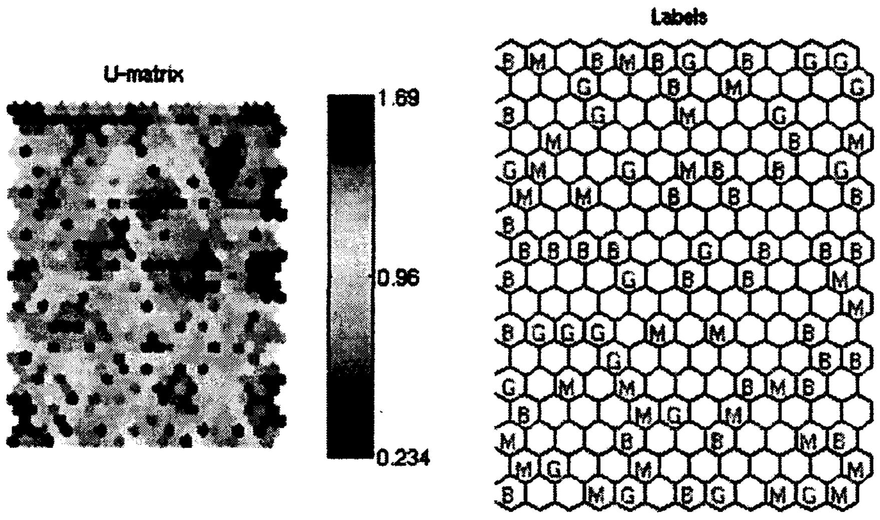


Figure 8.8 U-matrix and labels for 120 cases

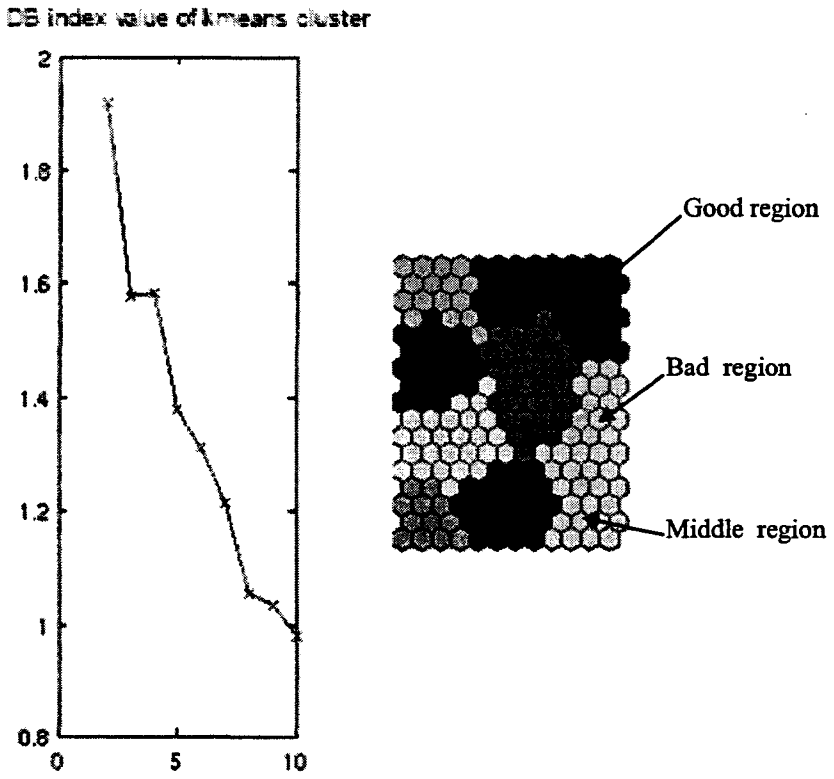


Figure 8.9 Davies-Boulding Index and SOM cluster for 120 cases

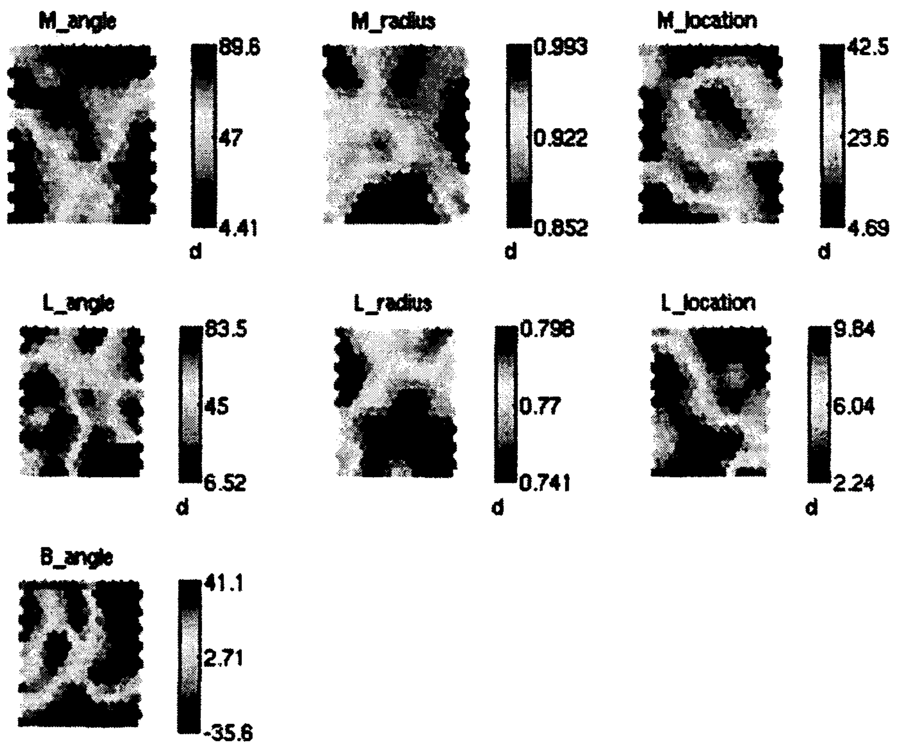
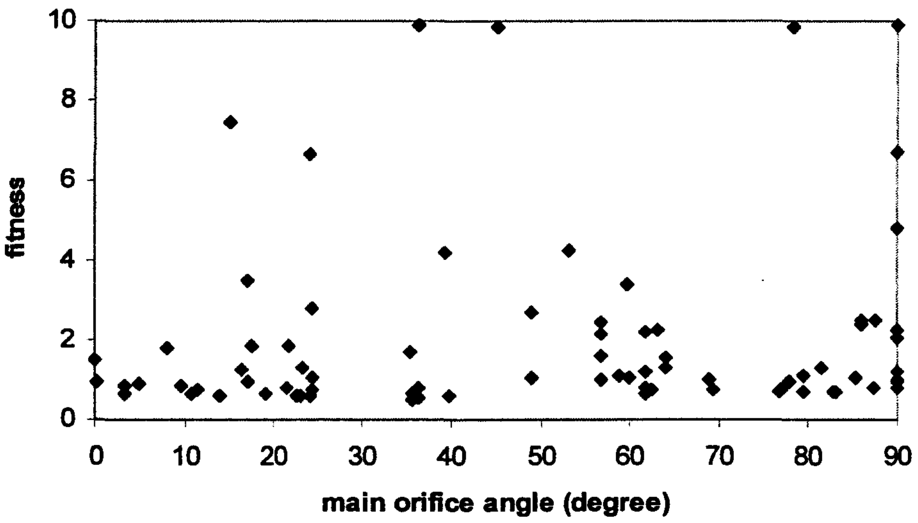
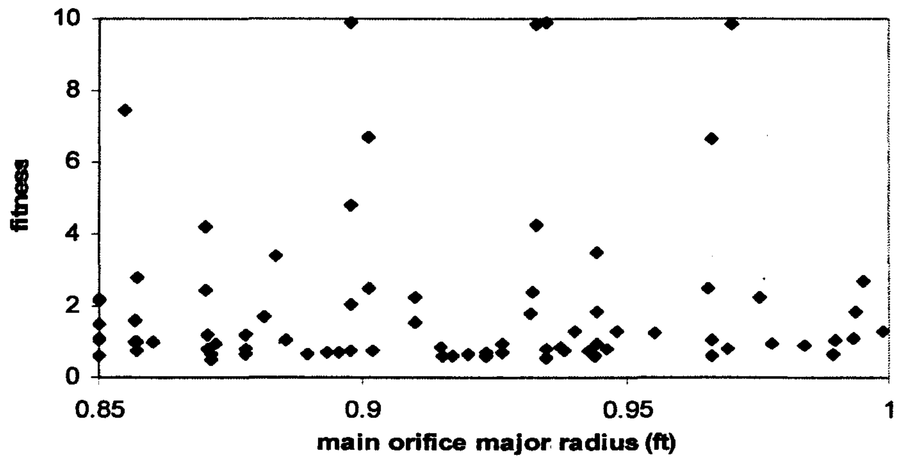
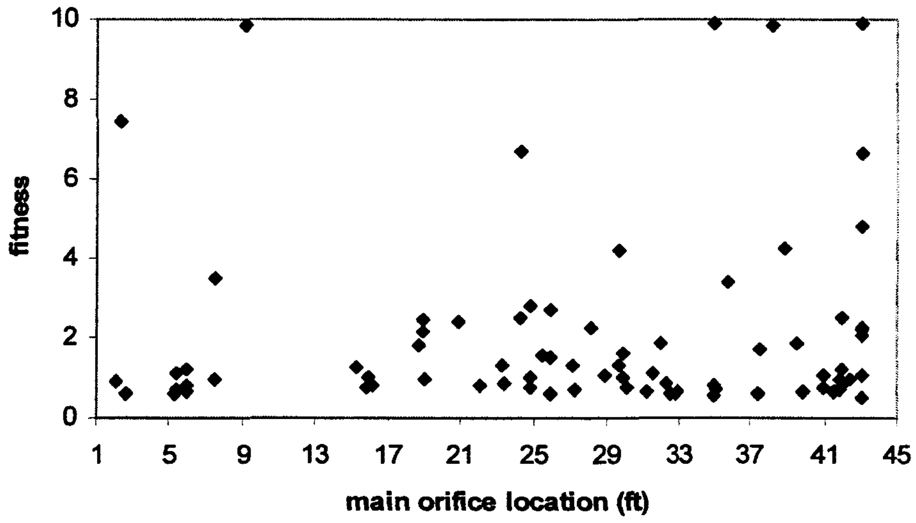
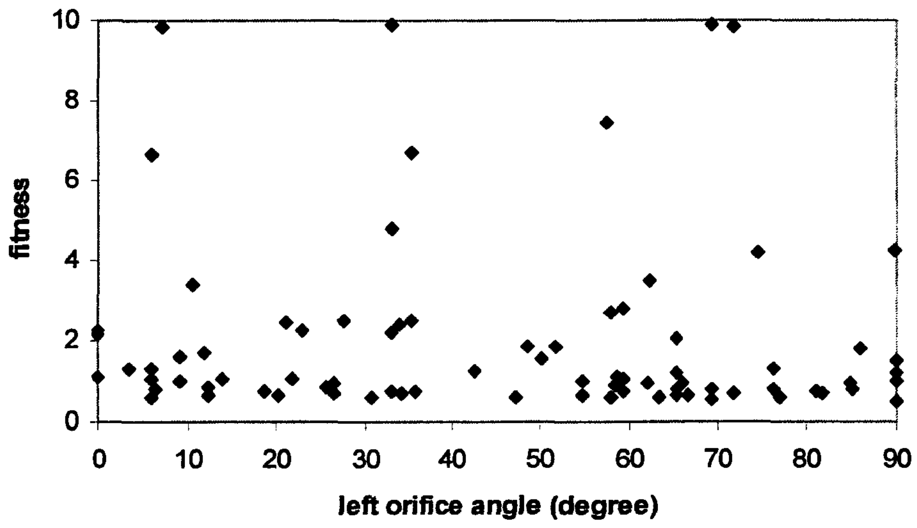


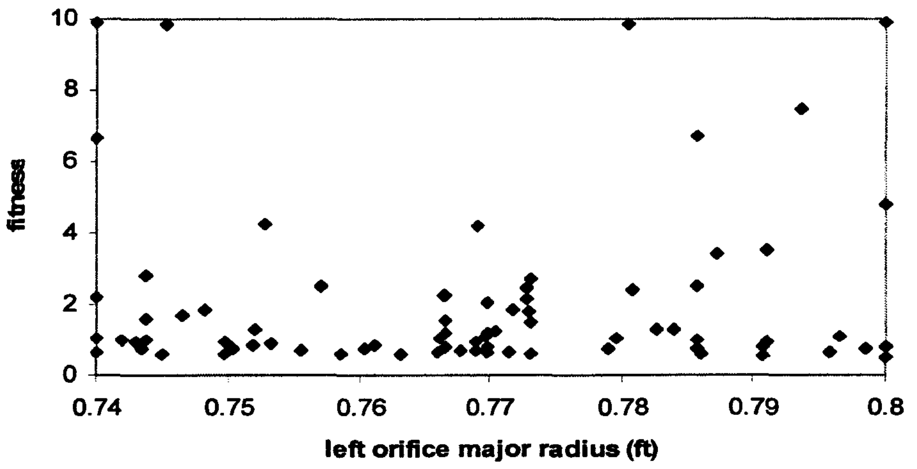
Figure 8.10 Component planes for 120 cases

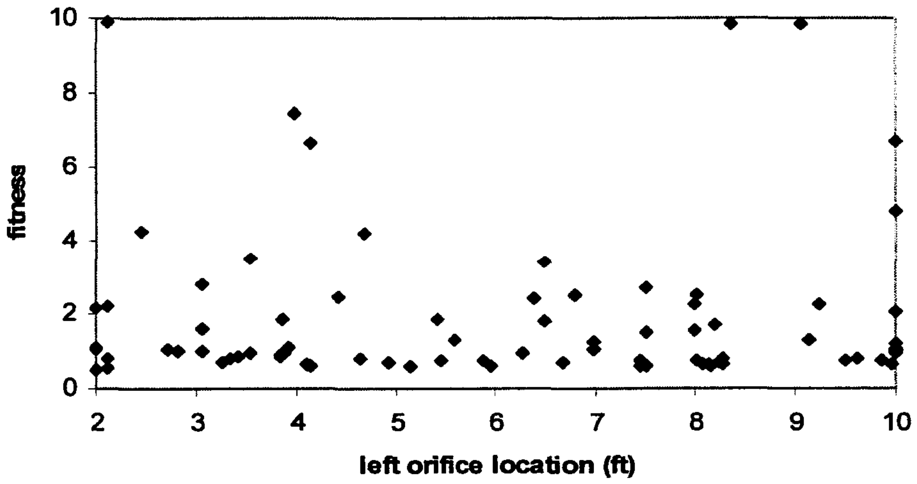



10
8
6
4
2
0
fitness
0.85
0.9
0.95
1
main orifice major radius (ft)


10
8
6
4
2
0
fitness
1
5
9
13
17
21
25
29
33
37
41
45
main orifice location (ft)


10
8
6
4
2
0
fitness
0
10
20
30
40
50
60
70
80
90
left orifice angle (degree)


10
8
6
4
2
0
fitness
0.74
0.75
0.76
0.77
0.78
0.79
0.8
left orifice major radius (ft)


10
8
6
4
2
0
fitness
2
3
4
5
6
7
8
9
10
left orifice location (ft)

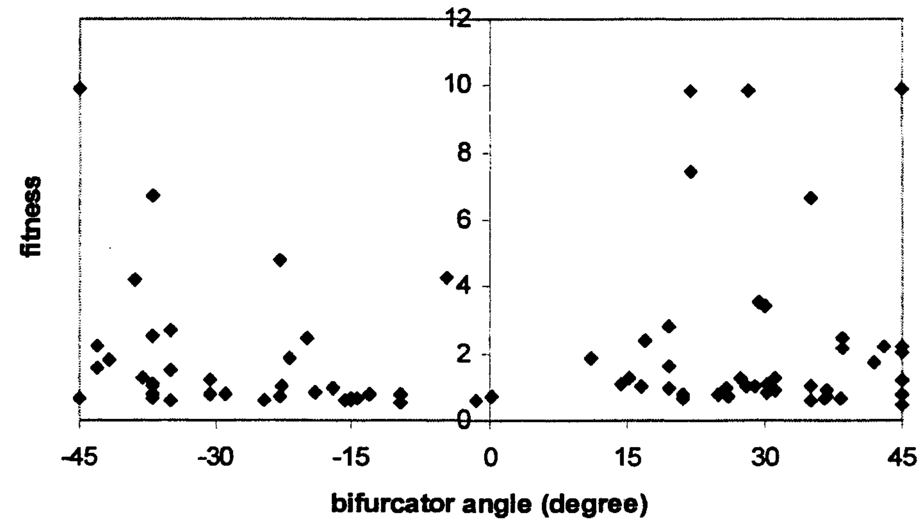


Figure 8.11 Correlations between design variables with fitness

The design optimization history can be seen from Figure 8.12, which monitored the development of objective function. Starting from a randomly generated initial parameter set with 32 individuals, with the help of the user, the fitness converged very quickly. As indicated in Figure 8.12, beyond the twelfth design iteration no significant changes take place. Therefore, we can consider that the optimizer obtains the optimum after 12 iterations (generations). In a matter of 12 (about five days) design iterations, the coal mass flow unbalance decreased from 15% (baseline model) to 1%.

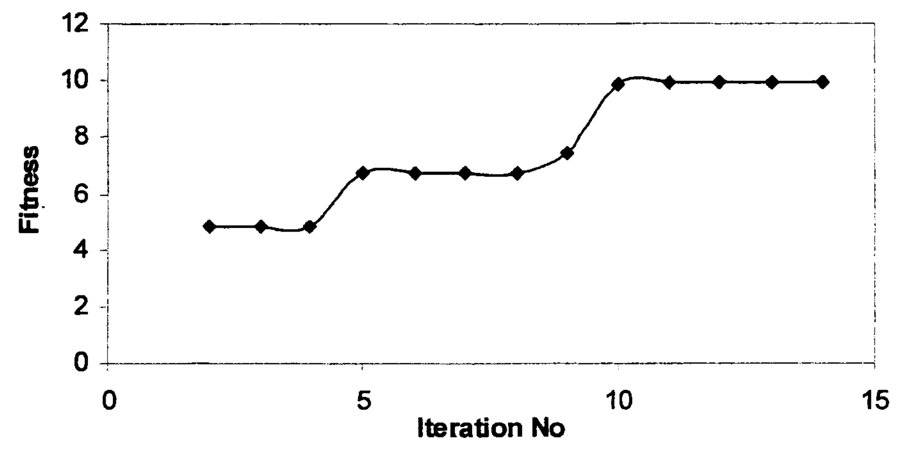
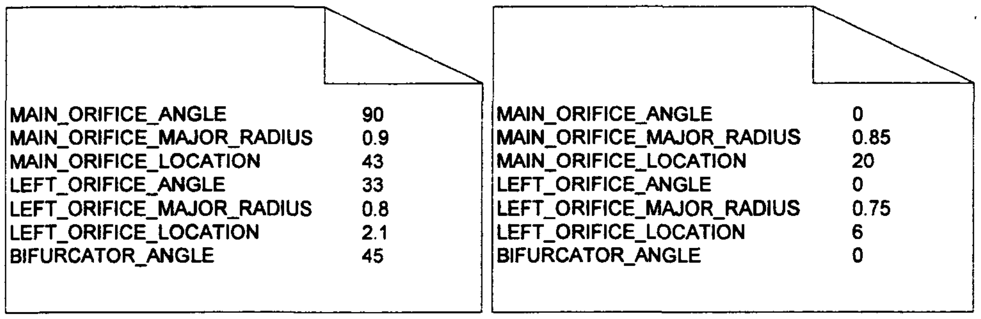


Figure 8.12 Iteration history for fitness

8.4 Comparison between Optimum and Baseline Designs

Figure 8.13 shows a configuration summary of the design variables for the one found as optimum and the baseline model. It shows that the baseline geometry greatly differs from the found optimum geometry. Figure 8.14 shows the geometry of the optimal pipe. Since all seven design variables are changed it is hard to define which one is dominant parameter. The main changes in orifice A (from circular shape to elliptical shape, from near the center of the long horizontal pipe to its end right before the second bend) affect the way the rope, which is generated after the first vertical-to-horizontal, enters the second bend. As seen from Figure 8.15, after the rope enters the second bend, it has a twisted shape. With the bifurcator rotated 45 degree, this twisted rope can enter the bifurcator through its center line. This will greatly help the bifurcator distribute particle flow evenly. To understand the effects of the orifice B (in the left branch) on the coal distribution, the optimum case is compared with a similar design (See Figure 8.16 for its geometry summary). The only geometry difference between these two is the orifice B. However, its coal flow rate unbalance increased from 1% to 5%.

This implied that the location of the orifice has a big impact on the pressure drop. This is mainly because the pressure fluctuation happened in the gas-solid flow. Unlike single-phase flow, the pressure is dynamically changed (pattern unknown so far) instead of linear increased with the pipe length for gas-solid flow. Therefore, the location of orifice is a very important variable to determine if the flow resistance is the same for the two branches which will in turn affect the flow rate balance. This proves the importance of using high-fidelity CFD solve instead of simply model when designing a high efficient power plant piping system.



MAIN_ORIFICE_ANGLE	90
MAIN_ORIFICE_MAJOR_RADIUS	0.9
MAIN_ORIFICE_LOCATION	43
LEFT_ORIFICE_ANGLE	33
LEFT_ORIFICE_MAJOR_RADIUS	0.8
LEFT_ORIFICE_LOCATION	2.1
BIFURCATOR_ANGLE	45

(a)

MAIN_ORIFICE_ANGLE	0
MAIN_ORIFICE_MAJOR_RADIUS	0.85
MAIN_ORIFICE_LOCATION	20
LEFT_ORIFICE_ANGLE	0
LEFT_ORIFICE_MAJOR_RADIUS	0.75
LEFT_ORIFICE_LOCATION	6
BIFURCATOR_ANGLE	0

(b)

Figure 8.13 the summary of design variables. (a) optimum (b) baseline

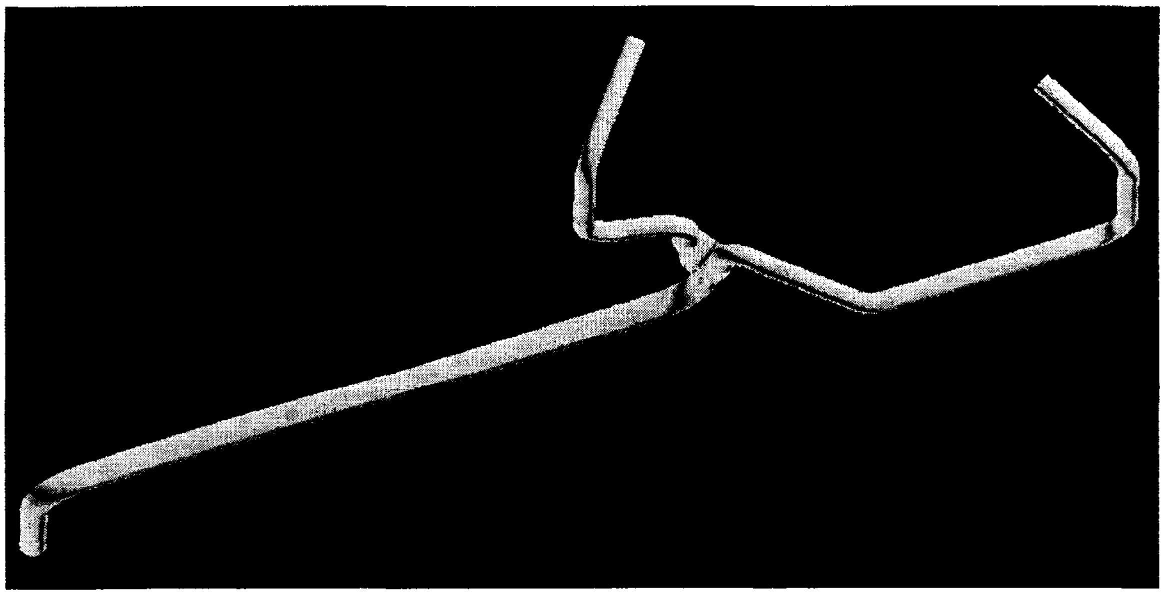


Figure 8.14 the geometry of the optimal pipe

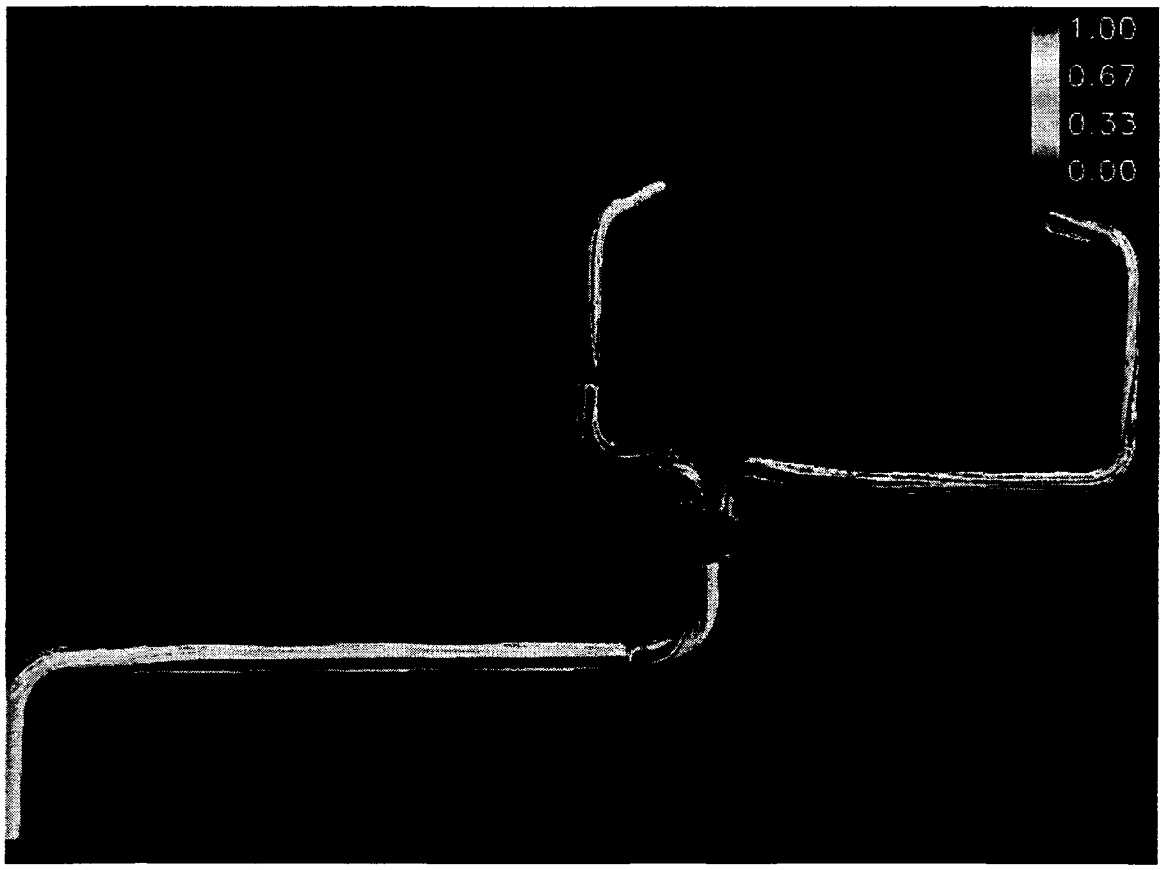
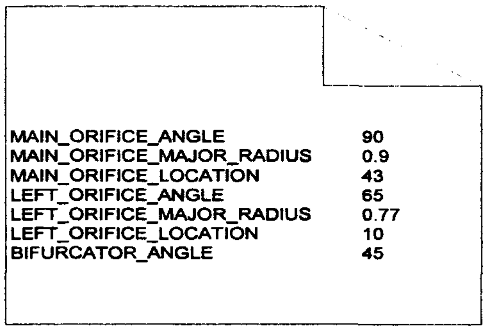
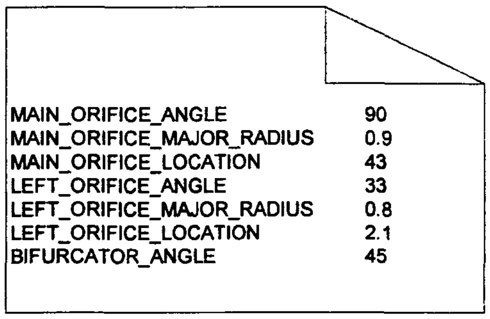


Figure 8.15 Particle distribution in the optimum pipe



MAIN_ORIFICE_ANGLE	90
MAIN_ORIFICE_MAJOR_RADIUS	0.9
MAIN_ORIFICE_LOCATION	43
LEFT_ORIFICE_ANGLE	65
LEFT_ORIFICE_MAJOR_RADIUS	0.77
LEFT_ORIFICE_LOCATION	10
BIFURCATOR_ANGLE	45

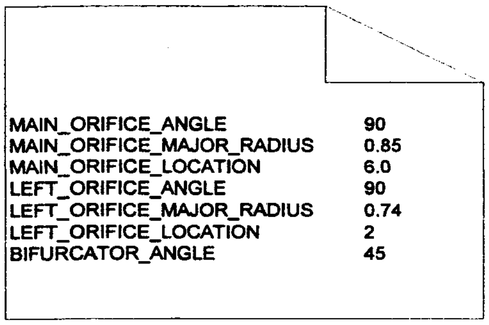
(a)



MAIN_ORIFICE_ANGLE	90
MAIN_ORIFICE_MAJOR_RADIUS	0.9
MAIN_ORIFICE_LOCATION	43
LEFT_ORIFICE_ANGLE	33
LEFT_ORIFICE_MAJOR_RADIUS	0.8
LEFT_ORIFICE_LOCATION	2.1
BIFURCATOR_ANGLE	45

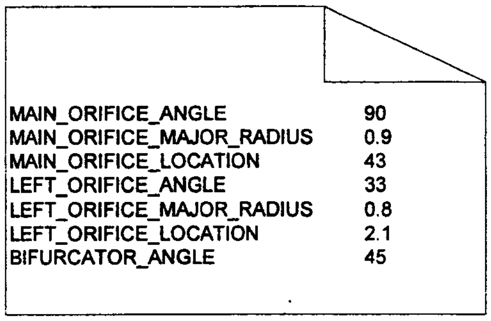
(b)

Figure 8.16 the summary of design variables for (a) case with same parameters for orifice A and bifurcator with the optimum case (b) optimum design



MAIN_ORIFICE_ANGLE	90
MAIN_ORIFICE_MAJOR_RADIUS	0.85
MAIN_ORIFICE_LOCATION	6.0
LEFT_ORIFICE_ANGLE	90
LEFT_ORIFICE_MAJOR_RADIUS	0.74
LEFT_ORIFICE_LOCATION	2
BIFURCATOR_ANGLE	45

(a)



MAIN_ORIFICE_ANGLE	90
MAIN_ORIFICE_MAJOR_RADIUS	0.9
MAIN_ORIFICE_LOCATION	43
LEFT_ORIFICE_ANGLE	33
LEFT_ORIFICE_MAJOR_RADIUS	0.8
LEFT_ORIFICE_LOCATION	2.1
BIFURCATOR_ANGLE	45

(b)

Figure 8.17 the summary of design variables for cases
(a) best particle distribution (b) optimum design

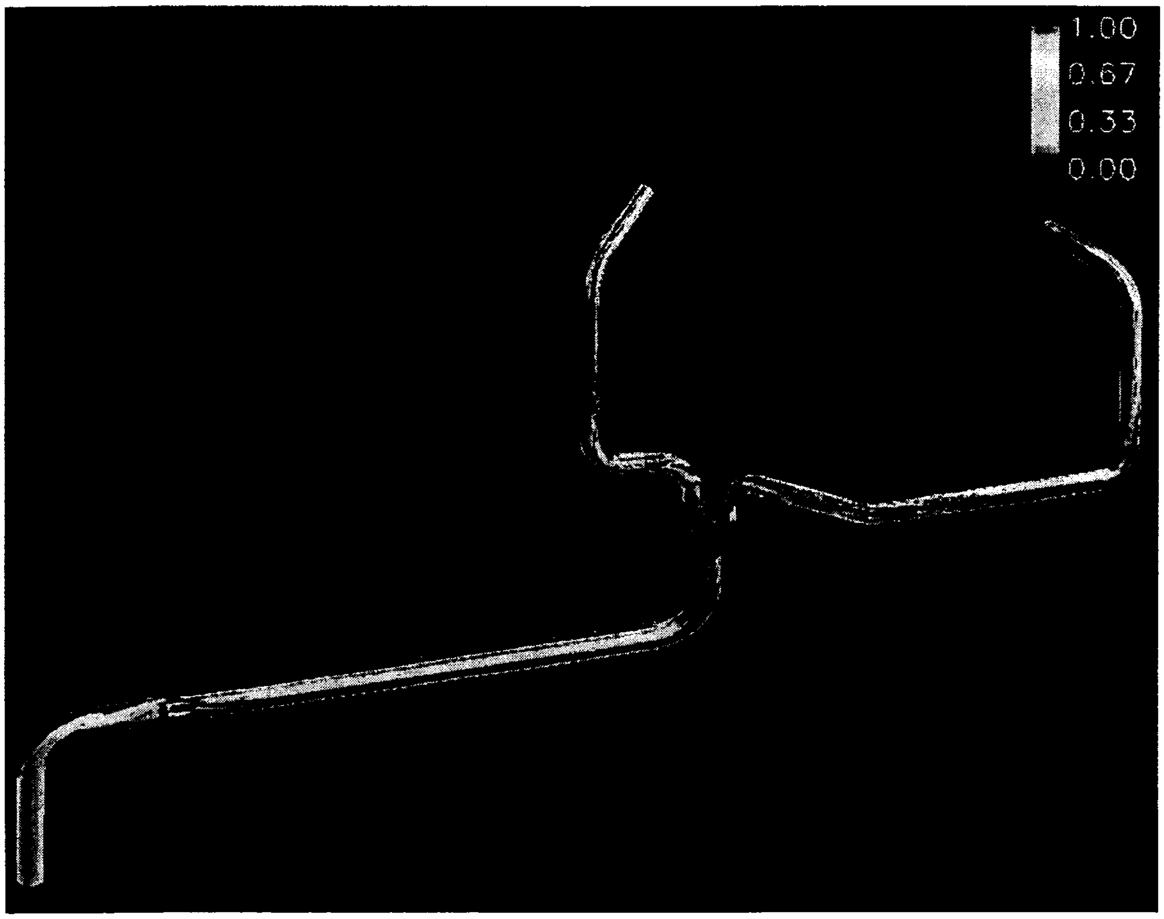


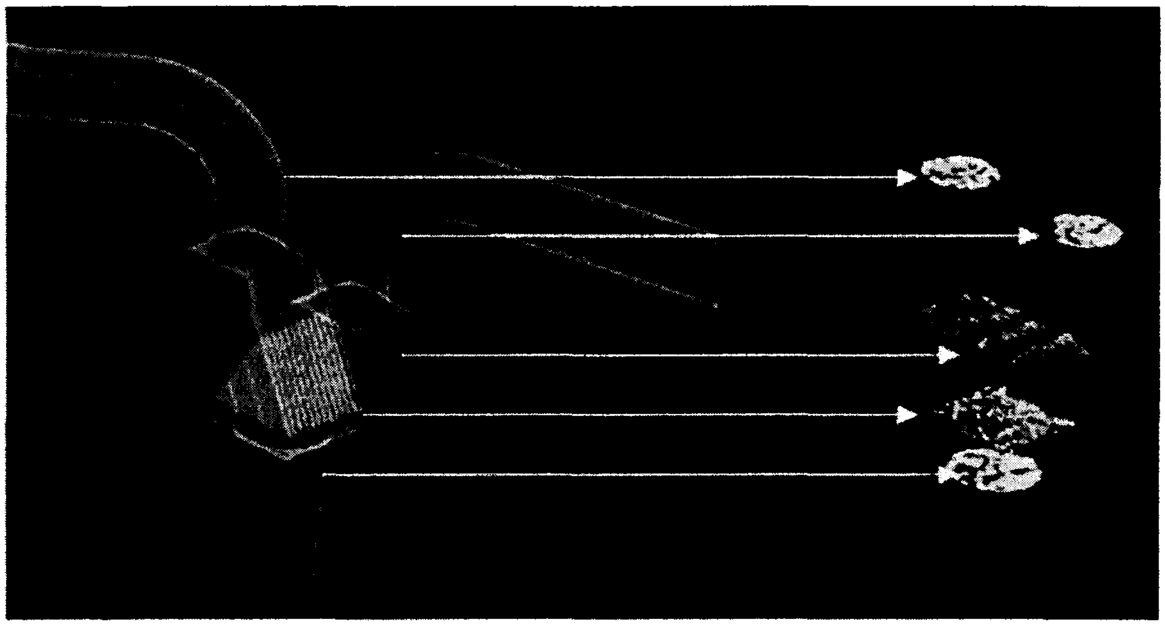
Figure 8.18 Particle distribution in the case with the best particle distribution

Another interesting observation is that the case with best particle distribution (with the best second objective function), which the geometry summary is shown in Figure 8.17), is not the optimum design case. Its whole particle distribution is shown in Figure 8.18. The comparison between particles distribution in the bifurcator section is shown in Figure 8.19. Since the research has shown that the bifurcator works better when the particle distribution can be tailored as uniform as possible. We initially expected the best particle distribution design will come with the best coal flow distribution. The difference between these two cases implies that when considering the performance of bifurcator, the test case should be selected such that it can reflect not only the real pipe's working condition but also its geometry features.

8.5 Conclusions

The developed interactive design tool is successfully applied to an optimization of coal pipe system. The baseline pipe was chosen from a real power plant. The specific components involved are two orifices and bifurcator. By tailoring the geometrical variables of these components, the coal flow imbalance can be reduced to less than 1% from original 15% based on the numerical simulation. The result indicates that the overall design methodology leads to a more efficient pipe system. Also, without adding more hardware into the pipe, this technology can be easily retrofitted into an existing coal pipe network.

Convergence was achieved using a practical reasonable time (five days) and the interaction between the optimization algorithm and the design engineers was maintained throughout the process. Throughout the course of the design process, a total of 12 generations were needed before the fitness converged. The result illustrates the feasibility of the proposed design environment for the coal pipe system design problem.


(a)

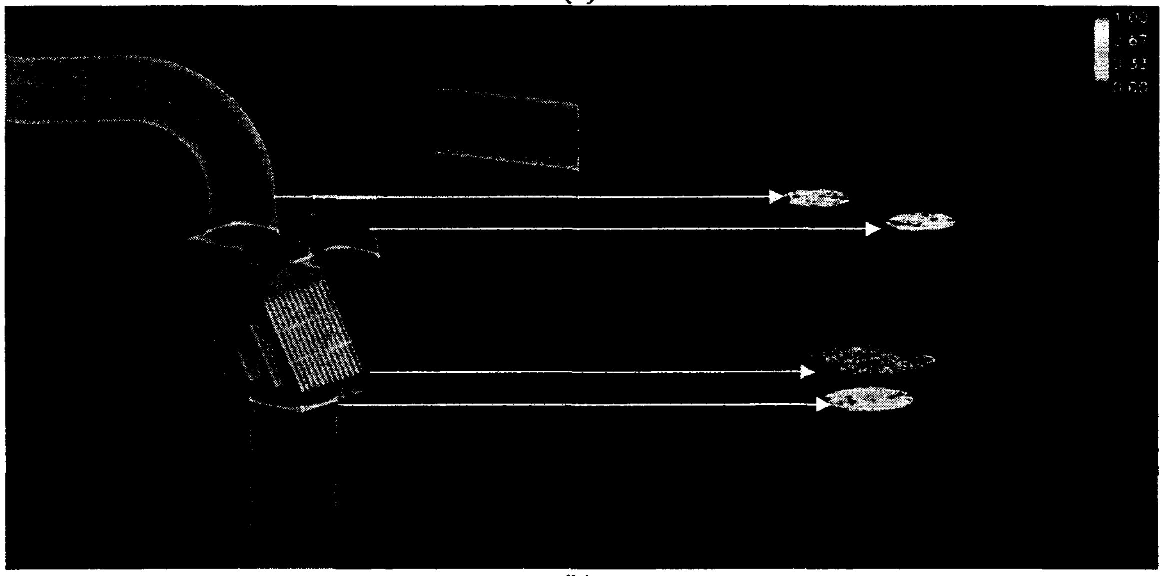

(b)

Figure 8.19 Contours of particle distribution in the bifurcation section.
(a) optimum (b) best particle distribution

Chapter 9 Conclusions and Future Work

9.1 Conclusions

For a coal-fired power plant, careful control of coal distribution can reduce unburned carbon and minimize excess air requirements, resulting in improved overall furnace efficiency and less adverse impacts on downstream emissions control equipment. However, pulverized coal is transported by primary air in a two phase flow regime making it difficult, if not impossible, in the past to be distributed evenly to individual burner. The results from literature show that the nature of particle distribution in the pipeline can be highly inhomogeneous depending on the pipe geometry, phase loading, and particle properties. A particular difficult type of flow phenomenon is called "roping" where the coal particles are concentrated together in a small region of the pipe cross-section after the gas-solid mixture flows through the bend. Since the traditional pipe design is based on the empirical model and without the consideration of particle distribution at all, improving the design tool is necessary. The objective of the research project documented in this thesis has been to develop and integrate methods of high-fidelity analysis and evolution optimization for complex design problems involving coal piping system of power plant.

It is well-known complex flow simulations are challenging and error-prone, and it takes a lot of engineering expertise to obtain valid solutions. When using CFD as the analysis tool, the first step is to make sure that the valid model can be built. The thesis starts with building a complete coal pipeline with complex components such as bifurcator using FLUENT^{TM} to understand the characteristic of particle distribution in a real pipe with multiple bends, orifices, and bifurcator. This pipe is chosen as our baseline design after its results have been compared with real data collected from power plant. The baseline simulation result shows that the coal particle stream after the second pipe is presented as a spiral. Because of this fact, the bifurcator suffered severe erosion damage and the coal flow distribution is 15% off balance.

Combining high-fidelity CFD models into population-based optimization tools presents perhaps the most comprehensive challenges in the field of optimization because the evaluation is very expensive. Therefore, reducing computational time is essential. In this thesis, we proposed a new interactive design environment to address this issue. Other than general benefits automated optimization approach can provide, the characteristics and capabilities of efficient of this tool can be summarized as the following three different features:

First, the tool has been implemented in a virtual engineering framework VE-Suite. Therefore, users can take advantage of all the capabilities VE-Suite provides. One of most important benefits that VE-Suite provides is that it allows engineers to carry out geometric modeling, performance analysis, and decision making in a seamless manner. With an easy-to-use GUI, the effort that users have to spend on setup the optimization process in both optimization algorithms and the problem to optimize is minimized. The designer can access the system information stream at any point in time and view the current state of the running process through this GUI too. This attach/detach functionality gives the user the ability to easily monitor the computational process. Also with minimum amount of coding, the designer can easily switch to other optimization algorithm. Although the test case is coal piping design in this thesis, we believe that the proposed design system is also highly suitable for other design applications.

Second, the system allows users to interact or guide the searching path as the design evolves. The goal here is to combine the computer's computational ability and human's super abstract thinking ability together to reduce the number of evaluated solutions in an optimization run. It also greatly helps the designer to understand the properties of the design so that they can trust and use it. In order to help users better interact with the design system, the tool not only provides the necessary controlling functionality that users can use to monitor the design process, but also a wide range of relevant information from basic EAs searching information to design space knowledge from previous evaluations based on SOM method which can be discussed and analyzed both subjectively and objectively by the designer.

Third, a fast calculation approach is used to reduce the time for single CFD case by introducing the fitness residuals as main convergence criteria for CFD simulation. The amount of time that can be saved is problem dependent. For the case when the fitness evaluation only needs small part of the information from the flow field, the time used for simulation of single case can be reduced by 95% as shown in our test case.

The developed interactive design tool is successfully applied to an optimization of coal pipe system. The optimum design conferred author's original idea that the proper pipe shape geometry can provide better coal flow distribution which is currently overlooked by design community. By tailoring seven geometrical variables and without adding new hardware, the coal flow imbalance can be reduced to less than 1% from original 15% based on the numerical simulation. This result illustrates that the proposed design tool can find excellent solutions to complex engineering problems. The fast convergence history indicates that the overall design methodology leads to a more efficient design system.

9.2 Future Work

When considering the nature of particle distribution in the pipeline, the work on coal pipe design optimization becomes a very complex research area, which leaves many interesting possibilities for future work. First, the results in this thesis need more in-depth analysis, and especially testing on real pipe application. Also, there are some further researches that could be pursued based on the work done in this thesis:

- While design tool presented in this thesis support it, we have not touched on the subject of connecting CAD package into the design system. Most current pipe design packages focus on pipe routing through CAD approach. It would be interesting to combine these two powerful design tools to speed up the pipe design.
- The calculation speed is still somewhat disappointing with respect to our expectation for "real-time" analysis and optimization. Since it is widely known that EAs can be used to take full advantage of a parallel computer structure, one can expect that by adding parallelizing functions to EAs search as well as the CFD calculation, the optimization time will be greatly reduced.
- Eventually, we hope this design tool can be used to facilitate multidisciplinary and collaborative product realization.

References

[1] Fan, L.S. and C. Zhu, *Principles of Gas-Solid Flows*, Cambridge University Press, Cambridge, United Kingdom, 1998.

[2] Littman, H., M.H. Morgan III, S.Dj. Jovanovic, J.D. Paccione, Z.B. Grbavcic, and D.V. Vukovic, "Effect of particle diameter, particle density and loading ratio on the effective drag coefficient in steady turbulent gas-solids transport," *Powder Technology*, 84:49-56, 1995.

[3] Yao, J., B. Zhang, and J. Fan, "An experimental investigation of a new method for protecting bends from erosion in gas-particle flows," *Wear*, 240:215-222, 2000.

[4] Tsuji, Y., Y. Morikawa, and H. Shiomi, "LDV measurements of an air-solid two-phase flow in a vertical pipe," *Journal of Fluid Mechanics*, 139:417-434, 1984.

[5] Flemmer, C.L., R.L.C. Flemmer, E.K. Johnson, and K.H. Means, "Roping paths in elbows," *ASME/Pressure Vessel & Piping Resources*, 193:105-109, 1990.

[6] Yilmaz, A. and E.K. Levy, "Formation and dispersion of ropes in pneumatic converying," *Powder Technology*, 114:168-185, 2001.

[7] Schallert, R. and E. Levy, "Effect of a combination of two elbows on particle roping in pneumatic conveying," *Powder Technology*, 107:226-233, 2000.

[8] Bilirgen, H., E. Levy, and A. Yilmaz, "Prediction of pneumatic conveying flow phenomena using commercial CFD software," *Powder Technology*, 95:37-41, 1998.

[9] Schneider, H., T. Frank, D.K. Pachler, and K. Bernert, "A numerical study of the gas-particle flow in pipework and flow splitting devices of coal-fired power plant," *10th Workshop on Two-Phase Flow Predictions*, 227-236, 2002.

[10] Huber, N. and M. Sommerfeld, "Modeling and numerical calculation of dilute-phase pneumatic conveying in pipe systems," *Powder Technology*, 99:90-101, 1998.

[11] Huber, N. and M. Sommerfeld, "Numerical calculation of dilute-Phase pneumatic conveying in complex pipe systems," *ASME Fluids Engineering Division Summer Meeting*, 1-9, 1997.

[12] Adams, B., M. Cremer, J. Valentine, and V. Bhamidipati, D. O'Connor, "Use of CFD modeling for design of NO_x reduction systems in utility boilers," *International Joint Power Generation Conference*, 695-702, 2002.

[13] Stopford, P.J., "Recent applications of CFD modeling in the power generation and combustion industries," *Applied Mathematical Modeling*, 26:351-374, 2002.

[14] Foucaut, J.M. and A. Taniere, "A new choice of primary parameters to study particle transport phenomena," *1997 ASME Fluids Engineering Division Summer Meeting*, 1-7, June 22-26, 1997.

[15] Parslow, G.I., D.J. Stephenson, J.E. Strutt, and S. Tetlow, "Investigation of solid particle erosion in components of complex geometry," *Wear*, 23:737-745,1999.

[16] Yilmaz, Ali, "Roping phenomena in lean phase pneumatic conveying," Doctor of Philosophy Dissertation, Lehigh University, 1997.

[17] Bilirgen, H., "Mixing and dispersion of particle ropes in lean phase pneumatic conveying," Doctor of Philosophy Dissertation, Lehigh University, 2000.

[18] McCorkle, D.S., "A new methodology for optimization of energy systems," Master of Science Thesis, Iowa State University, 2002.

[19] Svenningsen, K.H. and J.I. Madsen, "Optimization of flow geometries applying quasianalytical sensitivity analysis," *Applied Mathematical Modelling*, 20:214-224, 1996.

[20] Dornberger, R., P. Stoll, and D. Buche, "Multidisciplinary turbomachinery blade design optimization," *AIAA Paper* 2000-0838.

[21] Obayashi, S., K. Nakahashi, A. Oyama, and N. Yoshino, "Design optimization of supersonic wings using evolutionary algorithms," *European Community on Computational Methods in Applied Sciences (ECCOMAS)*, 1998.

[22] Quagliarella, D. and A. Vicini, "GAs for aerodynamic shape design I: general issues, shape parameterization problems and hybridization techniques," *Genetic Algorithms for Optimization in Aeronautics and Turbomachinery, 1999-2000, VKI Lecture Series*, 2000.

[23] Benini, E. and A. Toffolo, "A parametric method for optimal design of two-dimensional cascades," *Proceedings of the I MECH E Part A Journal of Power and Energy*, 215(4):465-473, 2001.

[24] Akl, W., A. Baz, and M. Ruzzene, "Virtual reality design of underwater shells with active stiffeners," *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2002.

[25] Schmitz, B., "Great expectations: the future of virtual design," *Computer-Aided Engineering*, 14(10):68-72, 1995.

[26] Rizk, M.H., "Optimization by updating design parameters as CFD iterative flow solutions evolve," *ASME FFD Multidisciplinary Application of Computational Fluid Dynamics*, 129:51-62, 1991.

[27] Li, J. and N. Satofuka, "Optimization design of a compressor cascade airfoil using a Navier-Stokes solver and genetic algorithm," *Proceedings of the Institution of Mechanical Engineers, Part A: Power and Energy*, 216:195-202, 2002.

[28] Cantu-Paz, E., "Markov chain models of parallel genetic algorithms," *IEEE Transactions on Evolutionary Computation*, 4(3):216-226, 2000.

[29] Hamalainen, J.P., R.A.E. Makinen, P. Tarvainen, and J. Toivanen, "Evolutionary shape optimization in CFD with industrial applications," *European Congress on Computational Methods in Applied Sciences and Engineering*, ECCOMAS 2000.

[30] Oksuz, O., I.S. Akmandor, and M.S. Kavsaoglu, "Aerodynamic optimization of turbomachinery cascades using Euler/Boundary-Layer coupled genetic algorithms," *Journal of Propulsion and Power*, 18(3):652-657, 2002.

[31] Anderson, J., "On engineering systems design: A simulation and optimization approach," Thesis, Linkopings University, Sweden, 1999.

[32] Makinen, R.A.E., P. Neittaanmaki, J. Periaux, and J. Toivanen, "A genetic algorithm for multiobjective design optimization in aerodynamics and electromagnetics," *European Community on Computational Methods in Applied Sciences (ECCOMAS)*, 1998.

[33] Van Dam, A., A.S. Forsberg, D.H. Laidlaw, J.J. LaViola, and R.M. Simpson, "Immersive VR for scientific visualization: A progress report", *Computer Graphics*, 20(6):26-52, 2000.

[34] Nelsen, J., "Sandia uses CFD software with adaptive meshing capability to optimize complex inlet design," *Journal Articles by Fluent Software Users*, accessed August 5, 2003, http://www.fluent.com/solutions/articles/ja060.pdf

[35] "NPARC alliance CFD verfication and validation Web Site," accessed August 5, 2003, http://www.grc.nasa.gov/WWW/wind/valid/tutorial/valassess.html

[36] Choi, T.J., "CFD shape optimization based on an adjoint variable formulation of the compressible Navier-Stokes equations," accessed August 5, 2003, http://www.me.cmu.edu/faculty1/stahovich/bennett/proceedings99/choi.pdf

[37] Holland, J.H., *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, University of Michigan Press, 1992.

[38] Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Maryland, 1989.

[39] Makinen. R., J. Periaux, and J. Toivanen, "Multidisciplinary shape optimization in aero-dynamics and electromagnetics using genetic algorithms," *International Journal of Numerical Methods*, 30:145-159, 2000.

[40] Jang, M. and J. Lee, "Genetic algorithm based design of transonic airfoils using Euler equations," *Collection of Technical Papers-AIAA/ASME/ASCE/ASC Structures, Structural Dynamical and Materials Conferences, Atlanta, GA*, 281-290, 2000.

[41] Quagliarella, D. and A. Vicini, "Viscous single and multicomponent airfoil design with genetic algorithms," *Finite Elements in Analysis and Design*, 37:365-380, 2001.

[42] Fabbri, G., "Optimization of heat transfer through finned dissipators cooled by laminar flow," *International Journal of Heat Fluid Flow*, 19:644-654, 1998.

[43] Schmit, T.S., A.K. Dhingra, F. Landis, and G. Kojasoy, "Genetic algorithm optimization technique for compact high intensity cooler design," *Journal of Enhanced Heat Transfer*, 3:281-290, 1996.

[44] Trigg, M.A., G.R. Tubby, and A.G. Sheard, "Automatic genetic optimization approach to two dimensional blade profile design for steam turbines," *Trans. ASME Turbomachinery*, 121:11-17, 1999.

[45] Fan, H-Y, "Inverse design method of diffuser blades by genetic algorithms," *Proceedings of the Institution of Mechanical Engineers, Part A: Power and Energy*, 212:193-205, 2002.

[46] Blaize, M., D. Knight, and K. Rasheed, "Automated optimal design of two-dimensional supersonic missile inlets," *Journal of Propulsion and Power*, 14:890-898, 1998.

[47] Zha, G., D. Smith, M. Schwabacher, K. Rashed, A. Gelsey, D. Knight, and M. Haas, "High performance supersonic missile inlet design using automated optimization," *Journal of Aircraft*, 34:697-705, 1997.

[48] Foster, G.F. and G.S. Dulikravich, "Three-dimensional aerodynamic shape optimization and gradient search algorithms," *Journal of Spacecraft Rockets*, 34:36-42, 1997.

[49] Poloni, C., A. Giurgevich, L. Onesti, and V. Pediroda, "Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics," *Computer Methods in Applied Mechanics and Engineering*, 186:403-420, 2000.

[50] McCorkle, D.S., K.M. Bryden, and C.G. Carmichael, "A new methodology for evolutionary optimization of energy systems," *Computer Methods in Applied Mechanics and Engineering*, Accepted for publication, 2003.

[51] Bryden, K.M., D.A. Ashlock, D.S. McCorkle, and G.L. Urban, "Optimization of heat transfer utilizing graph based evolutionary algorithms," *International Journal of Heat Fluid Flow*, 24:267-277, 2003.

[52] Madsen, J., W. Shyy, and R. Haftka, "Response surface techniques for diffuser shape optimization," *AIAA Journal*, 38:1512-1518, 2000.

[53] Shyy, W., P.K. Tucker, and R. Vaidyanathan, "Response surface and neural network techniques for rocket engine injector optimization," *AIAA Paper* 99-2455, 1999.

[54] Papila, N., W. Shyy, L. Griffin, and D.J. Dorney, "Shape optimization of supersonic turbines using global approximation methods," *Journal of Propulsion and Power*, 18:509-518, 2002.

[55] Jones, B.R., W.A. Crossley, and A.S. Lyrintzis, "Aerodynamic and aeroacoustic optimization of airfoils via a parallel genetic algorithm," *AIAA Paper* 98-4811, 1998.

[56] Wang, X. and M. Damodarn, "Optimal three-dimensional nozzle shape design using CFD and parallel simulated annealing," *Journal Propulsion and power*, 18:217-221, 2001.

[57] Diachin, D., L. Freitag, D. Heath, J. Herzog, W. Michels, and P. Plassmann, "Collaborative virtual environments used in the design of pollution control systems",

International Journal of Supercomputer Applications and High Performance Computing, 10:223-235, 1996.

[58] Ryken, M.J. and J.M. Vance, "Applying virtual reality techniques to the interactive stress analysis of a tractor lift arm." *Finite Elements in Analysis and Design*, 35:141-155, 2000.

[59] Kihonge, J.N., J.M. Vance, and P.M. Larochelle, "Spatial mechanism design in virtual reality with networking." *Journal of Mechanical Design*, 124:435-440, 2002.

[60] Perles, B. and J.M. Vance, "Interactive virtual tools for manipulating NURBS surfaces in a virtual environment." *Journal of Mechanical Design*, 124:158-163, 2002.

[61] Cruz-Neira, C., "Virtual reality overview." *ACM SIGGRAPH '93 Course Notes: Applied Virtual Reality*, ACM SIGGRAPH '93 Conference, Anaheim, California, 1993.

[62] Marcus, R.D., L. S. Leung, G. E. Klinzing, and F. Rizk, *Pneumatic Conveying of Solids: A Theoretical and Practical Approach*, Chapman & Hall, London, 1990.

[63] Crowe, C.T., M. P. Sharma, and D. E. Stock, "Particle-Source-in Cell (PSI-Cell) model for gas-droplet flows," *Journal of Fluids Engineering*, 99:325-332, 1977.

[64] Murphy, G., *Similitude in Engineering*, The Ronald Press Company, 1950.

[65] Johnson, P.E., "A multi-solver technique for virtual engineering analysis using evolutionary data segregation," Doctor of Philosophy Dissertation, Iowa State University, 2003.

[66] Schade, K.P., H.J. Erdmann, T. Hadrich, H. Schneider, T. Frank, and K. Bernert, "Experimental and numerical investigation of particle erosion caused by pulverized fuel in channels and pipework of coal-fired power plant," *Powder Technology*, 125:242-250, 2002.

[67] Lee, L.Y., T.Y. Quek, R.D., R.B. Madhumita, and C.Wang, "Pneumatic transport of granular materials through a 90° bend," *Chemical Engineering Science*, 59:4637-4651, 2004.

[68] Zhang, J. and J. Coulthard, "Theoretical and experimental studies of the spatial sensitivity of an electrostatic pulverized fuel meter," *Journal of Electrostatics*, 63:1133-1149, 2005, issue 12.

[69] Portela, L.M. and R.V.A. Oliemans, "Eulerian-Lagrangian DNS/LES of particle-turbulence interactions in wall-bounded flows," *International Journal for Numerical Methods in Fluids (Int. J. Numer. Meth. Fluids)*, 43:1045-1065, 2003.

[70] Shirolkar, J.S., C.F.M. Coimbra and M.Q. McQuay, "Fundamental aspects of modeling turbulent particle dispersion in dilute flows," *Progress in Energy and Combustion Science (Prog. Energy Combust.Sci).*, 22:363-399, 1996.

[71] Molerus, O., "The role of science in particle technology," *Powder Technology*, 122:156-167, 2002.

[72] Marcus, R.D, A.J. Dickson, and C.J. Rallis, "Some Aspects of Dilute Phase Pneumatic Conveying of Gas-solid Suspensions," *The South African Mechanical Engineer*, 26:131-140, 1976.

[73] Ljus, C., B. Johansson, and A. Almstedt, "Turbulence modification by particles in a horizontal pipe flow," International Journal of Multiphase Flow, 28:1075-1090,2002.

[74] Crowe, C.T., T.R. Troutt and J.N. Chung, "Numerical models for two-phase turbulent flows," *Annual Review of Fluid Mechanism (Annu. Rev. Fluid. Mech.)*, 28:11-43, 1996.

[75] Wang, Q., and K.D. Squires, "Large eddy simulation of particle-laden turbulent channel flow," *Physics of Fluids A*, 1207-1223, 1996.

[76] Rogallo, R.S. and P. Moin, "Numerical simulation of turbulent flows," *Annual Review of Fluid Mechanism (Annu. Rev. Fluid. Mech.)*, 16:99-137, 1984.

[77] Bernert, K., T. Frank, H. Schneider and K. Pachler, "Numerical simulation of disperse multiphase flows with an application in power engineering," *International Journal for Numerical Methods in Fluids* (Int. J. Numer. Meth. Fluids), 41:1253-1271, 2003.

[78] Crowe, C.T., "Two-fluid vs. trajectory models; range of applicability," *Gas-Solid Flows, ASME FED*, 35:91-96, 1986.

[79] Naik, S. and I.G. Bryden, "Prediction of turbulent gas-solids flow in curved ducts using the Eulerian-Lagrangian method," *International Journal for Numerical Methods in Fluids*, 31:579-600, 1999.

[80] Forkeer, S., S. Kingman, I. Lowndes, and A. Reynolds, "Characterization of the cross sectional particle concentration distribution in horizontal dilute flow conveying- a review," *Chemical Engineering and Processing*, 43:677-691, 2004.

[81] Tashiro, H., X. Peng, and Y. Tomita, "Numerical prediction of saltation velocity for gas-solid two-phase flow in a horizontal pipe," *Powder Technology*, 91:141-146,1997.

[82] Huber, N. and M. Sommerfeld. "Modeling and numerical calculation of dilute-phase pneumatic conveying in pipe systems," *Powder Technology*; 98:90-101, 1998.

[83] Marjanovic, P., A. Levy, and D.J. Mason, "An investigation of the flow structure through abrupt enlargement of circular pipe," *Powder Technology*, 104:296-303, 1999.

[84] Cartaxo, S.J.M and S.C.S. Rocha, "Object-oriented simulation of the fluid-dynamics of gas-solid flow," *Powder Technology*, 117:177-188, 2001.

[85] Sommerfeld, M., "Analysis of collision effects for turbulent gas-particle flow in a horizontal channel: part I. particle transport," *International Journal of Multiphase Flow*, 29:675-699, 2003.

[86] Amio Rautiainen, A., G. Stewart, V. Poikolainen, and P. Sarkomaa, "An experimental study of vertical pneumatic conveying," *Powder Technology*, 104: 139-150, 1999.

[87] Giddings, D., A. Aroussi, S.J. Pickering and E. Mozaffari, "A ¼ scale test facility for PF transport in power station pipelines," *Fuel*, 83:2195-2204, 2004.

[88] Tsuji, Y., Y. Morikawa, "LDV measurements of an air-solid two-phase flow in a horizontal pipe," *Journal of Fluid Mechanics (Journal of Fluid Mech.)*, 120:385-409, 1982.

[89] Bell, T.A., "Challenges in the scale-up of particulate processes-an industrial perspective," *Powder Technology*, 150:60-71, 2005.

[90] Chambers, A.J. and R.D. Marcus, "Pneumatic conveying calculations," *Proc. 2nd International Conference on Bulk Materials Storage, Handling and Transportation*, Wollongong, 49-52, 1986.

[91] Haim, M., Y. Weiss, and H. Kalman, "Turbulent gas-particles flow in dilute state: a parametric study," *Particulate Science and Technology*, 21:1-24, 2003.

[92] Cao, J. and G. Ahmadi, "Gas-particle two-phase turbulent flow in horizontal and inclined ducts," *International Journal of Engineering Science*, 38:1961-1981, 2000.

[93] Herbreteau, C. and R. Bouard, "Experimental study of parameters which influence the energy minimum in horizontal gas-solid conveying," *Powder Technology*,112:213-220, 2000.

[94] Deng, T., M. Patel, I. Hutchings, and M.S.A. Bradley, "Effect of bend orientation on life and puncture point location due to solid particle erosion of a high concentration flow in pneumatic conveyors," Wear, 258:426-433, 2005.

[95] Rizk, M.H, "Optimization by updating design parameters as CFD inerative flow solutions evolve," *Multidisciplinary Applications of Computational Fluid Dynamics*, FED-Vol. 129, ASME 1991.

[96] Haumesser, F.P., R.E. Thompson, T.A. Davey, and J.N. Kuipers, "Achieving uniform combustion using burner line orifices to balance coal flow distribution," *Proceedings of International Joint Power Generation Conference*, IJPGC2002-26093.

[97] Simula, O., J.Vesanto, and P.Vasara, "Analysis of industrial systems using the Self-Organizing Map," *Second International Conference on Knowledge-Based Intelligent Electronic Systems*, Adelaide, Australia, April 21-23, 1998.

[98] Davies, D. and D. Boulding, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:224-227, 1979.

[99] Anderson D., E. Anderson, N. Lesh, J. Marks, B. Mirtich, D. Ratajczak, and K. Ryall, " Human-guided simple search," *Proceedings of AAAI 2000*, Austin, Texas, USA, July 30-August 3.

[100] CORBA software, accessed March 27, 2006, http://www.omg.org

[101] Hong L., T. Mingxi, and J. Hamilton, "Supporting creative design in a visual evolutionary computing environment," *Journal Advances in Engineering Software*, 35:261-271, 2004.

[102] McCorkle, D.S., K.M. Bryden, and S.J. Kirstukas, "Building a foundation for power plant virtual engineering," *Proceedings of the 28th International Technical Conference on Coal Utilization and Fuel Systems*, Clearwater, FL, 118-127, 2003.

[103] Parker, S.G. and C.R. Johnson, "SCIRun: a scientific programming environment for computational steering," *Proceedings of Supercomputing'95*, 1995.

[104] SGI-OpenGL Performer software, accessed March 27, 2006,
http://www.sgi.com/products/software/performer

[105] TAO CORBA software, accessed March 27, 2006,
http://www.ece.uci.edu./~schmidt/TAO.html

[106] The Visualization ToolKit (VTK) software, accessed March 27, 2006,
http://public.kitware.com/VTK

[107] VRJuggler software, accessed March 27, 2006, http://www.vrjuggler.org

[108] wxWidgets software, accessed March 27, 2006, http://www.wxWindows.org

[109] Vladimir O. B, C. Christophe, G. Dipankar, Q. Gary, and N.V. Garret, "VisualDOC: a software system for general-purpose integraton and design optimization," *AIAA/ISSMO'2002*, Atlanta-Georgia, 2002.

[110] Xiao, A. and K.M. Bryden, "Virtual Engineering: A vision of the next-generation product realization using virtual reality technologies," *Proceedings of ASME Design Engineering Technical, Computers in Engineering Conference*, Saltlake City, Utah, DETC2004/CIE-57698.

[111] Parmee, I.C. and C.R. Bonham, "Cluster-oriented genetic algorithms to support interactive designer/evolutionary computing systems," *Proceedings of the 1999 Congress on Evolutionary Computation,* 1999.

[112] Hayashida, N. and H. Takagi, "Accleration of EC convergence with landscape visualization and human intervention," *Applied Soft Computing*, 1:245-256, 2002.

[113] Saleeb, A.F., J.R. Marks, T.E. Wilt, and S.M. Arnold,"Interactive software for material parameter characterization of advanced engineering constitutive models," *Advances in Engineering Software*, 35:383-398,2004.

[114] Malkawi, A.M., R.S. Srinivasan, Y.K. Yi, and R. Choudhary, "Decision support and design evolution: integration genetic algorithms, CFD and visualization," *Automation in Construction*, 14:33-44, 2005.

[115] Kuo, R.J., K. Chang, and S.Y. Chien, "Integration of Self-Organizing feature maps and genetic-algorithm-based clustering method for market segmentation," *Journal of Organization Computing and Electronic Commerce*, 14:43-60, 2004.

[116] Asselemeyer, T., W. Ebeling, and H. Rose, (Torsten Asslemeyer, Werner Ebeling, Helge Rose) "Evolutionary strategies of optimization," *Physical Review E*, 56:1171-1180, 1997.

[117] Salas, A.O. and J.C. Townsend, "Framework requirements for MDO application development," AIAA-98-4740.

[118] Jin, Y., M. Olhofer, and B. Sendhoff, "Managing approximate models in evolutionary aerodynamic design optimization," *Proceeding of IEEE Congress on Evolutionary Computation*, 1:592-599, 2001.

[119] Tufte, E.R., *The visual display of quantitative information*, Graphics Press, Cheshire, Conn., 1983.

[120] Arora, J.S., *Introduction to optimum design*, McGraw-Hill Book Company, 1989.

[121] Lapworth, L., "Challenges and methodologies in the design of axial flow fans for high-bypass-ratio, gas turbine engines using steady and unsteady CFD," *Advances of CFD in Fluid Machinery Design*, Professional Engineering Publishing, Bury St Edmunds, London, UK, 2003.

[122] Newman, P.A., "Observations regarding use of advanced CFD analysis, sensitivity analysis, and design codes in MDO,"

[123] Eskin, D., Y. Leonenko, and O. Vinogradov, "Engineering model of dilute pneumatic conveying," *Journal of Engineering Mechanics*, 130:794-799, 2004.

[124] MaCluskey, D. R. and C. Elgaard, "PIV investigations of turbulence and rope dispersal," *IMechE Symposium on Flow Field Diagnostics*, London, Feb., 1991.

[125] Zhang, J. and J. Coulthard, "Theoretical and experimental studies of the spatial sensitivity of an electrostatic pulverized fuel meter," Journal of Electrostatics.

[126] Levy, A., T. Mooney, P. Marjanovic, and D. Mason, "A comparison of analytical and numerical models with experimental data for gas-solid flow through a straight pipe at different inclinations," *Powder Technology*, 93:253-260, 1997.

[127] Ginestet, A., J.F. Large, P. Guigon, and J.M. Beeckmans, "A new classification of flow regimes in vertical and inclined pneumatic transport," *2nd Int. Symp. Reliable Flow of Particulate Solids*, Oslo, Norway, 1993.

[128] Morikawa, Y., Y. Kobayashi, M., Takada, and Y. Ueura, "Pressure losses of air-solid mixtures in the branching of pneumatic conveying," *Bulletin of the JSME*, 17:1272-1277, 1974.

[129] Klinzing, G.E., N.D. Rohatgi, C.A. Myler, S. Dhodapkar, and A. Zaltash, "Pneumatic transport of solids in an inclined geometry," *Canadian Journal of Chemical Engineering*, 67:237-244, 1989.

[130] Jablonowski, D.J., J.D. Bruner, B. Bliss, and R.B. Haber, "VASE: the visualization and application steering environment," *Proceedings of Supercomputing'93*, 560-569, 1993.

[131] Geist II, G.A., J.A. Kohl, and P.M. Papadopoulos, "CUMULVS: providing fault tolerance, visualization, and steering of parallel applications," *International Journal of Supercomputer Applications and High Performance Computing*, 11:224-235, 1997.

[132] Kohonen, T., *Self-Organizing Maps*, Springer Series in Information Sciences, Heidelberg, New York, 1995.

[133] Davies, D. and D. Boulding, "A cluster separation measure," *IEEE Translation on Pattern Analysis and Machine Intelligence*, 1:224-227, 1979.

[134] SOM ToolBox software, accessed May 7, 2006,

http://www.cis.hut.fi/projects/somtoolbox

[135] GNUPLOT software, accessed May 7, 2006, http://www.gnuplot.info

Acknowledgements

I owe a dept of gratitude to many people who have been important for my studies and the completion of this dissertation.

First and foremost, I would like to thank my PhD supervisor, Dr. Kenneth Mark Bryden. He has provided me with a free exchange of ideas, encouragement, advice, and moral as well as financial support. Above all, I owe a lot of gratitude for his guidance and always being very open-minded and supportive regarding my projects and research. His training, advice, and insight into things have been invaluable.

I also owe a debt of thanks to my fellow student members on the VE-Suite team Doug, Gerrick, Angran, Steve. C., Steve. G., Dave, Jared, Sunil, and Balau. This project would not have been possible without the help from you guys. It has been a privilege to be part of such a hard-working and accomplished team.

I thank my family back home for everything they did for me; their love and support from the first day of my life has been a constant source of inspiration.

Last but not least I would like to express my sincere heartfelt thank to my husband, Johnathan for his dedication and incredible patience through this process, and for exposing me to the field of software development.