2003

# AMoEBA: the adaptive modeling by evolving blocks algorithm

Peter Eric Johnson
*Iowa State University*

# AMoEBA: The adaptive modeling by evolving blocks algorithm

by

**Peter Eric Johnson**

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Mechanical Engineering

Program of Study Committee:
Kenneth Bryden, Major Professor
Daniel Ashlock
Robert Brown
Ron Nelson
Adrian Sannier

Iowa State University

Ames, Iowa

2003

Graduate College
Iowa State University

This is to certify that the doctoral dissertation of

Peter Eric Johnson

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 ..... v

List of Tables ..... vii

Nomenclature ..... viii

Acknowledgements ..... x

Abstract ..... xi

Chapter 1. Introduction ..... 1
1.1. Virtual Engineering ..... 3
1.2. Motivation ..... 7
1.3. Overview ..... 14

Chapter 2. Background ..... 16
2.1. Reduced Order Modeling Techniques ..... 16
2.1.1. Nonlinear Interpolation ..... 17
2.1.2. Response Surface Techniques ..... 20
2.2. Coupled Systems ..... 26
2.2.1. Domain Decomposition ..... 26
2.2.2. Zonal Methods ..... 28
2.2.3. Multi-Solver Solution Techniques ..... 30
2.3. Evolutionary Optimization Tools ..... 33
2.3.1. Evolutionary Algorithms ..... 34
2.3.2. Genetic Programming ..... 38
2.3.3. Evolutionary Computation and Thermal System/Component Design ..41

Chapter 3. AMoEBA—The Adaptive Modeling by Evolving Blocks Algorithm ..... 45
3.1. Example of Manual Geometric Decomposition and Solver Assignment ..... 47
3.1.1. Navier-Stokes Equations ..... 49
3.1.2. Boundary Layer Equations ..... 49
3.1.3. Inviscid Flow Equations ..... 50
3.1.4. Results ..... 51
3.2. Description of the AMoEBA Algorithm ..... 56
3.2.1. Demonstration of AMoEBA ..... 65
3.3. AMoEBA: Approximating Algebraic Expressions ..... 70
3.3.1. AMoEBA Description ..... 70
3.3.1.1. Fitness Function Details of the Data Segregation Trees ..... 71
3.3.1.2. The Approximator Evolutionary Algorithm ..... 72
3.3.2. Problem Description ..... 75
3.3.3. Results ..... 79

3.3.3.1. Conduction Heat Transfer Results ..... 79
3.3.3.2. Sampling Rate Study ..... 84
3.3.3.3. Laminar Pipe Flow Results ..... 86
3.3.4. Conclusions ..... 95
3.4. AMoEBA: Inverse Heat Conduction Problems ..... 96
3.4.1. Literature Review: Inverse Engineering ..... 96
3.4.2. Problem Description ..... 99
3.4.3. AMoEBA Description ..... 103
3.4.4. Results ..... 107
3.4.4.1. One-Dimensional Results ..... 107
3.4.4.2. Two-Dimensional Results ..... 112
3.4.5. Conclusions ..... 118
3.5. AMoEBA: Computational Fluid Dynamics (CFD) ..... 118
3.5.1. AMoEBA Description ..... 122
3.5.1.1. Fitness Function ..... 123
3.5.1.2. CFD Solvers ..... 126
3.5.1.3. Other Adaptations to AMoEBA ..... 131
3.5.2. Problem Description ..... 132
3.5.3. Results of Preliminary Studies ..... 135
3.5.3.1. Solver Timing Studies ..... 136
3.5.3.2. Grid Study ..... 142
3.5.4. Results of AMoEBA for Flow Through an Elbow ..... 157
3.5.4.1. Grid Size Study for AMoEBA Segregations ..... 163
3.5.4.2. Elbow Angle Study for AMoEBA Segregations ..... 166
3.5.4.3. Boundary Condition Study for AMoEBA Segregations ..... 169
3.5.4.4. Obstructed Flow Study for AMoEBA Segregations ..... 171
3.5.5. Conclusions ..... 175

Chapter 4. Conclusions and Future Work ..... 177

References ..... 185

# List of Figures

Figure 2.1. An example of a genetic programming parse tree....40

Figure 3.1. The four different domains of a pipe entrance region....48

Figure 3.2. Transformed grid for the laminar pipe flow problem....52

Figure 3.3. Comparison of laminar pipe flow results....55

Figure 3.4. A data segregation tree with resulting data configuration....58

Figure 3.5. Flow of control of a generational evolutionary algorithm....60

Figure 3.6. An example of crossover....61

Figure 3.7. An example of the chopping function....63

Figure 3.8. Diffusion data set used in AMoEBA....66

Figure 3.9. Two examples of AMoEBA solutions....69

Figure 3.10. Exact solution of the one-dimensional heat transfer problem....76

Figure 3.11. Exact solution of the two-dimensional laminar pipe flow problem....78

Figure 3.12. Progress of the segregation trees....81

Figure 3.13. Temperature profiles of the best segregation tree/approximator team....82

Figure 3.14. Four methods for increasing the sampling rates....85

Figure 3.15. Results of the polynomial version of AMoEBA....88

Figure 3.16. Best solution of the laminar flow problem after 1000 generations....90

Figure 3.17. Segregation scheme of the best solution after 1000 generations....91

Figure 3.18. Approximators of the best solution after 1000 generations....93

Figure 3.19. Best possible solution of the laminar pipe flow problem....94

Figure 3.20. Exact temperature profile for the one-dimensional inverse problem....100

Figure 3.21. Exact solution for the two-dimensional inverse problem.....101

Figure 3.22. Comparison of the exact, most fit (correct), and most fit (incorrect)....108

Figure 3.23. Evolution of one-dimensional structures.....111

Figure 3.24. Evolution of two-dimensional structures .....115

Figure 3.25. Material placements of three improper solutions.....116

Figure 3.26. Material placements of three proper solutions .....117

Figure 3.27. Geometry for the manually segregated elbow.....127

Figure 3.28. Geometry for the elbow used in AMoEBA.....133

Figure 3.29. Time and accuracy comparison of 12 solvers using 572,000 cells .....139

Figure 3.30. Time and accuracy comparison of 12 solvers using 72,000 cells .....143

Figure 3.31. Effect of changing axial cell sizes #1 .....150

Figure 3.32. Effect of changing axial cell sizes #2.....151

Figure 3.33. Results of different angular cell sizes for the elbow .....153

Figure 3.34. Results of different radial cell sizes for the elbow .....154

Figure 3.35. Results of grid study using different cell sizes.....156

Figure 3.36. Examples of the five types of AMoEBA segregation schemes found..160

Figure 3.37. List of the eight AMoEBA solutions in Lisp-like format .....161

Figure 3.38. Geometry for the obstructed flow analysis .....173

# List of Tables

Table 1.1. Key technologies for virtual engineering ..... 8

Table 3.1. Results of the sampling rate study ..... 87

Table 3.2. Evolutionary parameters used in the inverse version of AMoEBA ..... 105

Table 3.3. Results of the one-dimensional inverse problem ..... 110

Table 3.4. Results of the two-dimensional inverse problem ..... 113

Table 3.5. Results of the timing trials using 12 solvers and the 572,000-cell grid .... 141

Table 3.6. Results of the timing trials using 12 solvers and the 72,000-cell grid ..... 144

Table 3.7. Results of the preliminary grid study ..... 146

Table 3.8. Results of the AMoEBA trials for the elbow problem ..... 158

Table 3.9. Results of the grid study using the AMoEBA segregations ..... 164

Table 3.10. Results of the angle study using the AMoEBA segregations ..... 168

Table 3.11. Results of the boundary study using the AMoEBA segregations ..... 170

Table 3.12. Results of the obstruction study using the AMoEBA segregations ..... 174

# Nomenclature

| | |
|---|---|
| $A_i$ | area of cell $i$ |
| $A_{tot}$ | total area of the exit plane |
| $B$ | sum of the squared differences between sample points when these points fall in different regions |
| $f(F)$ | fitness function of structure $F$ |
| $F$ | structure |
| $i, j$ | indices for grid locations |
| $k$ | thermal conductivity |
| $m, n$ | number of faces in the radial and angular dimensions, respectively |
| $\dot{m}_i$ | mass flow through the sector $i$ of the "exact" case |
| $\dot{m}_i'$ | mass flow through the sector $i$ of the coarse grid case |
| $\dot{m}_{tot}$ | total mass flow through the pipe (the flow through all $n$ sectors) for the "exact" case |
| $\dot{m}_{tot}'$ | total mass flow through the pipe for the coarse grid case |
| $M$ | number of cells in the $x$ dimension |
| $n$ | number of sectors |
| $n_o$ | population size |
| $n_t$ | tournament size |
| $N$ | number of cells in the $y$ dimension |
| $N_{max}$ | maximum total number of nodes in a tree |
| $P$ | pressure |
| $P_{bv}$ | probability of boundary value mutation |
| $P_e$ | penalty function |
| $P_f$ | probability of flip mutation |
| $P_l$ | probability of leaf mutation |
| $P_n$ | probability of any mutation |
| $P_{st}$ | probability of random subtree mutation |
| $q_0''$, $q_1''$ | heat fluxes from the x = 0 and x = 1 faces, respectively |
| $r_o$, $R_o$ | radius of the pipe |

| | |
|---|---|
| $R_{co}$ | crossover rate |
| $Re$, $Re_d$ | Reynolds number |
| $s$, $t$ | arbitrary data set values |
| $s^0_{i,j}$ | current boundary condition (three different velocities, pressure, turbulent kinetic energy, and turbulence dissipation rate) of the face located at position $i$, $j$ |
| $s^1_{i,j}$ | updated boundary condition (three different velocities, pressure, turbulent kinetic energy, and turbulence dissipation rate) of the face located at position $i$, $j$ |
| $T$ | temperature |
| $T_1$, $T_2$, $T_3$, $T_4$ | polynomials found for the one-dimensional heat transfer problem |
| $u$, $U$, $v$, $V$ | velocity components |
| $u_o$, $U_o$ | inlet velocity |
| $u_1$, $u_2$, $u_3$, $u_4$ | polynomials found for the two-dimensional laminar pipe flow problem |
| $u_i$ | axial velocity in cell $i$ of the single model "exact" solution |
| $u'_i$ | axial velocity in cell $i$ of the candidate case |
| $\overline{u}$ | average velocity through the pipe |
| $U_{cl}$ | centerline velocity |
| $U_e$ | source term |
| $W$ | sum of the squared differences between sample points when these points fall in the same region |
| $x$, $y$ | rectangular coordinates |
| $x$, $r$, $\theta$ | cylindrical coordinates (section 3.1) |
| $x^*$ | normalized $x$ coordinate |
| $z$, $r$ | cylindrical coordinates (section 3.3) |
| $\eta$, $\xi$ | transformation coordinates |
| $\eta_{max}$ | maximum grid location |
| $\rho$ | density |
| $\nu$ | kinematic viscosity |
| $\theta$ | cylindrical coordinate, temperature |

# Acknowledgements

I would like to thank my advisor, Dr. Kenneth M. Bryden, for his guidance, his inspiration, and all that he has done to help me in my career. Without Dr. Bryden's foresight and experience, I would not have been able to pursue many of the interests contained in this dissertation.

I would also like to thank Dr. Dan Ashlock. Dr. Ashlock has been a constant source of ideas and a valuable resource both to my colleagues and me.

The research contained in this thesis was made possible through funding by Alliant Energy under the direction of Dr. Edmundo Vasquez. I would like to thank Dr. Vasquez for his advice regarding my research and for helping to provide an industrial perspective of the applications of this algorithm.

I would like to thank Dr. Robert Brown, Dr. Ron Nelson, Dr. Adrian Sannier, and Dr. Judy Vance for serving on my program of study committee. Their advice and ideas were helpful throughout my Ph.D. studies.

I would like to acknowledge each of Dr. Bryden's students for their time and energy throughout my career at Iowa State University. Their advice and ideas helped me during key moments in my career.

Finally, I would like to thank my family and friends for their support. They were a constant source of encouragement and helped me to persevere through many of the stressful times during my studies.

# Abstract

This dissertation presents AMoEBA, the Adaptive Modeling by Evolving Blocks Algorithm. AMoEBA is an evolutionary technique for automatic decomposition of data fields and solver/descriptor placement. By automatically decomposing a numerical data set, the algorithm is able to solve a variety of problems that are difficult to solve with other techniques. Two key features of the algorithm are its ability to work with discrete data types and its unique geometric representation of the domain. AMoEBA uses genetic programming generated parse trees to define data segregation schemes. These trees also place solver/descriptors in the decomposed regions. Since the segregation trees define the boundaries between the regions, discrete representations of the data set are possible. AMoEBA is versatile and can be applied to many different types of geometries as well as different types of problems. In this thesis, three problems will be used to demonstrate the capabilities of this algorithm. For the first problem, AMoEBA used approximated algebraic expressions to match known profiles representing a steady-state conduction heat transfer problem and the fully-developed laminar flow through a pipe. To further illustrate the versatility of the algorithm, an inverse engineering problem was also solved. For this problem, AMoEBA placed different materials in the segregated regions defined by the trees and compared this to known temperature profiles. The final demonstration illustrates the application of AMoEBA to computational fluid dynamics. In this implementation, AMoEBA segregated an elbow section of pipe and placed numerical solvers in the regions. The resulting solver networks were solved and compared to a known solution. Both the time and accuracy of the networks were compared to determine if a faster solution method can be found with a reasonably accurate solution. Although AMoEBA is adapted for each application, the core algorithm of AMoEBA is unaltered in each application. This illustrates the flexibility of the algorithm.

# Chapter 1.

# Introduction

A key issue in the analysis and design of engineering systems is complexity. The analysis of any system can have varying degrees of complexity throughout. For example, the solution of fluid flow through a pipe relies on tightly coupled, nonlinear equations. The addition of solid particles to this flow greatly increases the complexity. If these are coal particles fueling a pulverized coal furnace, the combustion of the coal introduces more complexity into the solution. To create a solution for the entire power plant system in which this furnace resides, the equation set increases dramatically, bringing thermodynamics equations, structural analysis, and electrical considerations into the equation set. To analyze these types of systems, engineers have developed a variety of different methodologies to deal with their varying degrees of complexity. A method often used is to uncouple the system into different components. For example, a coal-fired power plant can be divided into models for the coal transport system, the furnace, the turbine, and the other components, each solved separately. However, these models do not capture the interrelationships between components and the system response to various changes. To capture these effects, a system model is needed. This system model is too complex to incorporate high-fidelity models of each of the components with the computing power available today. This requires that separate, less complex component models be developed for use in systems models. Higher fidelity approaches are common for individual aspects of single components, but the choices are limited. Finite element analysis (FEA) can be used for structural analysis.

Computational fluid dynamics (CFD) is a useful technique for modeling a variety of fluid flows from simple, two-dimensional, laminar boundary layer flows to complex, three-dimensional, turbulent combustion processes. In cases where both fluids and structural analysis are needed for some components, generally the analyses are run separately as uncoupled analyses and combined manually.

For the specific case of power plant analysis, plant modeling software may be used to describe the system by reducing it to a system of equations where the equations correspond to the components of the plant. Computer aided design is useful for different problems, and this methodology is beginning to be integrated with different numerical approaches including CFD and FEA. For the most part, these modeling techniques are not coupled together and the calculations are slow and computationally expensive. For example, CFD involves the solution of tightly coupled nonlinear equations on large grids of vertices (~10 million). In many cases, the computation time to get a single answer to a given problem can take from three days to three months, and the problem must be recalculated if refinements are made to the boundary conditions or the geometry. Often the previous solution provides an initial guess for the new calculation, but even with a head start, this second step can still take weeks or even months to get a new answer, causing an interruption in the design process. Due to this lengthy turnaround time, high-fidelity modeling techniques (e.g. CFD) are not often used as design tools. Alternatively, the engineer can build the system, study the impacts that the different subsystems may have on each other, and make any changes that are necessary. This is a "build and try" approach and is, in effect, a full-scale modeling technique. Scale models can also be used with varying success, but they can be difficult to

work with and are expensive and time consuming to build. Equation set models can also be used to analyze the system response. These models depend on reduced order models and minimize the number of details that are used in the analysis. In the furnace example, these models may disregard the flowfield and combustion effects inside the furnace and represent this component as a heat exchanger only. Although these techniques create an adequate model of the system, the reduction in fidelity of the model decreases the number of variables that can be adjusted to make improvements on the design of the system and reduces the detailed understanding of the system.

For high-fidelity computational modeling to become a routine design and engineering tool, the solution speeds need to be increased by a factor of 100,000-1,000,000×. Technology has shown and continues to show improvements in computer hardware speed and memory. If hardware technology continues to double the speed of a processor every 18 months (Moore's Law), in ten years the processor speed will be increased by a factor of approximately 100×. To reach the ~100,000× speedup necessary for complex numerical models to be used as design tools, software improvements must also be made.

## 1.1. Virtual Engineering

The term "virtual reality" has entered the popular lexicon and consequently can have different meanings for different situations. Virtual reality as used here is "... immersive, interactive, multi-sensory, viewer-centered, three-dimensional computer generated environments and the combination of technologies required to build these environments"

[1]. The key aspect of virtual reality as used in engineering applications is combining high-fidelity models, user interaction, and three-dimensional visualization into the same virtual space, creating a model that acts as if it were real.

Virtual reality has become an enabling technology in a variety of engineering applications due to its ability to present three-dimensional, complex data and information in an intuitive and interactive manner. Similarly, the use of high-fidelity modeling has become a useful tool in the engineering community due to the increased computational power made available by technological advances. The combination of these computational techniques is a natural and powerful one. For example, the inherent three-dimensional nature of the flows involved in CFD lends itself to visualization and interaction via a virtual interface. Additionally, coupling between various components of a high-fidelity system can make identification of all the impacts of a design change challenging. By providing a virtual engineering platform, the user can directly interact with the individual components, and therefore develop a more intuitive feel for the design, operation, and maintenance of the overall system. When fully-developed, this virtual engineering platform will be used to train operators, to explore the impact of engineering design changes, to develop and optimize construction and maintenance plans, and to communicate between various groups both inside and outside of the company including maintenance, engineering, and management. A key aspect of this work is to create a physically driven model that is more than a static visualization of data (virtual cardboard). That is, the goal is to create virtual models of physically real systems that respond in the same way as their real counterparts. Reaching this goal will enable the user to interact with the virtual world in the same manner

as they would with a real prototype. These interactions will not only be a method for analyzing the system but also for making design changes and witnessing the impacts these changes make on the design.

As demands for improved performance levels in various products grow and the costs of computation continue to decrease, the importance of numerical modeling as a design/engineering tool will increase. In many complex flows, such as those involving various degrees of fluid flow, heat transfer, and combustion, the numerical solutions become nonlinear and tightly coupled. The complexity of these interactions allows very subtle changes in the boundary conditions associated with the model to cause significant effects on the performance of the product and its cost effectiveness. The development of a complete virtual engineering platform will simplify the analysis process and make it easier to reap the benefits of these potential performance increases.

From a purely economical point of view, numerical modeling is more cost-efficient than using physical prototypes for design. As much as 80% of the total cost of producing some products is used in the design stage where expensive physical prototypes are created, tested, and adapted [2]. Much of this cost could be avoided if numerical models could be used instead of physical models. However, today, computational techniques are not often implemented due to the difficulty of regenerating geometry changes and the long turnaround time for recomputing these models. The reduction of these time costs is an essential step in increasing the use of high-fidelity modeling earlier in the design process.

In the 1950s, a significant amount of time was required to graphically render anything but the most primitive computer image. As technology advanced, visual fly-throughs in which the user could move through a digital world on a predetermined path and see things from a preset point of view became possible. By the early 1990s, the combined advancement of hardware technology and software algorithms led to the development of virtual reality where the path of the user was completely arbitrary and the visualization was calculated on-the-fly in real-time*. This development enabled the user to enter and explore a computer rendered environment in the same manneras she/he would explore the real world.

The advances made in computation and analysis of numerical models have many parallels to those made in the world of computer graphics and virtual reality. Before the digital age, fluid dynamics and other numerical problems were solved through analytical approximations and similarity techniques [3]. Since the 1960s, numerical methods and CFD have grown in power and capability until they are now considered to be an essential part of the engineering analysis process for fluid flows. However, today CFD plays only a limited role in the design process, which typically depends on low-fidelity models and physical prototypes. In the future, more and more aspects of engineering design will be completed via detailed models. Just as the step from visual fly-throughs to fully independent virtual reality required software as well as hardware advances, so too will the jump from today's design processes to fully immersive, fully digital virtual engineering require hardware and software improvements.

* Here real-time is faster than the eye can perceive: 15-30 times per second.

To make the technological leap to virtual engineering, a number of steps must be implemented and, more importantly, a number of tools must be created. Table 1.1 shows the key technologies that are required for the development of these types of virtual engineering environments. Table 1.1 is not all-inclusive, but it presents the groundwork for the types of technologies that must be developed. The focus of this work will be on taking the first steps toward developing the tools that are essential in giving this virtual engineering platform the flexibility and speed necessary for real-time engineering analysis and query.

## 1.2. Motivation

Computational speed is important for design due to the significance of real-time* feedback. Real-time feedback is an important characteristic of the design environment because it creates an atmosphere suitable to the what-if type questions essential to the engineering process, maintains the creative momentum developed in this process, and allows engineers to develop an intuitive feel for the solution. In this situation, an engineer may not require precise, fine-tuned answers; rather, solutions that are accurate enough to provide direction and reliable enough to base engineering decisions on are all that is necessary. An important result from a design perspective is simply knowing whether one idea outperforms another. Reliable approximations to the altered numerical data set can provide the qualitative results that determine whether the changes were effective. Similarly, changes made to the system

* Here we define "real-time" as the small amount of time that the user must wait to see the results of the changes they have made, on the order of 5-30 seconds. This is not the typical "real-time" of virtual reality and gaming software, in which the graphics must be rendered fast enough for any delays to go unnoticed by the human eye.

**Table 1.1. Key technologies needed for the development of virtual engineering design tools. The bold categories represent the areas that will be covered in the work presented here [4].**

| Area | Virtual Environments | Physics Modeling | Engineering and Design | Human-Computer Interaction | High Performance Computing |
|---|---|---|---|---|---|
| Technologies | Visualization within virtual environments | High-fidelity modeling | Robust optimization tools | Haptics | Massively parallel computing |
| | Virtual collaboration | Multi-scale analysis | Virtual design tools | Portable devices | Massive data sets |
| | Multi-sensory reality | Coupled systems | Integrated hierarchy of models | Remote collaboration | Heterogeneous computing |
| | Augmented reality | **Real-time physics modeling** | **Real-time analysis tools** | Wireless engineering | Long-lived data sets |

may not require the recalculation of the entire domain. Uncoupling the regions that do not change significantly from the recalculation process can reduce the number of calculations that need to be made, increasing the speed of the recalculation and providing answers to engineers' questions in a significantly shorter period of time. Nonlinear interpolation is one solution for making these approximations. Currently, interpolation between and extrapolation from complex nonlinear systems are possible but highly problem-dependent. Additionally, nonlinear interpolation is generally used to develop a single answer to a complex problem rather than to interpolate the richness and fidelity of a complete high-fidelity model. Because of this, nonlinear interpolation is not currently useful as a general design tool for many engineering applications. For example, the tightly coupled nature inherent to the underlying equation set of fluid dynamics makes nonlinear interpolation techniques difficult to implement for these types of problems.

Another category of techniques for making reliable approximations of numerical data sets are response surface methods. A response surface is a model, usually algebraic, of the reaction of a system based on a number of different variables. By using a simplified algebraic representation based on a select number of full calculations of the system, the response of the variables can be recalculated more quickly than, and almost as accurately as, a high-fidelity model. One weakness of this method is that details of the system are lost in the simplification. Also, to obtain the same level of accuracy as high-fidelity techniques, many calculations of the system are required. This can take a large amount of time in the creation of the surface.

The challenge in creating a useful and effective engineering design tool is to enable changes to be made to a component and the response to be calculated with sufficient accuracy for an engineer to understand the answer. This requires speedup of high-fidelity models. Numerical models are often very large and require an excessive amount of compute time to calculate. For example, CFD models solve varying forms of the Navier-Stokes equations for any number of nodes (typically as large as 10 million for current models). In this case, simplifications such as boundary layer codes, inviscid flow approximations, the Bernoulli equation, etc. can be very accurate under certain circumstances, but are not generally used for large complex flows. Similarly, there are instances when other solvers (FEA, similarity techniques, etc.) perform adequately with significantly improved performance times. By implementing varying forms of these simplifications in the same algorithmic model, a multi-solver computational technique is created. Similar multi-solver methods are possible for different numerical models as well. Non-dimensional situations exist that would benefit from the segregation of the different components or systems involved. The power plant example discussed earlier could be automatically segregated into the furnace, turbine, and fuel transport components, and these could be further segregated into different subsystems of the individual components. Not only does dividing the entire complex model into smaller models reduce the complexity of the system, it also provides a framework for recalculating the problem after changes are made. Returning again to the power plant example, it may be determined that changes made to the turbine will have little impact on the coal transport piping network. If the entire system is segregated into smaller sub-systems, only the components that are affected by the changes need to be recalculated. At a finer level, the individual components themselves can be further segregated. For example, portions of the

furnace may change only slightly when new boundary conditions are applied at the burners. Changes made to an elbow in the downstream portion of the coal piping may have very little impact on the flowfield upstream of these changes. In these cases, recalculation of the entire furnace model and the entire coal piping is not necessary. Therefore, the computational resources available can be applied where they are most needed, reducing the time to solution of these models. Similarly, there may exist regions that can be modeled with more simplified solution methods. These regions may need to be calculated but may not require the detailed numerical modeling that other regions in the domain require. For example, the heat transfer from the flue gases in the furnace to the steam used to drive the turbine may not require a high-fidelity model. A simple response surface may be sufficient to transmit the energy information from the furnace to the steam at the level of accuracy that is needed. By creating a network of different solvers, each zone can receive the minimum attention and fidelity it requires to efficiently produce a solution that is within a known level of accuracy. The difficulty of producing such a multi-solver method is in the application: how does one decide which areas of the model require the most complicated solution techniques and which areas can be adequately approximated by a simplified model?

The multi-solver method in this problem statement resembles domain decomposition techniques. Domain decomposition is a method for dividing a numerical domain into more manageable pieces. The main purposes for making these divisions are for parallelization—sending each sub-region to a different processor—or for creating a mesh on a difficult geometry. In both cases, the same solver is used in each of the decomposed domains.

Combining different solvers as a solution technique is not a new idea for fluids problems. Before the onset of the digital computer, engineers devised unique methods of solving complicated flows. Because the computations were performed by hand, it was essential to reduce the number of calculations performed. Engineers developed methods for combining analytical solutions through similarity techniques [3]. A brief example of the application of these techniques is the simple, flat-plate boundary layer. Inside the boundary layer, viscous effects are essential to the development of the flow as well as to the solution of the wall boundary condition. Outside the boundary layer, viscous terms can be neglected as the overall momentum of the flow far outweighs any effects attributed to the viscosity of the fluid. Engineers in the 1940s and 1950s applied these observations by utilizing different analytic solution methods for the two different regions. Flow over an airplane wing is a natural progression from the flat-plate example described. In problems such as this, engineers use the same idea of using one solver in the boundary layer and a different solver for the outer, inviscid flow. However, these techniques, known as zonal methods, are mostly used in high-speed flows where knowledge of the flowfield can be used to determine where each solver is used. Multi-solver techniques for more complicated flows are also being studied. As with zonal methods, these techniques require user intuition for the placements of the different solvers, and therefore, the solver networks are not automatically generated.

The development of an automatic multi-solver technique is difficult. Not only does the technique require knowledge of the overall flowfield, but also knowledge of how well each

solver performs in the different regions. Without this previous knowledge of the division of the flow, applying many solvers to one computational model can be viewed as an optimization problem. To solve this optimization a variety of capabilities are required of the multi-solver technique. Two of these include:

- The ability to work with discrete inputs—the multi-solver technique should be able to use distinct solvers that are completely separated except for the variable information that is passed between them.
- The ability to automatically decompose the geometry of the system—the optimization scheme should be able to segregate the data set into regions in which the solvers are placed.

There are infinite numbers of distributions of how the solvers can be applied, but different combinations of solvers will perform at varying degrees of accuracy. The solver placements will also differ in the run time used to solve the given domain. Each of these criteria needs to be optimized, or at the very least an accuracy level must be defined and the multi-solvers that meet this level judged on their respective times to solution. Under these circumstances, the problem becomes a challenging, multi-objective optimization problem with a large search space and discrete inputs. Evolutionary computation is becoming more and more useful to engineers in solving difficult optimization problems such as this. Evolutionary algorithms are a type of evolutionary technique. They have been described as "... powerful global optimizers that can negotiate complex nonlinear search domains to provide optimal design solutions" [5]. The versatility of evolutionary algorithms is applicable to a variety of engineering-related problems ranging from the optimization of turbine blades [5] to controlling the rate of speed for a bank of presses in a sugar-beet plant [6]. Because of their

robustness, their ability to handle discrete inputs, and their ability to explore large search spaces, an evolutionary algorithm is the prime candidate for the multi-solver technique.

## 1.3. Overview

This dissertation presents the Adaptive Modeling by Evolving Blocks Algorithm (AMoEBA), an evolutionary algorithm based strategy for automatic data segregation and solver placement. AMoEBA uses a binary tree to automatically segregate a data set into regions and place different solution methods in these regions. The trees are evolved until a scheme is found that meets the criteria set forth by the user. In this thesis, the versatility of the AMoEBA technique is demonstrated on three different problems:

- the development of approximating algebraic expressions to match the solution of a one-dimensional temperature profile and a two-dimensional velocity profile,
- the solution of two different inverse conduction heat transfer problems, and
- the solution of turbulent pipe flow in an elbow.

In each case, AMoEBA automatically segregates the problem domain and places the appropriate solution methods in each of the regions.

Chapter 2 begins with a discussion of two types of reduced order models that are being used: nonlinear interpolation and response surfaces. Also included in the chapter is an overview of three different types of coupled systems: domain decomposition, zonal methods, and multi-solver techniques. The final section of this chapter briefly describes evolutionary algorithms and genetic programming. Background material of some examples

of the use of evolutionary techniques applied to thermal systems completes this section of Chapter 2.

In Chapter 3, an example of a manual multi-solver is presented. Following this section, AMoEBA is described in detail, and three applications of AMoEBA are discussed. The first of these is the development of approximating algebraic expressions to automatically match the solution of a one-dimensional steady-state heat conduction problem and a two-dimensional laminar pipe flow problem. This discussion introduces the reader to the terminology used throughout this dissertation and demonstrates how the AMoEBA algorithm can be applied to standard engineering problems. Applying AMoEBA to inverse engineering problems is the next section of this chapter. In this section, two different inverse heat conduction problems are solved. This application of AMoEBA is more complicated than the first and develops more of the techniques that are used in the algorithm. The final section applies AMoEBA to computational fluid dynamics by using the example problem of flow through an elbow section of pipe. After building the reader's knowledge of AMoEBA in the first two implementations, this application of AMoEBA is more complicated and makes use of the terminology and structure provided earlier in the dissertation.

Chapter 4 concludes this dissertation with an overall synopsis of the results of the three applications of AMoEBA, a restatement of the direction of AMoEBA, and the direction of future work in this area.

# Chapter 2.

# Background

As discussed in Chapter 1, several technologies have been developed to cope with the complexity of engineering systems. These technologies include: 1) reduced order modeling techniques that simplify the problem or decrease the fidelity to model the global characteristics of the system; 2) coupled systems that manually decompose the domain into more manageable pieces, applying different solution methods where necessary or automatically decomposing the system to solve the pieces in parallel using a single solution method; and 3) robust optimization tools that search a large portion of the optimization landscape and can be applied to a wide range of problems.

## 2.1. Reduced Order Modeling Techniques

Reduced order modeling techniques use global calculations to create a low-fidelity model of the domain. By focusing on the overall parameters of the system, these techniques avoid the computational costs of calculating the fine details of the numerical domain. Although this methodology is accurate in many situations, it lacks detail and precision, which limits its use as a numerical tool. By reducing the model to its global characteristics, these techniques eliminate high-fidelity details of the systems that may be useful parameters for design changes. Additionally, without the fine-scale details, the full aspect of design changes may not be understood, decreasing the effectiveness of these techniques as engineering tools. However, on a much broader scale and in areas where only the global

characteristics are important, these are very useful techniques that can provide fast, accurate solutions to that portion of the system. Several techniques have been developed for creating reduced order models. Two of these include nonlinear interpolation and response surfaces. In nonlinear interpolation, an attempt to reproduce the information between two or more points of a nonlinear data set is made by performing calculations on the points. Typically these calculations rely only on local information to predict the values between the known points. Response surfaces are similar in that they too are used to predict the information between known data points. Often response surface techniques utilize information throughout the domain to fit a surface to the known data instead of using only local information for the predicted solution.

### *2.1.1. Nonlinear Interpolation*

In the early stages of the design process, when many new ideas are implemented and tested, one solution to a question is usually not sufficient. A more useful design platform is one that allows engineers to make changes to the original prototype and witness the results of those changes soon afterward. Typically, this option is performed using physical models due to the extended time required to recalculate numerical simulations. By interpolating between known cases, the time required for recalculating computational models may be decreased. Unfortunately, many numerical models are nonlinear, and often the equations that produce these models are tightly coupled as in the case of CFD solutions. Interpolating between nonlinear data sets like these is difficult because of the coupled, nonlinear nature of the solutions. Nonlinear interpolation is not a general technique but a method that is problem dependent. The technique used to interpolate information in one data set may not

work for a different type of data set. Three examples of nonlinear interpolation will be discussed. Although the methods used are completely different, the nonlinear interpolation techniques are not interchangeable, which illustrates the non-generality of this solution technique.

The first example is the reconstruction of experimental data from the measurement of the topography of a body of water [7, 8]. Under most circumstances, this topography can be derived linearly from slope-image data. However, when the body of water becomes turbulent, the discontinuities that are created by the breaking waves turn the problem into a nonlinear one. By using a combination of genetic algorithms and Fast Fourier Transform integration to solve for Fourier series coefficients, researchers have been able to correctly reproduce the topographies from slope-image data even under turbulent conditions. The technique is based on the fact that Fourier series coefficients are the same for the derivative (slope) of the series as they are for the series itself. By using a genetic algorithm to solve for these coefficients, nonlinearities in the data, such as bubbles, noise, water spray, or missing data, can be modeled.

A second example of nonlinear interpolation has been used in predicting speech patterns [9]. The technique uses multi-layer artificial neural networks trained on each of the 37 phoneme symbols used. An error-back propagation algorithm was used to adjust the weights of the neural network during this training phase. The neural networks were used to match the phoneme symbols to 17 French sentences spoken five times by one male speaker and manually labeled. The neural networks use speech frames at times ..., n-2, n-1 and n+1,

n+2, ... in their nonlinear interpolation of the speech frame at time n. By using more of the information being presented, they are able to predict speech patterns more accurately than previous methods.

Nonlinear interpolation has been used in fluids problems as well [10, 11, 12]. Zhong, Weng, and Huang [10, 11] and Zhong, Adrian, and Huang [12] have used an interpolation technique that combines the constraints of the physical system with the interpolation scheme itself. This streamlines the solution process by making a one-step method instead of the two stage, iterative schemes that are most often used. However, since this process results in an overdetermined linear system, the system must be solved differently than a normal set of equations. A sliding window is used to solve this system. This has two benefits. The first allows the over specified system to be solved. The second is that since the interpolation is only carried out in the window of influence, smoothing effects are not introduced globally, which improves the accuracy of the technique.

Although these three cases adequately interpolate between their respective data sets, in the first two cases the methods used are not easily applied as general nonlinear interpolation techniques. Both examples use a large amount of known information to create a missing data point. In the water topography example, much of the data set was linear and known a priori. Reconstruction of this topography relied on the known data set to fill in the missing data, a technique that is not useful for predicting an entirely new data set based on known information of one or two original data sets. Similarly, the speech recognition example used information contained at previous time steps as well as later time steps in the interpolation

scheme. In all three techniques, the interpolation schemes were used to fill in missing data for a given data set. In each case the schemes were successful; however, the techniques do not provide an adequate method for extrapolating known information to new boundary conditions or to geometry changes. These are the characteristics that are required of a virtual engineering design tool.

### *2.1.2. Response Surface Techniques*

A response surface is generally thought of as an experimental optimization technique, but it is a technique that is often used in numerical situations as well. The idea of a response surface is to create a surface based on the performance of a numerical model or experiment with respect to any number of parameters. If this surface is approximated, the optimum of the model, or simply any location on the surface, can be found more easily using the surface and the input parameters than by trying to optimize the complicated system it is modeling. As with other reduced order techniques, response surfaces can result in accurate solutions to the problem but at the expense of flexibility and robustness. Unforeseen changes to the system are difficult, if not impossible, to incorporate without creating a new surface, and creating a surface based on all the variables in the system can be very time-consuming. Despite these restrictions, response surface techniques are useful for fast solutions of a system, and depending on the time and effort used in the creation of the surface, they can be very accurate as well. Recently, response surface approximations have become popular for modeling complicated numerical systems [2, 13-19]. The complexity of these data sets is reduced by the approximated solution, allowing the computations to be made much faster.

The surface itself is created by modeling an approximation (usually low-order polynomials are used) of the dependent variable based on a number of independent parameters. An example of a response surface would be to approximate the yield of a chemical equation. Trials would be made based on variations in temperature, reaction time, and pressure. Once the surface is created, the yield of any combination of temperature, reaction time and pressure within the bounds examined can then be found more easily by using the approximation instead of performing the actual calculation or experiment. Global reaction models using Arrhenious relationships are an example of this type of reduced order models.

Another example is to use a response surface to approximate the behavior of a flowfield far downstream of a complex region. This section of the flowfield may not change much and may be easy to calculate. A database of a number of different inlet conditions to this region would be stored and fitted with a polynomial to create the surface. Once a new complex region is solved, the response surface section would be found by using the new inlet conditions to the response surface approximation. In this case, an optimum of the surface would not necessarily be important; recomputing new locations on the surface quickly would be the end goal.

One weakness of response surfaces includes their use in general cases. Since the surfaces are created from preliminary knowledge of the system, they become focused on the particular problem for which they are designed. Difficulties can also arise when response surfaces are used in highly discontinuous domains. Because the surface is an approximation, they tend to smooth out most of the sharp peaks and valleys of a

discontinuous region. This aspect is very useful for eliminating the noise contained in many numerical data sets and is an advantage over derivative-based search algorithms that struggle with noisy data sets [13]. A problem with this technique arises when the discontinuities of a data set are important. To obtain the fine precision necessary in resolving these features, a large number of experiments must be run, increasing the time spent on the surface creation and decreasing the usefulness of this tool.

Applications of response surfaces have included the minimization of the takeoff gross weight of a High-Speed Civil Transport (HSCT) [2], the shape optimization of a diffuser [13], and the optimization of the strength of bonding leads to electronic equipment [14]. In the case of an HSCT configuration, low-fidelity aerodynamic models generate the response surface [2]. The response surface is used to evaluate performance or aerodynamic constraints to be used by a high-dimensional, tightly constrained, multidisciplinary design optimization procedure. Twenty design variables that define the size and shape of the wing and body of the plane are used in the optimization. The response surface was created from low-fidelity models to evaluate up to 50 performance- or aerodynamic-related constraints. Examples of these constraints include emergency landing conditions for which it is assumed that the aircraft lands 5000 ft above sea level at 145 kn, carrying 50% of its initial fuel weight. To calculate the constraints based on this information, estimates must be made of aerodynamic forces and moments, stability and control derivatives, and center of gravity and inertia of the aircraft. The surface is used to obtain calculations of the constraints when necessary in the optimization procedure. This technique is based in part on variable complexity modeling, which combines the accuracy of high-fidelity models with the

efficiency of simpler simulations. In this case, the low-fidelity model consists of the Euler equations with viscous drag corrections. As the responses of these low-fidelity simulations are modeled, terms that are found to have little effect on the surface are eliminated, leaving the more important terms. Because of this, fewer CFD analyses are required for computing the coefficients of the models. The drawback to this approach is the possibility of overlooking subtle influences of certain important nonlinear effects; however, the results have shown that the time saved outweighs this minor loss of accuracy.

Another example application of response surface techniques is for diffuser shape optimization [13]. Diffusers are often used to convert dynamic pressure into a static pressure rise by reducing the velocity of the flow. Maximizing the pressure recovery of a diffuser is a benchmark problem for CFD. Solutions to diffuser problems have the nature of converging quickly, and solutions to these components can be found for many different flow conditions. Two methods were used in the creation of the response surface for this problem. The first used two design variables by defining the two-dimensional diffuser shape with a fourth order polynomial (the two endpoints of the polynomial are given). The second case used five design variables and defined the shape by a B-spline using the two endpoints, the five design variables, and a prescribed inlet slope (fully-developed flow is assumed at the entrance to the diffuser). Although there seems to be little difference between these two cases, from the perspective of the response surface there is a large difference: the number of variables used in the creation of the surface. The creation of the response surface for this problem begins by limiting the search space to exclude poor designs. Enforcing these limits not only simplifies the complexity but also reduces the time

spent on creating the surface. Once the limits are defined, the surface is created by choosing a representative number of designs with a variety of parameters. The number of designs analyzed depends on the complexity of the approximating surface. For example, for a quadratic polynomial, (n+1)*(n+2)/2 terms are required to produce a fit of the response. For the two design variable case, this yields six terms, whereas 21 terms are required for the five design variable case. In this research, both cases produced similar results for the diffuser shape optimization problem. Both optimum shapes were able to find a slight improvement over a gradient-based search optimization technique. The difference between the response surface approximation technique and the gradient-based technique can most likely be attributed to the robustness of the response surface method.

Response surfaces are designed to create a simplified approximation of a complex domain, typically to obtain a more easily optimized search space. Even with a polynomial approximation, polymodal optimization landscapes can result from response surfaces of complex problems. In cases such as these, genetic algorithms have been used to find the optimum of the complicated surface that results [14]. One such application receives a surface definition as an input to the genetic algorithm combined with the various controlling parameters for the genetic search. Three optimization landscapes were used to test this combination of a genetic algorithm and a response surface. The first test of this application used a known mathematical function as the search space. With this function as the input surface, the genetic algorithm was able to find the optimum after 5,825 generations with most of this time spent climbing the correct hill. The second case studied utilized a commercial response surface optimization software package, Design Expert, to

both generate the response surface input into the genetic algorithm and to locate an optimum solution to the surface to compare with the genetic algorithm's solution. The response optimized was the mean pull strength of leads bonded to a die. The bonding machine attaches leads to a die based on the temperature, the bonding force, and the bonding time. These variables are constrained to vary within predetermined ranges. Destructive testing is used to determine the mean pull strength of a bonding process, and this is the response to be monitored. Using this response surface, the genetic algorithm outperformed the commercial software. The final case using the genetic algorithm optimized response surface was based on the same problem, but in this case both the mean pull strength and the minimum pull strength were optimized. Two different surfaces are created, one for the mean pull strength (the same surface as the second problem) and one for the minimum pull strength. This time Design Expert produced slightly better results, but it is expected that changing parameters in the genetic algorithm could improve this technique for this final problem.

In each of these examples, the reduced order modeling techniques that were used (nonlinear interpolation or response surface) adequately modeled the system at the expense of fidelity. Although this produces fast accurate solution methods, in essence these techniques convert the model to a black box that is detached from the details of the data set. For the cases presented, this is acceptable since only one solution is required. However, for design problems in which many solutions are required, the reduction in fidelity can make multiple solutions difficult, if not impossible, to calculate.

## 2.2. Coupled Systems

Coupled systems include domain decomposition, zonal methods, and multi-solver solution techniques. Each of these decomposes a model into different regions. In domain decomposition, models are decomposed to parallelize the solution of the model or to create a grid for complicated geometries. Zonal methods use this technique, typically for high-speed flows, to place different solution methods in the decomposed regions. The combination of the different methods allows the user to use less grid to achieve a similar degree of accuracy on the flowfield. Multi-solver solution techniques are very diverse, as will be shown by the three examples to be described later. The common thread in each of these is that different solution methods are connected to form a larger network of solvers.

### *2.2.1. Domain Decomposition*

Domain decomposition is a mathematical technique that has been used for many years to solve complicated partial differential equations (PDEs) [20]. In the late 1980s, this technique became popular among CFD experts due to its ability to segregate the computational domain into manageable pieces either for meshing complicated geometries or for parallelization of their code [20]. These techniques were necessary to solve flows in complicated geometries while retaining structured grids on the subdomains [21]. By dividing the geometry into smaller pieces, complicated geometries can be meshed with irregularities safely falling in the overlap regions. The size of these overlap regions has a large influence on the accuracy and speed of the solution: a larger overlap often results in a better answer, but it may require longer or shorter computation times. EZGrid (Expert Zonal Grid generator) is an example of an early application of automated domain

decomposition techniques for grid generation [22]. By automating the decomposition process, less time is spent in this stage of the CFD analysis.

Another major use for decomposing a numerical domain has been for grid refinement in certain areas of the flowfield. An unfortunate side effect of this is the grid imbalance, which disrupts the parallelization scheme. In most cases, domain decomposition is imposed to divide a large problem into many subproblems of similar size to send them to different processors. Refinement of some of these regions causes the other segregations to be solved more quickly, weakening the efficiency of the parallelization strategy. An alternative to refinement is to use a moving grid where the grid points are moved to important regions, refining these regions while coarsening other regions. This technique is not as effective as grid refinement, but is much easier to control for parallelization because the matrix size stays the same and transformations can be used to account for the grid movement. Ribbens [23] addressed the issue of local grid refinement by creating an adaptive decomposition scheme that is applied during the choice of the decomposition itself. This strategy results in "... a convenient and efficient high-level mechanism for adaptively choosing the decomposition" [23].

Domain decomposition has recently been used to advance the optimization capabilities of genetic algorithms. By decomposing the search space, separate regions can be studied simultaneously with the goal of decreasing the time spent in the optimization process [24]. This hybrid genetic algorithm uses evolutionary techniques to find the "hills" of the search space and decomposition methods to segregate the search space into these hills and the rest

of the domain. The hills are climbed by a local search routine, and the genetic algorithm explores the rest of the domain. The local optima are found more quickly with the local search routine and the genetic algorithm is used to find the best areas for applying these routines. In a broad sense, this is similar to the technique in this thesis in which CFD solvers are placed in the domain and used to solve a fluid flow instead of placing a local search routine for solving an optimization problem, as in [24].

### *2.2.2. Zonal Methods*

Zonal methods are strategies based on domain decomposition and draw many parallels to the work presented in this dissertation. Often they are used to refine certain areas of a decomposition for more efficient grid generation [25]. They are popular in the aerodynamic community for solving complex, compressible, high-speed flows past objects. Accurate results for these types of problems typically require the full Navier-Stokes equations to be solved on very large grids. Good approximations can be found on smaller grids by using viscous/inviscid coupling [25-28]. In viscous/inviscid coupling, a boundary layer code is used near the surface of the objects on a coarser grid than would be required by the Navier-Stokes equations. Outside of this boundary layer, the Euler equations can be used due to the inviscid nature of the flow in this region. A much coarser grid is used in this region as well, reducing the memory load required. Both the storage benefits and the computational speedup of this method have made complicated transonic flows solvable in reasonable time frames.

One interesting implementation of the Euler/boundary layer method has been as a design tool for the transonic re-entry flow of the space shuttle [29]. The extreme conditions that arise during re-entry make reproduction of these conditions in wind and shock tunnels impossible. Creating dependable numerical models is also difficult for these circumstances due to the complexity of combining accurate physical and chemical models at a reasonable computational cost. By solving the necessary aerothermochemistry equations with a coupled Euler/boundary layer method, the flowfield is modeled within reasonable error bounds and in a reasonable compute time when compared with the alternative of using the full Navier-Stokes equations for the fluid calculations.

A more detailed tool using these techniques uses both the viscous/inviscid coupling technique described previously and the full Navier-Stokes equations [30-32]. The tool combines the Euler/boundary layer solvers in regions of weak viscous/inviscid interaction and uses the Navier-Stokes equations to solve the more complicated regions in which the viscous/inviscid interaction is strong. An example would be in the transonic flow past an airfoil. For this example, the Navier-Stokes equations are used in the boundary layer region of the shock as well as in the wake flow. By using three solvers instead of the two mentioned earlier, the accuracy is improved and more difficult problems become tractable.

A unique application of the viscous/inviscid solvers discussed previously has been to use a genetic algorithm with this technique [33]. In the design of aircraft wings, there are many different conditions that need to be addressed. Two of these are to maximize the lift of the wing during takeoff conditions while minimizing its drag during cruising conditions. In

their research, Quagliarella and Vicini [33] created a genetic algorithm that varied geometric parameters of wing designs and then called two different solvers to analyze the wing under these two conditions. For the maximum lift condition a viscous/inviscid interaction method was used, while for the cruising drag an Euler flow solver and an integral boundary layer routine were used. By using multiple solvers, the optimization can analyze geometries very quickly depending on the conditions applied. Also, by matching the complexity of the solver to the appropriate conditions, highly accurate, complex solvers are only used when absolutely necessary, reducing the time for the optimization process.

Each of these techniques uses viscous/inviscid coupling of the boundary layer equations and the Euler equations. These have been used for unique applications such as the space shuttle analysis, with the full Navier-Stokes equations as in the transonic flow study, or with a genetic algorithm in the maximized lift example. However, in each of these cases, previous knowledge of the flow was used to assist in the placement of the different solvers. Zonal methods provide a preview of the capabilities of a multi-solver method without the automation that is necessary for use in engineering design.

### *2.2.3. Multi-Solver Solution Techniques*

Some zonal methods utilizing different simplifications of the Navier-Stokes equations have been applied to CFD problems, typically in high-speed flows. Broader uses of the zonal method have gone beyond using different subsets of a complete set of equations to solve problems but have combined differing models and solution techniques to solve an engineering problem. Three examples of these types of applications are presented here.

1. Many complicated physical systems require multiple PDEs to accurately create numerical models of these systems. In most cases, these models are designed specifically for the system to be modeled. An alternative is to use existing code for the separate PDEs and combine them in ways that allow the physical laws to be preserved. An important part of this network of PDE solvers is the connection between them. One method for connecting these solvers is an interface relaxation method described in detail in [34]. This method begins by solving the PDEs themselves. Then, based on the values of the solutions, the interface conditions are calculated using a relaxation formula. Both are iterated until the system converges. The relaxation formula that is used to compute the interfaces of the PDE network is obtained from knowledge of the physics that define the system. In general, the types of calculations that are made along the interfaces require the values and derivatives to be known or calculated. Two techniques are presented in detail in [34] for creating an application that can solve such a PDE network.
2. Researchers at ALSTOM Power [35] have developed a unique method for solving fluids-related problems. In the past, their work for improving their Selective Catalytic Reduction systems has been limited to large, expensive physical models. These models took long periods of time to create, and even scaled-down models were expensive to build, typically eating up 50% of a project's budget just to get to a point where initial data could be recorded. Once the physical models were created, they were relatively easy to operate, modify, and retest. Also, as with any physical model, they were inherently "converged." An alternative method is to use CFD to model the flow through the system. These types of computational models are much quicker to set up initially,

but they take a long time to find a solution. Changing the geometry of a CFD model is also difficult, and time is again lost when a new solution is queried. ALSTOM Power's solution is to combine the two techniques to take advantage of the strengths of each while minimizing their weaknesses. In the system being examined, there are segments that require many adaptations before the desired $NH_3$:$NO_x$ ratio distribution is achieved. Other areas in the system are fairly stable, and geometric changes are not necessary in these regions. The hybrid model that has been used by ALSTOM Power numerically models the entrance region to the system using CFD. The exit conditions of this model are then used as a basis for the inlet flow into a physical model. This is a scaled model of the middle section of the system from a point upstream of the ammonia injection grid through the first catalyst layer. Data from the exit of this model is recorded and used as an inlet condition into a final CFD model of the exit of the system. The physical model of the middle region is easily adaptable and can be adjusted to obtain the desired ratio. The CFD models reduce the cost of the entire system since they can be run on available computing resources. Space is also conserved by reducing the size of the physical model. Indications have shown a 25-30% reduction in cost with a similar time savings in the project schedule.

3. Stephen Niksa [36] uses a multi-solver technique for calculating the formation of $NO_x$ for different fuels. Based on a converged CFD model of a chemically reacting flow, the domain is segregated into different regions depending on the mass fraction of the combustibles. Each region is then further represented by a number of continuously stirred tank reactors (CSTRs). The resulting reactor network is evaluated to determine the $NO_x$ formation throughout the entire flowfield. This technique is very similar to the

one presented in this thesis, but there are differences also. In Niksa's work, the flowfield is interactively segregated and the CSTRs are applied based on the engineer's analysis capability and the residence time distributions. In the project described in this dissertation, the goal is to evolve a segregation scheme with the proper solver assignment placed automatically. The overall purposes of these two projects also differ. The goal of Niksa's research is directed toward a specific problem, for which a tailored solution method has been developed; whereas the goal of the research presented in this thesis is to develop a more generalizable technique.

Each of these multi-solver methods combines different types of solution techniques into a conglomerate solver. The PDE network does so out of necessity, depending on the physical system being represented. The ALSTOM Power physical/numerical system does so to reduce cost and turnaround time. Niksa's method does so to get an approximate answer to a complicated problem in an acceptable turnaround time. Although in each case this technique of combining different solution mechanisms to create a multi-solver is successful for the application, each method requires user input to define the boundaries between the various solvers. A needed extension to this methodology is to automatically create the solver network and the segregation scheme that defines the boundaries between them.

## 2.3. Evolutionary Optimization Tools

The study of optimization has been developing and improving for the last 40 years [37]. With the advancement of current computation resources, the techniques developed by researchers have become more and more useful for many types of engineering design problems. These techniques range between the standard methods of the golden section

algorithm and Newton's Method, to the steepest descent method and conjugate gradient techniques, to more complicated and robust techniques such as the simplex method and the branch and bound method. Each of these optimization techniques excel at solving one type of problem or another. Problems also exist in which one or more of these techniques struggle. In general, most of these methods are gradient-based and depend on unimodal landscapes or on additional techniques to assist in locating the correct hills of polymodal spaces.

One area of optimization that performs well on polymodal problems is evolutionary computation. Evolutionary computation covers a broad category of techniques that follow nature's lead in locating optimum solutions to difficult problems. Evolutionary computation has provided engineers with a variety of very robust optimization techniques. The research in this dissertation utilizes two types of evolutionary computation techniques: evolutionary algorithms [38] and genetic programming [39-44]. A short description of evolutionary algorithms will be provided, followed by a brief overview of genetic programming. Evolutionary algorithms have been particularly popular for a variety of engineering optimization problems. The examples discussed here focus on numerical modeling of energy systems, but the techniques are used in many other areas of engineering as well.

### *2.3.1. Evolutionary Algorithms*

In this section, a brief overview of evolutionary algorithms will be given. A more in-depth discussion can be found in [38]. Evolutionary techniques, in general, are very robust and typically explore a much larger portion of the search space than other optimization

techniques. Based on the Darwinian notion of the survival of the fittest, evolutionary algorithms start with a completely randomized population of candidates, or structures. In engineering problem solving situations, these structures are typically trial solutions to the system being optimized. The evolutionary process begins with the *selection of parents*. A number of selection methods exist, and they often involve some degree of randomness. One selection method is to choose the "best" two structures in the population to become the parents. Determining which structures are "best" is problem-dependent and will be explained later. Once parents are selected, *mating* happens. In this dissertation, mating consists of three steps that create two children. The first step is to copy the parents. A secondary operation, *crossover*, is then applied to the duplicated structures. Crossover is a process that takes a piece (or pieces) of each child and swaps them, creating two new structures that are unlike either of the two parents yet share similarities with both. The third step is to mutate, introducing small changes. A good example for describing evolutionary algorithms is the problem of matching a binary string [45]. If the goal is to find an eight-character binary string consisting of all ones, two parents selected from a random population may be: 01001010 and 10101001. An example of a crossover scheme for this system is to randomly choose a position between two bits (e.g. between three and four) and swap the tails of the strings from this point on. After a crossover operation of this nature, the children (the duplicated parents) become 01001001 and 10101010. This technique creates new structures by exchanging pieces of current structures. It also conserves *diversity*, an important quality in evolutionary algorithms. Diversity is a characteristic that estimates how well the optimization landscape is being searched. It is also a measure of how different the structures of a population are. A diverse population will explore more of

the landscape than a population that contains structures with very similar properties, creating a more robust optimization routine. An operation that enhances diversity in the algorithm is *mutation*. As with crossover, mutation is problem-specific and can take on many forms. An example of a mutation operation for the string problem is to randomly select one bit (position five) and flip it. In this case, 01001001 becomes 01000001. Here the mutation was not helpful as the new string is further from the desired string of 11111111.

Once the new structures are created in the system, an assessment of their ability to solve the problem must be made. How close a structure is to the optimum result is termed the *fitness* or the *fitness value* of the structure. The *fitness function* is the function used to calculate the fitness value of a structure. For the string example, a logical fitness function is to calculate the number of bits that are one. Using this function, the two children in this example, 01001001 and 10101010, have fitness values of three and four, respectively, and the second structure is more fit than the first.

Once the new structures have been created and their quality assessed, these new structures reenter the evolving population in the final step of an evolutionary algorithm, *replacement*. This step involves the method of replacing the new structures (the children) back into the population. Unlike populations in the real world, those in the computational realm are usually held at steady state. It is easiest to work with a constant population size. Therefore, as children are created, other structures must be deleted. This is similar to parent selection, except in this case, the structures chosen will be removed instead of duplicated. Since the "best" or most fit structures were chosen to be parents, a logical choice may be to choose

the "worst" or least fit structures to be deleted. These structures are then removed and the newly created children take their place in the population. Although in this case the two most fit structures are chosen to be duplicated and the two least fit structures are deleted, these are not necessarily the best choices. Depending on the problem being solved, there exists the possibility that structures with very low fitness values contain important pieces to the genetic puzzle. Removing these structures from the population results in a loss of diversity and can be counter-productive to the search. More common methods of selection and replacement involve varying degrees of randomness to give structures with low fitness values but potentially useful genetic material a chance to reproduce or at least a chance to survive until the next generation. When a large population is used, this amounts to ensuring that some low fitness structures will survive based on probabilistic statistics. Although in some forms of replacement the parents cannot be replaced, in others the parents must go through the replacement process just like the rest of the population. Similarly, in some cases the children must go through the replacement process as well. In these cases, the introduction of new genetic material into the system is not guaranteed.

The mating scheme described here is one type of evolutionary search called a *steady-state evolutionary algorithm*. The process of selecting parents, mating, and replacing the children is called a *mating event*. A different type of evolutionary algorithm is a *generational evolutionary algorithm*. The term generational is given because of the method in which structures are created and destroyed. In the mating events of a steady-state evolutionary algorithm, only two structures from the entire population are chosen for reproduction, and only two structures are deleted. A generational algorithm chooses enough pairs of parents

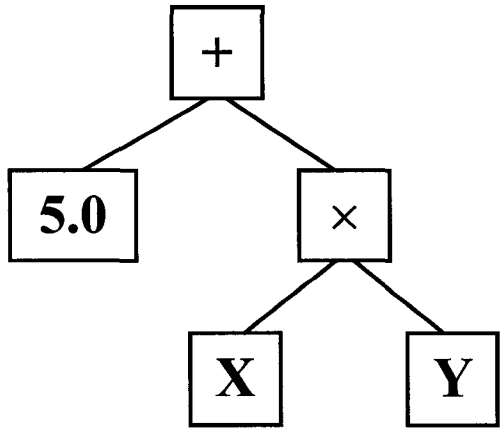
that half of the population will be duplicated and half of the population will be replaced by the newly created children. The process of selecting multiple pairs of parents, mating, and replacing the pairs of children is called a *generation*. In one generation, every structure in the population has either been selected as a parent or involved in the replacement process. Generational algorithms are similar to carrying out enough simultaneous mating events to replace half the population. Carrying out these "mating events" simultaneously restricts the children from one mating event from mating with a deleted structure from a different mating event. For example, consider a population of eight structures in a steady-state evolutionary algorithm: A, B, C, D, E, F, G and H. A and B are chosen as parents, they create two children: A′ and B′, and these children replace structures G and H. The new population will then be A, B, C, D, E, F, A′ and B′. After this mating event, the new structures, A′ and B′, could potentially mate with A, B, C, D, E, or F in the next mating event. In a generational evolutionary algorithm, using the same population, with A and B chosen as one set of parents, C and D chosen as another, and the children A′, B′, C′ and D′ replace E, F, G, and H, the new population would be A, B, C, D, A′, B′, C′, and D′. Unlike in the earlier case, neither child A′, B′, C′, nor D′ would have the opportunity to mate with E or F. Depending on the problem being solved, this can have a significant impact on the performance of the algorithm.

### *2.3.2. Genetic Programming*

Genetic programming is an evolutionary technique that evolves functions [39-44]. These functions are defined by parse trees—the internal nodes of the trees are operations (+, −, ×, ÷, exp, log, etc.) and leaves or terminals of the trees can be constants, variables, or

experimental parameters. An example of a typical genetic programming tree, or parse tree, is illustrated in Fig. 2.1. Also included in the figure is a more economical way of displaying a tree. The Lisp-like notation used in Fig. 2.1 recursively displays the nodes of the tree in parentheses creating a more compact representation. Parse trees can be used as evolutionary algorithms in the same way as other structures, but since they are variably sized, more flexibility is allowed in the structures, making more diverse problems tractable than with typical evolutionary algorithms. The robustness of the technique comes at a cost. The flexibility that is useful for solving a large variety of problems creates a much larger search space than that of a comparable evolutionary algorithm, increasing the difficulty of the search. To search the pertinent areas of the search space, the user can create a problem specific language (set of operations) to be used in the parse trees. This language limits the operators and leaves of the parse trees to only those that are useful for the problem being solved. If the proper language or operation set is used, this technique has the effect of substantially reducing the search space.

As in evolutionary algorithms, the operations of crossover and mutation are useful for exploration of the search space. In genetic programming, the typical crossover operation is called *subtree crossover*. In subtree crossover, a node from each parent structure is selected and the two nodes are swapped along with the nodes and leaves below the selected nodes (the subtrees). Mutation for genetic programming trees consists of randomly selecting a node in the tree, deleting it and its subtree, and replacing it with a new randomly generated subtree. More information on the details of this technique and a variety of different uses can be found in [39-45].




a.) 5.0 + XY

b.) (+ 5.0 (× X Y)

**Figure 2.1. An example of a genetic programming parse tree, the corresponding algebraic representation (a.), and the tree in Lisp-like notation (b.).**

*2.3.3 Evolutionary Computation and Thermal System/Component Design*

There are many applications in which evolutionary computation has been utilized in thermal system/component design calculations. The most popular example is to optimize a design by analyzing a large number of numerical models of potential designs. The robustness of an evolutionary algorithm allows many characteristics of the designs to be optimized at the same time. This aspect makes evolutionary algorithms very useful for fluids problems because the nonlinearities inherent to fluid motion often create a polymodal optimization landscape that is difficult to explore using standard derivative-based techniques.

Trigg et al. [5] have created a fully automatic program that uses a genetic algorithm to optimize turbine blade designs. The application starts from a randomized set of structures, each containing a 17-character genetic code that defines the blade profile. Each structure's profile is then meshed and its performance is evaluated. This process has been used to find two-dimensional blade designs that reduce blade profile loss from 10-20% over unoptimized blade designs. Similarly, Fabbri [46] and Schmit et al. [47] have used genetic algorithms to optimize the geometry associated with thermal systems components. In both cases, an evolutionary technique was used to change the fin profile used in heat removal from computer chips. In the first example, a 45% improvement in the global heat transfer coefficient was obtained compared to the maximum obtainable from a rectangular fin. In the second case, compact high intensity coolers were optimized. The results of the genetic algorithm-based optimization procedure improved both the thermal and hydraulic

performance of the chip over the empirical "best" design previously reported in the literature.

Another approach for optimizing a fluid dynamics problem has been to use a genetic algorithm and the conjugate gradient method. Poloni et al. [48] used a combination of a genetic algorithm and the conjugate gradient method to fine-tune their optimization problem. Derivative-based techniques, such as the conjugate gradient method, perform well on problems with one distinct global optimum but are not well suited for multi-modal problems. Genetic algorithms can find the optimum on single mode problems but at a much slower rate than derivative-based techniques. One of the benefits of a genetic algorithm is its ability to explore more of the search space, even when a local optimum has already been found. The technique used by Poloni et al. uses a multi-objective genetic algorithm to search the design space, a conjugate gradient algorithm to optimize the problem in the near-field, and a neural network for the sensitivity calculations necessary for using these two strategies together. This technique was used to optimize the shape of a fin keel wing of a sail yacht by performing CFD calculations for each structure in the population. The new designs have optimized the criteria given and may soon be tested experimentally to determine their physical performance.

Bryden et al. [49] have used an evolutionary algorithm to optimize the design of the EcoStove, a cook stove designed to reduce the harmful effects of open flame cooking in third world countries. The surface temperature of the original EcoStove design was not spatially uniform, making it poorly suited for its main purpose. An evolutionary algorithm

was created that generates baffles in the flowfield below the surface of the stove and then calls a CFD solver to analyze the effectiveness of the new design. The optimized designs have been shown to have a usable surface area of approximately 80-90% compared to the 30% usable surface area of the baffle-free original design.

This work has been advanced recently to reduce the time costs associated with any evolutionary algorithm-optimized CFD problem. Evolutionary algorithms work by analyzing many designs and comparing the results. For the cook stove research, one trial called for 4000 stove designs to be analyzed through CFD. Even with a grid that has been coarsened as much as possible, an individual CFD call lasted between two and three minutes to reach full convergence. To most CFD experts this is very fast, but when used in an evolutionary algorithm in which 4000 designs are analyzed, that adds up to eight days of compute time to obtain a single optimized design. Researchers have found that by training a neural network to determine when to cut out of a CFD run, drastic time savings can be achieved. This work has found that designs similar in quality to the original research have been found in 1/6 to 1/8 of the time [50].

Milano et al. [51] used an evolutionary algorithm to optimize the placements of actuators that are used to control surface-mounted vortex generators that make corrections for the unsteadiness of the flows involved. The goal of the research of Milano et al. was to create significant performance improvements by controlling the flow past an object using active devices placed on the objects. The application created was tested on the benchmark problem of flow past a circular cylinder. Using the optimized active controlled cylinder, the

researchers were able to achieve drag reductions of up to 60%, outperforming the best methods previously reported in the fluid dynamics literature for this problem. This application illustrates how a genetic algorithm can optimize the placement of the actuators, a complex problem that would be difficult for many other optimization techniques.

Instead of using CFD as an analysis tool and a genetic algorithm as the optimizer, Fan et al. [52] used a genetic algorithm to mimic the analysis capabilities of CFD. This system uses the natural correlation between fluxes and forces to create organisms (fluxes) from chromosomes (forces). In this situation, the fitness function for the organisms is how well the fluxes satisfy physical laws of conservation. By using this fitness function, the organisms evolve to the correct solution of the flowfield. The method has been verified on a simple CFD problem and encouraging results were found.

In each of the papers described, similar tools are used for different goals. Computational fluid dynamics and evolutionary computation techniques were combined in each of the examples above, but in each case, they were used differently. Like those above, this paper will discuss how an evolutionary algorithm can be used to call different solution methods. However, where these examples aim to optimize a design, the goal in the research presented in this thesis is to optimize the multi-solver system itself to allow for quicker recalculation of previous designs.

# Chapter 3.

# AMoEBA–The Adaptive Modeling by Evolving Blocks Algorithm

The goal of this research is to create a tool for engineering problem solving that segregates a domain into regions that can be assigned solvers or modelers. The combined network of solvers or modelers should accurately model or rapidly solve the original domain and provide a scheme for recalculating the domain after changes are made to the model. These changes may be geometry changes, changes to the physical properties of the system, or changes to the boundary conditions. The segregation scheme may result in some of the regions going unchanged for some parameter changes of the model, reducing the amount of computation required. To meet these goals as an effective technique for engineering problem solving, a number of criteria should be met.

- The algorithm should be robust and be able to be easily modified to solve a number of different engineering problems.
- The criteria used to automatically create the segregation scheme should be defined by the user so that a variety of engineering problems are not only tractable but can also be solved in different ways at the discretion of the user.
- The algorithm should not be limited to any degree or type of problem. AMoEBA should be able to solve one-dimensional, two-dimensional, or multi-dimensional problems as well as problems that are non-dimensional, such as a power plant, a car engine, an airplane, etc., in which the segregations defined by AMoEBA are based

on locations in the system and may not necessarily be defined by a geometric plane or line.

These criteria ensure that a robust and versatile engineering tool that provides a segregation framework for data analysis, domain manipulation, and recalculation for the design of engineering systems is developed.

This chapter first describes an example of a manually decomposed and manually placed multi-solver. A description of the AMoEBA algorithm in detail follows this discussion. The majority of this chapter is spent on three different engineering applications of AMoEBA. Demonstrations of these applications include:

- A polynomial implementation that models two different engineering data sets. This section will allow the reader to become familiar with the terms and the actions associated with this algorithm while presenting the capabilities of the technique at an introductory level.
- An inverse heat conduction problem solving method. This demonstration is more complex than the first and also applies the algorithm in a new direction from both the data segregation problem and the polynomial implementation.
- The tightly coupled, nonlinear data sets of computational fluid dynamics for single-phase flow in an elbow section of pipe. This implementation develops the framework that will enable the user to make changes to the geometry, to alter the boundary conditions, or to place obstructions in the flowfield and witness the impacts that these changes have on the model.

In each of these demonstrations, AMoEBA automatically decomposes the domains and places solvers in the resulting segregations. To help understand the automated technique, the following section will demonstrate how a multi-solver can be created manually, using previous knowledge of the flowfield.

## 3.1. Example of Manual Geometric Decomposition and Solver Assignment

An example problem that can be divided into different solution methods is the numerical calculation of laminar flow in the entrance region of a circular pipe. As with flat plate flow, a boundary layer begins to develop at the entrance of the pipe (Fig. 3.1). Outside of this boundary layer, the flow behaves as an inviscid fluid. However, unlike the flat plate, this outer flow is accelerated as the boundary layer begins to fill the pipe, thinning the boundary layer. Eventually the boundary layer grows until the entire pipe is filled with the viscous boundary layer. At this stage the flow is not quite the parabolic Poiseuille flow observed in fully-developed flow. A short region known as the filled region exists between the point when the boundary layer fills the pipe and the fully-developed stage [53]. Three solvers are required to calculate the flow from the entrance until it is fully-developed. The boundary layer equations are used for the flow inside of the boundary layer with an inviscid solver for the outer flow. Once the flow reaches the filled stage, the full Navier-Stokes equations must be used to resolve the subtle changes in the flow in this region. In the fully-developed stage, a full solver is unnecessary as the flow continues in the parabolic Poiseuille flow until a turn or another outside influence disrupts it. Although the goal of this research is to be able to automatically segregate CFD domains into regions that require the same solution techniques, it is useful to divide the laminar entrance flow into its proper regions

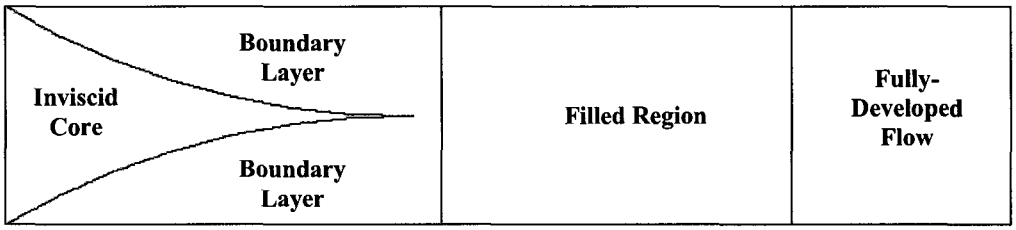



**Fig. 3.1. The four different domains of the development of the flow in the entrance region of a pipe.**

interactively to illustrate the effectiveness of the multi-solver strategy. This is also a useful exercise to gain experience and understanding of this type of solution procedure as well as a better grasp on the coupling functions between the different solvers. The following method describes how the entrance flow problem can be numerically calculated.

### *3.1.1. Navier-Stokes Equations*

Of the solvers necessary to compute the laminar entrance flow, the incompressible, steady-state Navier-Stokes equations are the most complicated and the most robust. These are a slightly reduced form of the full Navier-Stokes equations, and in two dimensions in cylindrical coordinates they are:

continuity:
$$\frac{\partial u}{\partial x}+\frac{1}{r}\frac{\partial (rv)}{\partial r}=0 \qquad [3.1]$$

x-momentum:
$$u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial r}=-\frac{1}{\rho}\frac{\partial P}{\partial x}+\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)\right) \qquad [3.2]$$

r-momentum:
$$u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-\frac{1}{\rho}\frac{\partial P}{\partial y}+\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial v}{\partial y}\right)-\frac{v}{r^2}\right) \qquad [3.3]$$

### *3.1.2. Boundary Layer Equations*

Although it is possible to compute the entire entrance flow with the Navier-Stokes equations described, it is useful to reduce these equations further by eliminating terms that become insignificant in specific regions. Due to the no-slip condition on the pipe wall and the assumption that the flow is overwhelmingly unidirectional, the shear forces due to viscous effects have a very small effect on the flow in the $x$-direction inside the boundary

region. This can be shown by performing an order of magnitude analysis on the Navier-Stokes *r*-momentum equation. This type of analysis shows that every term other than the pressure term is very small in comparison and therefore can be considered insignificant to the development of the flow. Making this approximation reduces the equations to the boundary layer equations shown here:

continuity: $$\frac{\partial u}{\partial x}+\frac{1}{r}\frac{\partial(rv)}{\partial r}=0 \qquad [3.4]$$

x-momentum: $$u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial r}=U_e\frac{\partial U_e}{\partial x}+\nu\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)\right) \qquad [3.5]$$

r-momentum: $$\frac{\partial P}{\partial r}=0 \qquad [3.6]$$

*3.1.3. Inviscid Flow Equations*

In the entrance region outside of the boundary layer, viscous effects are negligible, leading to a flattened velocity profile. In this region, the boundary layer equations can be further reduced to the inviscid flow equations by neglecting terms associated with the viscosity of the fluid. These become:

continuity: $$\frac{\partial u}{\partial x}+\frac{1}{r}\frac{\partial(rv)}{\partial r}=0 \qquad [3.7]$$

x-momentum: $$u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial r}=U_e\frac{\partial U_e}{\partial x} \qquad [3.8]$$

r-momentum: $$\frac{\partial P}{\partial r}=0 \qquad [3.9]$$

A further approximation can also be made in the outer flow with only a minor reduction in accuracy. Instead of solving the outer flow using the inviscid flow equations, a mass flow analysis can be used to determine the velocity of the fluid at the centerline of the pipe. The flattened profile of the inviscid region allows this generalization to be made with only very minor effects on the accuracy. This is the technique that is used for the solution presented here. To calculate the outer flow in this way, an iterative procedure is used in which an assumption is first made on the outer flow. Using the predicted outer flow, the boundary layer is calculated, and from this new boundary information a better estimate on the outer flow is found. Eventually this converges to give an accurate representation of the centerline velocity.

### *3.1.4. Results*

In practice, it is useful to solve the above equation sets on a transformed grid to ensure the proper grid refinement in the boundary layer. If there are not enough nodes in the boundary layer, errors will occur early and propagate through the solution until it eventually diverges. A Levy-Lees transformation has been used in this case to transform the $x$ and $r$ coordinates to the variables $\xi$ and $\eta$ by the following equations:

$$\xi = x \qquad [3.10]$$

$$\eta = \eta_{\max} - (R_0 - r)\sqrt{\left(\frac{\mathrm{Re}}{x}\right)} \qquad [3.11]$$

where $\eta$ goes from 0 to $\eta_{max}$ and $\eta_{max}$ = 6.5. This transformed grid places points in the boundary layer that develops on the pipe wall, but initially does not extend completely to the centerline of the pipe (Fig. 3.2).

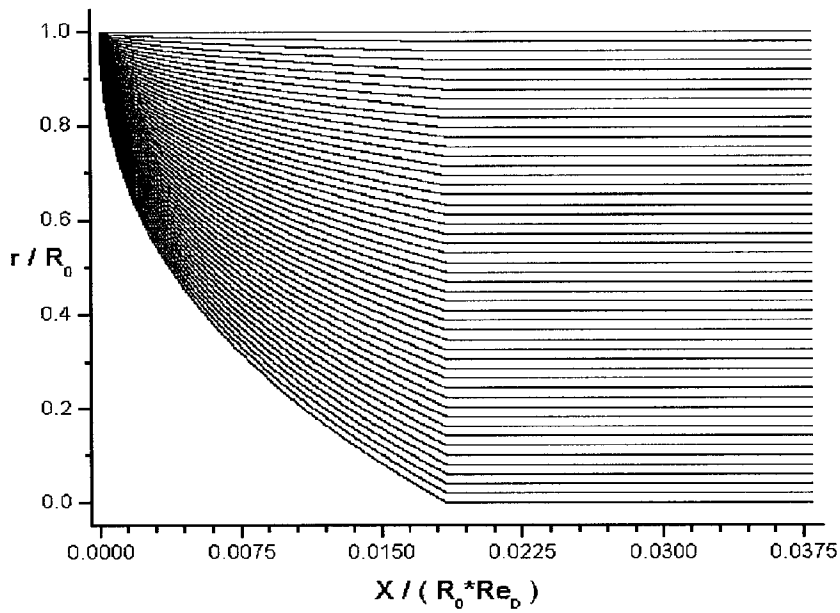



**Fig. 3.2. Transformed grid for the laminar pipe flow problem. Notice how the grid does not extend to the centerline ($r/R_0$) before approximately 0.018 on the x-axis.**

The boundary layer equations are discretized such that the radial direction becomes fully implicit. Using the following boundary conditions, the solution to the boundary layer equations is marched in the axial ($x$) direction as the centerline velocity is found:

$$U(x=0,r)=u_0 \qquad [3.12]$$

$$V(x=0,r)=0 \qquad [3.13]$$

$$U(x,r=R_0)=0 \qquad [3.14]$$

One more boundary condition is necessary to solve the equation set presented. For flat plate flow, the final boundary condition is the wall condition of the vertical velocity component $V(x,y=0)=0$. Based on the discretization of the equation set for cylindrical coordinates, the solver becomes unstable when marched forward in the axial direction and backward in the radial direction from the wall to the centerline. One solution to this is to recognize that when the grid fills the entire pipe, the radial velocity component does not change with respect to the radial direction at the centerline. This provides the boundary condition for the inner edge after the boundary layer grid fills the pipe. For the region before the grid fills the pipe, a shooting algorithm is used to determine the radial velocity component at the edge of the grid.

Several options are available for the coupling function between the inviscid flow region and the boundary layer region. These are:

Option 1. Guess a radial velocity component of $V = 0$ at the boundary between the solvers in the early portion of the grid and make the necessary corrections until the

proper boundary is found. This results in premature convergence of the radial velocity profile.

Option 2. Use the radial velocity component at the boundary from the previous $x$ step as an initial guess. This disrupts the stability of the solution and causes poor performance in both the radial velocity component and the axial velocity component calculations.

Option 3. Underrelax the radial velocity from the previous $x$ step. This underrelaxation has a significant impact on the performance of the model.

The boundary layer equations were solved based on Option 3 as the initial guess for the outer flow. This centerline velocity value was then updated based on a mass flow analysis of the boundary layer solution. At each axial step, this procedure was iterated until there was very little change in the centerline velocity.

Results of this computational technique based on the geometric decomposition and solver assignment of the entrance flow in a pipe can be seen in Fig. 3.3 compared with experimental results from [54] and numerical data from [53] to illustrate the accuracy of previously published results. As can be seen in Fig. 3.3, the multi-solver technique produced slightly better results than the published numerical data. This technique also produced a similarly shaped profile when compared to the numerical data with an overall shift in the transition point to the filled region ($x^* = 0.036$ with $U_{cl} = 1.774*U_o$ for [53] versus $x^* = 0.032$ with $U_{cl} = 1.786*U_o$ for the multi-solver results – $x* = x/(R_o \cdot Re_d)$.

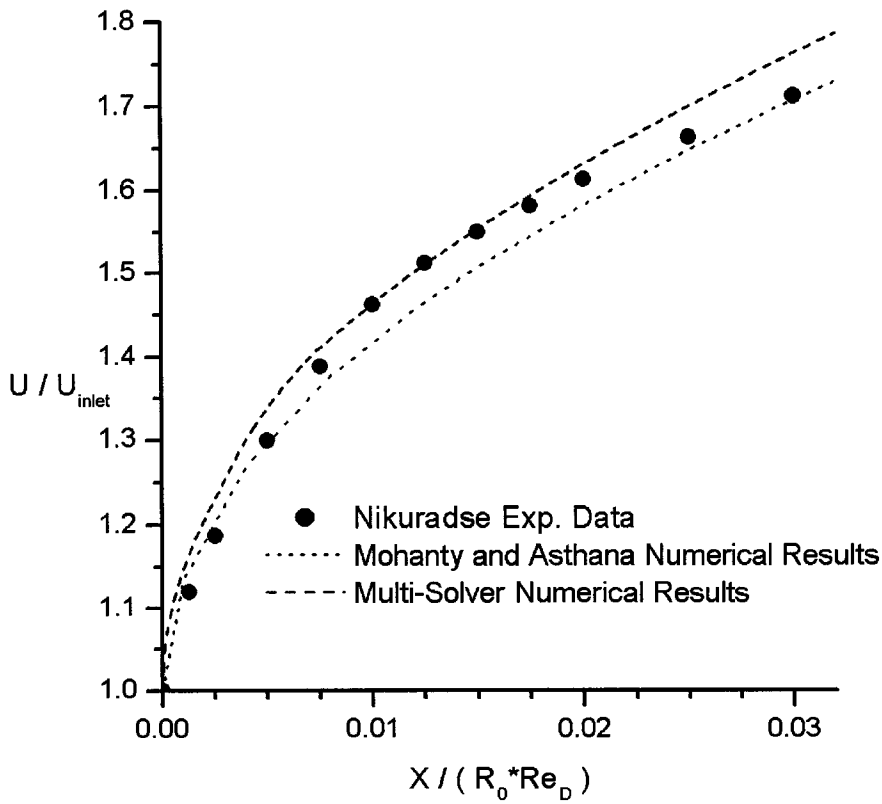



**Fig. 3.3. Comparison among Nikuradse's experimental data [54], Mohanty and Asthana's numerical results [53], and the numerical results of the multi-solver method.**

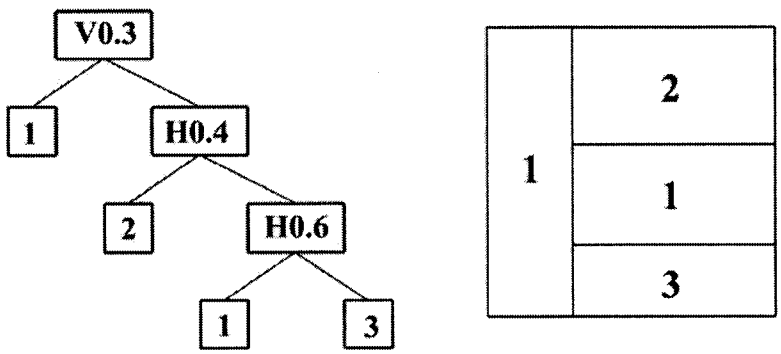
## 3.2. Description of the AMoEBA Algorithm

The Adaptive Modeling by Evolving Blocks Algorithm, AMoEBA, is a versatile evolutionary technique designed to automatically decompose the domain of an engineering problem into coherent blocks based on parameters established by the user. AMoEBA utilizes a genetic programming-generated tree to segregate a computational domain into a number of sub-regions. An evolutionary algorithm then optimizes a population of these structures based on criteria determined by the intended application. Potential applications of this algorithm include inverse heat transfer problems, computational fluid dynamics, energy systems modeling, and a variety of other engineering problems, as well as applications that are not engineering related, e. g. room layout.

In AMoEBA, the structures of the evolutionary system are binary trees that define a segregation scheme and solver/descriptor for a given data set. The trees, or segregation trees, are similar to the function encoding trees of genetic programming [39-44]. Like the trees of genetic programming, each segregation tree consists of nodes and leaves, the leaves being assignments of data to regions. At each internal node of the tree, there exists an integer value defining the type of segregation to be made. For example, a zero signifies that a vertical segregation will be made, and a one signifies a horizontal segregation for a two-dimensional system. For higher dimensions, the number of directions of segregations available is increased. The direct application to non-dimensional problems is not explicitly addressed in this research, but adaptation of AMoEBA to these types of problems is discussed in the "Future Work" section of Chapter 4. Each internal node also contains a floating-point value that defines the location of the segregation. The floating-point value

ranges between zero and one to define a segregation as a percentage of the entire domain for the top node or a percentage of the sub-region for internal nodes. A node may assign the left and/or right region to a terminal leaf value (integer) or to another node. If another node is assigned, the data set is segregated further. Fig. 3.4 shows an example of the data segregation encoding structure as well as the compact Lisp-like notation of the tree. In this example, the leaves define solvers/descriptors to be placed in the segregated regions. The top node defines a vertical partition that creates a 30:70 split of the domain starting from the left. Solver/Descriptor 1 is placed to the left of this partition (in the 30% region) and another node describes the right side of the partition. On the right, a horizontal partition creates a 60:40 split of the right region and Solver/Descriptor 2 is placed in the newly formed top region (the 40% region). Another horizontal partition divides the bottom region into Solvers/Descriptors 3 and 1 at a 40:60 ratio; Solver/Descriptor 3 is placed in the bottom region (the 40% region) and Solver/Descriptor 1 is placed in the top region (the 60% region). Note that at each level the partitioning scheme is local; to find the actual position of this final partition, the tree must be traversed from the leaf to the top node. In this example, the final boundary is located at the line $y = (0.4 \times 0.6) = 0.24$ measured from the bottom of the domain. The solvers/descriptors in the leaves of the trees can describe solvers to be used in the case of multi-solver decomposition, materials in the case of an inverse engineering problem, or any other spatially distributed quantity of interest.

Although the AMoEBA algorithm is more complex than the general evolutionary algorithm discussed in Chapter 2, the basic principles are still incorporated. AMoEBA uses a generational evolutionary algorithm as the optimization technique and has the same flow-




(V0.3 1 (H0.4 2 (H0.6 1 3)))

**Fig. 3.4. A data segregation tree, the resulting solver/descriptor configuration, and the Lisp-like representation of the tree. This example shows a two-dimensional tree making vertical (V) and horizontal (H) segregations in the domain and placing Solver/Descriptors 1, 2, and 3 in the resulting partitions.**

of-control as a typical generational evolutionary algorithm (Fig. 3.5). Due to the flexibility of evolutionary algorithms, many details associated with the evolutionary process differ from problem to problem. The first step of AMoEBA is to randomly generate the initial population. Depending on the application of AMoEBA, the number of structures in the initial population varies. Once the initial structures are created, their fitness values are evaluated. Again, due to the diverse types of problems that AMoEBA is used to solve, the fitness function may be significantly different in each case, so these will be discussed in detail for each problem. After evaluating the fitness values, parents are selected. In AMoEBA, the selection of parents and the replacement of the children are controlled by placing the structures into groups, called *tournaments*. In most applications of AMoEBA, a tournament of size four is used. In each tournament, the two most fit structures become the parents and the two least fit structures are deleted; i. e., they are replaced by the children. Because the tournament groups are created randomly, the most fit half of the population is not guaranteed to be selected as parents and the least fit half has a chance to survive. As with the generic evolutionary algorithm, the children are created by duplicating each parent. Before they are returned to the population, the operations of crossover and mutation are applied. Because of the tree configuration of these structures, *subtree crossover* is used [39-44]. In the first step, a node from each of the children is randomly selected. The selected nodes and the subtrees below these nodes are then swapped, creating two completely new trees (Fig. 3.6). This is a very disruptive form of crossover due to the structure naturally associated with the trees, and it has even been categorized as a macro-mutation based on this disruptive nature [55]. Due to the potentially drastic effects of this crossover operation, in most forms of AMoEBA there is only a 50% probability that crossover will occur for

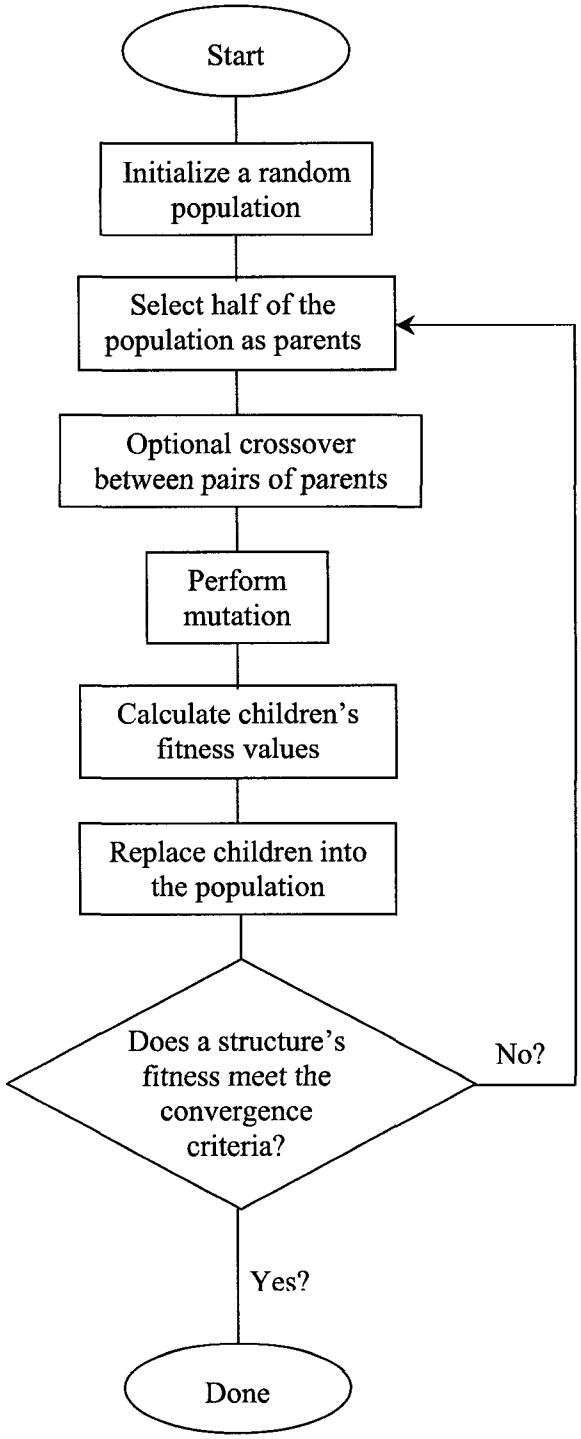



**Figure 3.5. The flow of control of a typical generational evolutionary algorithm.**

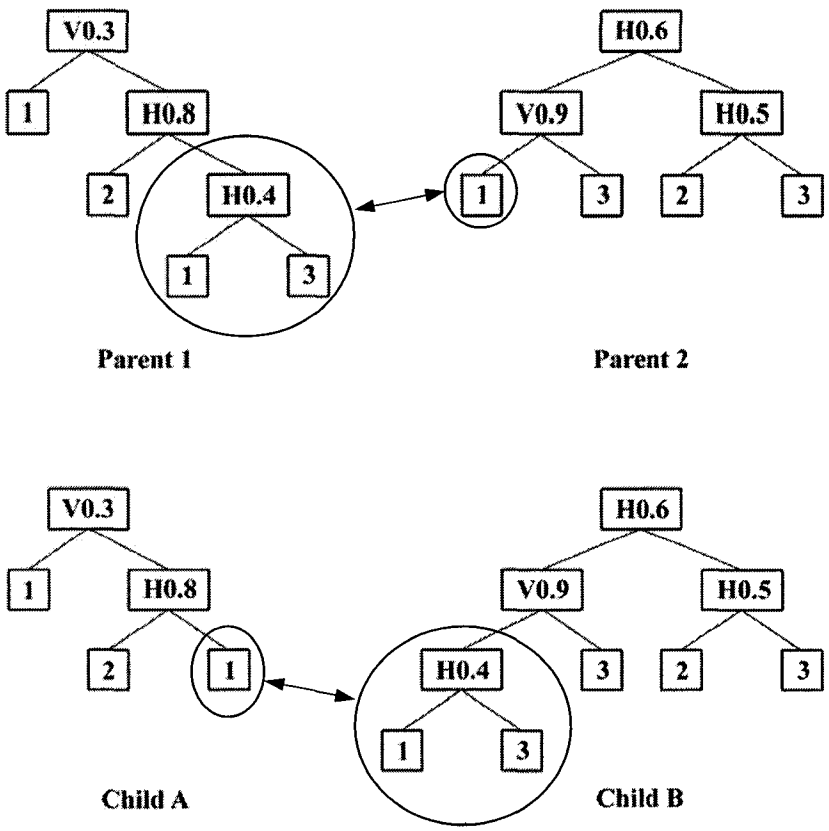



**Fig. 3.6. An example of crossover. A node or leaf from each parent is selected and swapped, resulting in two new children.**

each pair of children. Because there is no discrimination in this crossover operation between upper nodes in the tree and nodes that are closer to the leaves, there is a reasonable chance for nodes with a large number of subtrees to be swapped with nodes that have only a few subtrees. This creates one tree that is smaller than average, but more importantly, the other tree is likely to be larger than average. With no limit placed on the size, these trees can grow geometrically, creating finer and finer divisions in the data set until eventually the divisions are so fine that they are finer than the size of the cells used to define the data set. To combat this increase in tree size, a *chopping* operation is introduced. After crossover, if a tree contains more than a set number of nodes, then a subtree of the root node is randomly selected and promoted to be the new root (Fig. 3.7). This chopping operation prevents geometric increase of the tree size and is also used when the original population is created to ensure that all trees have fewer nodes than a certain limit. A *pruning* operation is used to enforce a minimum region width for the trees as well. The prune function checks each node for the widths of the right and left regions. If these regions are smaller than a minimum value and if the offending region is assigned another node, the node is replaced by a random leaf. This function helps to limit the trees from creating smaller and smaller subdivisions in the domain.

The different types of mutation for the segregation trees are independent of the application. However, the parameters that determine how often each type is used are dependent on the problem being solved. There are four different types of mutation for the segregation trees. These are:

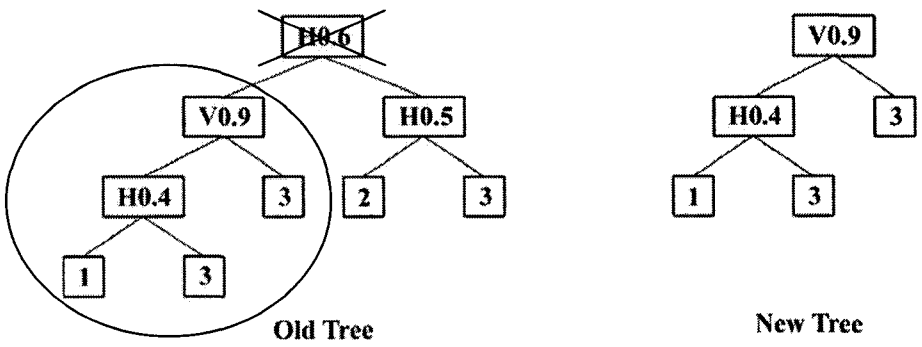



**Fig. 3.7. An example of the chopping function. In this case, a node from the old tree is randomly selected and promoted to be the new tree.**

- *Random subtree mutation*—This operation randomly selects a node in the tree, removes it, and then replaces that node with a randomly generated subtree of the same size. This, like subtree crossover, is very disruptive to the trees and therefore is used sparingly.
- *Flip mutation*—This operation is also potentially disruptive for the segregation trees. In flip mutation, the integer value that defines the direction of the segregation line is flipped to the opposite direction. For example, if a node calls for a horizontal line, flip mutation would use the same floating-point value for the line but flip it to a vertical line instead. This form of mutation is only used in cases that have two or more dimensions.
- *Boundary value mutation*—This operation is less disruptive than the previous two operations and is the most essential of the mutation operations towards the end of the optimization. Boundary value mutation randomly chooses a floating-point value between -0.05 and 0.05 and adds it to the segregation line definition, shifting the boundary by a random fraction. This operation is reflective if the shift causes a boundary to extend outside of the domain. Often, near the end of the evolutionary search, the most fit structure is very close to the optimum solution. Many times a slight difference in one of the segregation locations is all that prevents a structure's fitness value from meeting the criteria. Subtle shifts in the boundaries of the segregation scheme are useful in obtaining a structure that has a fitness value that meets the stopping criteria.
- *Leaf mutation*—This operation is also less disruptive than flip mutation or random subtree mutation. Leaf mutation changes the value, or the assignment given by the

leaf. This mutation operation is the only form of mutation that is somewhat problem- as opposed to dimension-dependent since, even though the method in which the leaves are assigned is the same, the leaf assignments are different for each application of the AMoEBA modeling routine. In the three applications studied in detail in this research, the leaf assignments are different solvers or solution methods that, when combined, calculate an entirely new data set that can be compared to the original data set.

It is common in evolutionary algorithms to incorporate mutation operations that can make large jumps throughout the optimization landscape as well as operations that can "dial in" on the solution once a structure gets close to the optimum. In AMoEBA, this school of thought has been adapted. Random subtree mutation and flip mutation have the potential of being very chaotic because they make significant changes to the structures. Boundary value mutation introduces more subtle changes that can slowly step the population in the right direction. Leaf mutation falls somewhere in between. This operation only changes one solver/descriptor of the tree so it does not cause the disruptions that the first two operations can, but this is a more significant change to the structure than boundary value mutation.

### *3.2.1. Demonstration of AMoEBA*

The goal in this application of AMoEBA is to segment a numerical data set (Fig. 3.8) into regions that have more in common with themselves than with any other regions in the domain. For example, consider a data set consisting of mostly one value, "*s*," with a handful of significantly larger values, "*t*," scattered throughout. If this data set were to be segregated using this implementation of AMoEBA, it would be divided into regions

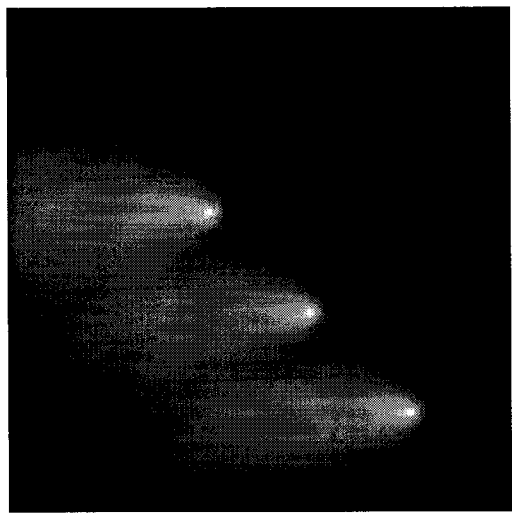


**Fig. 3.8. The diffusion data set used in the first implementation of the AMoEBA algorithm.**

containing the value "*s*" and regions containing the value "*t*." Although this implementation may draw parallels to pattern recognition, that is not necessarily the intended purpose for the application. The goal of pattern recognition is to locate regions that can be classified into predetermined categories or feature sets. With AMoEBA, the data set is segregated into regions, but these regions are not restricted to any categories other than those created by AMoEBA. There are instances where an AMoEBA segregation may resemble pattern recognition schemes and vice versa, but these instances are unintentional. AMoEBA may also be mistaken for a clustering algorithm. In fact, the technique is similar to clustering but, if anything, it is a significantly more restricted form of clustering. The goal of clustering is to be able to sort a variable into clusters of data that are similar. Different values that lie between like data points are placed in different clusters. The dependent variable in the data set used in the AMoEBA algorithm could be clustered using this technique, and the three peaks of the data set in Fig. 3.8 would be placed in a single cluster. The difference is that in AMoEBA, the geometry is also considered. Because valleys of lower values separate these peaks in the geometry, placing them in the same region would not be useful for the AMoEBA segregation schemes. This geometrical restriction of the data set is not possible and not the intended purpose of clustering algorithms.

In each implementation of the AMoEBA algorithm, various parameters (mutation rates, population size, tournament size, etc.) are adjusted to improve the performance of the algorithm. The fitness function is also different from application to application. In this utilization of AMoEBA, the fitness function is a calculation based on the values of the data set inside a given region and those values outside of that region. The data set itself is a

200×200 grid of values and is normalized to extend between zero and one in the $x$ and $y$ directions. The values throughout the domain are also normalized between zero and one and represent an artificially created, directionally biased diffusion data set. To calculate the fitness of a structure, $F$, 5000 pairs of sample points are selected. $B$ is then calculated to be the sum of the squared differences between the sample points when the two points fall in different regions. $W$ is calculated similarly for when the points fall in the same region. The fitness, $f(F)$, of the structure, $F$, then, is:

$$f(F) = \frac{1+W}{1+B} \qquad [3.15]$$

This fitness function has the property that, as the difference between points in different regions increases and the difference between points in the same region decreases, the fitness of the structure goes toward zero. AMoEBA was used to solve the data segregation problem starting from 30 different random populations. Each trial was run until a structure in the population had a fitness value of less than 0.075. This relatively low value was chosen based on preliminary studies of the algorithm. Fig. 3.9 presents two different solutions to the data segregation problem. As Fig. 3.9 shows, AMoEBA is able to decompose the data set into multiple regions. The underlying data set can also be seen in Fig. 3.9, which illustrates the effectiveness of the AMoEBA partitioning scheme. Further research into the capabilities of this implementation of AMoEBA as well as new research on the effects of imposing a graph structure on the mating scheme can be found in [56, 57].

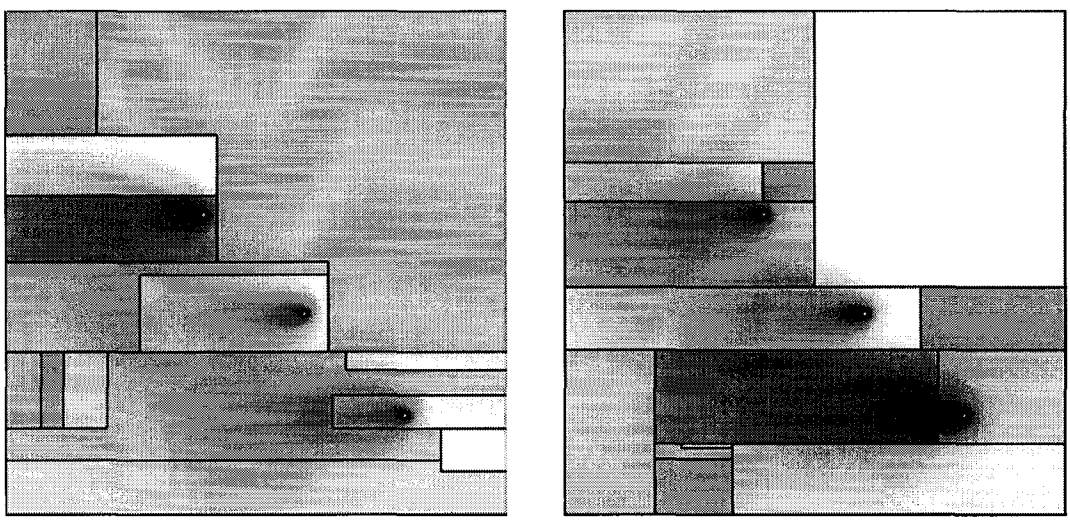


**Fig. 3.9. Two examples of AMoEBA solutions to the data segregation problem of a diffusion data set.**

## 3.3. AMoEBA: Approximating Algebraic Expressions

As demonstrated, AMoEBA can segregate a numerical data set based on criteria chosen by the user. In this section, the AMoEBA algorithm is applied to an engineering problem by modeling a data set using approximating algebraic expressions that are co-evolved with the segregation trees [58]. The trees of AMoEBA segregate the engineering data set into different sub-domains, and instead of comparing the values in these segregated regions to each other as in the previous application, new values are generated using polynomials as the algebraic expression. The polynomials assigned to each region are co-evolved with the segregation trees, creating a codependency between the trees and the polynomials. By creating polynomial models of portions of the data, the data can be described in a more compact method and can be recreated more easily. The long-term goal for the research described in this dissertation is to be able to quickly recalculate engineering data sets after changes are made to existing geometries or boundary conditions. This implementation of AMoEBA presents an initial methodology for achieving this goal.

### *3.3.1. AMoEBA Description*

In each application described, AMoEBA is modified to accommodate the different structures and variables of the particular problem being solved. For this implementation, AMoEBA is converted into a co-evolutionary system using two populations, one of which is a population of the binary trees that segregate the data. These trees are as described previously. The leaves of the segregation trees are integers, and, for this application, they correspond to approximators that model the data in the subdivisions defined by the tree. These approximators are the second population in the system. In this case they are

polynomials, but in theory they could be coded for performing any applicable computation. The codependency of the two populations is determined by the fitness functions of the two different types of structures. The fitness function of the approximators is determined by how often each approximator is used by the trees in the current population. The codependency is further enhanced by the fitness function of the segregation trees. The fitness value of a particular tree is dependent on how well the tree/approximator team models the original data set. The goal of the tree/approximator team is to decrease the squared error between the data set created by the team and the original data set. To meet this goal, the correct segregation scheme, the proper approximator placements in this segregation scheme, and suitable coefficients for these approximators must be found by one of the teams in the population.

#### 3.3.1.1. Fitness Function Details of the Data Segregation Trees

The domain in this implementation is similar to the domain used in the previous case. A 200×200 grid is used, and it is normalized to the range [0, 1] in both directions. However, unlike the previous demonstration of AMoEBA, the dependent variable is not normalized for this problem. Each tree defines a segregation scheme and the placement of the approximators. To compute the fitness value of a tree, sample points are chosen randomly throughout the domain. Each sample point lies in a region corresponding to a leaf of the segregation tree. Each leaf is assigned a polynomial approximator, and the result of evaluating the polynomial at the sample point coordinates is compared to the exact data set at these coordinates. The sum of the squared error between the sample point results and the exact data set is the fitness value for the data segregation tree. A new random set of sample

points is created for each generation, and each tree is tested on the same set of randomly generated sample points. The number of sample points was originally set to 5000 points, or 1/8 of the total number of points in the data set (200×200 = 40,000). In later generations, the average fitness of the population is good, and determining which structures are better than others becomes more and more difficult. Structures that "get lucky" and have good solutions in the regions in which the sample points fall, but do not perform well in regions that go unchecked, tend to reproduce. Although these structures often die out quickly, they tend to disrupt the efficiency of the algorithm. To improve on the algorithm, a study was performed on the number of sample points initially used in the fitness function and on how the sampling rate is increased.

3.3.1.2. The Approximator Evolutionary Algorithm

A long-term goal for AMoEBA is to allow the data segregation trees to point to a variety of modeling techniques such as similarity techniques, interpolation techniques, finite element analysis, and CFD computations. In this implementation, polynomials with variable degrees have been used as the approximators. These polynomials are limited to ten terms and have random degrees (between constant and ninth order) and random coefficients. Both the degree and coefficients are evolved parameters. The random coefficients are floating point values and range between –800 and 800 or –5.0 and 5.0 for the two different problems solved. Forty polynomials are used in the approximator population, and these are evolved after every tenth tree generation. It was determined in preliminary studies that the algorithm ran best when trees were allowed to evolve based on more stable, or static, approximators.

By evolving the approximators at a slower rate than the trees, the trees are able to evolve around the polynomials and the time to solution is reduced.

The evolutionary process for the approximators is different from the evolutionary process for the trees. The variables associated with the polynomials are the degree of the polynomial and the coefficients of each term. The strong dependence of the polynomials on the first few coefficients makes them unsuitable for the more traditional mating scheme demonstrated in the binary string example of choosing two parents, duplicating them, and swapping portions of the children with each other. Instead, an asexual system was created in which parents are selected based on their fitness values and are simply duplicated with no crossover between any of the parent structures. The 25% of the population with the best fitness are selected as parents, and mutated copies of these parents replace the lowest 25%. Because there is no crossover, new structures are created only through mutation. There are three types of approximator mutation, each having a 25% probability of occurrence together with a 25% chance of no mutation. In the first mutation operation, a random coefficient of the approximator is selected and a random floating-point value in the range between –150.0 and 150.0 or –0.05 and 0.05 for the two different problems is added to the coefficient. This type of mutation can have a significant effect if one of the lower ordered terms is selected or an almost negligible effect if a higher order term is selected for mutation. A different type of mutation selects a random coefficient and multiplies it by negative one. This mutation is similar to the previous coefficient mutation in that it can have a large impact on the polynomial or only slightly affect the approximator, depending on the order of the coefficient selected. The other mutation operation adjusts the degree of the polynomial.

This mutation operation has an equal probability of increasing the degree of the polynomial by one, decreasing it by one, or leaving it unmutated. If the degree is increased by one, a random coefficient is generated for that term in the polynomial. Increasing the degree is disallowed if the degree is at the maximum limit and the mutation then has an even probability of decreasing the degree or leaving it unchanged. Similarly, if the polynomial is a constant, the mutation has an equal chance of leaving it a constant or increasing it to a linear function.

During preliminary testing, it was learned that placing the mutated children into the lowest 25% would doom them to be used as infrequently as the approximators they replace. Since the approximator fitness value is calculated based on the number of times they are used by a tree, the mutated copies of the parent approximators would be placed in the fewest number of leaf positions, making it difficult for these approximators to obtain higher fitness values. On the contrary, placing the new polynomials into the parent positions would be very disruptive to the tree/approximator teams and could slow the progress of the algorithm. Instead, not only are the children placed in the deleted approximator positions, but a small percentage of them also take their parents' places in some tree leaves. By finding the leaves pointing to the parents and randomly replacing them with their children (at an average occurrence of one in four), the children are given a fair chance without doing excessive damage to the parents or the rest of the population.

*3.3.2. Problem Description*

Two different engineering data sets are used to demonstrate the capabilities of this implementation of AMoEBA. The first of these focuses on one-dimensional heat transfer through a wire. The system is at steady state and conduction is the only mode of heat transfer. The boundary conditions used for the numerical calculation of this data set are 650 K at one end and 400 K at the opposite end. To more easily adapt to the AMoEBA algorithm, the domain consists of a 200×200 grid with both dimensions extending from zero to one. Because this is a one-dimensional system, the solution temperatures are the same in the *y*-direction. The analytical steady-state temperature profile for this system is:

$$T = 650 - 250x \qquad [3.16]$$

where $T$ is the temperature of the wire and $x$ is the linear distance from the 650 K end of the wire and is normalized to the range [0, 1]. Fig. 3.10 illustrates the temperature profile for this implementation of AMoEBA. In this implementation, the coefficients of the initial approximators are chosen randomly from the range –800 to 800 with the random coefficient mutation range of –150 to 150. The goal of this problem is to demonstrate not only AMoEBA's effectiveness at decomposing the domain, but also the polynomial solvers' ability to evolve to a polynomial that is similar to the analytical solution. Because the exact analytical solution can be modeled by a single polynomial, the expectation for the AMoEBA solution is to find a single segregation with a profile similar to the analytical solution on both sides. Each AMoEBA tree is required to contain at least one node, i.e. one division, which is why the expected solution contains two different subregions even though the entire domain can be modeled by a single polynomial.

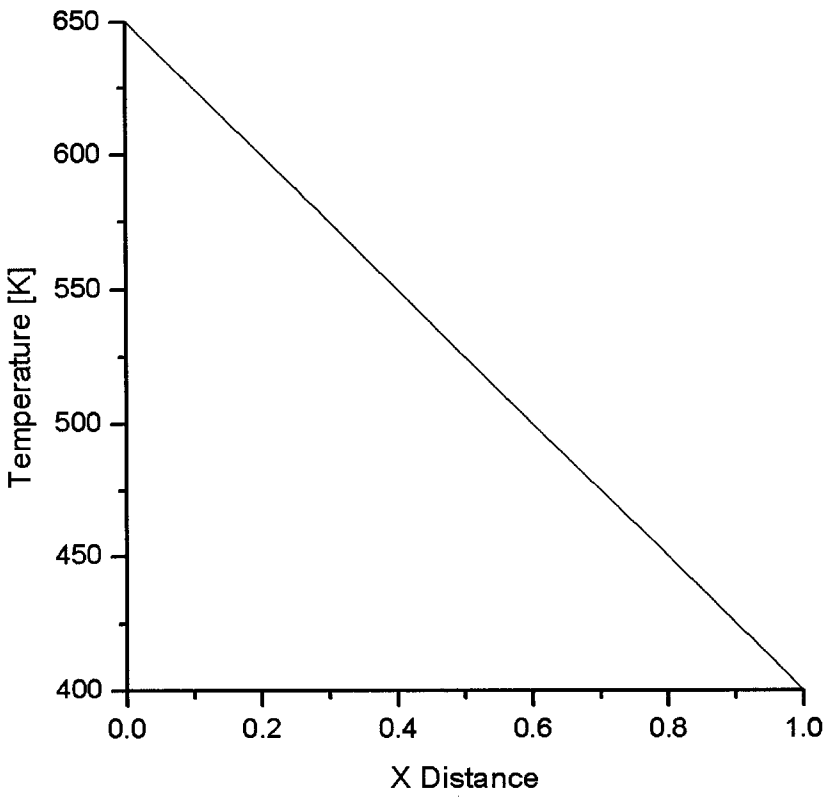



**Fig. 3.10. The exact solution for the one-dimensional heat transfer problem.**

The second of the two engineering problems solved by this implementation of AMoEBA involves the velocity profile of two-dimensional, fully-developed laminar flow in a straight pipe. The exact solution of this problem is a parabola with the maximum velocity located at the center of the pipe and the minimum velocity (zero) located at the wall:

$$u(z,r)=\frac{2.0*\bar{u}}{r_0^2}\left(r_0^2-r^2\right) \qquad [3.17]$$

where $u$ is the velocity, $\bar{u}$ is the average velocity, $r_0$ is the radius of the pipe, $r$ is the radial distance from the centerline of the pipe, and $z$ is the axial distance. For the domain used with AMoEBA (normalized to the range [0, 1] in the $x$ and $y$ dimensions), the analytical result is transformed into a polynomial using an average velocity of 0.5 with $x$ as the transformed radial dimension and $y$ as the axial dimension.

$$u(x)=0+4x-4x^2 \qquad [3.18]$$

The resulting velocity profile can be seen in Fig. 3.11. For this case, the initial coefficients are chosen randomly from the range of –5.0 to 5.0 with random coefficient mutation values chosen from the range of –0.05 to 0.05. It is clear by the analytical solution (Eq. 3.17), the exact polynomial (Eq. 3.18), and Fig. 3.11 that this problem is really independent of the axial distance. However, in the solution of this problem, the polynomial approximators are two-dimensional instead of the one-dimensional approximators used in the heat transfer solution and the segregation trees are allowed to divide the domain in the two different dimensions. Although the problem can be solved by using one-dimensional approximators, its difficulty is increased by the inclusion of the second dimension in the approximator and segregation tree evolutionary scheme.

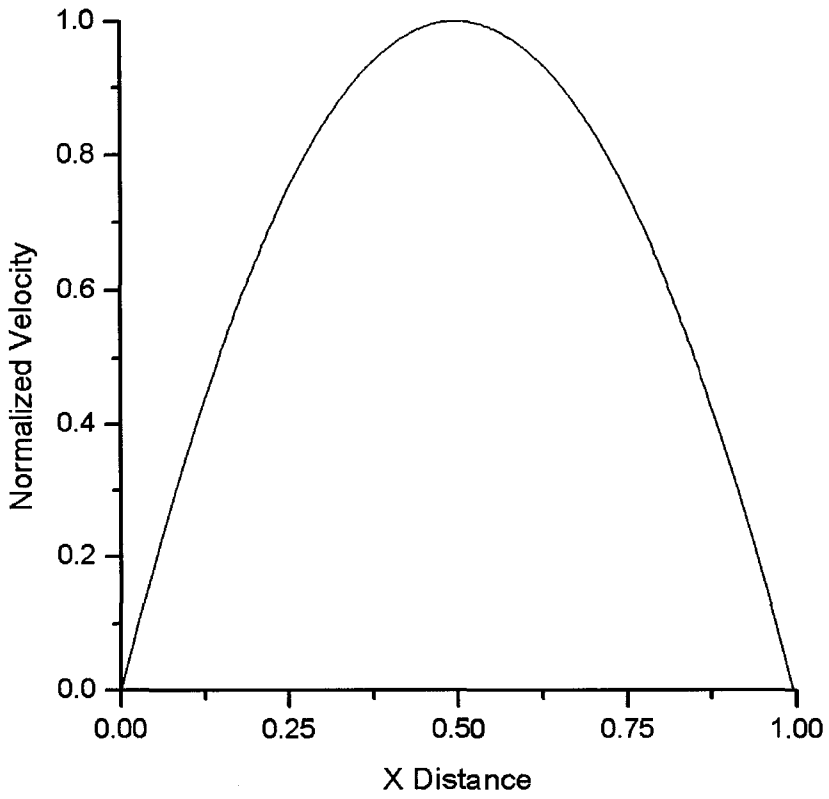



**Fig. 3.11. The exact solution for the two-dimensional laminar pipe flow problem.**

*3.3.3. Results*

Results of the heat transfer problem using this form of AMoEBA have illustrated that the polynomial/tree combination is capable of finding reasonable solutions to this problem. Based on the results of this initial solution, it was theorized that an increasing sampling rate would improve the efficiency of the algorithm. It was unclear, however, how quickly the number of samples should be increased and what number of initial sample points should be used. To determine a more efficient sampling rate for the laminar pipe flow problem, the heat transfer version of AMoEBA was converted to use two-dimensional polynomials and two-dimensional segregation trees. Using this implementation, two different types of sampling rate increases were studied, each starting from two different initial sampling rates and each resulting in the entire grid being sampled by the $1000^{th}$ generation. The results of this study were then used to solve the laminar pipe flow problem using the two-dimensional polynomials and the two-dimensional segregation form of AMoEBA.

3.3.3.1. Conduction Heat Transfer Results

Because of the co-evolution of two populations, two experiments were required to test the functionality of the populations. The effectiveness of the tree population was determined by hard-coding the equation used to generate the original data set (the analytical solution) into the initial approximator population. If the trees are evolved correctly, the trees with leaves pointing to the actual function would outperform any other trees in the population. Similarly, if the approximator evolutionary scheme is correct, the population of polynomials should evolve to different strains of the hard-coded analytical solution.

Thirty different random initial populations were tested to provide statistical relevance. The best structure out of each of these populations was recorded after 100, 250, 500, and 1000 generations. Fig. 3.12 shows the progression of the evolution based on the number of segregations made by the most fit structure. As expected, the number of trees with only one segregation increased as the number of generations increased. This alone does not demonstrate that the segregation trees are evolving correctly. The approximators applied to the regions of the best solutions in each generation demonstrate successful evolution of the segregation trees. After 100 generations, the best tree in each of the 30 random populations contained at least one reference to Eq. 3.16. As evolution progressed, this number fell to 14 trees after 500 generations and seven trees after 1000 generations. However, this does not mean that the trees de-evolved or became less fit. When the approximators of the trees that did not point to Eq. 3.16 were analyzed, they were found to be either identical copies or slight mutations of Eq. 3.16. This discovery shows that as the evolution continued, the actual function was chosen as a parent and that later in the evolution, copies of the analytical function were chosen as parents until a large percentage of the approximator population was similar to the analytical solution introduced into the system in the initial population.

The approximator population was further tested by running the code without any special adjustments. With the trees working, the approximators are expected to evolve to good approximations of the data set. Given enough time, the actual function, or slight variations of Eq. 3.16, are expected to be found. Fig. 3.13 illustrates the improvement in performance of the data segregation tree/approximator combinations without the help of the actual

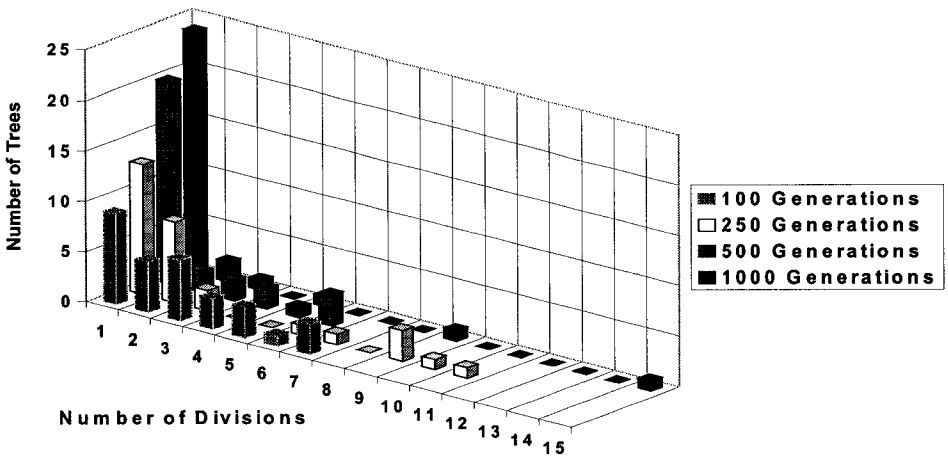



**Fig. 3.12. The progress of the segregation trees when evolved with Eq. 3.16 hard-coded into the program.**

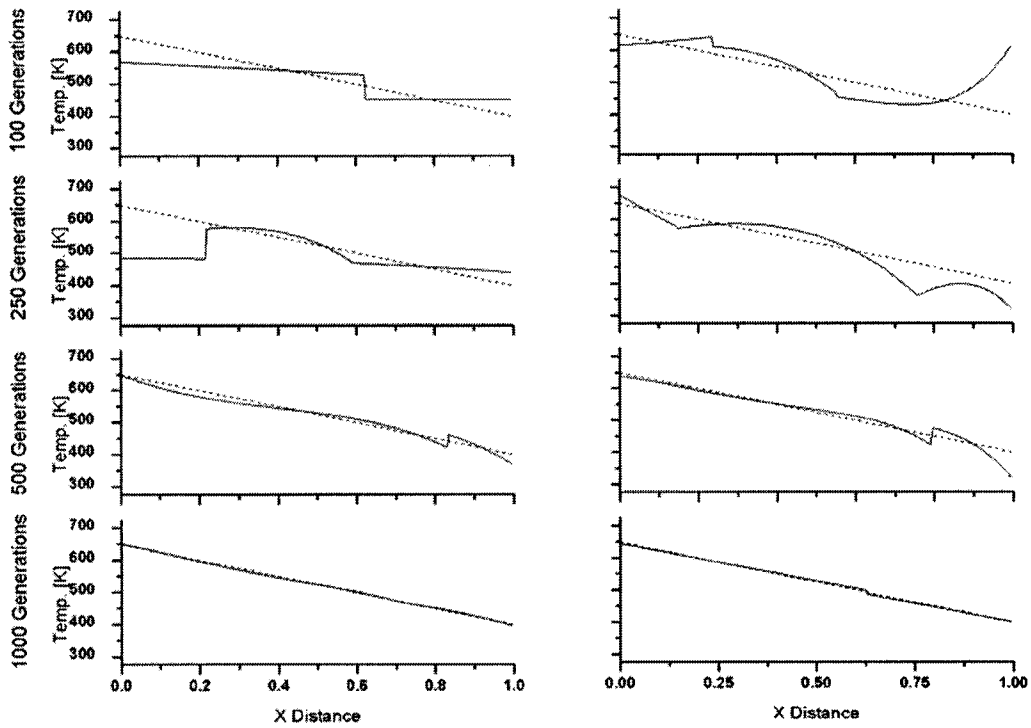



**Fig. 3.13. Temperature profiles of the best segregation tree/approximator combination after 100 generations, 250 generations, 500 generations, and 1000 generations for runs #14 (left) and #25 (right). The dotted line represents the actual temperature profile.**

function in the population. Again, AMoEBA found that smaller trees outperformed larger ones. Eqs. 3.19 through 3.22 show the approximators that were used in the 1000th generation of two sample trials: trial 14 (Eqs 3.19 and 3.20) and trial 25 (Eqs. 3.21 and 3.22).

$$T_1 = 652.02 - 260.06x - 200.5x^2 + 569.07x^3 - 147.67x^4 - 320.74x^5 - 27.71x^6 \quad [3.19]$$

$$T_2 = 652.02 - 279.9x - 209.32x^2 + 507.53x^3 - 147.67x^4 - 128.69x^5 \quad [3.20]$$

$$T_3 = 645.465 - 234.305x \quad [3.21]$$

$$T_4 = 634.18 - 234.305x \quad [3.22]$$

Although each of these equations is only used in part of the domain, the errors from the exact polynomial temperature profile have been calculated for the entire domain. The absolute value of the difference between the exact profile and the polynomial is normalized by the exact value at each of the 200 grid points in the domain. The errors for Eqs. 3.19 and 3.20 were significantly higher (7.1% and 2.9%, respectively) than the errors for the other two equations, Eqs. 3.21 and 3.22 (2.0% and 2.8%, respectively). However, when the error of the equations is calculated for only the regions in which it is used, the results are reversed. In this case, the errors of Eqs. 3.19 and 3.20 are 0.69% and 0.26% for a total error of 0.95%, whereas the errors of Eqs. 3.21 and 3.22 are 0.56% and 0.48% for a total error of 1.04%. This shows the effectiveness of the AMoEBA trees as they were able to place the equations in the regions in which they perform best, reducing an equation that had 7.1% error for the entire domain to only 0.69% error by placing it in the correct region (Eq. 3.19).

3.3.3.2. Sampling Rate Study

For the sampling rate study, the AMoEBA algorithm was adapted to include the ability to generate a $y$-polynomial combined with the original $x$-polynomial. The $y$ components of the new polynomials are again randomly generated up to tenth order with randomly generated coefficients. The same range of –800.0 to 800.0 is used to generate the random coefficients and the same mutation operators are adapted for the $y$ components. Similarly, the sampling function was adapted to increase with the number of generations. Two types of increases were studied: a linear increase from the initial sampling rate to the maximum number of nodes in the system, 40,000; and an exponential increase for the same range. Each of these increases starts from two different initial sampling rates and increases until, after 1000 generations, all 40,000 points are used in the fitness analysis. The initial sampling rates were varied to determine if having a lower sampling rate initially could save time. The one-dimensional heat transfer case that was first solved uses 5000 points as its sampling rate. This number of points was chosen as one of the initial sampling rates, and 500 points were used as the second initial sampling rate studied. Fig. 3.14 illustrates the four different sampling rate increases used.

To make a valid comparison, the fitness values were normalized to the sampling rate. Thirty independent trials were made with random initial populations. The most fit 20% (top six) of each final population is used in the comparisons for each sampling rate function/initial sampling rate combination. The average fitness value for these top six cases was calculated for the four different combinations: linear increase with 500 initial sampling points, linear increase with 5000 initial sampling points, exponential increase with 500 initial sampling

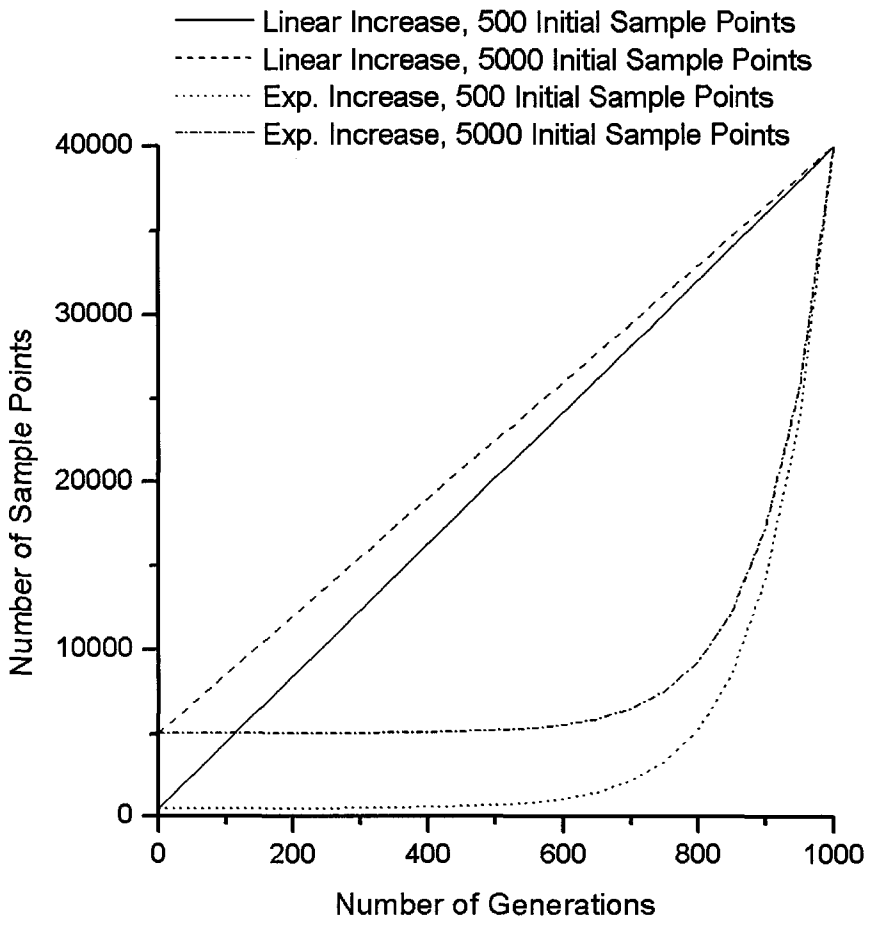



**Fig. 3.14. Four methods for increasing the sampling rate in the polynomial version of AMoEBA.**

points, and exponential increase with 5000 initial sampling points. The results of these calculations are shown in Table 3.1 with the overall best structure's fitness for each case and the wall clock time required to perform the 1000-generation trials. It can be seen that an exponential increase with 5000 initial sample points created the best overall structure, although the best average fitness value was found by the linearly increasing case with 500 initial sample points. The difference in time to solution for these two cases is significant. The exponential case with 5000 initial sample points completed 1000 generations in less than half the time required to complete the same number of generations using 500 initial sample points and a linear increase. These results show that if linear increasing is used, it is more efficient to start at a lower number, and this will yield as good or better structures. However, if speed is more important, using an exponential increase in the sampling rate and starting at a higher number may be best. Fig. 3.15 illustrates the resulting data sets after 1000 generations of the best structure for all 30 trials for each of the four different sampling functions studied. The results are three-dimensional, but to better compare the data sets to the exact temperature profile, the $y$ dimension is flattened. This only affects the results of the exponential case starting from 500 initial sample points, which has a slight dependence on the $y$ variable. All three of the other cases were able to find results that were independent of the $y$ dimension.

#### 3.3.3.3. Laminar Pipe Flow Results

Using the results of the sampling rate study, the laminar pipe flow problem was solved. In this problem, both the $x$- and $y$-polynomials are used and all coefficients are randomly generated between the range of –5.0 and 5.0. Thirty trials of different random populations

**Table 3.1: Results of 30 initially random populations after 1,000 generations for the sampling rate study. For the second column, the best structure in the six best runs was selected, and these six fitness values were averaged.**

| | Fitness of the best structure of all 30 runs | Average of the best structure of the six best runs | Time to complete 1000 generations of 30 different runs [hh]:[mm] |
|---|---|---|---|
| Exponentially increasing w/ 500 initial sample points | 842,793 | 3,533,082 | 23:30 |
| Exponentially increasing w/ 5,000 initial sample points | 58,685 | 1,191,359 | 39:23 |
| Linearly increasing w/ 500 initial sample points | 67,048 | 583,044 | 99:07 |
| Linearly increasing w/ 5,000 initial sample points | 287,193 | 2,225,594 | 113:36 |

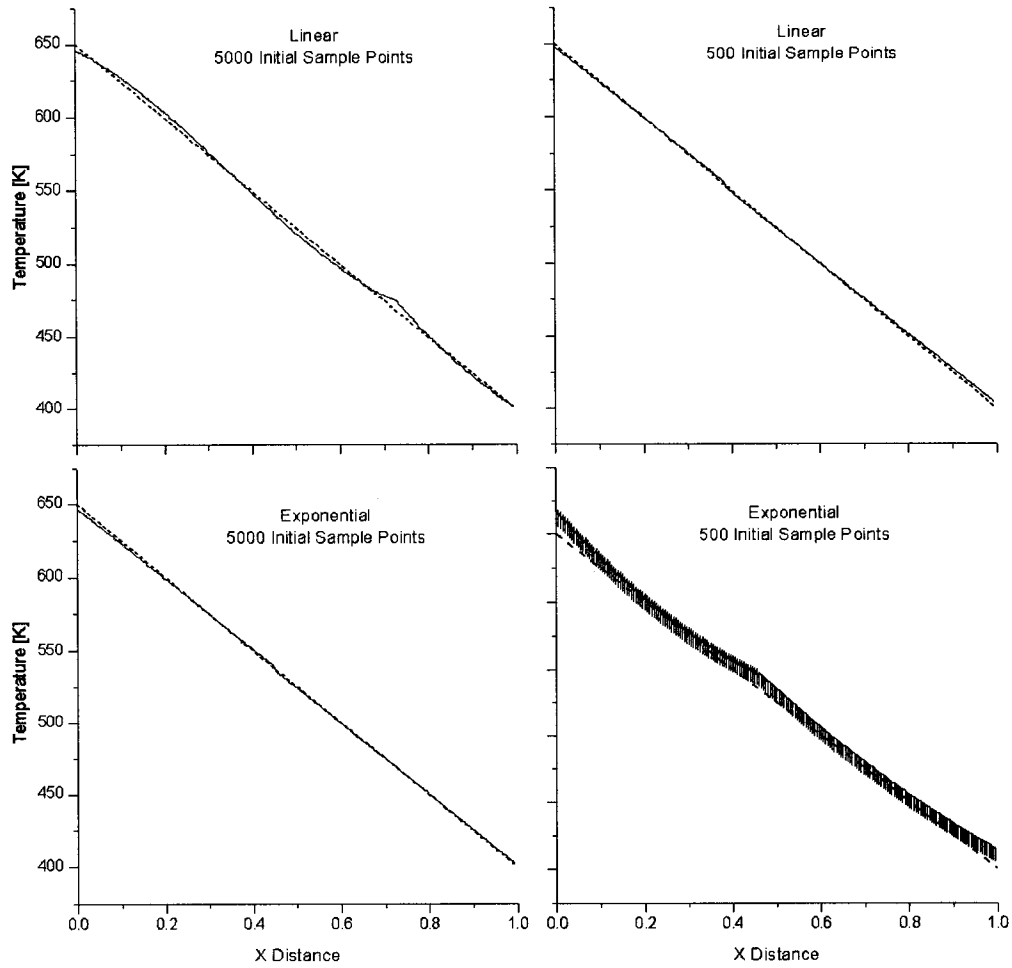



**Fig. 3.15. Results of the polynomial version of AMoEBA after 1000 generations for the four sampling rates studied.**

were evolved for 1000 generations for the exponential increase case, starting from 5000 initial sample points. This case was calculated on a faster machine than the previous sampling rate study cases. The first machine was a 24-node Silicon Graphics 400 MHz processor computer whereas the faster machine is an 18-node Linux Networxs dual 1 GHz and 1.2 GHz Athalon processor cluster. Because of this difference, the time necessary to evolve the polynomials and the trees was much faster than the 39:23 hours used in the sampling study. On the faster computer, the algorithm required 17:08 hours to calculate the 1000 generations. Fig. 3.16 illustrates the most fit solution after 1000 generations of the 30 initially random populations used in the study of the laminar pipe flow problem. As in the sampling rate study, the domain has been flattened and is viewed so that the profile of the exact solution can be seen. This solution used four segregations with a different polynomial in each of the four regions. Fig. 3.17 shows the segregation scheme found for this structure with the following equations (Eqs. 3.23-3.26) assigned to the regions:

$$\begin{aligned} u_1 = {} & 1.70713(yconst) + 0.454834y - 1.8911(xconst) + 3.79704x \\ & - 0.16645x^2 - 2.43788x^3 + 4.3286x^4 - 0.122183x^5 \\ & - 2.66701x^6 + 1.33453x^7 - 0.542511x^8 \end{aligned} \qquad [3.23]$$

$$\begin{aligned} u_2 = {} & 1.5588(yconst) + 0.467271y - 1.91978(xconst) \\ & + 3.75648x - 0.183913x^2 - 2.44945x^3 + 4.30893x^4 \\ & - 0.122183x^5 - 2.70747x^6 + 1.33453x^7 \end{aligned} \qquad [3.24]$$

$$\begin{aligned} u_3 = {} & 1.67818(yconst) - 1.91978(xconst) + 3.73099x \\ & + 0.183913x^2 - 2.44945x^3 - 4.30893x^4 - 0.0963959x^5 \\ & - 2.70747x^6 + 1.33453x^7 - 0.542511x^8 \end{aligned} \qquad [3.25]$$

$$\begin{aligned} u_4 = {} & -3.98626(yconst) + 4.21456(xconst) \\ & + 3.66057x - 3.94965x^2 \end{aligned} \qquad [3.26]$$

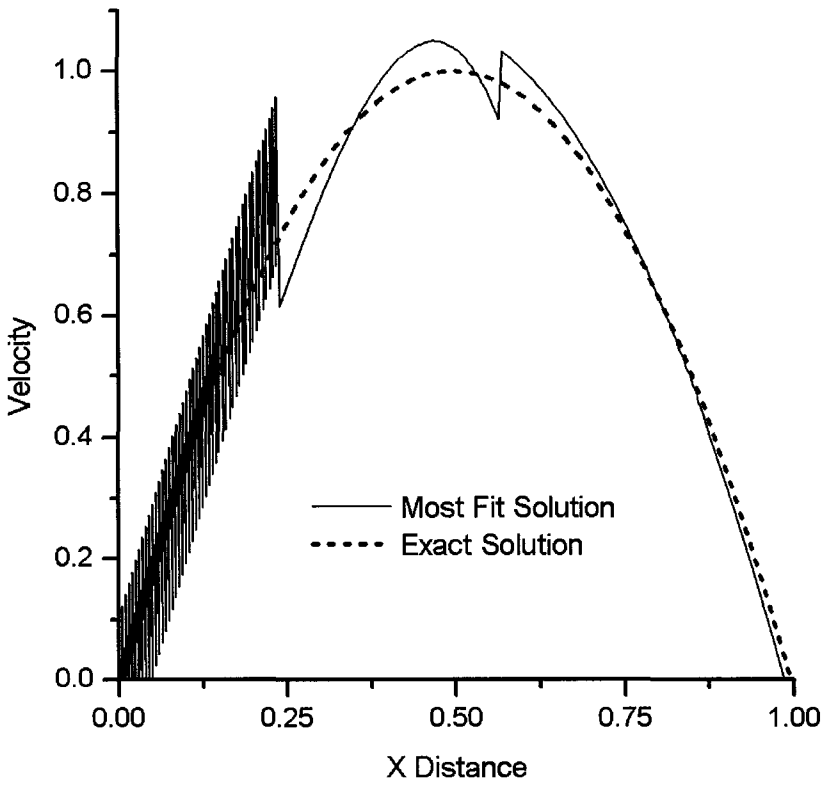



**Fig. 3.16. The best solution for the laminar pipe flow problem after 1000 generations. The shaded region on the left is the flattened section of the solution.**

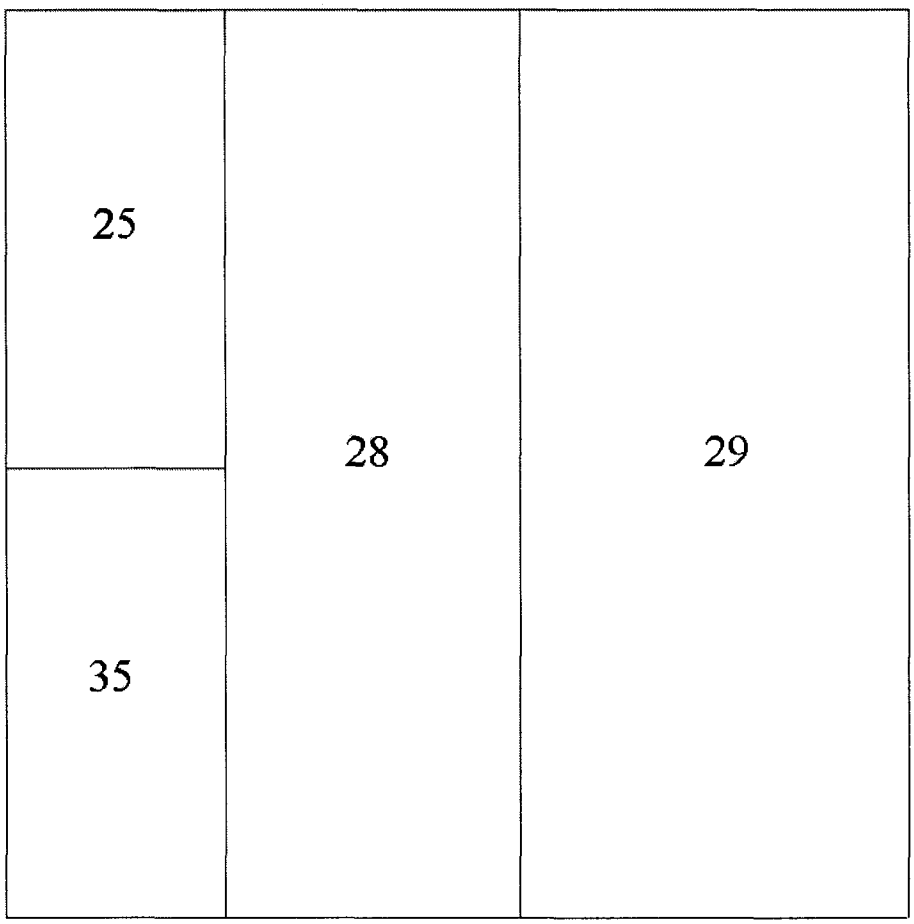



**Fig. 3.17. The segregation scheme of the best solution for the laminar pipe flow problem after 1000 generations.**

Fig. 3.18 illustrates a plot of each of the four polynomials used. In the figure, the linear $y$-dependency of Eqs. 3.23 and 3.24 is represented by illustrating the minimum and maximum lines of the equations to compare the approximators in the plane of interest. As shown in the figure, Eqs. 3.25 and 3.26 are similar to the exact solution of Eq. 3.17, but Eqs. 3.23 and 3.24 are only helpful in the regions in which they are assigned by the segregation tree. This demonstrates the effectiveness of the evolution of the segregation trees while pointing out the shortcomings of the evolved polynomials. Even though the polynomial expressions of Eqs. 3.23 and 3.24 are only beneficial in certain areas of the domain, AMoEBA was able to place these approximators in the proper regions. One interesting point to note is that Eqs. 3.24 and 3.25 share some similarities. The only major differences between these two equations is that Eq. 3.24 has a first order $y$ term, Eq. 3.25 has an eighth order $x$ term, the second and fourth order $x$ terms are basically off by a sign, and the fifth order $x$ term coefficients are slightly different. With this many terms and only a handful of differences, it is clear that these have come from the same lineage of polynomials.

Since the best solution after 1000 generations resulted in a poor approximation of the velocity profile, AMoEBA was run for 10,000 generations to determine the best performance of the algorithm on this problem. Fig. 3.19 illustrates the best solution found by AMoEBA for the laminar pipe flow problem. As shown in Fig. 3.19, the difference between the AMoEBA solution and the exact profile is almost indistinguishable. Although this solution required 10,000 generations to reach this level of performance, it demonstrates the effectiveness of the algorithm. It is anticipated that adjustments to the evolutionary

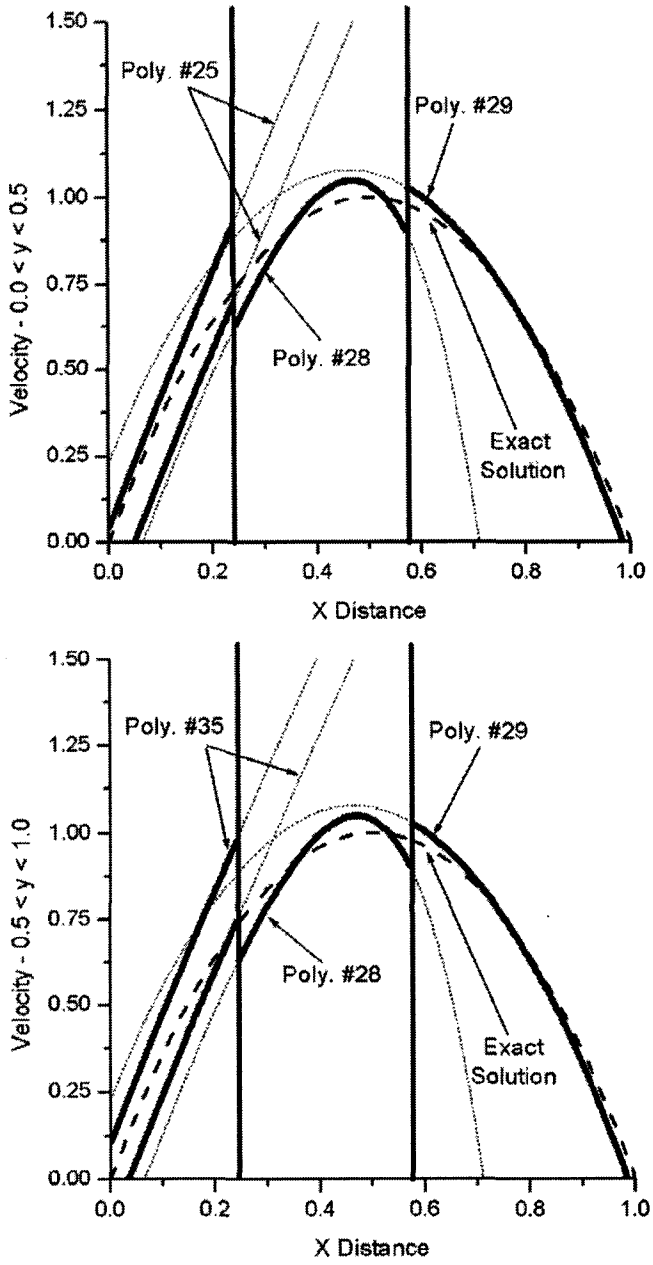



**Fig. 3.18. Complete plot of the approximators used in the best solution after 1000 generations of the laminar pipe flow problem. The bold lines are where the approximators are applied in the segregation scheme and the dotted lines are the extensions of these polynomials into other regions. The dashed line is the analytical solution.**

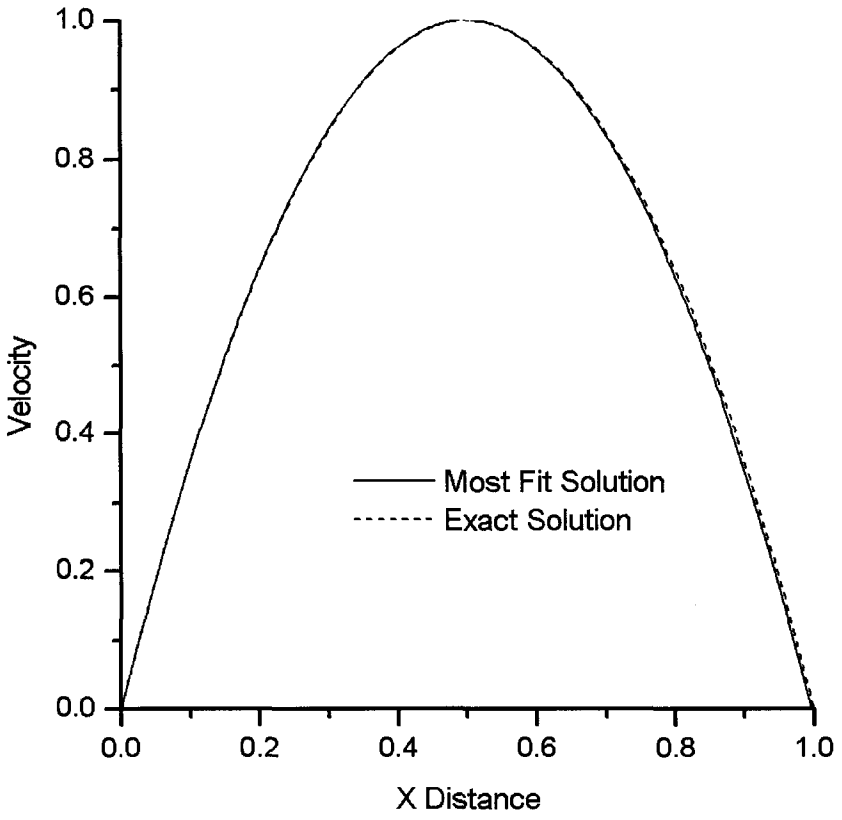



**Fig. 3.19. The best solution for the laminar pipe flow problem after 10,000 generations.**

parameters of the algorithm would decrease the number of generations required to reach this level of performance.

### *3.3.4. Conclusions*

The concept of adaptive modeling using evolving blocks can be applied to a simple one-dimensional problem. For the heat transfer problem, the data segregation trees were able to find that the best division of the domain was to have as few segregations as possible. The approximators also evolved to find polynomials that mimic the behavior of Eqs. 3.16 and 3.17. The sampling rate study showed that if a linear increase in the sampling rate is used, it is best to start from a smaller number of sample points; whereas if an exponential increase is used, a larger number of initial sample points is best. For the laminar pipe flow problem, the segregation trees were able to place good approximators in the proper regions and were able to divide the domain into smaller regions when good approximators were not available for that area of the domain. The evolved polynomials were less successful in the solution of this problem, but when the algorithm was given more time to find a solution, it performed much better. In this implementation of AMoEBA, polynomials were used because of the problems being solved. Although AMoEBA shows potential for data segregation in this implementation, the evolved polynomials are slow compared to more standard polynomial fitting techniques. It is anticipated that better segregations would be found with more advanced polynomial fitting techniques and the time to solution would also be reduced. The polynomials used in this implementation were useful in that they illustrated the effectiveness of a coevolutionary system applied to the AMoEBA algorithm. In the "Future

Work" section in Chapter 4 of this dissertation, issues regarding the use of AMoEBA with coevolved solution methods will be addressed.

## 3.4. AMoEBA: Inverse Heat Conduction Problems

Inverse solution of engineering problems involves the calculation of the properties and/or the boundary conditions based on knowledge of the dependent variable throughout the domain. This process is in contrast to direct solutions in which the dependent variable is calculated throughout the domain based on known boundary conditions and known physical properties. Although direct solutions are the most common technique in engineering problem solving, situations in which severe conditions make it impossible to use proper measurement tools or in which making these measurements affects the properties being measured require the use of an inverse technique. Inverse problems can be more difficult to solve than direct problems because often, these problems are ill posed and may have multiple solutions.

### *3.4.1. Literature Review: Inverse Engineering*

Researchers have used a variety of techniques to solve inverse problems in many different engineering situations. Huang and Chin [59] have used the conjugate gradient method to solve a two-dimensional inverse heat conduction problem (IHCP). In their study, temperature readings taken at appropriate locations and times on the surface of a non-homogeneous medium were used to estimate its time and spatially dependent thermal conductivity. Their approach was to view the IHCP as an optimization problem with the solution as the peak of the optimization landscape. Based on the results of two different test

cases, they concluded that their approach allowed them to successfully estimate functions for predicting non-homogeneous thermal conductivities with a very fast rate of convergence.

Previously Tervola [60] had used a technique based on knowledge of the temperature profile as well. The temperature dependent thermal conductivity of a homogeneous material was found in this study. Similar to [59], temperatures at certain points in the material were known and used in the solution. A piecewise polynomial was used to model the thermal conductivity. The resulting optimization problem was then solved using the Davidon-Fletcher-Powell method [60]. Two example problems were used to test the method and a practical problem, the determination of the thermal conductivity of sand in a mould, was also solved.

Others have used inverse techniques similar to the ones described to solve problems in heat transfer [61-64]. These techniques range from the identification of rock thermal conductivities through different layers [61] to the determination of physical properties and boundary conditions in casting situations [64]. Although most treat the IHCP as a minimization of the error between a candidate solution and the real solution, few have used techniques that search the entire landscape. Despite the fact that inverse problems can have multiple solutions, most have used the standard hill-climbing techniques that are best for unimodal problems.

Fan [65] has presented a more exhaustive approach by solving an inverse engineering problem, the design of diffuser blades, using a combination of genetic algorithms and neural networks. Direct methods are most commonly used in these designs and involve creating a temporary blade, computing and analyzing the resulting flowfield, and making changes to improve the blade's performance. This approach is iterative and can become very time-consuming. The inverse technique can be used when there are expected conditions (surface velocity or pressure distribution) for the ideal blade. Fan's solution uses these conditions as the goal and solves the inverse problem for the geometry of the blade profile. The genetic algorithm creates candidate blade profiles which are analyzed by a back-propagation neural net. The resulting flowfields are then compared to the target conditions and evolved until the ideal blade design is found.

AMoEBA has been adapted to be used for the inverse solution of heat transfer engineering design problems [66, 67]. In the two demonstrations presented here, the temperature profile and materials involved in a one-dimensional and a two-dimensional heat conduction problem are known, but the placement of these materials and the heat fluxes on the boundaries of the domains are unknown. Instead of making comparisons between the regions created by the data segregation trees or co-evolving polynomial approximators with the segregation trees, AMoEBA uses the evolving binary trees to define the geometry and assign different heat conduction solvers to each region. Each solver corresponds to a different material with the properties matching the known materials of the system. The multi-solver systems defined by the trees are solved directly. The resulting data sets are then compared with the known temperature profile. The least squared error between the

known profile and the proposed profiles are calculated and used to determine the fitness values of the new designs. When one of the trees' fitness values reaches the convergence criterion, the material placement scheme of the real system is found and boundary conditions matching the problem definition are identified. This process of adaptive blocks modeling applies to many inverse types of engineering problems because it can simultaneously optimize the domain decomposition and the equations describing the physics within the decomposed blocks.

### *3.4.2. Problem Description*

The application of the AMoEBA algorithm to inverse heat conduction problems is demonstrated on one-dimensional (Fig. 3.20) and two-dimensional (Fig. 3.21) steady state heat conduction through three different materials. Neither the heat flux nor the sizes and placements of the materials are known. The exact placements of these materials are shown in Figs. 3.20 and 3.21 with their respective temperature distributions. In both cases, a square domain is used with constant temperatures of 650 K at the left face ($x$ = 0 m), 400 K at the right face ($x$ = 1.0 m), and adiabatic boundary conditions on the top and bottom faces ($y$ = 1.0 m and $y$ = 0 m). The domain is normalized to the range [0, 1] in both the $x$ and $y$ directions.

In the one-dimensional case, Material 1 ($k$ = 10 W m$^{-1}$ K$^{-1}$) extends from $x$ = 0 m to 0.25 m, Material 2 ($k$ = 80 W m$^{-1}$ K$^{-1}$) from $x$ = 0.25 m to 0.75 m, and Material 3 ($k$ = 8 W m$^{-1}$ K$^{-1}$) from $x$ = 0.75 m to 1.0 m (Fig. 3.20). Although this is a one-dimensional problem, it is not trivial due to the thermal conductivities being used. The difference between the thermal

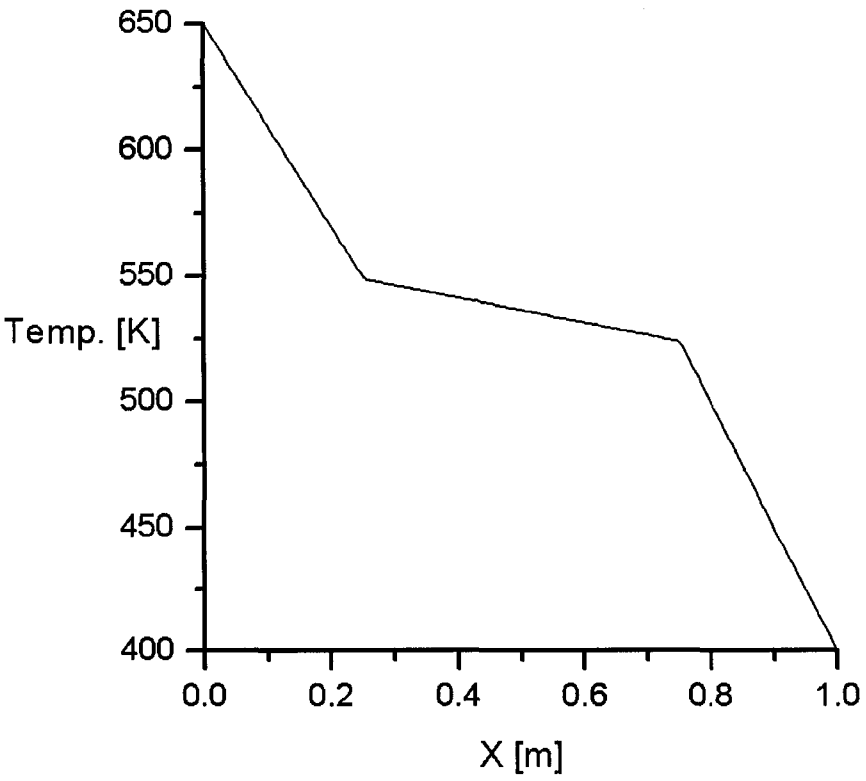



**Fig. 3.20. The exact temperature profile for the one-dimensional inverse heat conduction problem.**

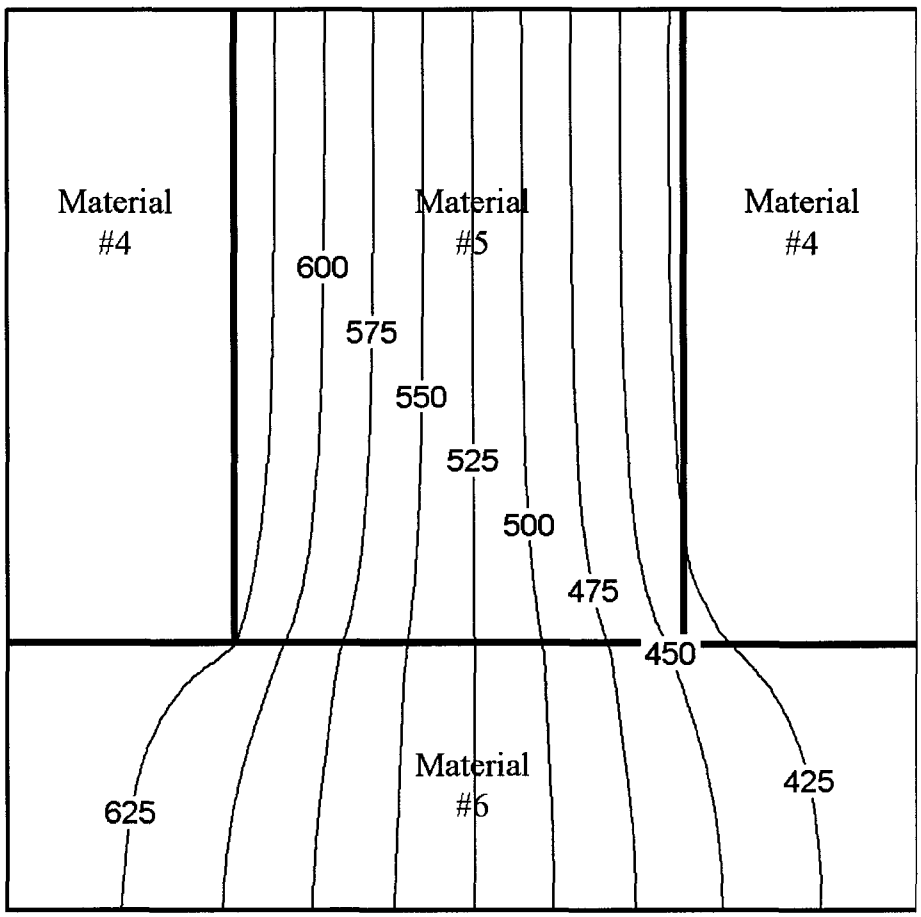



**Fig. 3.21. The exact material placement and contour lines for the two-dimensional inverse heat conduction problem.**

conductivities of Materials 1 and 3 is small compared to the difference between either of these and the Material 2 conductivity. Consequently, candidate configurations that have the correct boundary placement but place the materials in a 1-2-1 pattern instead of a 1-2-3 pattern provide local optima and may still result in an apparently good solution despite the error in the third position. For the two-dimensional problem, the material placement is divided along the $y = 0.3$ m line. Above this line, the domain consists of Material 4 ($k = 80$ W m$^{-1}$ K$^{-1}$) from $x = 0$ m to $x = 0.25$ m and from $x = 0.75$ m to $x = 1.0$ m with Material 5 ($k = 5$ W m$^{-1}$ K$^{-1}$) placed in between from $x = 0.25$ m to $x = 0.75$ m. Below the dividing line of $y = 0.3$ m is Material 6 ($k = 30$ W m$^{-1}$ K$^{-1}$) (Fig. 3.21). In both cases, the exact solution was used to generate a 200×200 grid of 40,000 temperatures as input data for the inverse heat conduction problem.

At the boundaries between the materials, a discontinuity in the slope of the profile occurs due to the property change. This discontinuity can present problems depending on the solution method used. To determine the proper placement of the materials, an inverse engineering problem must be solved. As the solution to this inverse problem is found, the residual, defined by the sum of the square of the difference between the candidate temperature profile, $\theta_{i,j}$, and the exact profile, $T_{i,j}$, over each point in the domain goes toward zero.

$$\text{Residual} = \sum_{j=0}^{M}\sum_{i=0}^{N}\left(T_{i,j} - \theta_{i,j}\right)^2 \qquad [3.27]$$

where $M$ and $N$ are the number of points in the $x$ and $y$ directions. The approach taken in this dissertation is to minimize the residual by evolving a population of candidate material

configurations. The material placements for the candidate configurations are generated by AMoEBA and are solved directly to obtain their individual thermal profiles, $\theta_{i,j}$. The residual is then calculated and used in the evaluation of the fitness functions for the candidate solutions.

### *3.4.3. AMoEBA Description*

To incorporate the heat conduction solvers into AMoEBA, a number of changes had to be made to the algorithm. Because the materials in the problem are given, all parameters associated with the solvers are known a priori. Therefore, the solvers do not need to be evolved like the approximators are in the polynomial implementation. However, a problem that was not necessary to address with the polynomials is critical with the heat conduction solvers. In the polynomial version of AMoEBA, the boundaries between the solvers are allowed to be discontinuous; there is no special attention given to matching the polynomials on either side of the boundary. This may seem to be useful due to the discontinuities in this problem. However, the discontinuities associated with the polynomials are in the dependent variable itself, whereas the discontinuities in this problem are with the dependent variable's first derivative. Ignoring the internal boundaries is not possible in this application of AMoEBA because the results of the heat conduction solvers depend on the boundary conditions. The fitness function for the heat conduction solver implementation of AMoEBA is also different from the polynomial case. In the polynomial version, the time spent in evaluating the fitness function at the 5000 sample points is significant when compared with the time required to calculate the value of the polynomial at these points. Because the heat conduction solvers take a considerably longer time to create the data set to be compared

with the exact solution, the time spent evaluating 5000 sample points is of the same order as the time spent evaluating all 40,000 points in the data set. Another major difference between the two implementations is the fact that the polynomials are evolved and are not based on physical laws or parameters. The heat conduction solvers, on the other hand, are designed to obey physical laws. Because of this, the candidate data sets are tested to determine if energy is conserved, a necessary quality in the solution to this problem.

Because the two-dimensional problem is much more difficult to solve than the one-dimensional problem, certain evolutionary parameters were adjusted to fine-tune the search process for each problem. The evolutionary parameters for both problems can be seen in Table 3.2. Adjustments have been made to the population size ($n_o$), the tournament size ($n_t$), the maximum total number of nodes in a tree ($N_{max}$), the crossover rate ($R_{co}$), and the probabilities of the different types of mutation operations (probability of any mutation–$P_n$, probability of random subtree mutation–$P_{st}$, probability of flip mutation–$P_f$, probability of boundary value mutation–$P_{bv}$, and probability of leaf mutation–$P_l$).

Initially, 30 separate trials of the randomly generated structures (tree/solver combinations) were created as the starting populations for the two AMoEBA searches. A fitness criterion of 0.5 was used in both cases to determine when an exact answer to the inverse heat transfer problem had been found. Finding an exact answer depends both on obtaining a suitable match between the temperature profiles of the candidate and exact solutions, and on the conservation of energy for the candidate solution to ensure physically realistic results. Because of this, the fitness of a structure is the product of an energy balance penalty and the

**Table 3.2. Evolutionary parameters used in AMoEBA for the one-dimensional (1-D) and the two-dimensional (2-D) inverse heat conduction problem.**

| | 1-D Problem | 2-D Problem |
|---|---|---|
| Population Size ($n_o$) | 72 | 104 |
| Tournament Size ($n_t$) | 4 | 8 |
| Maximum Number of Nodes ($N_{max}$) | 24 | 8 |
| Crossover Rate ($R_{co}$) | 50% | 5% |
| Probability of Mutation ($P_n$) | 5:6 | 6:7 |
| Probability of Subtree Mutation ($P_{st}$) | 1:6 | 1:7 |
| Probability of Flip Mutation ($P_f$) | N/A | 1:7 |
| Probability of Leaf Mutation ($P_l$) | 1:6 | 1:7 |
| Probability of Boundary Value Mutation ($P_{bv}$) | 1:2 | 3:7 |

sum of the squared error between the structure's data set and the actual temperature profile. This is:

$$Fitness = (1 + P_e) * \left[ \sum_{i=0}^{M} \sum_{j=0}^{N} (T_{i,j} - \theta_{i,j})^2 \right] \qquad [3.28]$$

where $P_e$ is the energy balance penalty, $T_{i,j}$ is the exact temperature, and $\theta_{i,j}$ is the candidate solution temperature. Both $T_{i,j}$ and $\theta_{i,j}$ are evaluated at point $i,j$. The quantity in the bracket is simply the residual from Eq. 3.27. For the energy to be balanced, the heat flux entering the domain from the $x$ = 0 m end ($q_0''$) and the heat flux leaving the domain from the $x$ = 1 m end ($q_1''$) must be equal. Any nonzero difference in these values equates to an energy imbalance in the system. For the penalty function, the heat fluxes are calculated for the candidate solution and their difference is normalized, first by the $q_0''$ value and then by the $q_1''$ value. The absolute values of these normalizations are then compared, and the maximum of these two normalizations is the penalty function:

$$P_e = \max\left( abs\left( \frac{q_1'' - q_0''}{q_1''} \right), abs\left( \frac{q_1'' - q_0''}{q_0''} \right) \right) \qquad [3.29]$$

The reason that there are two different normalizations is because neither the left-face heat flux nor the right-face heat flux is the correct heat flux. By using the maximum of the two normalizations, we ensure that the actual energy imbalance is less than that used in the penalty function.

### *3.4.4. Results*

Both the one- and two-dimensional cases were tested on 30 trials, each using a different initial population. The different populations were used to ensure statistical relevance due to the initially random structures. In each case, the algorithm was permitted to run until the fitness criteria had been reached or until the algorithm had completed a maximum of 2500 generations. If the algorithm did not find a structure that met the criteria within 2500 generations, it was determined that it would not find the structure in a reasonable amount of time for that initial population.

#### 3.4.4.1. One-Dimensional Results

For the one-dimensional problem, all 30 trials were able to find structures that met the fitness criteria of 0.5 in the allocated number of generations. Twenty-six of the 30 trials were able to find the proper material placement with the other four having slightly different configurations. It is interesting to note that each of the four solutions that did not find the proper material placement resulted in approximately the same material configuration. Fig. 3.22 shows the material placements of the most fit of the four incorrect segregation schemes (Trial 25), the most fit of the 26 solutions with the proper material configuration (Trial 2), and the exact solution. Fig. 3.22 also shows that there is very little difference in the material placements between the correct and incorrect solutions. The only difference is the small strip of Material 1 that is placed between Materials 2 and 3. The boundaries between materials for the exact solution are located at $x = 0.25$ m and $x = 0.75$ m. For the correct segregation scheme presented in Fig. 3.22 (Trial 2), these boundaries are located at $x = 0.2540$ m and $x = 0.7496$, corresponding to errors of 1.6% and 0.16%, respectively. The

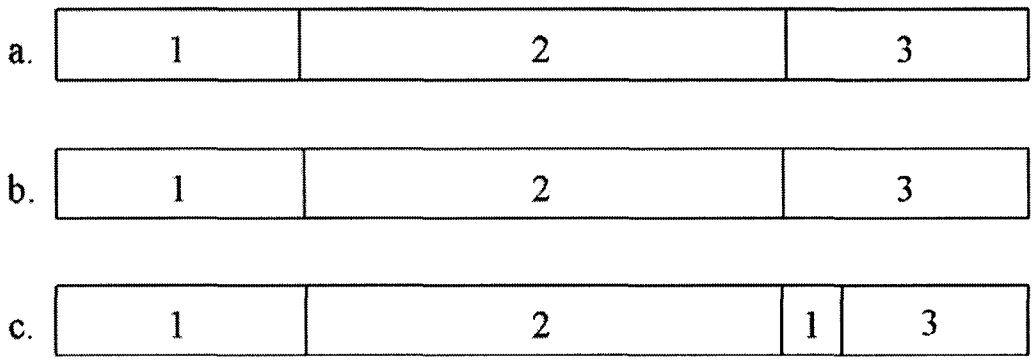



**Fig. 3.22. The exact material placement (top), the most fit of the correct material placements (middle), and the most fit of the incorrect material placements (bottom).**

incorrect solution (Trial 25) has similar boundaries, which are located at $x$ = 0.2555 m (2.2% error), $x$ = 0.7434 m, and $x$ = 0.7986 m, with the extra strip of Material 1 placed between these last two values. The results of these material placements give fitness values of 0.037 for the correct solution and 0.273 for the incorrect solution. A complete table showing the fitness values and times to solution for each of the 30 trials is given in Table 3.3.

For the 30 trials studied, the solution time ranged from 13 generations to 89 generations with an average time to solution of 42.1 generations and a standard deviation of 20.6. Trial 12 found the correct configuration the fastest with a solution time of 13 generations. Trial 24 was the slowest of the trials that found the correct configuration. Eighty-six generations were required to find the correct segregation scheme. Trial 26 was the slowest of all the trials. After its 89 generations, it still did not find the correct solution even though it did find a structure that satisfied the fitness criteria. The evolution of the best structures' fitness values for these three cases can be seen in Fig. 3.23. It should be understood that this is not the evolution of the solution of a single structure but of the best solution in the population for each generation. In each of the three cases presented, the starting fitness values of the best structures are approximately the same: ~100. As shown, the fitness values move forward as a series of major and minor steps. In Trial 12, the evolutionary process finds the correct material placement and boundaries in the final major step. Trial 24 shows a more typical pattern in which the correct material configuration of 1-2-3 is found at generation 68 but the boundaries are slightly incorrect ($x$ = 0.2604 and $x$ = 0.7526). The evolutionary process then slowly tweaks these boundaries into place ($x$ = 0.2466 and $x$ = 0.7526). These

**Table 3.3. The final fitness values of the best structure of each of the 30 trials for the one-dimensional inverse heat transfer problem, including the number of generations required to reach the fitness criterion and whether the correct material placement was found.**

| Trial Number | Generations To Solution | Final Fitness | Incorrect Material Placement |
|---|---|---|---|
| 0 | 24 | 0.284 | |
| 1 | 30 | 0.192 | |
| 2 | 47 | 0.037 | |
| 3 | 45 | 0.495 | |
| 4 | 34 | 0.374 | |
| 5 | 59 | 0.233 | |
| 6 | 21 | 0.199 | |
| 7 | 16 | 0.326 | |
| 8 | 26 | 0.420 | |
| 9 | 66 | 0.254 | |
| 10 | 43 | 0.219 | |
| 11 | 48 | 0.498 | |
| 12 | 13 | 0.368 | |
| 13 | 19 | 0.220 | |
| 14 | 66 | 0.220 | |
| 15 | 62 | 0.345 | X |
| 16 | 68 | 0.271 | |
| 17 | 24 | 0.333 | |
| 18 | 38 | 0.371 | |
| 19 | 29 | 0.475 | |
| 20 | 16 | 0.355 | |
| 21 | 54 | 0.228 | |
| 22 | 45 | 0.395 | X |
| 23 | 20 | 0.485 | |
| 24 | 86 | 0.245 | |
| 25 | 57 | 0.273 | X |
| 26 | 89 | 0.496 | X |
| 27 | 33 | 0.485 | |
| 28 | 51 | 0.329 | |
| 29 | 33 | 0.289 | |

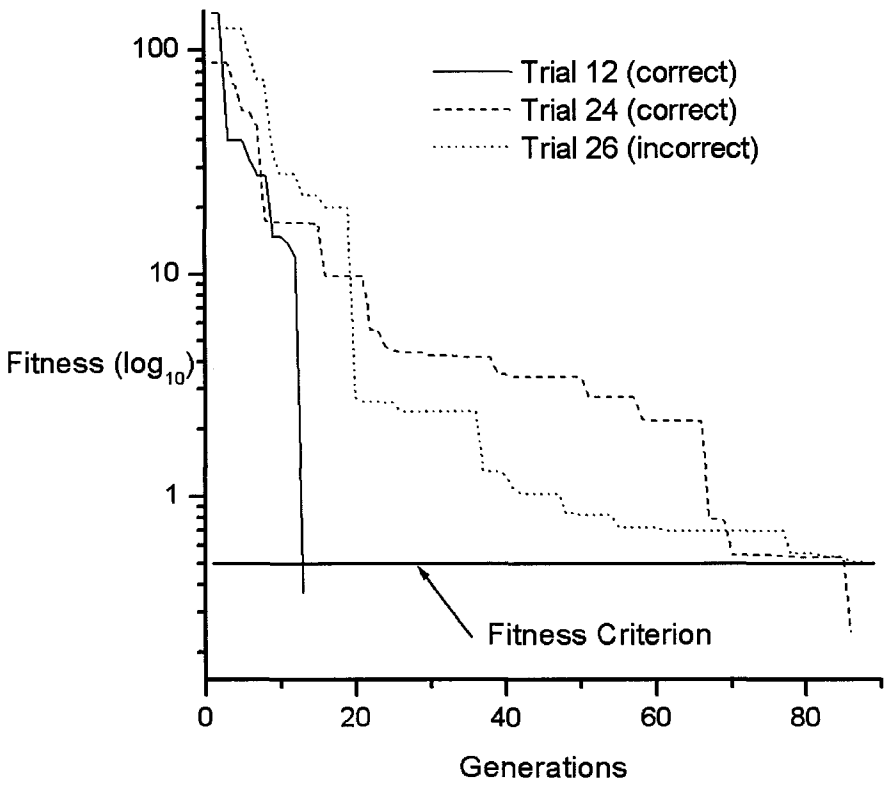



**Fig. 3.23. Evolution of the fastest (Trial 12) and the slowest (Trial 24) of the trials that were able to find the correct solution and the slowest of all the trials (Trial 26).**

small variations in the boundary locations occur through boundary value mutation. Based on the evolutionary parameters, boundary value mutation only occurs 1/6 of the time for the one-dimensional case and 1/7 of the time for the two-dimensional case and can only improve the best structure when it is not crossed over with a co-parent. Even then, the mutation could move the boundary too much or in the wrong direction, depending on the random floating-point value generated. Due to the potentially disruptive behavior of the genetic programming-generated segregation trees, fine-tuning these evolutionary parameters is difficult. Increasing the rate of boundary value mutation may be helpful for one trial, but may slow the evolution of other trials.

3.4.4.2. Two-Dimensional Results

Of the 30 initial trials of the AMoEBA algorithm on the two-dimensional inverse heat conduction problem, only 18 were able to evolve to solutions that met the fitness criterion of 0.5. As this was not adequate for statistical reasons, 15 more trials were generated and, of these, 12 were able to reach fitness values less than the 0.5 criterion within the 2500 generation run limit that was enforced. The two different studies resulted in 60% and 80% solution rates, respectively, with an overall solution rate of 66.7% (30 of the 45 total trials met the criteria). Table 3.4 summarizes the time to solution of each trial, their resulting fitness values, and whether the proper material configuration was obtained. Of the 30 populations that converged to a solution, the fastest (Trials 19 and 25) completed the search in 75 generations and the slowest (Trial 20) took 1905 generations. The average time to solution of the cases that converged was 549 generations with a standard deviation of 513. Looking only at the 20 cases that converged in less than 500 generations, the average is 237

**Table 3.4. Fitness values and the number of generations required to reach the convergence criterion for all 45 trials of the two-dimensional inverse heat conduction problem including whether the best structure was able to find the correct material placement.**

| Trial Number | Generations to Solution | Final Fitness | Incorrect Material Placement |
|---|---|---|---|
| 0 | 320 | 0.394956 | |
| 1 | MAX | 0.631043 | X |
| 2 | MAX | 1.45117 | X |
| 3 | 1275 | 0.487194 | X |
| 4 | MAX | 0.630993 | X |
| 5 | 810 | 0.475753 | X |
| 6 | 465 | 0.467608 | X |
| 7 | MAX | 0.929588 | X |
| 8 | MAX | 0.929763 | X |
| 9 | MAX | 0.90851 | X |
| 10 | 435 | 0.458513 | |
| 11 | 195 | 0.497371 | |
| 12 | MAX | 1.82436 | X |
| 13 | 125 | 0.423296 | |
| 14 | MAX | 1.36568 | X |
| 15 | 1110 | 0.422012 | |
| 16 | 485 | 0.422015 | |
| 17 | 105 | 0.422016 | |
| 18 | 120 | 0.447247 | |
| 19 | 75 | 0.446024 | |
| 20 | 1905 | 0.487544 | X |
| 21 | MAX | 0.631103 | X |
| 22 | MAX | 0.790428 | X |
| 23 | 1125 | 0.475359 | |
| 24 | 90 | 0.497374 | |
| 25 | 75 | 0.422019 | |
| 26 | 815 | 0.475365 | |
| 27 | MAX | 0.630994 | X |
| 28 | MAX | 0.929765 | X |
| 29 | 1205 | 0.394955 | |
| 30 | 435 | 0.393723 | |
| 31 | 415 | 0.436202 | |
| 32 | 125 | 0.497368 | |
| 33 | 315 | 0.474209 | |
| 34 | 150 | 0.369233 | |
| 35 | 150 | 0.497373 | |
| 36 | 80 | 0.474204 | |
| 37 | MAX | 0.552746 | X |
| 38 | 135 | 0.47464 | |
| 39 | MAX | 0.631072 | X |
| 40 | 1660 | 0.423287 | |
| 41 | 445 | 0.499397 | X |
| 42 | MAX | 0.929929 | X |
| 43 | 620 | 0.457445 | |
| 44 | 1205 | 0.463838 | |

generations with a standard deviation of 156. A good example solution of this average is Trial 11 (195 generations). The ten cases that took more than 500 generations had an average of 1173 generations with a standard deviation of 388. Good example solutions of this case are Trials 29 and 44 (1205 generations each). The evolution of the best structure's fitness for the fastest (Trial 19 and 25), the slowest (Trial 20), and the three average cases (Trials 11, 29, and 44) can be seen in Fig. 3.24.

The most fit of the final results (Trial 34) obtained a fitness value of 0.369. The fitness value of the least fit of the final solutions (Trial 41) is 0.499. The average fitness value of the final solutions is 0.453 with a standard deviation of 0.037. Of the 30 cases that found a solution, 25 found solutions that almost exactly match the proper material configuration. Fig. 3.25 illustrates three of these solutions that were able to locate the proper material placement as well as the exact material placement in this system. The average fitness value of the cases that were able to find the proper material placement is 0.446 with a standard deviation of 0.037.

Fig. 3.26 shows three of the five incorrect configurations with the exact material placement. Unlike in the one-dimensional case, there is little similarity between the incorrect configurations found. The range of fitness values of the incorrect configurations is between 0.499 and 0.468. The average of the fitness values of these structures is 0.483 with a standard deviation of 0.012. This average is significantly higher than the average of those structures that found the proper material placement. It should be noted that none of the cases that did not converge were able to find the proper material configuration. In other

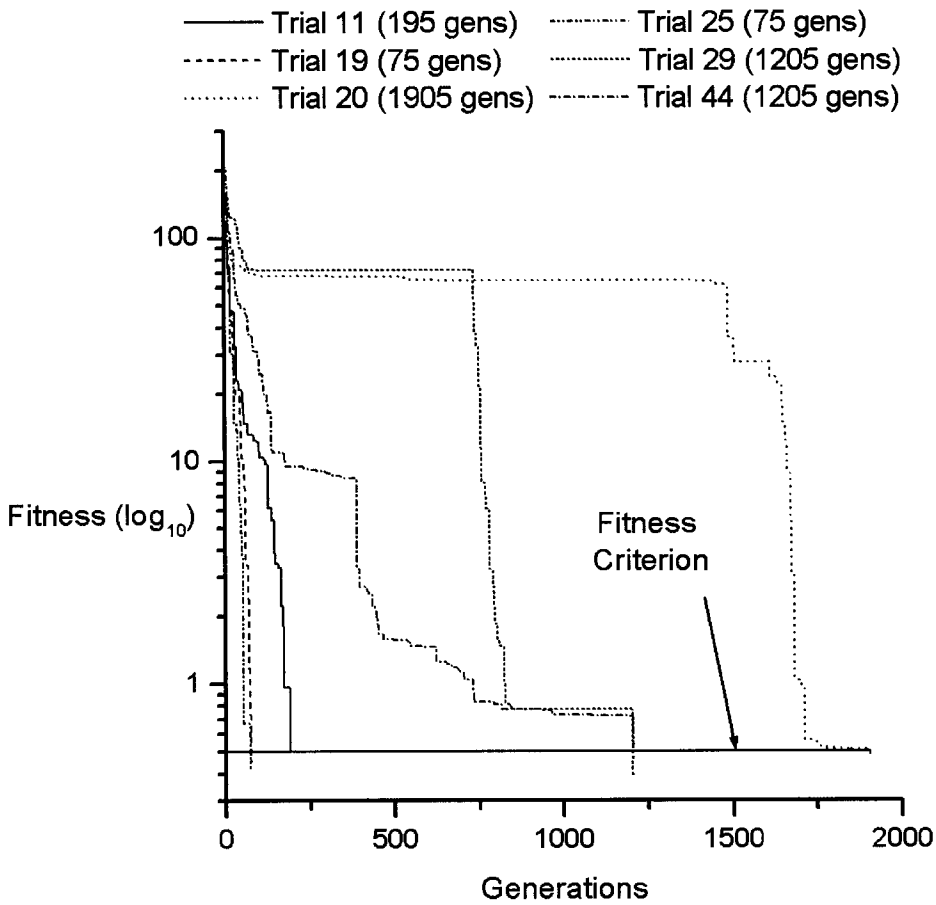



**Fig. 3.24. The evolution of the best structure's fitness value for the fastest (Trial 19 and 25), the slowest (Trial 20), and the three average cases (Trials 11, 29, and 44) of AMoEBA's solution of the two-dimensional inverse heat conduction problem.**

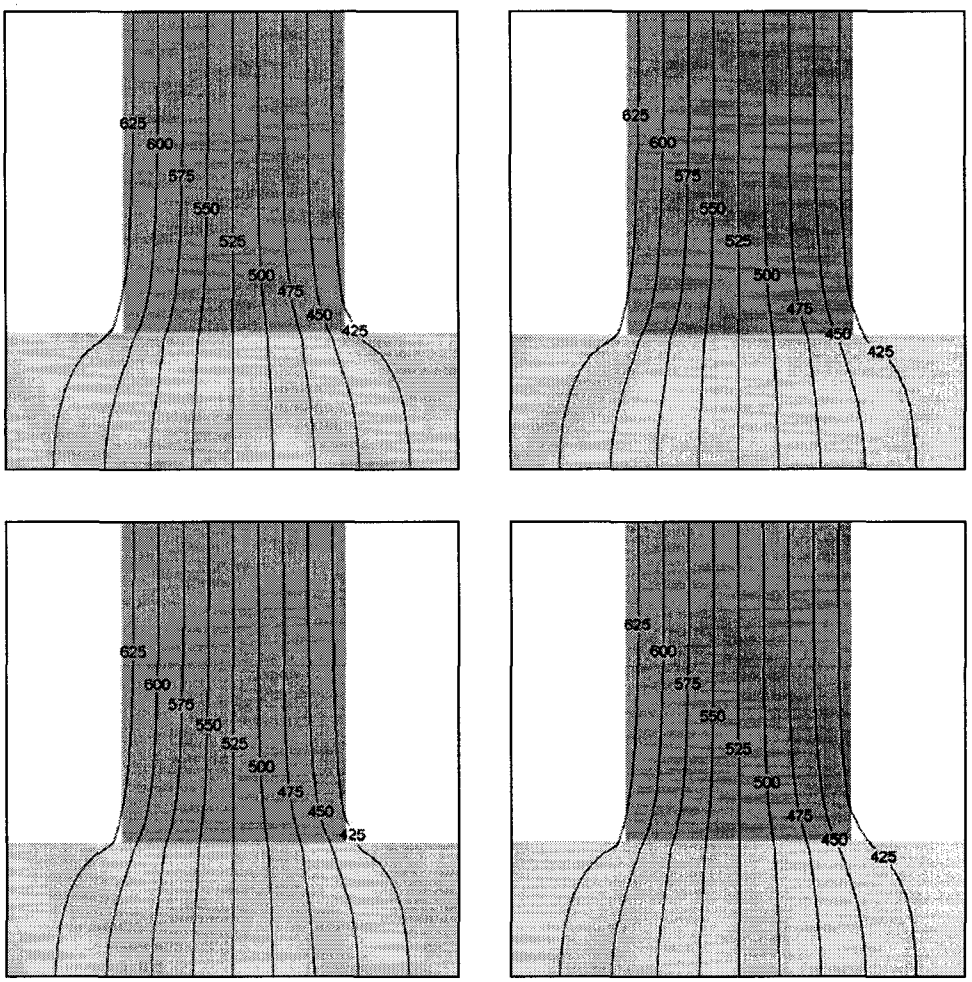



**Fig. 3.25. Material placements and contour maps of three properly placed solutions (Trial 19–upper left, Trial 30–upper right, and Trial 35–lower left) and the exact material placement (lower right). The color code is: Material 4 = white, Material 5 = dark gray, and Material 6 = light gray.**

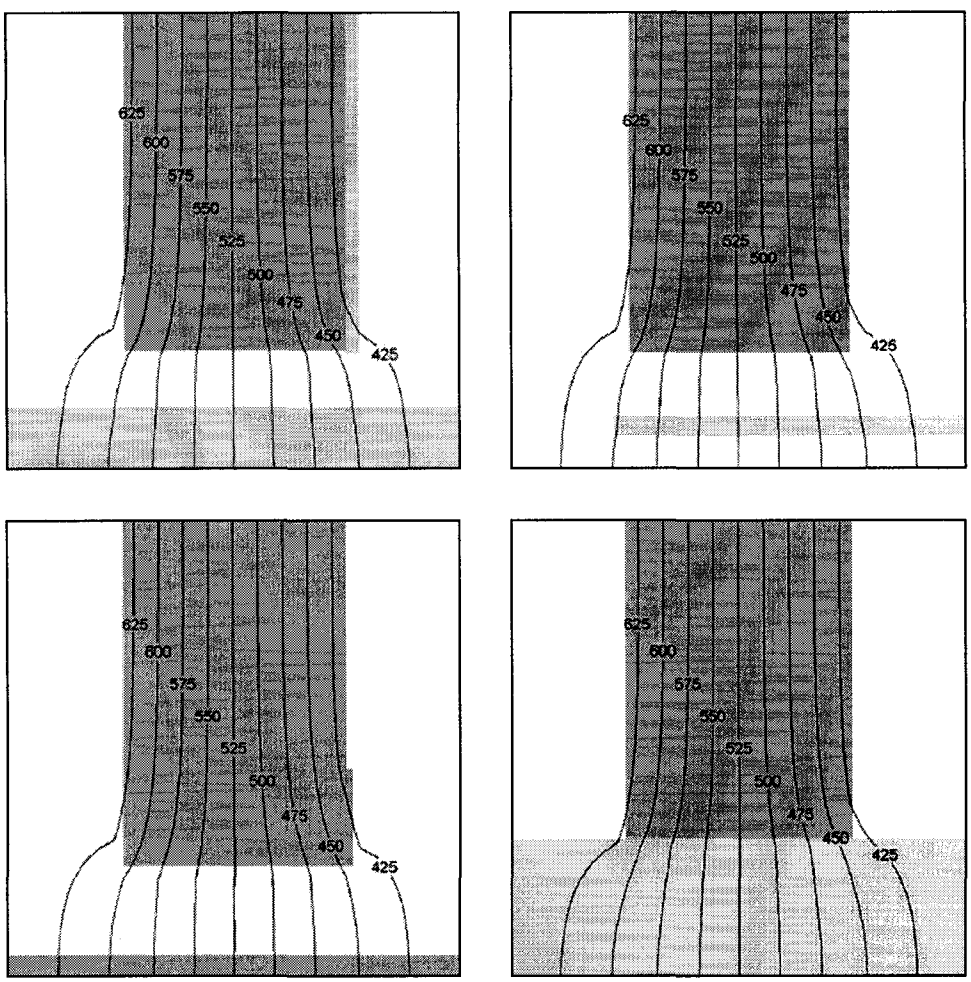



**Fig. 3.26. Material placements and contour maps of three improperly placed solutions (Trial 5–upper left, Trial 20–upper right, and Trial 41–lower left) and the exact material placement (lower right). The color code is: Material 4 = white, Material 5 = dark gray, and Material 6 = light gray.**

words, trials that did not reach the fitness criteria did not find the proper material configuration. The improper placements show that for this problem, there are multiple answers based on the way the problem is solved and for the criteria used. In this case, the solution of this problem depends greatly on the accuracy of the direct solutions of the candidate placements. A stricter fitness criterion and/or a stricter convergence criterion for the direct solution may eliminate the possibility of these structures meeting the criteria, but at a larger cost in computation time.

### *3.4.5. Conclusions*

Solutions of one- and two-dimensional inverse heat conduction problems using evolutionary data segregation techniques have been demonstrated. The algorithm evolves until a suitable partitioning scheme is found based on the known thermal profile and the known materials involved. In the one-dimensional case the proper material placement was found 86.7% of the time, and in the two-dimensional case the materials were placed correctly 83.3% of the time that a solution was found. With more compute time and stricter criteria, it is anticipated that these values will improve. The primary advantages of this algorithm are its ability to handle discrete rather than continuous data and geometries and its potential ability to handle complex geometries in the same manner as simple geometries. In the "Future Work" section of Chapter 4 of this dissertation, the extension of this algorithm to more complicated and three-dimensional geometries will be addressed.

## 3.5. AMoEBA: Computational Fluid Dynamics (CFD)

This version of AMoEBA uses the same segregation trees and evolutionary algorithm to

segregate a large domain into smaller sub-domains. The main differences again lie in the types of solvers that are then assigned to these regions and in the fitness function used to determine whether one segregation scheme/solver placement is better than another. The goal for this application of AMoEBA is to be able to segregate a computational fluid dynamics problem into regions containing different solution methods. The two main reasons for segregating a CFD domain are for faster solution times and for improved accuracy.

- In the first case, the concern is to be able to calculate a CFD data set, make changes to the boundary conditions or the geometry, and be able to recalculate the data set as quickly as possible with a known level of accuracy. The important part is to establish a "known level of accuracy" that is accurate enough to be a basis for engineering decision making. If an engineer can receive feedback from the model very quickly with an understanding of the level of accuracy, she/he can base judgements on the model and make the changes that he/she feels are necessary. For example, if the main concern of the model is only the exit conditions, the engineer may feel that accuracy of fine details in the flowfield is not as important as long as the exit conditions are predicted as accurately as necessary. By relaxing the accuracy in the inner portion of the flowfield, a faster solution to the problem can be found. Without this understanding of the accuracy, the engineer will not have confidence in the model and the model will lose value since sound judgements can no longer be made based on the data presented.
- The second reason for dividing a CFD domain into a network of solution methods is for improved accuracy over single models. There are a variety of turbulence models

being used in computational analysis of fluid flows. Each is designed for a specific purpose and performs best when used for these flow conditions. For example, the k-ε RNG (renormalized group) turbulence model is more accurate than other turbulence models when used for calculating flows that have a high degree of swirl. However, flows that are more unidirectional are best calculated using the standard k-ε turbulence model. When an engineer is confronted with a flow that has a high degree of swirl in some areas and a more directional flow in others, she/he must decide in which area(s) he/she is willing to sacrifice the accuracy. By segregating the flowfield, the engineer is allowed to place the correct solver in the areas where they are needed most. This reason for segregating a CFD domain will not be addressed in detail but is mentioned here for completeness and is discussed in the "Future Work" section of Chapter 4.

In this discussion, the main goal is to be able to calculate and recalculate a flowfield as quickly as possible within a known level of accuracy. To accomplish this, AMoEBA will be used to segregate the flowfield and place different turbulence modeling techniques as well as other solution methods in the sub-regions. Accuracy is judged based on the abilities of the solver networks to match the outlet velocity profile to an exit profile from a single model solution. Speed is a major priority in the algorithm because a CFD data set must be calculated for every structure in the population. Grid studies have been performed to determine the coarsest grid that can be used while still retaining the level of accuracy necessary to make the following assumption:

> If a coarse grid with a known level of accuracy is used to determine the proper segregation scheme and solver placement, the same scheme and placement will correspond to a different known level of accuracy when higher fidelity models are used.

For example, if the coarse grid solver placement scheme is within 3% of the coarse grid model used in the comparisons, then the same placement scheme used in a fine grid may be within 1% of the fine grid model used in the comparison. It may also fall to within 5% of the fine grid model, but as long as a correlation that gives the engineer a bound on the maximum amount of error in the fine grid solution is made, the segregation scheme will be successful. A justification of this assumption based on the AMoEBA results will be provided.

It is important to clarify the goal of AMoEBA. Evolutionary techniques often sacrifice speed for robustness. In this case, AMoEBA is a slow algorithm because many direct solutions of the candidates must be made. The long-term goal is for the algorithm to develop a segregation scheme that can be used in a virtual engineering design tool. This tool will allow the engineer to make changes to a design and witness the impacts of these design changes. For this system, the analyst will set up AMoEBA using an initial model of the design. AMoEBA segregation schemes will be evolved until a suitable scheme is found. In this case, there will be no time savings for the analyst due to the time that must be spent in the original design analysis and AMoEBA evolution. Once an acceptable scheme is found, the engineer can visualize the solution to the design, make changes, and recalculate the flow based on the segregation scheme. In this stage, the engineer will experience the

increased speed that the results of AMoEBA provide. The segregation schemes found by AMoEBA create an enabling tool for the engineer, a tool that will be the foundation of a virtual engineering design environment.

### *3.5.1. AMoEBA Description*

Significant changes were made to the AMoEBA algorithm to incorporate the CFD solvers. As with the two previous implementations of AMoEBA, both the fitness function and the placement of the solvers were adapted. Instead of basing the segregation on one variable, as has been done in the other applications, the CFD version of AMoEBA must be dependent on six different variables: radial velocity, angular velocity, axial velocity, pressure, turbulent kinetic energy, and turbulence dissipation rate. The types of problems to be solved in this case are flows through various elbow sections of pipe. Details of the case to be segregated will be discussed later. The goal in the pipe flow problem is to minimize the fluctuations in the exit velocity profile. This is a precursor to the larger goal of designing pipe systems that minimize the uneven flow distributions that occur due to elbows. The goal of this implementation is to use AMoEBA to determine the best segregation scheme that can reproduce the exit profile accurately in a shorter period of time than using a single model. In this case, the accuracy throughout the entire pipe is of less concern than the accuracy at the exit. For this reason, the fitness function is based on the results of the outlet velocity profile of the candidate solution.

3.5.1.1. Fitness Function

It is important to establish a feel for the accuracy of each solver network and to be able to compare this accuracy to known conditions. To this end, the fitness function is calculated, not by the sum of the squared error as in previous cases, but by a percentage error. For this calculation, the absolute value of the difference in the axial velocity between the candidate and exact solutions is first weighted by the fractional area of the cell, then summed over each cell in the cross-sectional area of the exit. The resulting value is normalized by the average axial velocity through the pipe to yield an area-weighted per cell average of the error in the candidate solution (Eq. 3.30).

$$Error = \frac{\sum_{i=0}^{N} \frac{A_i |u_i - u_i'|}{A_{tot}}}{\overline{u}} \qquad [3.30]$$

where $A_i$ is the area of cell $i$, $A_{tot}$ is the total area of the exit plane, $u_i$ is the axial velocity in cell $i$ of the single model "exact" solution, $u_i'$ is the axial velocity in cell $i$ of the candidate case, and $\overline{u}$ is the average velocity through the pipe. For the extreme circumstance of zero velocity in the candidate solution, the summation term is the average velocity through the pipe resulting in an error of 100%.

Since time is also important in the solution of a fast and accurate segregation scheme, a secondary fitness is also used. During each calculation of the candidate solver networks, the time spent on each iteration of each solver is summed to give the total time spent on that structure. This does not include the time spent on creating the grid and transferring the information from the solvers to the boundaries. These time costs are overhead for having

more than one solver and are excluded from the total time for each network. Without excluding these time costs, the expense of adding segregations outweighs any time savings of using more than one solver for the grid sizes used in AMoEBA. The reason for this is that for any evolutionary algorithm to work efficiently, the solution of the structures must be very fast. Since 500 generations for a population size of 64 is not uncommon, and since for each generation half of the population must be solved, a solution time of one minute per structure would take 11 days for the structures to be calculated alone. A grid study was performed to determine the speed of different sizes of grids for the pipe flow. The maximum size studied was 572,000 cells, and this case took over 39 hours to converge. The minimum size, the size used in AMoEBA, was 1800 cells, which only took 15.47 seconds to converge. At the maximum grid size level, having multiple solvers reduces the time to solution significantly because the overhead of creating and meshing the geometry is of a much lower order than the time spent on the solution of the flowfield itself. (As with the grid study, this has been analyzed and a broader discussion of the results of both of these studies will be presented later.) However, when coarser grid sizes are used, the overhead of creating and meshing the geometry is of the same order as the flowfield calculations and, because of this, the speedup is lost at these grid sizes unless the overhead is removed from the total time.

Implementing two different fitness functions into the same algorithm can be challenging. The difficulty lies in ensuring that both fitness functions are influencing the solution to the degree necessary. The accuracy of the CFD solutions is important to a certain point, but afterwards it is only important to have an understanding of the accuracy and to be sure that

the structures are within a certain level of accuracy. For this reason, an initial limit can be used for this fitness criterion. As two structures' data sets are compared, they are first compared on their accuracy to the "exact", single model data set. If both are within a defined accuracy range, 5% for example, they are then judged on their times to solution. The algorithm continues until a structure is found that is both within the accuracy criterion and has been solved within a set time limit. In this application of AMoEBA, 5% was chosen arbitrarily as a reasonable level of error for the accuracy fitness criterion for the structures.

Transmission of information throughout the pipe is another important aspect of the segregated domain. In flow through a pipe, information travels downstream due to convection and only travels upstream through diffusion. The large difference in the time scales of these two effects allows the assumption that downstream changes will have only a slight effect on the flowfield upstream. This means that changes made to an elbow will only slightly affect the flowfield before this elbow but can result in significant changes after the elbow. If the pipe is segregated, only those segregations that contain the altered elbow and are downstream of the elbow need to be recalculated. If the flowfield continues far enough downstream of the elbow, the later segregations may not change either since the flow will return to the fully-developed stage after a certain distance. For this reason, having more segregations is useful since it potentially reduces the amount of the flowfield that must be recalculated after changes are made. Influencing AMoEBA to evolve to more segregations is an added challenge that has been met by rewarding those candidate solutions that have more regions. Trees are only rewarded when they contain four or more segregations. Both the timing fitness and the accuracy fitness values are rewarded by dividing each fitness

value by a constant reward of 4.5. This value was chosen based on preliminary studies of the algorithm. The rewarded fitness values are only used when comparisons are made to other structures. In determining the convergence of the algorithm, unrewarded fitness values are used. This ensures that a segregation scheme is found that meets the convergence criteria. If the rewarded fitness values and times are used for determining convergence, the actual accuracy and time to solution of the structure is not known.

3.5.1.2. CFD Solvers

The solution methods used in the previous two implementations of AMoEBA were written specifically for the application and could be adjusted as necessary to work better with the format in which the AMoEBA code is written. In this application, commercial CFD software is used for almost all of the solution techniques. Therefore, the AMoEBA algorithm is adapted to work better with the commercial software. Star-CD™ is the CFD software used in this application due to its ability to be run in a non-graphical mode from a script file listing the commands to be used in the application.

A C++ code was created to manually study the interactions between two different CFD solution methods through a common boundary. This code is referred as the "manual multi-solver." This code is used for a number of the preliminary studies, such as the grid study and the solver timing study to be addressed later. The manual multi-solver automatically generates a three-dimensional cylindrical entrance region of pipe followed by a predefined toroidal elbow section of pipe with a straight exit section. The model is segregated into two different regions: an entrance region and the elbow and exit region (Fig. 3.27). The two

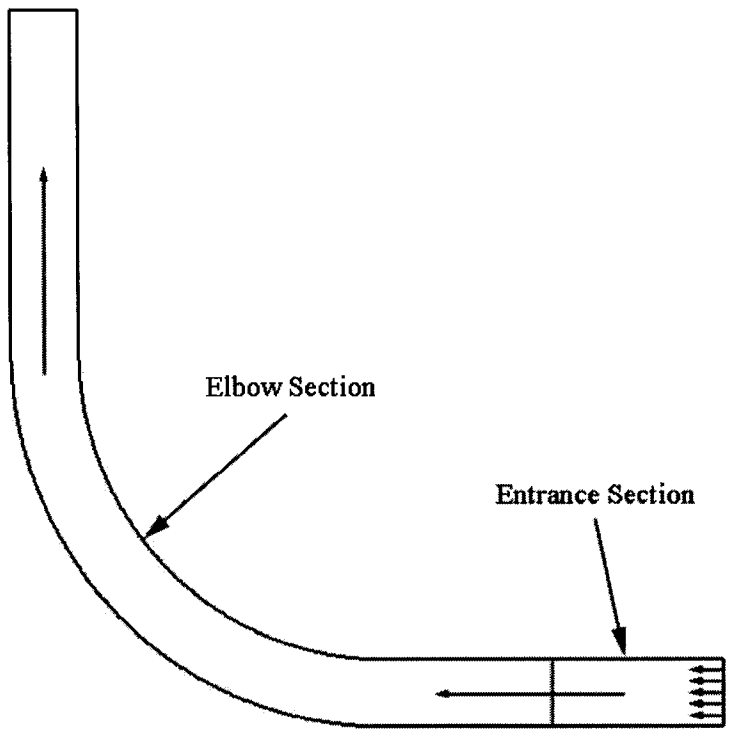



**Fig. 3.27. The geometry for the manually segregated elbow used in the manual multi-solver.**

different sections are then calculated using the same solution technique or two different solution methods. These are calculated according to the following:

- The application first calls Star-CD™ to generate the entrance section to the pipe. The geometry is created and meshed and the boundary faces are defined. The boundary conditions are applied based on the initial inlet and outlet conditions. For the outlet boundary of the first solver, a pressure outlet is used so that the pressure is the mode of information transfer from the downstream solver to the upstream solver. The inlet to the first solver does not change throughout the algorithm and is the same as the inlet to the "exact" single model solution. The boundary conditions are applied differently than for most CFD calculations. Since the boundary conditions between the two solvers are dynamic, they are defined by a user file that can be changed each time the CFD solver is called. This boundary condition user file is created by the manual multi-solver to automate the coupling of these regions. This user file is opened and read by Star-CD™ before each iteration.

  A second file, the data file, is written by the CFD solver and read by the manual multi-solver so that the boundary conditions can be recalculated and the boundary condition file can be updated. Once the boundary conditions are defined, the turbulence model, the number of iterations, the molecular properties of the flow (viscosity, density, specific heat, and the thermal conductivity), the under-relaxation factors, and the convergence criteria are chosen. Finally, an option is selected by the manual multi-solver. This option allows the Star-CD™ model to output the data file after each iteration for each cell defined in the model. This data file is written

according to an output user file. Like the boundary condition user file, this output user file is also created by the manual multi-solver. The data file written by Star-CD™ according to the output user file is read by the manual multi-solver and is used to calculate the new boundary conditions to be used to define the boundaries in the boundary condition file. The case is then saved, compiled, and run for the set number of iterations.

- The manual multi-solver creates the second solver model in exactly the same way as it did the first solver model. Boundary conditions begin as the global boundary conditions: initially, the inlet and exit conditions are the same as the "exact", single model solution. In this case, the second solver is the exit to the entire model; therefore, its exit condition is always the same as the global exit condition. The exit to the "exact" solution is an outlet boundary condition instead of a pressure boundary. Outlet boundaries differ from pressure outlets in that no information is given for the flow beyond the extent of the domain. In pressure outlets, the pressure value assigned tells the CFD software the pressure of the flowfield that lies outside of the calculated domain.

The overall solution process of the two model system is as follows. The entrance is calculated and provides entrance boundary conditions to the second region. The second region is calculated and then provides the outlet pressure boundary conditions for the first region. This occurs by updating the second solver's data file: CellFile2.dat. The boundary conditions are defined at the faces of the last layer of cells in that particular model. A boundary is defined at each cell face, and each boundary is calculated by taking the average

value of each of the six variables at the center of the last cell of the first solver and the first cell of the second solver models. To improve on the accuracy, this value could be weighted to emphasize the first or second model, but since the goal is to be accurate for a variety of different conditions, the average value is left neutral. These new boundary conditions are written to the first solver's boundary condition file to be read by the Star-CD™ solver. The first model is then calculated again using the new boundary conditions read from the boundary condition file, and the first model's cell file, CellFile1.dat, is updated. The boundary conditions for the second model are then calculated in the same manner as they were for the first model using CellFile1.dat instead of CellFile2.dat. The updated boundary conditions are written to the boundary condition file and the second model is recalculated for the prescribed number of iterations. After each solver is calculated, the residual level is checked for each of the six variables to determine if that solver has converged or not. Convergence of the complete network is determined not only by the individual solvers' convergence, but also on the convergence of the boundary conditions. The maximum difference between the current boundary conditions and the updated boundary conditions is calculated, and once this maximum value falls below a given criterion, 0.001 for the cases studied, and each individual solver has converged, the entire system is complete.

$$\underset{\substack{1\le i\le m\\ 1\le j\le n}}{Max}\left|s_{i,j}^{0}-s_{i,j}^{1}\right|<0.001 \qquad [3.31]$$

where $s_{i,j}^{0}$ is the current boundary condition (three different velocities, pressure, turbulent kinetic energy, and turbulence dissipation rate) of the face located at position $i,j$, $s_{i,j}^{1}$ is the updated boundary condition, $m$ is the number of faces in the radial dimension and $n$ is the number of faces in the angular dimension.

A variety of solution methods are provided for AMoEBA to assign to each partition. These include a high Reynolds number k-ε solver; the k-ε RNG solver; and two constant turbulent viscosity solvers, one with a turbulent viscosity of 0.02 and the other with the value 0.00125. These solvers were chosen based on the preliminary solver study to be discussed later. Each of these solvers is used with the Star-CD™ software and requires geometry to be created and meshed each time that model is used in a solver network. A final solution technique is also used as an option and does not require geometry to be created. This technique simply transfers the velocity profile and the values of the three other variables at the inlet to this region to the outlet of the segregation without changing their values. Although this solver is the least accurate of the solvers used, it is very quick, and, when used sparingly, it is able to reduce the time to solution for a network of solvers without a large loss in accuracy.

3.5.1.3. Other Adaptations to AMoEBA

The domains used in the other implementations of AMoEBA were each 200×200 data sets normalized to the range [0, 1] in each dimension. In this application, an elbow section of pipe will be analyzed. Instead of adapting AMoEBA, the geometry of the pipe along its length, excluding the elbow section, was mapped to the range [0, 1]. The elbow was not included because it was assumed that segregations placed in the elbow itself would not be helpful. Not only is the pipe geometry transformed to this normalized range, but the AMoEBA segregations are discretized to force the partitions to be applied at predetermined intervals so that every cell in each model matches up with the grid used in the "exact" case.

The original form of AMoEBA places segregations according to floating point numbers and not on an integer basis. To force the segregations to be placed between cells, this discretization scheme was implemented after the trees were created instead of altering the structure of the segregation trees themselves. Also, for this implementation, segregations are made in the axial direction only.

Another AMoEBA function that has been adapted for the CFD version is the prune function. In its original form, this function is used as a lower limit on the size of the segregations made. The function checks to see if the right and left subtrees of a node are larger than a certain value. If the left or right regions are too small, they are turned into leaves instead of subtrees. Since it is important that the regions created by the segregation trees are larger than at least one cell size, this function has been revised. The original form still placed a solver in the region that was determined to be too small. The new function looks at the left and right regions of both the left and right subtrees of a node. If either of these second level regions is too small, then the entire left or right subtree becomes a new random leaf. This operation is successful in preventing regions that are smaller than a cell size from being created.

### *3.5.2. Problem Description*

This application of AMoEBA focuses on fluid flow through a 90° elbow section of a cylindrical pipe. The diameter of the pipe is 0.5 m, and there are inlet and outlet sections of straight pipe, each 5.0 m in length. The radius of curvature of the elbow is 5.0 m, creating a total centerline length of 17.85 m (5 m + 7.85 m + 5 m) in the axial direction. Fig. 3.28

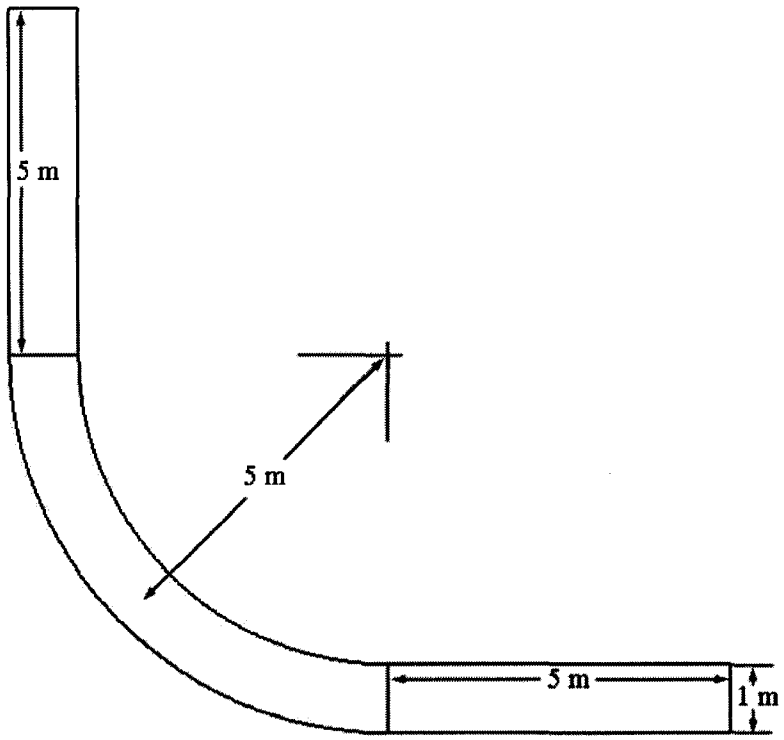



**Fig. 3.28. The geometry of the elbow used in AMoEBA.**

illustrates the pipe that is used in this implementation of the AMoEBA algorithm. The inlet velocity for this problem is a uniform flow of 1.0 m/s. The dynamic viscosity of the fluid is $2.5\times10^{-4}$ N·s/m$^2$, and the density used is 1.0 kg/m$^3$, resulting in a kinematic viscosity of $2.5\times10^{-4}$ m$^2$/s.

Before segregating this problem with AMoEBA, two preliminary studies were performed. The first of these was to determine which solution methods are fastest for this problem and whether multiple solvers can compete with single solution methods in both time and accuracy. The second study focused on the grid size of the domain. AMoEBA requires that a very coarse grid be used to generate the many candidate solutions necessary for the evolutionary algorithm in a reasonable amount of time. However, if the solutions based off of the coarse grids are not also reasonably accurate at finer levels, the segregation schemes found will have little value to the engineer.

The results of the grid study were used to identify the appropriate grid size for use with AMoEBA. Using this grid size, eight segregation schemes were found using AMoEBA. These segregation schemes were then tested using the following four studies:

- A grid study to determine the segregation schemes' ability to transfer from the coarse grid on which they were originally created to three different levels of finer grids,
- A geometry study to determine whether the segregation schemes retain their usefulness on different elbow angles,

- A similar study using different inlet velocities to determine the usefulness of the schemes under different flow characteristics, and
- A study to test the assumption that information travel in the upstream direction is negligible. To test this assumption, the flow is obstructed in the latter part of the elbow. Using a segregation scheme, only the regions in which the obstruction is located and those regions downstream of the obstruction are calculated. The time and accuracy of these solutions are compared to the "exact" solution.

For each of these studies, the segregated schemes were compared to "exact" single k-ε RNG model solutions to illustrate the accuracy of the scheme and to make a comparison of their times to solution.

*3.5.3. Results of Preliminary Studies*

Two of the studies performed utilize the manual multi-solver that divides a CFD data set into two pieces and applies the same CFD solver technique to each model or places two different CFD solvers in the two regions. Because the segregations are predetermined, comparisons can be made between the different models in the segregations with little to no influence from the segregation scheme itself. This preliminary study was used to ensure the effectiveness of the internal boundary conditions, to analyze the speed of the different solvers when used in a divided grid, and to check the code used to call the commercial CFD software package. As in the AMoEBA algorithm, this manual multi-solver is able to define the geometry, place a mesh on the completed geometry, define boundary conditions for the model, and compile and run the completed case, all using the commercial code.

Two studies were performed using the manual multi-solver. As discussed earlier, they are:

- A timing study to determine the speed of the different solvers when a single solver is used for the entire grid, when a single solver is used on a divided grid, and when two different solvers are combined into one solution.
- A grid study to determine the correlation between grid size, speed, and accuracy.

The details of how each study was performed as well as the results of each study follow. All geometry has been reduced by placing a symmetry boundary along the centerplane of the pipe. This simplification allows the algorithm to use half as many cells with the same level of accuracy. The information learned from these studies is used in the AMoEBA algorithm for the solution of the elbow pipe problem.

3.5.3.1. Solver Timing Studies

The time to solution for a variety of different solution methods was measured to determine which solution methods were faster, whether a divided solution is faster than a complete solution, and to discover if a combination of solvers would outperform a single solver. The solvers used in this study were:

- a high Reynolds number k-ε turbulence model (complete and split cases: a complete case refers to using this solution method for the entire model, a split case refers to dividing the elbow model into two pieces—an entrance section and the elbow itself—and using the same solver for both regions);
- a low Reynolds number k-ε turbulence model (complete and split cases);
- a k-ε RNG turbulence model (complete and split cases);

- a combination of the low Reynolds number k-ε solver in the entrance section and the k-ε RNG solver in the elbow section, Combo-1;
- a combination of the k-ε RNG solver in the entrance section and the low Reynolds number k-ε solver in the elbow section, Combo-2;
- a combination of the high Reynolds number k-ε solver in the entrance section and the k-ε RNG solver in the elbow section, Combo-3;
- a combination of the k-ε RNG solver in the entrance section and the high Reynolds number k-ε solver in the elbow section, Combo-4;
- a combination model using two different constant turbulent viscosity values from the complete low Reynolds number k-ε turbulence model, Const-1—the turbulent viscosity in the entrance region of this case is approximately 0.025 and approximately 0.002 in the rest of the domain; and
- a combination model using two different constant turbulent viscosity values from the complete k-ε RNG turbulence model case, Const-2—the turbulent viscosity in the entrance region of this case is approximately 0.02 and approximately 0.00125 in the rest of the domain.

The complete versions of these cases were calculated on individual nodes of an 18-node Linux Networxs dual 1 GHz and 1.2 GHz Athalon processor cluster. The creation and meshing of the geometry for the complete cases were not included in the time to solution. The split versions of the above cases were also calculated on the same machines and did include the cost of creating and meshing the individual models. Each case was run with the same number of cells—572,000 cells (40 cells in the radial dimension, 10 cells in the

angular dimension, and 1430 cells in the axial dimension) in one study and 72,000 cells (20 cells, 5 cells, and 720 cells) in the second study—and three trials were made of each case. For the 572,000-cell cases, the Linux cluster was devoted solely to this study. This was not the case for the 72,000-cell cases. However, for this part of the study, the same loading on the machine was maintained for each solver studied. Similarly, every study performed except the 572,000-cell solver study was run with the machine on this loaded condition so that comparisons of runtimes would be valid. There are two reasons for using times from the loaded machine in the studies. The first is that in real-world engineering situations, it would be difficult to devote an entire machine to one application. The second is that because the studies were performed in a large research organization, devoting an entire machine to a study could only be done for the 572,000-cell cases.

Fig. 3.29 illustrates the time to solution and the accuracy of the 12 different solvers studied. Every solver was within 5% error of the complete k-ε RNG solver except for the constant turbulent viscosity solvers, the complete and split forms of the low Reynolds number solver, and the Combo-2 case, which uses the low Reynolds number solver for the majority of the pipe. As is expected, the least complicated solution method—the two constant turbulent viscosity cases—were completed in the least amount of time: 25:58 average for the case using the constants from the low Reynolds number k-ε solver results and 25:46 average for the constants from the k-ε RNG results. The next two cases to finish were the split versions of the k-ε RNG turbulence model (31:56 average) and the high Reynolds number k-ε solver (32:36 average). On average, these cases finished 23.9% and 19.82% faster than their complete counterparts, respectively. The combination cases using these two

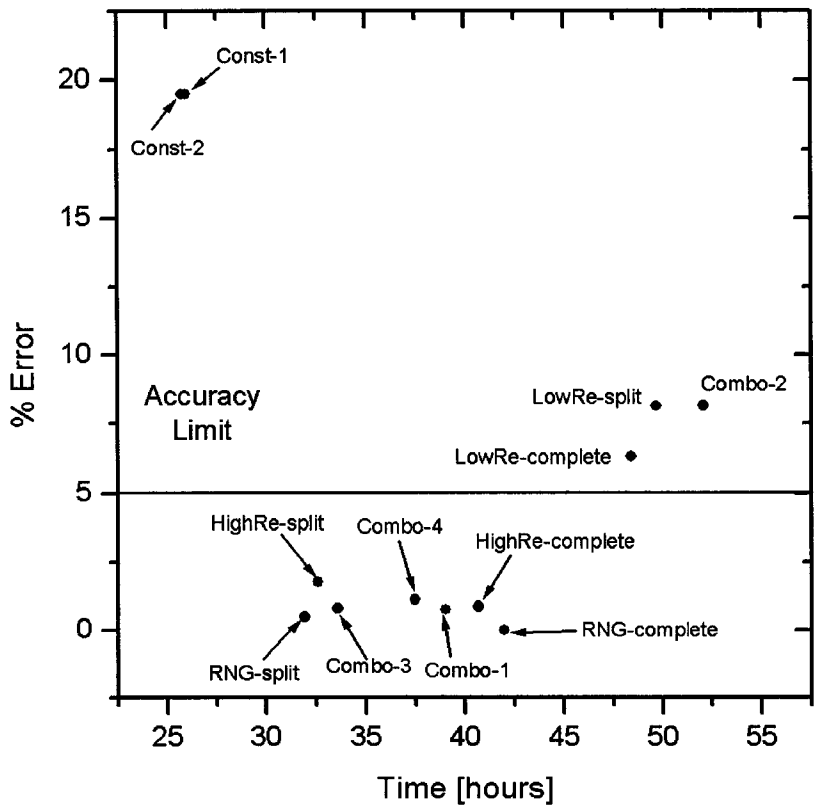



**Fig. 3.29. A comparison of the time to solution and accuracy of the 12 different solutions methods in the 572,000-cell timing study.**

solvers also finished relatively quickly. Combo-3, the case using the k-ε RNG model in the elbow region and the high Reynolds number model in the entrance region, required an average time of 33:36 to reach convergence. Combo-4, the opposite placement of these solvers, required slightly more time on average (37:28). The times to solution for each case can be seen in tabular form in Table 3.5.

Also included in Table 3.5 are the accuracy values for each different solution scheme. These errors are calculated using the same error function that AMoEBA uses as its accuracy fitness function (Eq. 3.30). In this case, the accuracy is determined by using the complete k-ε RNG model as the "exact" solution. Included in the table is the accuracy that would be achieved if the inlet conditions were transmitted, unaltered, to the exit. This is included to provide perspective for the errors of the different solvers. For example, when compared with the even profile, the two different constant solver combinations (19.5% error each) were only slightly more accurate than the evenly distributed result (23.9% error). This can be compared with the next best solvers, the Combo-2 case, that had an error of 8.14% and the split version of the low Reynolds number solver at 8.11%. The most accurate solver was the split version of the k-ε RNG solver (0.494%), followed closely by the two different combination cases that included the k-ε RNG solver in the elbow section: Combo-1 (0.773%) and Combo-3 (0.820%). Of the cases that did not include the k-ε RNG solver, the high Reynolds number complete and split cases were the most accurate at 0.863% and 1.79%, respectively. It is no surprise that the low Reynolds number and the constant turbulent viscosity solvers performed poorly. The former solver is best used for slower flows than the 1.0 m/s used in this problem. The constant turbulent viscosity solver is an

**Table 3.5. Results of the three timing trials, the average time to solution, and the area-weighted error method used in AMoEBA for the 12 different solvers studied using the 572,000 cell grid.**

| Solver | Trial 1 | Trial 2 | Trial 3 | Average Time | % Error |
|---|---|---|---|---|---|
| Complete k-ε RNG | 39:27 | 44:04 | 42:22 | 41:58 | - |
| Split k-ε RNG | 29:07 | 30:22 | 36:19 | 31:56 | 0.49 |
| Complete High Re k-ε | 41:28 | 40:41 | 39:48 | 40:39 | 0.86 |
| Split High Re k-ε | 34:09 | 33:11 | 30:27 | 32:36 | 1.79 |
| Complete Low Re k-ε | 44:30 | 51:22 | 49:18 | 48:23 | 6.30 |
| Split Low Re k-ε | 43:54 | 54:05 | 51:00 | 49:40 | 8.12 |
| Combo-1 | 38:58 | 39:22 | 38:42 | 39:01 | 0.77 |
| Combo-2 | 52:13 | 55:26 | 48:36 | 52:05 | 8.14 |
| Combo-3 | 30:06 | 33:44 | 36:57 | 33:36 | 0.82 |
| Combo-4 | 37:24 | 37:40 | 37:21 | 37:28 | 1.13 |
| Const-1 | 27:10 | 25:37 | 25:08 | 25:58 | 19.5 |
| Const-2 | 26:51 | 25:22 | 25:06 | 25:46 | 19.5 |
| Even Distribution | N/A | N/A | N/A | N/A | 23.9 |

approximation to the flow and was expected to perform as it did, with low accuracy but a faster solution.

The same timing study was also performed at a smaller grid size to determine if the speedup found on the fine grid would also exist at a more reasonable grid size. This grid size uses 20 cells in the radial dimension, five in the angular dimension, and 720 in the axial dimension, totaling 72,000 cells for the entire model. Fig. 3.30 illustrates the comparison of error versus time for the 12 solver schemes studied for this grid. Also, Table 3.6 lists each solver with their respective times to solution, the errors from the complete k-ε RNG case, and the error associated with an evenly distributed velocity profile. The accuracies of the different solvers are very similar to those found on the finer grid. Those solvers that were within 5% of the complete k-ε RNG model on the fine grid are also within 5% of the complete k-ε RNG model on this coarser grid. The main difference with the solvers on the 72,000-cell grid is in the time to solution. As mentioned, as the grid size decreases, the overhead time required to create and mesh the geometry associated with split models increases with respect to the time required to solve the models. Because of this effect, the cases using more than one solver and those single solver cases using a split grid require more time to solve than the complete cases.

3.5.3.2. Grid Study

A grid study in which a number of variables were examined was performed to determine the size of the grid to be used with AMoEBA. The most important of these studies was to determine the minimum number of nodes that could be used while still maintaining the

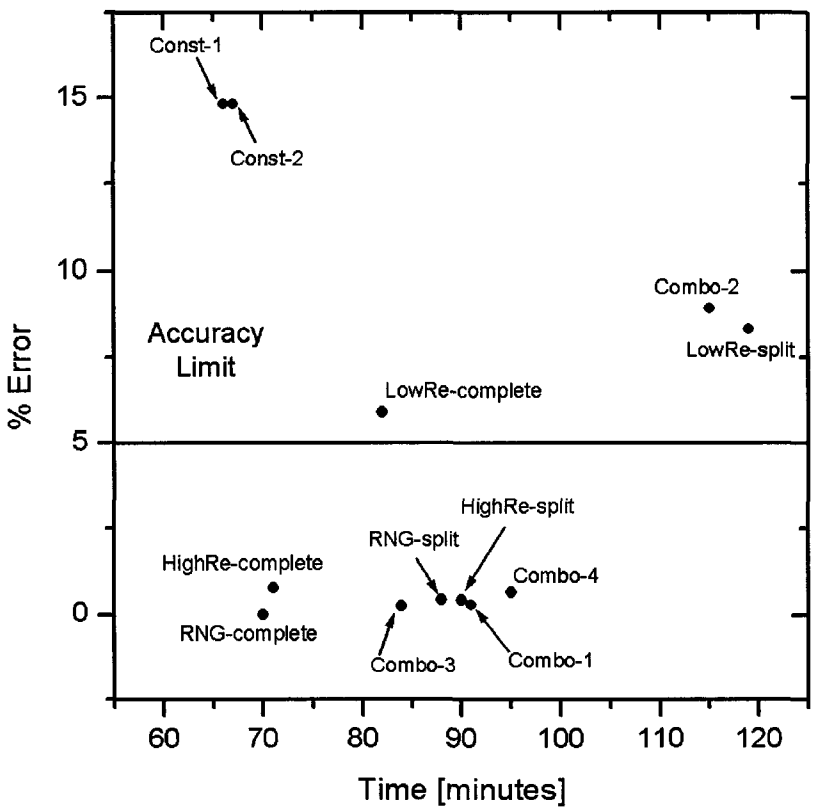



**Fig. 3.30. A comparison of the time to solution and accuracy of the 12 different solutions methods in the 72,000-cell timing study.**

**Table 3.6. Results of the three timing trials, the average time to solution, and the area-weighted error method used in AMoEBA for the 12 different solvers studied using the 72,000 cell grid.**

| Solver | Trial 1 | Trial 2 | Trial 3 | Average Time | % Error |
|---|---|---|---|---|---|
| Complete k-ε RNG | 1:13 | 1:07 | 1:09 | 1:10 | - |
| Split k-ε RNG | 1:29 | 1:25 | 1:29 | 1:28 | 0.44 |
| Complete High Re k-ε | 1:16 | 1:07 | 1:10 | 1:11 | 0.79 |
| Split High Re k-ε | 1:31 | 1:27 | 1:31 | 1:30 | 0.42 |
| Complete Low Re k-ε | 1:27 | 1:16 | 1:23 | 1:22 | 5.90 |
| Split Low Re k-ε | 2:02 | 1:53 | 2:01 | 1:59 | 8.30 |
| Combo-1 | 1:34 | 1:29 | 1:31 | 1:31 | 0.29 |
| Combo-2 | 2:03 | 1:47 | 1:56 | 1:55 | 8.91 |
| Combo-3 | 1:27 | 1:17 | 1:27 | 1:24 | 0.24 |
| Combo-4 | 1:33 | 1:28 | 1:43 | 1:35 | 0.67 |
| Const-1 | 1:09 | 1:02 | 1:09 | 1:07 | 14.82 |
| Const-2 | 1:10 | 1:02 | 1:06 | 1:06 | 14.82 |
| Even Distribution | N/A | N/A | N/A | N/A | 22.30 |

integrity of the exit velocity results. This included determining which parameters are the most important—cells in the radial direction, angular direction, or axial direction—and which values are the most efficient for the defined application. Accuracy and time to solution are both very important in this study. If each solution cannot be found within approximately one minute and 30 seconds, AMoEBA will take far too long to make this project a reasonable application. On the other hand, if these coarse models are not accurate enough, they are also not useful, and the segregation schemes AMoEBA is able to find will be meaningless at the fine grid levels.

Each of the mesh sizes studied was solved using Star-CD™, and every calculation was performed on the same Linux cluster used in the timing study. For each case, Star-CD™ measures the clock time used for the solution as well as the processor time. Each of these is shown in the comparisons of the various grid sizes (Table 3.7). These times measured are for the calculation time of the model only; time spent on creating and meshing the geometry is not included in the study. The accuracy has been measured in a slightly different manner than that used in the accuracy fitness function. In the fitness function, the candidate solutions have the same grid size as the "exact" solution to which they are being compared. For this reason, each structure's solution can be compared on a cell-by-cell basis. In this study, each grid is compared to the case that contains the maximum number of cells: this case is assumed to be the most accurate of all the grid sizes in the study. Since the grid sizes being analyzed have fewer cells than this "exact" case, every cell of the coarse meshes will not necessarily match up with a cell from the "exact" case. Instead, an alternative accuracy

**Table 3.7. Complete data–time to solution and accuracy–for the grid study. Times marked with a * are from split models and the times for six of the grid sizes were unavailable.**

| Radial Cells | Angular Cells | Axial Cells | Total Cells | Time | % Error |
|---|---|---|---|---|---|
| 7 | 5 | 36 | 1,260 | 38 sec* | 2.83 |
| 5 | 5 | 54 | 1,350 | 27.61 sec | 4.91 |
| 8 | 5 | 36 | 1,440 | 47 sec* | 2.04 |
| 9 | 5 | 36 | 1,620 | 48 sec* | 1.37 |
| 10 | 5 | 36 | 1,800 | 50 sec* | 0.80 |
| 10 | 5 | 36 | 1,800 | 15.47 sec | 0.99 |
| 5 | 5 | 72 | 1,800 | 19.54 sec | 4.93 |
| 5 | 5 | 90 | 2,250 | 29.30 sec | 4.94 |
| 10 | 5 | 54 | 2,700 | 33.93 sec | 1.00 |
| 5 | 5 | 122 | 3,050 | 45.69 sec | 4.95 |
| 10 | 5 | 72 | 3,600 | 41.14 sec | 0.98 |
| 5 | 5 | 150 | 3,750 | 0:01:10.63 | 4.96 |
| 10 | 5 | 90 | 4,500 | 0:01:03.41 | 0.97 |
| 5 | 5 | 180 | 4,500 | 0:01:21:48 | 4.96 |
| 10 | 5 | 122 | 6,100 | 0:01:15.74 | 0.97 |
| 10 | 5 | 150 | 7,500 | 0:02:55 | 0.97 |
| 5 | 10 | 180 | 9,000 | 0:03:09 | 5.08 |
| 10 | 5 | 180 | 9,000 | 0:03:14 | 0.97 |
| 5 | 15 | 180 | 13,500 | - | 5.14 |
| 15 | 5 | 180 | 13,500 | - | 0.38 |
| 5 | 20 | 180 | 18,000 | - | 5.18 |
| 10 | 10 | 180 | 18,000 | 0:09:33 | 0.94 |
| 20 | 5 | 180 | 18,000 | 0:10:27 | 0.34 |
| 10 | 15 | 180 | 27,000 | 0:17:23 | 1.09 |
| 15 | 10 | 180 | 27,000 | - | 1.06 |
| 10 | 20 | 180 | 36,000 | 0:31:50 | 1.18 |
| 20 | 10 | 180 | 36,000 | 0:27:09 | 0.88 |
| 20 | 5 | 360 | 36,000 | 0:27:26 | 0.32 |
| 15 | 15 | 180 | 40,500 | - | 1.40 |
| 15 | 20 | 180 | 54,000 | - | 1.58 |
| 20 | 15 | 180 | 54,000 | 0:46:22 | 1.22 |
| 20 | 20 | 180 | 72,000 | 1:19:32 | 1.39 |
| 20 | 10 | 360 | 72,000 | 0:49:29 | 0.89 |
| 20 | 5 | 720 | 72,000 | 1:18:27 | 0.28 |
| 20 | 10 | 720 | 144,000 | 2:20:47 | 0.87 |
| 25 | 10 | 900 | 225,000 | 5:31:12 | 0.59 |
| 30 | 10 | 1080 | 324,000 | 10:39:33 | 0.34 |
| 40 | 10 | 1430 | 572,000 | 39:23:14 | - |

method has been used. This method compares the mass flow rate through different sectors of the outlet circular area. The exact calculation used for this error estimation is:

$$Error = \frac{1}{n} * \sum_{i=1}^{n} \left[ \frac{\left| \frac{\dot{m}_i}{\dot{m}_{tot}} - \frac{\dot{m}'_i}{\dot{m}'_{tot}} \right|}{\frac{\dot{m}_i}{\dot{m}_{tot}}} \right] \qquad [3.32]$$

where $n$ is the number of sectors used, $\dot{m}_i$ is the mass flow through the sector $i$ of the "exact" case, $\dot{m}'_i$ is the mass flow through the sector $i$ of the coarse grid case, $\dot{m}_{tot}$ is the total mass flow through the pipe (the flow through all $n$ sectors) for the "exact" case, and $\dot{m}'_{tot}$ is the total mass flow through the pipe for the coarse grid case. The "exact" case to which all other mesh sizes are compared for this study is the grid with 40 cells in the radial dimension, 10 cells in the angular dimension, and 1430 cells in the axial dimension, which yields 572,000 total cells. Since the coarsest mesh used had five sectors, $n = 5$ for each case in the study. For those cases with $n = 10$, 15, or 20, the flow through groups of two, three, and four sectors, respectively, were summed to calculate the error. These error calculations are shown in Table 3.7.

In flow through an elbow, the general trend is for the flow to be swept to the outside of the elbow due to the centrifugal acceleration. The future application of this project is to be able to design an elbow section of pipe that reduces the uneven exit distribution that occurs due to this effect. The industrial application of this is in pneumatic transportation of coal particles through a pipe. In two-phase flow such as this, the phenomenon of roping occurs. This roping is a result of the difference in momentum between the particles and the driving

fluid. As the mixture enters an elbow, this difference in momentum results in the approximately even distribution of particles becoming significantly uneven as the centrifugal acceleration forces the particles to the outside of the elbow. This uneven distribution can cause major problems in the flow rate of the solid if the mixture is divided shortly after the elbow. Similarly, in the case of coal transportation for a coal-fired furnace, the uneven distribution can result in poorer combustion if the roping continues into the burner region of the furnace. By studying the velocity profile of the exit plane of the pipe, changes can be made to the elbow to improve this distribution. With this goal in mind, comparing the exit profiles by calculating the difference in mass flow rates through different sectors of the exit plane is a reasonable accuracy measure. The main concern of the accuracy of the different mesh sizes is to determine how well each size will perform when comparing the exit distribution of the flow. By dividing the exit planes and comparing the mass flow rates through the different divisions, the different models are judged on how the flow is distributed through the five sectors, which is representative of the velocity distribution.

Each dimension of the mesh can have a different influence on the accuracy of the model. To study the effects of each dimension, 28 different grid sizes were tested, from the "exact" case of 572,000 cells to the coarsest mesh of 1350 cells. The number of cells in the radial dimension ranged from five to 40 cells; in the angular direction the range was five to 20 cells; and in the axial direction a variety of cell sizes were used from 36 to 1430 cells.

The number of cells in the axial direction had the least amount of influence on the accuracy of the model (Figs. 3.31 and 3.32). Figs. 3.31 and 3.32 illustrate the effect of changing axial cell numbers. Fig. 3.31 shows the accuracy and time to solution for two different radial sizes (five and ten radial cells) with five angular cells and six different axial cell sizes (54, 72, 90, 122, 150, and 180). Although there is a significant difference in the time to solution for the different numbers of cells, there is very little difference in the errors. For the five-radial-cell case, the accuracy only ranges between 4.90% and 4.96%. The error range is 0.970% to 0.995% for the ten-radial-cell case. Fig. 3.32 shows a similar graph of the accuracy and time to solution for two different angular cell sizes (five and ten angular cells) with 20 radial cells and three different axial cell sizes (180, 360, and 720). Although only three different axial cell sizes were calculated for these radial and angular cell sizes, the results are similar to those in Fig. 3.31. The five-angular-cell case had a range of errors between 0.283% and 0.343% and the ten-angular-cell case ranged between 0.872% and 0.893% error. In each of these four cases, the time to solution varied consistently with the total cell size of the grid. From these results it can be concluded that the number of cells in the axial direction has little effect on the accuracy of the model.

Due to the relatively small cross-sectional area of the pipe compared to its length, four different angular cell sizes were tested in the grid study. These were each tested with an axial cell size of 180 cells and radial cell sizes of five, ten, 15, and 20 cells. Five, ten, 15, and 20 cells were placed in the angular dimension for these cases as well. Only one axial cell size was used since it was found that the accuracy had very little dependence on the fidelity of the model in this dimension. Again, the time to solution for these cases varied

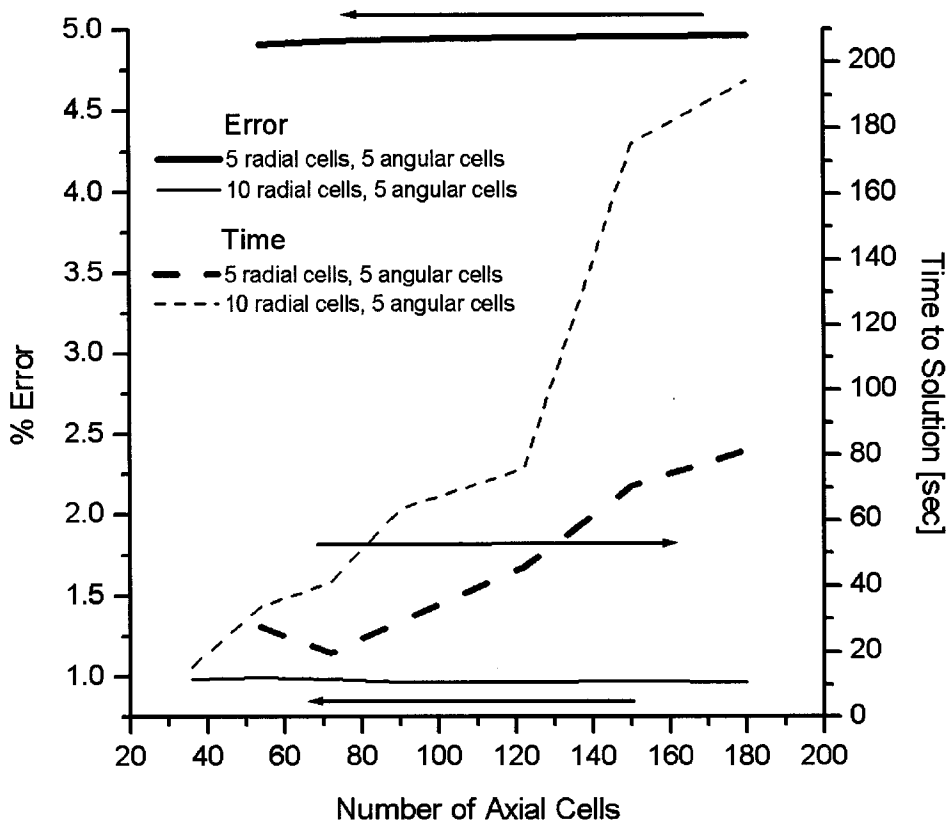



**Fig. 3.31. The effect of changing axial cell numbers based on five and ten radial cells and five angular cells for the grid study of the elbow.**

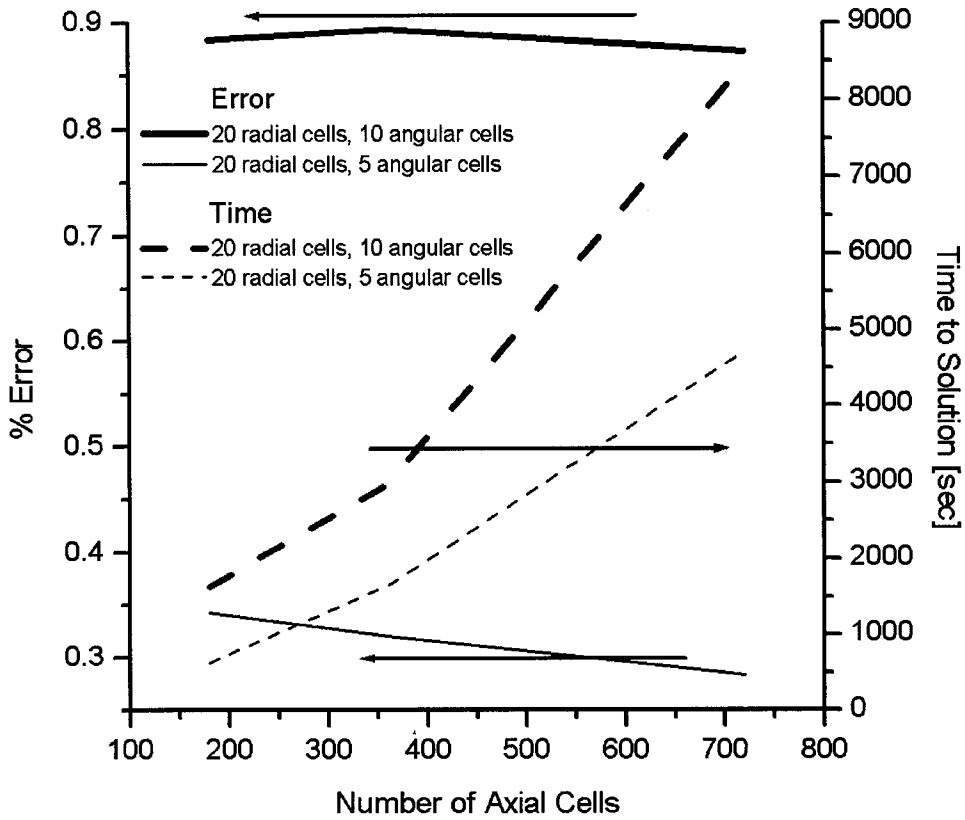



**Fig. 3.32. The effect of changing axial cell numbers based on 20 radial cells and five and ten angular cells for the grid study of the elbow.**

consistently with the total number of cells in the mesh. Fig. 3.33 shows the results of this part of the study. It is interesting to see that as the number of cells increases in the angular dimension, there is a notable decrease in the accuracy, i.e., an increase in the error. This is counterintuitive but can be explained by the fact that the "exact" case contains 40 cells in the radial dimension and ten cells in the angular dimension. This corresponds to a 4:1 aspect ratio. In the angular cell study, the mesh sizes with 20 radial cells and five angular cells (4:1 aspect ratio) outperformed the models with 20 radial cells and ten (2:1), 15 (4:3), and 20 (1:1) angular cells. Similarly, the case with 15 radial cells and five angular cells (3:1 aspect ratio) was more accurate than those models with ten (3:2), 15 (1:1), and 20 (3:4) angular cells. In this case, the aspect ratio is not the same as the exact case but it is closer to the 4:1 aspect ratio than any of the other cases. Since the accuracy is based on five sectors of the circular cross-sectional exit plane, the "aspect ratio" anomaly was not as noticeable in grid sizes with less than 15 radial cells.

Four different radial cell sizes were used in the study for this dimension. Radial cell sizes of five, ten, 15, and 20 cells were used on meshes with five, ten, 15, and 20 angular cells and 180 axial cells. Fig. 3.34 illustrates how the accuracy changes with respect to a change in the radial cell size. For the five-angular-cell case, the error decreases by a factor of 5.1 when the radial cell size changes from five radial cells to ten radial cells with a factor of 2.3 increase in time. A similar decrease factor of 5.4 is found in the ten-angular-cell case with a corresponding 3.0 factor time increase. Although speed is crucial in the AMoEBA algorithm, the accuracy gained by using ten cells in the radial dimension outweighs the expense of having more cells. A second study on the radial cell size was made in the range

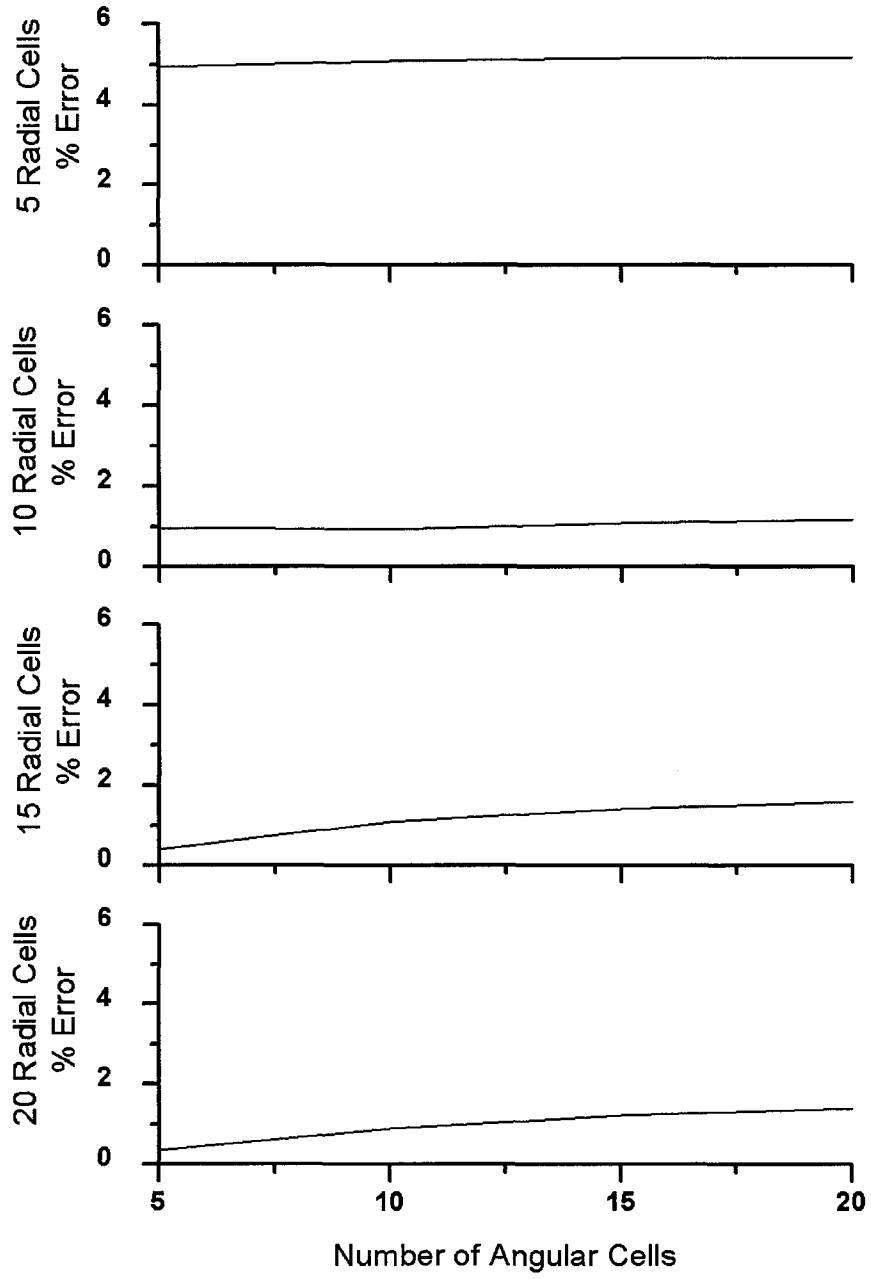



**Fig. 3.33. The results of different angular cell sizes for the elbow. For five and ten radial cells, the change in error is minimal and the error increases slightly for the 15 and 20 radial cell cases.**

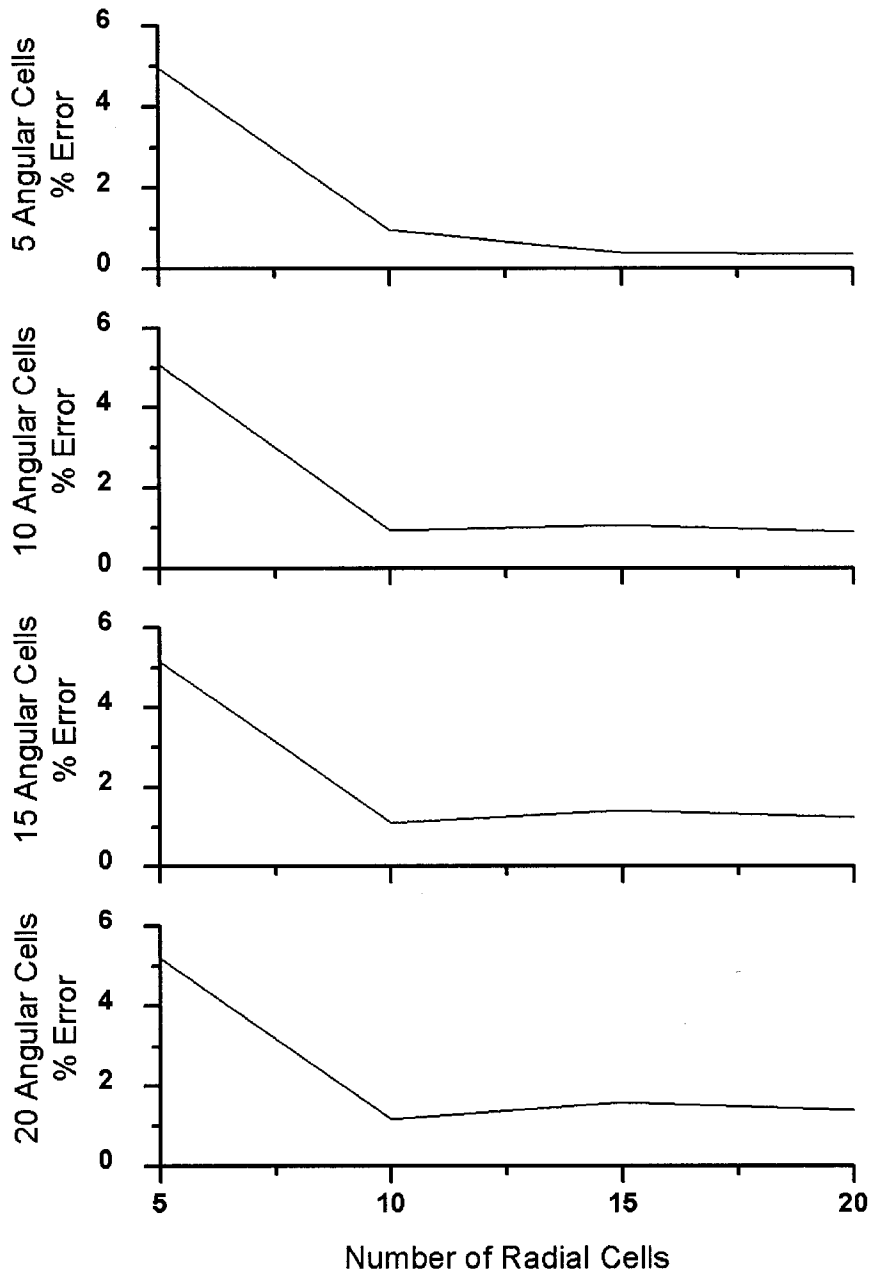



**Fig. 3.34. The results of different radial cell sizes for the elbow. In each case, a major change in error occurs when going from five to ten radial cells.**

of cell sizes used in the AMoEBA algorithm. Instead of using a single model, these cases were run on split models to analyze the added cost of generating and meshing two geometries. The mesh sizes in this study used a single axial cell size of 36 cells, a single angular cell size of five cells, and seven, eight, nine, and ten radial cell sizes. A much smoother error curve results (Fig. 3.35) in this study. However, based on the earlier study, only a slight decrease in error is expected beyond the ten radial cells used here. At this level, there is still a significant decrease in error from the seven-radial-cell case to the ten-radial-cell case (a factor of 3.6), whereas the increase in time is small (approximately 38 seconds to 50 seconds—a factor of 1.3 difference). Of the three different dimensions, the decrease in error between five and ten radial cells is by far the most significant change in error.

To study this change in error further and to test grid sizes that are fast enough to be used in AMoEBA, a number of smaller trials were made. Five and ten radial cells and five angular cells were used with five different axial cells sizes (54, 72, 90, 122, and 150). As with the previous cases, the time to solution varied with cell size, but the accuracy only varied between the five- and ten-radial-cell cases. The average error for the five cases with five radial cells was 4.94% with a standard deviation of 0.02%. In the ten-radial-cell case, the average error decreased significantly to 0.979% with a standard deviation of 0.011%.

The speed for the all the cases involved in the grid study held true to the expectations. As the number of cells increases in the pipe geometry, so does the time to solution. Changes in radial, angular, or axial cell size only affects the speed inasmuch as the total number of cells

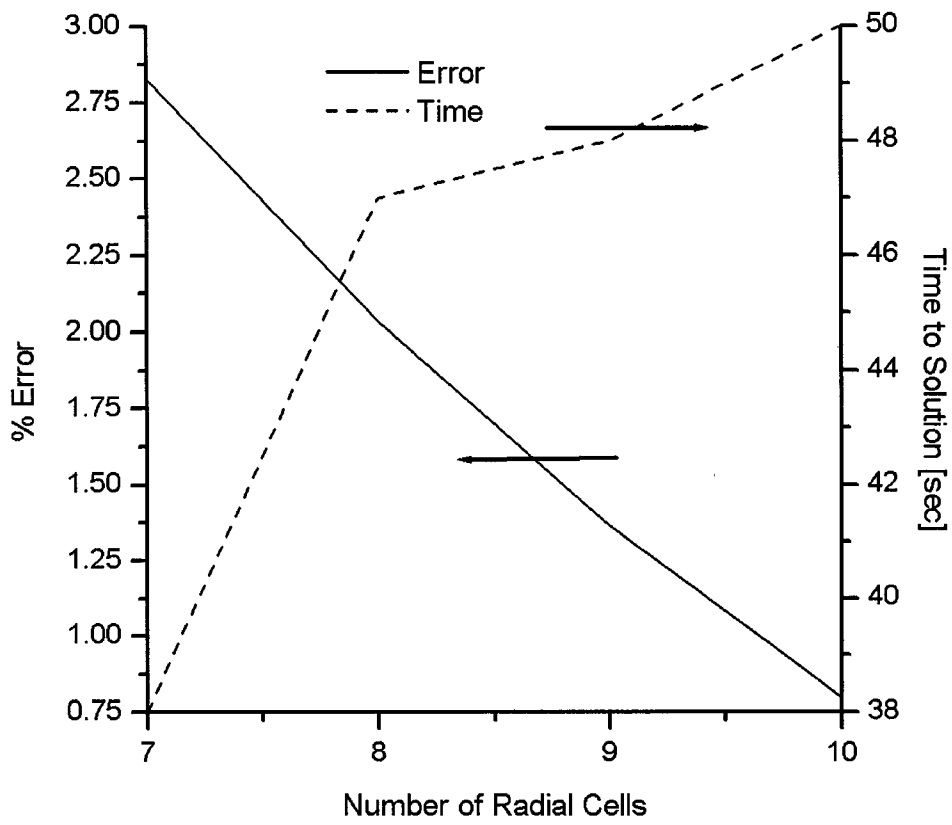



**Fig. 3.35. The results of the grid study using five angular cells, 36 axial cells and seven, eight, nine, and ten radial cells and using the k-ε RNG solver on two different models.**

affects the speed. Since the accuracy changed only slightly with respect to the axial dimension, the size of the mesh was reduced as much as possible in this direction. Due to the significant change going from five radial cells to ten cells, ten cells were chosen for the radial dimension. Based on the results of the accuracy and speed study of the various grid sizes, a 10×5×36 cell grid was chosen for the AMoEBA algorithm. The accuracy of this size of grid is within 0.986% of the "exact" grid and the time to solution for a single model on this size of grid was 15.47 seconds, well within the range of one minute and 30 seconds required by AMoEBA.

### *3.5.4. Results of AMoEBA for Flow Through an Elbow*

Ten segregation schemes for the 1800-node grid for the elbow flow problem were found using AMoEBA, each starting from different random populations of structures. Each case was run until the stopping criteria were reached. Because two fitness functions are used, one for time and one for accuracy, two convergence criteria were necessary. When a structure's accuracy fitness value falls below 5% and the time required to solve this segregation scheme is less than 100 seconds, convergence is obtained. A generation limit was also enforced. If a population had not found a structure that met the convergence criteria within 70 generations, it was determined that it would not find a solution in a reasonable amount of time. Cases 6 and 7 were each stopped after reaching the 70-generation limit. Table 3.8 lists each case, the fitness value (accuracy) of the best structure, the number of generations needed to reach the stopping criteria, and the times required to calculate the multi-solver CFD solutions. Case 1 was allowed to run beyond the stopping criteria to determine if any interesting structures could be found after reaching the 100-

**Table 3.8. Accuracy, number of generations required, and the time to solution for seven cases of AMoEBA. Case 1 was allowed to run after reaching convergence to further explore the search space. Cases 6 and 7 did not reach the convergence criteria of 5% error and 100 seconds within the limit of 70 generations.**

| Case | % Error | Generations | Time [seconds] |
|---|---|---|---|
| 1a | 1.31 | 18 | 89.1 |
| 1b | 1.68 | 50 | 79.2 |
| 2 | 0.67 | 39 | 81.9 |
| 3 | 0.82 | 37 | 99.5 |
| 4 | 1.06 | 34 | 71.9 |
| 5 | 1.04 | 9 | 72.5 |
| 6 | 5.41 | 70+ | 95.3 |
| 7 | 1.55 | 70+ | 240.5 |
| 8 | 0.91 | 10 | 96.2 |
| 9 | 1.29 | 3 | 82.8 |

second criterion. Thirty-two generations after finding a scheme that met the stopping criteria, a faster solution was found. This second scheme has also been included in the analysis.

Five different solver placement types were found using the eight converged cases. Four of these were found only once, and the fifth placement was found in the best structures of four of the eight cases. Fig. 3.36 illustrates the five types of segregation schemes found by each random population. The two cases that did not find a solution that met the stopping criteria each had found a two-solver scheme. This solver placement is included in Fig. 3.36. Fig. 3.37 lists the Lisp-like notation of the segregation trees for all ten cases. As shown in Figs. 3.36 and 3.37, every segregation scheme found by AMoEBA utilized the inlet-to-outlet mapping solver for the entrance to the elbow. This is a reasonable solution to the problem as it shows there is only a slight difference at the exit of the domain when the placement of the inlet is moved closer to the elbow itself. Five of the eight solutions found and both of the cases that were prematurely stopped placed the inlet within three cells of the elbow. Of the cases that had converged, this still resulted in less than 1.5% error of the single model solution. Another similarity in the solver placements is that none of the schemes found used either of the constant turbulent viscosity solvers. This is not surprising since these solvers obtained the poorest accuracy of all the commercial solvers. The speeds of the solvers are their greatest strength, but their accuracy is approximately the same order as the inlet-to-outlet solver. Compared to this solver, which does not need to create a grid and, therefore, is almost instantaneous, the constant turbulent viscosity solvers are slow.

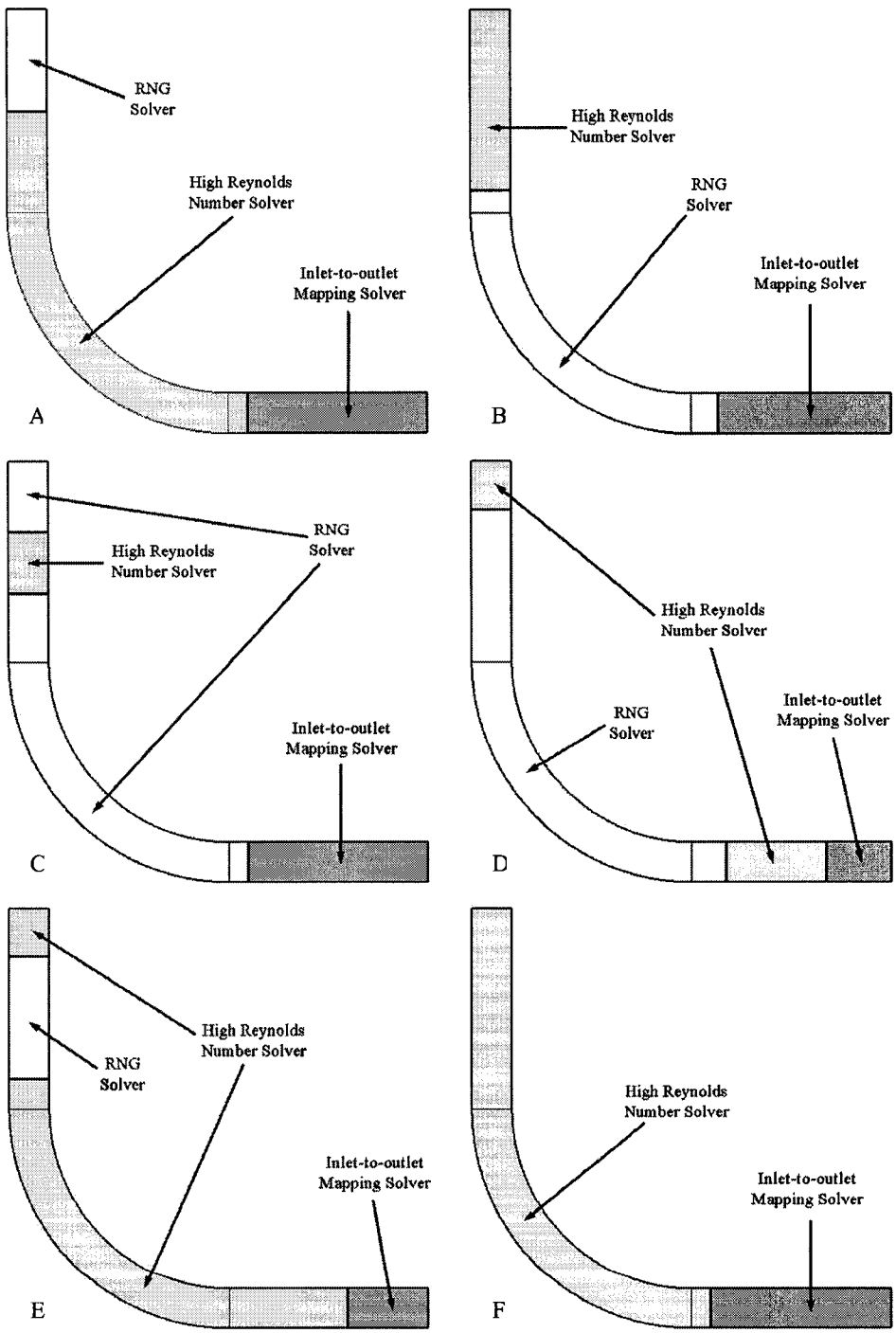



**Fig. 3.36. Examples of the five different types of segregation schemes found by the eight converged AMoEBA cases (A-E) and an example of the segregation scheme of the two unconverged cases (F).**

| Lisp-like format of each case | Segregation Type (Fig. 3.36) |
|---|---|
| Case 1a: (0.685744 (0.677495 4 0) (0.452028 1 0)) | C |
| Case 1b: (0.555489 (0.428379 4 1) (0.649083 0 1)) | E |
| Case 2: (0.840473 (0.526626 (0.409497 4 1) 0) 1)) | D |
| Case 3: (0.384398 (0.381505 4 1) (0.588765 1 0)) | A |
| Case 4: (0.106946 4 (0.140715 4 (0.461898 (0.415415 4 1) 0))) | A |
| Case 5: (0.319276 4 (0.637608 (0.350867 4 1) 0) | A |
| Case 6: (0.354832 4 1) | F |
| Case 7: (0.386131 4 1) | F |
| Case 8: (0.148131 (0.340349 (0.421359 4 1) 0)) | A |
| Case 9: (0.337392 (0.357571 (0.430415 4 0) 1)) | B |

**Fig. 3.37. A list of the eight AMoEBA solutions and the best solutions of two cases that did not converge in Lisp-like format.**

To determine the performance of the segregation schemes found, four studies were made.

- The first of these was to compare the accuracy of the schemes on different grid sizes to show that the schemes found on the coarse 1800-node grid used by AMoEBA are still applicable on more traditionally sized meshing schemes. For this study, three more grid sizes were used. The times and accuracies of the segregation schemes are compared for each of the four grid sizes.
- A second study explores the segregation schemes' versatility on different geometries. The ultimate goal of this research is to be able to manipulate numerical data sets and recalculate the changes very quickly. For the AMoEBA segregation schemes to be useful in this context, they must be able to adapt to the changes that may be made to the geometry. To show this capability, this study compares the time and accuracy of the AMoEBA segregation schemes when used on five different angles of the pipe elbow.
- Similarly, changes may also be made to the boundary conditions. To examine the AMoEBA segregations' performance on different boundary conditions, five more inlet velocities have also been studied. Again the time to solution and accuracies of the segregation schemes have been recorded as well as the time to solution of a single model using a single k-ε RNG solver.
- As mentioned earlier, the difference in the effective rates of diffusion and convection often allows the assumption that changes made to the flowfield in one section have minimal effects upstream of these changes. To test this assumption and to show the effectiveness of the AMoEBA segregations, an obstruction was placed in the exit region of the 90° elbow. The solution was then found using the full model

of the elbow starting from the converged results. A solution was also found using one of the AMoEBA segregation schemes and only calculating those regions that are downstream of the obstruction and the region in which the obstruction lies. The accuracy of this second case and the time to solution was then compared to the full model values.

3.5.4.1. Grid Size Study for AMoEBA Segregations

For this study, three more grid sizes were used: 18,000 cells (20 radial, five angular, 180 axial), 36,000 cells (20 radial, five angular, 360 axial), and 72,000 cells (20 radial, five angular, 720 axial). The eight segregation schemes found by the different random populations were each tested on the three new grid sizes to demonstrate how the segregation schemes translate from the coarse grid used by AMoEBA to the different grid sizes. Since most CFD analyses use much larger grid sizes than that used by AMoEBA, this test is important in determining if AMoEBA is an effective algorithm. Both the accuracy and the times to solution for the schemes are compared. For accuracy, the cell to cell comparison used by AMoEBA is calculated to determine the error at the exit profile from a single model case of the same size using the k-ε RNG solver. Included in the study is the accuracy of an evenly distributed flow to give the reader perspective for the accuracy of the models.

Table 3.9 lists the times to solution of the eight different segregation schemes with the error calculations for the different grid sizes as well as the time required to solve the single model solution and the accuracy associated with an even profile at the exit. In every case but

**Table 3.9. Accuracy and times to solution for the grid study performed on the eight different segregation schemes found by AMoEBA.**

| | Grid #1 1800 cells | | Grid #2 18,000 cells | | Grid #3 36,000 cells | | Grid #4 72,000 cells | |
|---|---|---|---|---|---|---|---|---|
| Case | % Error | Time [sec.] | % Error | Time [h]:[m]:[s] | % Error | Time [h]:[m]:[s] | % Error | Time [h]:[m]:[s] |
| 1a | 1.31 | 89.1 | 0.89 | 0:16:27 | 0.99 | 0:30:34 | 1.09 | 1:18:01 |
| 1b | 1.68 | 79.2 | 0.59 | 0:19:01 | 0.61 | 0:39:44 | 0.66 | 1:36:56 |
| 2 | 0.67 | 81.9 | 0.34 | 0:19:00 | 0.35 | 0:33:01 | 0.53 | 1:31:50 |
| 3 | 0.82 | 99.5 | 0.44 | 0:19:54 | 0.44 | 0:38:34 | 0.36 | 1:36:11 |
| 4 | 1.06 | 71.9 | 0.97 | 0:14:51 | 0.93 | 0:30:49 | 0.64 | 1:13:46 |
| 5 | 1.04 | 72.5 | 1.15 | 0:14:15 | 1.06 | 0:29:22 | 0.60 | 1:11:16 |
| 8 | 0.91 | 96.2 | 0.92 | 0:13:56 | 0.86 | 0:28:46 | 0.55 | 1:21:56 |
| 9 | 1.29 | 82.8 | 0.80 | 0:13:39 | 0.82 | 0:29:40 | 0.87 | 1:18:02 |
| Single | - | 15.5 | - | 0:11:19 | - | 0:31:36 | - | 1:41:52 |
| Even | 17.6 | N/A | 23.5 | N/A | 23.5 | N/A | 23.3 | N/A |

Cases 5 and 8, the accuracy of the scheme improved as the grid size was increased. Case 5 (1.04% error) created slightly worse exit profiles at grid sizes of 18,000 cells (1.14% error) and 36,000 cells (1.06% error) than on the grid size used by AMoEBA. Case 8 (0.91%) resulted in improved accuracy on the 36,000-cell (0.86%) and 72,000-cell (0.55%) grids and approximately the same level of accuracy on the 18,000-cell grid (0.92%). None of the solutions were faster than a single model on the grid size used with AMoEBA (1800 cells). When the segregation schemes were used on the next largest grids, only the fastest schemes (~13-14 min.) converged in approximately the same order of time as the single model solution (11 min. 19 sec.). As the size of the grid was increased, the performances of the segregation schemes improved. On the 36,000-cell grid, five of the eight schemes were slightly faster than the single model solution; and on the 72,000-cell grid, every scheme was faster, with the fastest converging 30% faster than the single case.

The change in performance on the grid study is due to the manner in which the segregated schemes are calculated. As mentioned, these studies were performed on a loaded machine. As AMoEBA runs, many segregation schemes must be solved. Recall that information is passed between the segregated regions through the boundary conditions that are written to and read from a file. The time required for this file access is the same for any two cases that have the same number of cells in the cross-sectional area of the pipe, and it is not significantly smaller when this number of cells is reduced. Because of this, as the grid size is reduced in the axial dimension, the total number of cells decreases, the time to solve the individual models decreases, but the time to access the boundary condition files stays the same and a different type of overhead cost is introduced. At some intermediate grid size

(somewhere near 36,000 cells), the order of magnitude of the time required to obtain a solution becomes approximately the same as that of the overhead costs of the problem. For grid sizes larger than 36,000 cells, the overhead from using more than one model is outweighed by the faster solution methods used in the segregated system and a multi-solver system can be completed in less time than a single solver. For smaller grid sizes, the overhead becomes significant and it is difficult to find any segregation scheme that would outperform a single model solution on these grid sizes. It should be noted that the file access that causes the overhead is unnecessary for complete models since the boundary conditions are not changing under these circumstances. Therefore, these models do not have the overhead costs that a segregated solution has.

The accuracies of the different schemes on the different grid sizes range from 0.34% to 1.68%. This error is reasonable and relatively low when compared to the errors that even distributions would give at the exit for these grid sizes. The largest disparity between errors for the different grid sizes for one scheme was only 1.09% (Case 1b). Since the errors in the segregated results are not dramatically increasing for the different grid sizes, it is clear that the grid size does not adversely affect the results of the AMoEBA segregation schemes.

#### 3.5.4.2. Elbow Angle Study for AMoEBA Segregations

For AMoEBA to be considered an effective algorithm, the segregation schemes that it produces must be useful when changes are made to the geometry. To study these effects, the segregation schemes were applied to six different angles of the pipe elbow. Since the AMoEBA segregation schemes are based on a 90° elbow, angles of 75°, 60°, 45°, 30°, and

15° were also used in this study. For each of these cases, the accuracy of the segregated solutions has been compared to the single k-ε RNG model results. The times were also recorded and have been included in the comparison (Table 3.10).

For each segregation scheme, the time to reach convergence on the different angles was less than the time to solve the single model solution. Case 5 was the fastest for each angle, reaching convergence 59.6%, 49.8%, 45.5%, 44.0%, 35.7%, and 30.0% faster than the single model for the 15°, 30°, 45°, 60°, 75°, and 90° elbows, respectively. However, the error of the solution from the Case 5 scheme on the 15°, 30°, and 45° elbows increased as the departure from the 90° case increased (6.97%, 6.11%, and 3.67% errors). For these same three angles, the faster segregation schemes (Cases 1a, 4, 8, and 9) performed similarly to Case 5. The other three cases had smaller errors but were slightly slower than these faster cases. The fastest of these three cases was Case 2. This segregation scheme still converged significantly faster than the single model solution with a significantly smaller error of 1.59%. Although the faster schemes obtained errors greater than the 5% value chosen initially, this does not mean the schemes are unacceptable. Recall that the 5% error criterion was chosen arbitrarily. It does not correspond to any physical law of the system. The reason for the study is to understand the level of accuracy that is maintained by the AMoEBA segregation schemes when geometry changes are made. In this case, using the fastest scheme (Case 5), an engineer can be sure that any changes made to the pipe angle will result in a solution with a reasonable amount of error. This study helps to provide an understanding of this level of accuracy.

**Table 3.10. Accuracy and times to solution for the angle study performed on the eight different segregation schemes found by AMoEBA.**

| | 15° | | 30° | | 45° | |
|---|---|---|---|---|---|---|
| Case | % Error | Time | % Error | Time | % Error | Time |
| 1a | 6.47 | 0:29:57 | 5.51 | 0:37:24 | 3.39 | 0:46:29 |
| 1b | 2.43 | 0:43:28 | 1.85 | 0:53:55 | 0.87 | 1:02:39 |
| 2 | 1.59 | 0:41:06 | 1.19 | 0:49:42 | 0.56 | 0:56:02 |
| 3 | 1.46 | 0:47:00 | 1.20 | 0:53:26 | 0.72 | 1:01:48 |
| 4 | 5.07 | 0:28:07 | 4.18 | 0:37:19 | 2.33 | 0:45:38 |
| 5 | 6.97 | 0:22:27 | 6.11 | 0:30:25 | 3.67 | 0:41:03 |
| 8 | 5.00 | 0:30:14 | 4.13 | 0:38:00 | 2.37 | 0:50:54 |
| 9 | 6.10 | 0:26:07 | 5.08 | 0:35:21 | 3.14 | 0:46:30 |
| Single | - | 0:55:52 | - | 1:00:51 | - | 1:15:14 |
| Even | 25.7 | N/A | 24.1 | N/A | 22.7 | N/A |

| | 60° | | 75° | | 90° | |
|---|---|---|---|---|---|---|
| Case | % Error | Time | % Error | Time | % Error | Time |
| 1a | 1.52 | 0:54:55 | 1.07 | 1:05:16 | 1.09 | 1:18:01 |
| 1b | 0.66 | 1:18:01 | 0.84 | 1:25:43 | 0.66 | 1:36:56 |
| 2 | 0.42 | 1:05:25 | 0.53 | 1:18:05 | 0.53 | 1:31:50 |
| 3 | 0.57 | 1:12:07 | 0.69 | 1:24:41 | 0.36 | 1:36:11 |
| 4 | 0.93 | 0:54:59 | 0.73 | 1:11:10 | 0.64 | 1:13:46 |
| 5 | 1.56 | 0:49:28 | 0.80 | 0:59:38 | 0.60 | 1:11:16 |
| 8 | 0.93 | 0:56:24 | 0.66 | 1:08:32 | 0.55 | 1:21:56 |
| 9 | 1.33 | 0:53:40 | 0.91 | 1:06:34 | 0.87 | 1:18:02 |
| Single | - | 1:28:20 | - | 1:32:50 | - | 1:41:52 |
| Even | 22.7 | N/A | 23.0 | N/A | 23.3 | N/A |

3.5.4.3. Boundary Condition Study for AMoEBA Segregations

Changing boundary conditions for a CFD flowfield is a useful technique for analyzing a fluid design. Understanding the effects that these changes have on the model is important for a CFD analyst. To determine the AMoEBA segregation schemes' ability to adapt to different boundary conditions, segregated models were calculated using inlet velocities of 0.25 m/s, 0.5 m/s, 1.0 m/s, 1.5 m/s, 2.0 m/s, and 5.0 m/s. Similar calculations were made using a single k-ε RNG model. This single model solution will be considered the "exact" solution for this study. In all cases, the times were recorded and the errors between the single model and the segregated models were calculated (Table 3.11).

The time required to solve the "exact" solutions increased slightly with increasing velocity. This is due to the initialization that is used in all cases. In each case studied in this dissertation, the domain is initialized to a zero velocity condition. For the CFD solution to reach the final converged results, the values on the grid must be iterated until the solution is found. For higher speed flows, this often requires more iterations, increasing the time to solution for these cases. As with the single model solutions, the segregated solutions also exhibited this increase in time for the faster inlet velocities. Overall, the segregation schemes performed well. For inlet velocities of 1.0 m/s or slower, every segregated solution was faster than the single model solution. For 1.5 m/s, seven of the eight were faster; for 2.0 m/s, six of eight were faster; and for 5.0 m/s, five of eight were faster. Cases 5 and 9 were the fastest schemes. Case 5 was the fastest for inlet velocities of 0.5 m/s (1:09:39–24.3% faster than the single model solution), 1.0 m/s (1:11:16–30.0% faster), and 1.5 m/s

**Table 3.11. Accuracy and times to solution for the boundary condition study performed on the eight different segregation schemes found by AMoEBA.**

| | 0.25 m/s | | 0.5 m/s | | 1.0 m/s | |
|---|---|---|---|---|---|---|
| Case | % Error | Time | % Error | Time | % Error | Time |
| 1a | 0.38 | 1:09:21 | 0.42 | 1:14:51 | 1.09 | 1:18:01 |
| 1b | 0.72 | 1:22:18 | 0.60 | 1:26:07 | 0.66 | 1:36:56 |
| 2 | 0.55 | 1:21:13 | 0.50 | 1:22:33 | 0.53 | 1:31:50 |
| 3 | 0.42 | 1:26:17 | 0.44 | 1:29:01 | 0.36 | 1:36:11 |
| 4 | 0.71 | 1:03:26 | 0.28 | 1:12:39 | 0.64 | 1:13:46 |
| 5 | 0.78 | 1:07:25 | 0.36 | 1:09:39 | 0.60 | 1:11:16 |
| 8 | 0.70 | 1:10:43 | 0.29 | 1:19:32 | 0.55 | 1:21:56 |
| 9 | 1.07 | 1:03:21 | 0.51 | 1:11:03 | 0.87 | 1:18:02 |
| Single | - | 1:28:55 | - | 1:32:01 | - | 1:41:52 |
| Even | 34.2 | N/A | 28.7 | N/A | 23.3 | N/A |

| | 1.5 m/s | | 2.0 m/s | | 5.0 m/s | |
|---|---|---|---|---|---|---|
| Case | % Error | Time | % Error | Time | % Error | Time |
| 1a | 1.25 | 1:21:51 | 1.33 | 1:25:06 | 1.96 | 1:25:12 |
| 1b | 0.89 | 1:36:32 | 0.93 | 1:42:12 | 0.81 | 1:50:05 |
| 2 | 0.41 | 1:33:19 | 0.39 | 1:33:59 | 0.63 | 1:51:12 |
| 3 | 0.62 | 1:41:25 | 0.74 | 1:46:35 | 0.45 | 2:01:51 |
| 4 | 1.11 | 1:19:03 | 1.28 | 1:23:37 | 1.35 | 1:25:46 |
| 5 | 1.32 | 1:16:05 | 1.56 | 1:20:25 | 1.89 | 1:28:13 |
| 8 | 1.01 | 1:16:56 | 1.29 | 1:22:44 | 1.40 | 1:32:18 |
| 9 | 0.89 | 1:20:27 | 1.09 | 1:19:10 | 2.15 | 1:24:56 |
| Single | - | 1:36:59 | - | 1:41:49 | - | 1:46:34 |
| Even | 20.5 | N/A | 18.3 | N/A | 12.7 | N/A |

(1:16:05–21.6% faster), whereas Case 9 was the fastest for inlet velocities of 0.25 m/s (1:03:21–28.9% faster), 2.0 m/s (1:19:10–22.2% faster), and 5.0 m/s (1:24:56–20.2% faster).

Accuracy values for every scheme were less than 2.5% for the six different inlet velocities used in the study. Although the error values changed for the different velocities, no trend was found for the different schemes. In some cases, the error increased as the velocity increased; in others, the error changed only slightly; and in others, the error decreased with an increase in velocity. The important result is that all errors were within a reasonable level of accuracy that changed very little for the different boundary conditions used in the study.

3.5.4.4. Obstructed Flow Study for AMoEBA Segregations

Perhaps the most important part of segregating a CFD flowfield is reducing the amount of grid that must be recalculated when changes are made. An example of this is when changes are made at one location in the flowfield that have little to no effect on the regions that are upstream. In situations such as these, it may not be worthwhile to calculate these "unchanging" regions. This study will examine the effects of introducing an obstruction in the latter portion of the 90° elbow. Depending on the location of the barrier, only those regions that are downstream or contain the barrier will be calculated. As has been used in previous studies, the "exact" case in the comparison will be single models of the elbow using the k-ε RNG solver. The obstruction will be an orifice plate that is placed 1.0 m from the exit of the flow. Four different diameters are used for the open area of the orifice plate: 0.95 m, resulting in a 9.75% decrease in the cross-sectional area of the pipe; 0.9 m–19%

decrease; 0.85 m–27.75% decrease; and 0.8 m–36% decrease. Fig. 3.38 illustrates the problem description using the full model with the four sizes of the orifice plates used in the study.

Table 3.12 lists the time to solution, accuracy, and number of cells in the segregated region used in the study of the eight AMoEBA segregation schemes, as well as the accuracy of an even distribution and the time to solution for the single model "exact" case. Since the orifice plate was placed 1.0 m from the exit, each segregation scheme required only one model to be solved. This significantly reduced the time to solution. The slowest solution was still 80% faster than the single model solution, and the fastest AMoEBA solution was 98% faster. The accuracy levels of the segregated solutions were within the arbitrarily chosen limit of 5% error for all cases except for four cases on the 36% restricted flow. This is due to the excessive restriction in the flowfield very close to the exit of the model. The even flow distribution in this case has an error of 61.66%, which is much larger than the 23.5% error on the unrestricted model and 25.22% for the 9.75% restricted flow. Of the four cases that are within the 5% error on all obstructions, Case 5 is the fastest for the two smaller restrictions (95.5% and 94.9% for the 9.75% and the 19% restricted flows, respectively) and Case 3 is the fastest on the two larger restrictions (95.1% and 93.3% for the 27.75% and the 36% restricted flows, respectively).

The three smallest regions (Cases 1a, 1b, and 2) are three of the four cases with the most error. The other four cases (Cases 3, 4, 5, 8) have errors that are significantly lower than these cases. It is not surprising that the smaller regions would perform the poorest in this

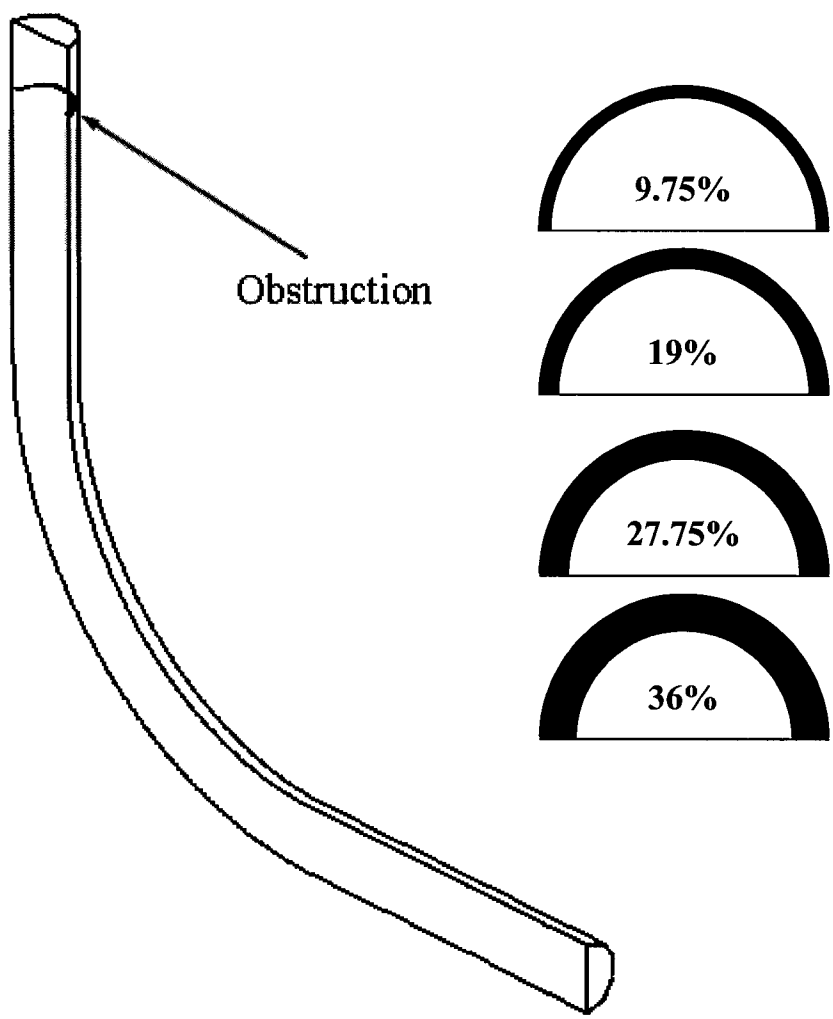



**Fig. 3.38. An example of the geometry of the obstructed flow analysis. The insets along the right illustrate the four orifice plate sizes used in the study.**

**Table 3.12. Accuracy and times to solution for the obstruction study performed on the eight different segregation schemes found by AMoEBA. Also included is the grid size of the segregated model.**

| | 9.75% Restriction | | 19% Restriction | | 27.75% Restriction | | 36% Restriction | | |
|---|---|---|---|---|---|---|---|---|---|
| Case | % Error | Time [h]:[m]:[s] | % Error | Time [h]:[m]:[s] | % Error | Time [h]:[m]:[s] | % Error | Time [h]:[m]:[s] | Grid Size |
| 1a | 1.06 | 0:02:12 | 1.04 | 0:01:55 | 1.25 | 0:02:02 | 5.59 | 0:05:39 | 6900 |
| 1b | 0.76 | 0:01:54 | 1.46 | 0:02:10 | 4.20 | 0:02:25 | 8.45 | 0:03:36 | 6200 |
| 2 | 0.41 | 0:01:27 | 1.23 | 0:02:02 | 4.10 | 0:02:22 | 8.60 | 0:03:16 | 6400 |
| 3 | 0.78 | 0:13:32 | 0.57 | 0:12:13 | 0.33 | 0:03:36 | 2.30 | 0:05:26 | 10,100 |
| 4 | 0.93 | 0:07:11 | 0.86 | 0:07:36 | 0.93 | 0:07:52 | 2.40 | 0:11:38 | 16,500 |
| 5 | 1.20 | 0:02:54 | 1.07 | 0:03:37 | 1.24 | 0:04:14 | 2.46 | 0:07:08 | 9,900 |
| 8 | 0.92 | 0:04:14 | 0.82 | 0:04:28 | 0.90 | 0:04:51 | 2.42 | 0:07:17 | 12,300 |
| 9 | 0.76 | 0:06:57 | 1.30 | 0:08:14 | 4.44 | 0:07:57 | 9.38 | 0:09:20 | 17,000 |
| Single | - | 1:07:21 | - | 1:10:06 | - | 1:12:54 | - | 1:22:34 | 72,000 |
| Even | 25.22 | N/A | 30.84 | N/A | 43.48 | N/A | 61.66 | N/A | N/A |

study because the inlet to these cases is closer to the obstruction than the larger models. As the inlet gets closer to the obstruction, the obstruction will have more of an influence on the upstream conditions. What is surprising is that the largest grid, Case 9, was the least accurate on the two largest obstruction cases. It is unclear why this occurs. Another interesting case is Case 3. The size of the final region for this case is not overly large or small, and the accuracy is very good for each of the four obstructions used in the study. It is in the top half for accuracy for the smallest obstruction and the most accurate for the other three obstruction sizes. The interesting feature of this case is the time to solution. In almost all of the other cases, including the full case, the time to solution slowly increases as the obstruction increases. In Case 3, there is a noticeable difference in this trend. For the two smaller sizes, Case 3's time to solution is significantly higher than any of the other segregated cases. For the other two obstruction sizes, the time to solution for this case is more reasonable compared to the others. Again, it is not clear why this occurs.

### *3.5.5. Conclusions*

This section has illustrated that the evolving segregation trees of AMoEBA are capable of partitioning a CFD flowfield and placing the proper solution methods in the segregated regions. The grid study of the AMoEBA segregations showed that the segregated solutions found by AMoEBA were within a reasonable amount of error for the three different grid sizes studied, and as the grid size was increased, these schemes became faster than using a single model solution. The changing angle study pointed out that not all of the AMoEBA segregation schemes were accurate for all angles in the study, but those that were within the error limit improved the time to solution for the flowfield. The accuracy levels of the

solvers in the boundary condition study were not affected as much as in the angle study. Every segregation scheme was within 2.5% error of the single model solutions. All but three schemes solved the models in less time than the single model case, and the three that were slower were only slower on the fastest inlet condition in the study. The obstruction study was the most significant of those performed. This study illustrated the effectiveness of the AMoEBA segregation schemes for CFD. By segregating the domain, changes can be made to this problem in the latter half of the pipe, and only those regions that are affected by the change require recalculation. In this case, time savings of 80-98% were found with accuracy levels still within reasonable limits.

# Chapter 4.

# Conclusions and Future Work

This thesis has presented the Adaptive Modeling by Evolving Blocks Algorithm, AMoEBA, and has demonstrated that AMoEBA is a versatile evolutionary tool that automatically locates optimum geometrical decompositions of numerical domains. Based on the application, the decomposed regions are assigned approximating algebraic expressions, materials, or numerical solvers and the solution of this network is compared to a known data set. The user defines the function that AMoEBA uses to determine the optimum segregation schemes as well as how the comparison is made between the known information and the candidate solutions. Many different types of problems can be solved using the AMoEBA algorithm. The following three have been presented in this dissertation to illustrate the versatility of the algorithm:

- The polynomial implementation applied to a steady-state conduction heat transfer problem and a fully-developed pipe flow problem, demonstrating the use of approximating algebraic expressions.
- The inverse engineering application for one-dimensional and two-dimensional conduction heat transfer problems, demonstrating material placement.
- The CFD solution method for the analysis of flow through a three-dimensional elbow section of pipe, demonstrating the placement of numerical solvers.

In each of these applications, AMoEBA partitioned the data sets/flowfields into reasonable segregation schemes with the appropriate solver/descriptor placements. Although the fitness

function and the solver/descriptors were changed for each implementation, the core algorithm of AMoEBA is unaltered in each application. This illustrates the flexibility of the algorithm.

In the polynomial version of AMoEBA, the segregation trees/approximators successfully matched two analytical engineering results automatically. Because the exact solutions used for this case were polynomials themselves, the ideal decomposition of the domain uses only one segregation. This ideal solution is only possible if the approximators have evolved to match the analytical solution to the problem. In the heat transfer problem, the approximators were able to match the exact solution within a reasonable amount of error; therefore, the segregation trees evolved to segregate the domain into the minimum number of regions. For the laminar pipe flow problem, the approximators took more generations to match the proper polynomial. Because of this, the trees illustrated their ability to work with the tools available. By placing the approximators in regions in which they adequately modeled the exact solution, the trees were able to achieve better fitness values than if one approximately correct polynomial was used over the entire domain. This demonstrates the effectiveness of a coevolutionary system using AMoEBA.

In the inverse problem implementation, AMoEBA automatically decomposed a domain, placed materials in the subregions, and compared the resulting solutions to the known temperature profiles. In the one-dimensional problem, AMoEBA found the exact material placement scheme over 85% of the time for the 30 different trials. Similarly, the appropriate material placement and segregation scheme was found 25 out of 30 times for the more

difficult two-dimensional problem. The algorithm has the potential to be applied to many different kinds of inverse problems as well. The two main qualities of AMoEBA that are useful in these types of problems are:

- The ability to work with discrete data: the trees define the boundaries that are placed between the regions instead of generating a continuous representation of the domain.
- Geometric decomposition capabilities. AMoEBA has the ability to work with complex geometries that may be difficult for other inverse techniques.

For inverse problems, AMoEBA places distinct materials in automatically generated segregation schemes and solves the candidate solutions directly. The materials are not limited to the solids that have been shown in this implementation. For example, placing fluids in the decomposed regions could be used in convection problems. Similarly, the inverse problem-solving version of AMoEBA is not limited to heat transfer problems. The segregation schemes generated can be applied to any system, and the discrete placement of the solver/descriptors can also be helpful in numerous types of inverse problems.

For the CFD problem of flow through an elbow, AMoEBA automatically decomposed flow through a pipe that included an elbow. The segregation schemes found by AMoEBA were very consistent. Each took advantage of the inlet-to-outlet solver and retained the level of accuracy required by placing a proper mixture of the k-ε RNG solver and the high Reynolds number k-ε solver in the rest of the pipe, while avoiding the poor accuracy of the fast constant turbulent viscosity solvers. This implementation was more complicated than the other two because of the number of variables involved (six) and the complexity of the

solution. The end result is an automatically generated multi-solver scheme that can be used to solve the given problem (flow through an elbow) using a variety of different grid sizes, geometries, and boundary conditions. The multi-solver scheme can even perform well when more complicated changes are made to the flowfield, as shown in the obstruction study. These investigations were used to illustrate the effectiveness of the AMoEBA algorithm for this implementation. In this case, the technique is designed as a means for an analyst to develop an engineering tool for use by others. Instead of analyzing a few different solutions and picking the best one to be passed on to the design team, AMoEBA is used to analyze many segregated solutions and evolve to an acceptably accurate, fast segregation scheme. For this type of application, this scheme can then be passed on to the engineers as an enabling tool for efficient engineering design. The goal is not to reduce the amount of work or time spent in the analysis stage, but to develop a method that will recalculate changes that are made to numerical models in the design process quickly, maintaining the creative momentum that drives good engineering designs.

The versatility of the AMoEBA algorithm makes it adaptable to many different engineering applications, from computational fluid dynamics to inverse engineering design problems. One long-term goal of this algorithm is to be able to use the segregated regions to solve computational data sets faster. This can be done either by using a combination of fast solvers or, as shown in the CFD version of AMoEBA, by recalculating only those regions that are necessary to obtain a solution that is within the accuracy range defined by the problem. A logical, potential improvement to AMoEBA is to utilize the automatic domain decomposition features of the algorithm for parallelization. The main concern with

implementing this advancement to the algorithm would be the effect on the boundary conditions between the regions. In the current version of AMoEBA, because of the potentially irregular sizes of the segregated regions, parallelization of the structure solutions would not be optimized. Despite this, the time to solution could be greatly reduced. As mentioned in Chapter 2, AMoEBA is not limited to only two dimensions. The extension of AMoEBA to three or more dimensions is not difficult. However, illustrating the effectiveness of the algorithm on multi-dimensional problems would be more complex than the demonstrations made in this research. A more interesting direction of the algorithm may be towards non-dimensional problems, such as large engineering systems that have many different components—power plants, automobiles, airplanes, etc. Creating a way to use AMoEBA to automatically uncouple these types of systems will be a useful technique in the analysis of the system as a whole.

Specifically, for the inverse implementation of AMoEBA, a different direction is to add a convection heat transfer solver. Through this addition, a whole new range of inverse engineering problems are introduced, significantly increasing the complexity of the problems being solved. The convection heat transfer solver may be as simple as supplying a known heat transfer coefficient or as complicated as solving a CFD flowfield to obtain the direct solutions to the candidate material placements. A further extension would be to use the knowledge gained from the coevolutionary polynomial implementation and evolve the thermal conductivities and the heat transfer coefficients of the materials involved. This would allow the solver to be used in situations where the materials are not known and only the temperature profile of the system is known. The inverse implementation of AMoEBA

could also be used on different types of inverse problems. For example, particle image velocimetry (PIV) is used for tracking particles experimentally. PIV can be expensive; therefore, it is used sparingly—usually only patches of information are gathered from the system being investigated. Recreating the entire flowfield based on these patches of information is a difficult inverse technique that could be tackled by AMoEBA. In this problem, the known patches of information could be used as comparisons to the candidate segregation schemes of AMoEBA.

The most logical extensions to the CFD implementation of AMoEBA are the introduction of different solution techniques. Two new solution techniques are addressed. The first is an adaptation of the inlet-to-outlet mapping solver that has been used in this research. This solver takes the values of the cells at the inlet to the region and directly transfers them to the outlet cells. The values of the inlet cells can be considered a matrix and the outlet cells a different matrix. A different way of looking at this solver is that the inlet matrix is being multiplied by the identity matrix to get the outlet cell matrix. Instead of using the identity matrix, a different matrix could be used to transform the inlet values. This type of solver would still be much faster than a CFD solution since, instead of iterating to obtain a solution, only one matrix multiplication would need to be performed. Different transformation matrices could be used to represent different pipe components—elbows, diffusers, nozzles, orifice plates, etc. Understanding and generating these transformation matrices is a difficult task but would be a major addition to AMoEBA.

The second new solution method is to implement a symbolic partial differential equation solver. Kirstukas et al. [68] have developed a method for the symbolic solution of differential equations. By avoiding the use of numerical techniques, errors associated with the discretization schemes are eliminated. The technique uses genetic programming techniques and is similar to the AMoEBA algorithm in that it has two phases. The first phase, finding the analytic solutions to the differential equations, is slow. The second phase is much faster since, once the analytic solution is known, new solutions can be found very quickly using different boundary conditions.

As mentioned in Section 3.5, a second reason for segregating a CFD domain is for improved accuracy. For this implementation, changes to the AMoEBA algorithm are required. In AMoEBA's current state, the boundary conditions between the solvers are only concerned with the six variables: radial velocity, angular velocity, axial velocity, pressure, turbulent kinetic energy, and turbulence dissipation rate. The information transferred by these variables is only the constant value information; derivative information is not transferred. Although this could potentially reduce the accuracy of the segregated solution, this was not a concern since time was the main priority in this implementation. For the improved accuracy implementation, first and possibly second derivative information should be passed between the models to ensure that the segregated solution is an improvement over a single model solution. The relaxation technique for calculating the interface conditions between PDE solvers discussed in Chapter 2 could be used for this modification to AMoEBA. By making these adaptations, AMoEBA could be used as an improvement in

accuracy over single model solutions, a much different implementation than the three that were demonstrated.

# References


[1] Cruz-Neira, C., "Virtual Reality Overview," *ACM SIGGRAPH '93 Course Notes: Applied Virtual Reality*, ACM SIGGRAPH '93 Conference, 1993.

[2] Knill, D. L., A. A. Giunta, C. A. Baker, B. Grossman, W. H. Mason, R. T. Haftka, and L. T. Watson, "Response Surface Models Combining Linear and Euler Aerodynamics for Supersonic Transport Design," *Journal of Aircraft*, 36(1):75-86, 1999.

[3] Rosenhead, L., ed., *Laminar Boundary Layers*, Oxford University Press, Oxford, England, 1963.

[4] McCorkle, D. S., K. M. Bryden, S. J. Kirstukas, "Building a foundation for power plant virtual engineering," 2003 Clearwater Conference, 63-71, 2003.

[5] Trigg, M. A., G. R. Tubby, and A. G. Sheard, "Automatic Genetic Optimization Approach to Two-Dimensional Blade Profile Design for Steam Turbines," *Journal of Turbomachinery, Transactions of the ASME*, 121(1):11-17, 1999.

[6] Vavak, F., K. Jukes, and T. C. Fogarty, "Adaptive Combustion Balancing in Multiple Burner Boiler Using a Genetic Algorithm with Variable Range of Local Search," *Proceedings of the 7th International Conference on Genetic Algorithms*, 719-726, 1997.

[7] Daida, J. M., et al., "Measuring Small-Scale Water Surface Waves: Nonlinear Interpolation & Integration Techniques for Slope-Image Data," *Proceedings of the 1996 International Geoscience and Remote Sensing Symposium*, Part 4 (of 4), 2219-2221, 1996.

[8] Daida, J. M., et al., "Measuring Topography of Small-Scale Water Surface Waves," *Proceedings of the 1995 International Geoscience and Remote Sensing Symposium*, Part 3 (of 3), 1881-1883, 1995.

[9] Gong, Y and J.-P. Haton, "Non-linear Vector Interpolation by Neural Network for Phoneme Identification in Continuous Speech," *Proceedings of the 1991 International Conference on Acoustics, Speech and Signal Processing–ICASSP '91*, 121-124, 1991.

[10] Zhong, J., J. Weng, and T. S. Huang, "Vector Field Interpolation in Fluid Flow," *International Conference on Digital Signal Processing*, 323-329, 1991.

[11] Zhong, J., J. Weng, and T. S. Huang, "Robust and Physically-Constrained Interpolation of Fluid Flow Fields," *ICASSP '92*, 3:III 185-188, 1992.

[12] Zhong, J., R. J. Adrian, and T. S. Huang, "Interpolation of Multidimensional Random Processes with Application to Fluid Flow," *ICASSP '93*, 5:V 181-184, 1993.

[13] Madsen, J. I., W. Shyy, and R. T. Haftka, "Response Surface Techniques for Diffuser Shape Optimization," *AIAA Journal*, 38(9):1512-1518, 2000.

[14] Khoo, L. P. and C. H. Chen, "Integration of Response Surface Methodology with Genetic Algorithms," *International Journal of Advanced Manufacturing Technology*, 18(7):483-489, 2001.

[15] Ahn, T., H.-J. Kim, C. Kim, and O.-H. Rho, "Transonic Wing Design Using Genetic Optimization of Wing Planform and Target Pressures," *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 4:IV 469-474, 1999.

[16] Venter, G., R. T. Haftka, and J. H. Starnes Jr., "Construction of Response Surface Approximations for Design Optimization," *AIAA Journal*, 36(12):2242-2249, 1998.

[17] Papila, M. and R. T. Haftka, "Response Surface Approximations: Noise, Error Repair, and Modeling Errors," *AIAA Journal*, 38(12):2336-2343, 2000.

[18] Zheng, Y. and P. K. Das, "Improved Response Surface Method and its Application to Stiffened Plate Reliability Analysis," *Engineering Structures*, 22(5):544-551, 2000.

[19] Ankenman, B. and B. McDaniel, "A Test Bed for Comparing Response Surface Methodologies," *Proceedings of the 1997 Industrial Engineering Research Conference*, IERC, 292-297, 1997.

[20] Gropp, W. D. and D E. Keyes, "Domain Decomposition Methods in Computational Fluid Dynamics," *International Journal for Numerical Methods in Fluids*, 14(2):147-165, 1992.

[21] Brakkee, E., P. Wesseling, and C. G. M. Kassels, "Schwarz Domain Decomposition for the Incompressible Navier-Stokes Equations in General Coordinates," *International Journal for Numerical Methods in Fluids*, 32(2):141-173, 2000.

[22] Andrews Vogel, A., "Automated Domain Decomposition for Computational Fluid Dynamics," *Computers & Fluids*, 18(4):329-346, 1990.

[23] Ribbens, C. J., "On Adaptive Domain Decomposition with Moving Subdomains," *Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, 322-329, 1991.

[24] Weile, D. S. and E. Michielssen, "The Use of Domain Decomposition Genetic Algorithms Exploiting Model Reduction for the Design of Frequency Selective Surfaces," *Computer Methods in Applied Mechanics and Engineering*, 186(2):439-458, 2000.

[25] Coelho, P. J. and M. G. Carvalho, "Application of a Domain Decomposition Technique to the Mathematical Modeling of a Utility Boiler," *International Journal for Numerical Methods in Engineering*, 36(20):3401-3419, 1993.

[26] Muller, U. R. and H. Henke, "Computation of Subsonic Viscous and Transonic Viscous-Inviscid Unsteady Flow," *Computers & Fluids*, 22(4-5):649-661, 1993.

[27] Khalid, M. and M. Mokry, "A Scheme for Two-Dimensional Euler and Boundary-Layer Interactive Solutions," *Canadian Aeronautics and Space Journal*, 38(3):116-124, 1992.

[28] Bush, R. H., P. G. Vogel, W. P. Norby, and B. A. Haeffele, "Two-Dimensional Numerical Analysis for Inlets at Subsonic Through Hypersonic Speeds," *Journal of Propulsion and Power*, 4(6):549-556, 1988.

[29] Wuthrich, S., M. L. Sawley, and G. Perruchoud, "The Coupled Euler/Boundary Layer Method as a Design Tool for Hypersonic Re-entry Vehicles," *Zeitschrift fuer Flugwissenschaften und Weltraumforschung/Journal of Flight Sciences and Space Research*, 20(3):137-144, 1996.

[30] Schmatz, M. A., F. Mannoyer, and K. M. Wanie, "Numerical Simulation of Transonic Wing Flows Using a Zonal Euler, Boundary-Layer, Navier-Stokes Approach," *Zeitschrift fuer Flugwissenschaften und Weltraumforschung/Journal of Flight Sciences and Space Research*, 13(6):377-384, 1989.

[31] Schmatz, M. A., "Simulation of Viscous Flows by Zonal Solution of the Euler, Boundary-Layer, and Navier-Stokes Equations," *Zeitschrift fuer Flugwissenschaften und Weltraumforschung/Journal of Flight Sciences and Space Research*, 11(4-5):281-290, 1987.

[32] Wanie, K. M., M. A. Schmatz, and F. Mannoyer, "A Close Coupling Procedure for Zonal Solutions of the Navier-Stokes, Euler, and Boundary-Layer Equations," *Zeitschrift fuer Flugwissenschaften und Weltraumforschung/Journal of Flight Sciences and Space Research*, 11(6):347-359, 1987.

[33] Quagliarella, D. and A. Vicini, "Viscous Single and Multicomponent Airfoil Design with Genetic Algorithms," *Finite Element in Analysis and Design*, 37(5):365-380, 2001.

[34] Drashansky, T., "A Software Architecture of Collaborating Agents for Solving PDEs," Master of Science Thesis, Purdue University, 1995.

[35] Anderson, D. K. and M. J. Kozlak, "Leveraging Physical and Computation Flow Modeling Techniques in the Optimization of Commercial Selective Catalytic

Reduction System Designs," *Proceedings of the 2002 International Joint Power Generation Conference*, 2002.

[36] Niksa, S., G. Liu, L. G. Felix, and P. Vann Bush, "Advanced CFD Post-Processing for Pulverized Fuel Flame Structure and Emissions," *Proceedings of the 2002 International Joint Power Generation Conference*, 2002.

[37] Vanderplaats, G. N., *Numerical Optimization Techniques for Engineering Design*, Vanderplaats Research & Development, Colorado Springs, CO, 1999.

[38] Back, T., U. Hammel, and H.-P. Schwefel, "Evolutionary Computation: Comments on the History and Current State," *IEEE Transactions on Evolutionary Computation*, 1(1):3-17, 1997.

[39] Koza, J. R., *Genetic Programming*, The MIT Press, Cambridge, MA, 1992.

[40] Koza, J. R., *Genetic Programming II*, The MIT Press, Cambridge, MA, 1994.

[41] Koza, J. R., *Genetic Programming III*, Morgan Kaufmann, San Francisco, CA, 1999.

[42] Kinnear, K. E., ed., *Advances in Genetic Programming (Complex Adaptive Systems)*, The MIT Press, Cambridge, MA, 1994.

[43] Kinnear, K. E. and P. J. Angeline, eds., *Advances in Genetic Programming, Vol. 2 (Complex Adaptive Systems)*, The MIT Press, Cambridge, MA, 1996.

[44] Spector, L., W. B. Langdon, U. M. O'Reilly, and P. J. Angeline, eds., *Advances in Genetic Programming, Vol. 3 (Complex Adaptive Systems)*, The MIT Press, Cambridge, MA, 1999.

[45] Ashlock, D. A., *Math 378 Lecture Notes*, accessed July 31, 2003, http://www.math.iastate.edu/danwell/ma378/math378.html

[46] Fabbri, G., "Optimization of Heat Transfer Through Finned Dissipators Cooled by Laminar Flow," *International Journal of Heat and Fluid Flow*, 19(6):644-654, 1998.

[47] 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(4):281-290, 1996.

[48] 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(2):403-420, 2000.

[49] 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 and Fluid Flow*, 24:267-277, 2003.

[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*, in press, 2003.

[51] Milano, M., P. Koumoutsakos, X. Giannakopoulos, and J. Schmidhuber, "Evolving Strategies for Active Flow Control," *Proceedings of the IEEE Conference on Evolutionary Computation, ICEC, v 1, 2000, Proceedings of the 2000 Congress on Evolutionary Computation CEC 00*, 212-218, 2000.

[52] Fan, H.-Y., R. Yam, and C. Dang, "A Preliminary Study of the Use of Genetic Algorithms in Flowfield Numerical Analysis," *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, 215(2):203-212, 2001.

[53] Mohanty, A. K. and S. B. L. Asthana, "Laminar Flow in the Entrance Region of a Smooth Pipe," *Journal of Fluid Mechanics*, 90(3):433-447, 1979.

[54] Nikuradse, J., In *Applied Hydro- and Aerodynamics* by L. Prandtl and O. G. Tietjens, Dover Publications, New York, 27, 1957.

[55] Angeline, P. J., "Subtree Crossover: Building Block Engine or Macromutation?," *Genetic Programming 1997: Proceedings of the Second Annual Conference*, 9-17, 1997.

[56] Ashlock, D. A., K. M. Bryden, P. E. Johnson, and D. S. McCorkle, "Improving Data Segregation with a Graph Based Evolutionary Algorithm," *Intelligent Engineering Systems Through Artificial Neural Networks, Proceedings of the Artificial Neural Networks in Engineering Conference (ANNIE 2002)*, 417-422, 2002.

[57] Ashlock, D. A., K. M. Bryden, P. E. Johnson, and D. S. McCorkle, "A Data Segregation Strategy Using Graph Based Evolutionary Algorithms," submitted to the *International Journal of Smart Engineering System Design.*

[58] Johnson, P. E., K. M. Bryden, D. A. Ashlock, and E. Vasquez, "Evolving Cooperative Partial Functions for Data Summary and Interpolation," *Intelligent Engineering Systems Through Artificial Neural Networks, Proceedings of the Artificial Neural Networks in Engineering Conference (ANNIE 2001)*, 405-410, 2001.

[59] Huang, C.-H. and S.-C. Chin, "A Two-Dimensional Inverse Problem in Imaging the Thermal Conductivity of a Non-Homogeneous Medium," *International Journal of Heat and Mass Transfer*, 43(22):4061-4071, 2000.

[60] Tervola, P., "A Method to Determine the Thermal Conductivity from Measured Temperature Profiles," *International Journal of Heat and Mass Transfer*, 32(8):1425-1430, 1989.

[61] Ponzini, G., G. Crosta, and M. Giudici, "Identification of Thermal Conductivities by Temperature Gradient Profiles: One-Dimensional Steady Flow," *Geophysics*, 54(5):643-653, 1989.

[62] Sridhar, L. and K. A. Narh, "An Inverse Method for the Determination of Thermal Conductivity and Thermal Contact Resistance: Simulated Results," *Proceedings of the 7th AIAA/ASME Joint Thermophysics and Heat Transfer Conference–Part 3 (of 4)*, 267-273, 1998.

[63] Shenefelt, J. R., R. Luck, R. P. Taylor, and J. T. Berry, "Solution to Inverse Heat Conduction Problems Employing Singular Value Decomposition and Model-Reduction," *International Journal of Heat and Mass Transfer*, 45(1):67-74, 2002.

[64] Drezet, J.-M., M. Rappaz, G.-U. Grun, and M. Gremaud, "Determination of Thermophysical Properties and Boundary Conditions of Direct Chill-Cast Aluminum Alloys Using Inverse Methods," *Metallurgical and Materials Transactions A: Physical Metallurgy and Materials Science*, 31(6):1627-1634, 2000.

[65] Fan, H.-Y., "An Inverse Design Method of Diffuser Blades by Genetic Algorithms," *Proceedings of the Institute of Mechanical Engineers: Part A*, 212:261-268, 1998.

[66] Johnson, P. E., K. M. Bryden, and D. A. Ashlock, "Solution of an Inverse Heat Conduction Problem Using Evolutionary Data Segregation Techniques," accepted for presentation at the *2003 ASME International Mechanical Engineering Congress and Exposition (IMECE '03)*.

[67] Johnson, P. E., K. M. Bryden, and D. A. Ashlock, "Solution of a 2-D Inverse Heat Conduction Problem Using Evolutionary Data Segregation Techniques," accepted for presentation at the *Artificial Neural Networks in Engineering Conference (ANNIE 2003)*, 2003.

[68] Kirstukas, S. J., K. M. Bryden, and D. A. Ashlock, "Evolving Solutions of Differential Equations Using Analytical Derivatives," *Intelligent Engineering Systems Through Artificial Neural Networks, Proceedings of the Artificial Neural Networks in Engineering Conference (ANNIE 2002)*, 275-280, 2002.