ACCELARATING COMPUTATIONAL FLUID DYNAMICS BASED AEROELASTIC ANALYSIS USING DISTRIBUTED PROCESSING

By

ANTHONY ANDREW BOECKMAN

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

2001

Submitted to the Faculty
of the Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2003

ACCELARATING COMPUTATIONAL FLUID DYNAMICS BASED AEROELASTIC ANALYSIS USING DISTRIBUTED PROCESSING

Thesis Approved:

[illegible]

Thesis Advisor

[illegible]

[illegible]

[illegible]

Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to Dr. Andrew Arena. Without his guidance and assistance, I could not have completed this work. I want to thank the other members of my thesis committee, two excellent instructors, Dr. P. M. Moretti and Dr. R. D. Delahoussaye.

I wish to thank the department of Mechanical and Aerospace Engineering, whose grant of assistantships helped support me during my graduate education. The support of the National Air and Space Administration's Space Grant program was invaluable as well. Special thanks goes to NASA's Dryden Flight Research Center for their support of this work.

I want to thank my parents, Tony and Jean, for their support, encouragement and teaching me to accept challenges. Finally to my sister, L. C., who never fails to amaze me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

a_∞	$\Rightarrow$	Speed of Sound
ARMA	$\Rightarrow$	AutoRegressive Moving Average
CAE	$\Rightarrow$	Computational AeroElasticity
CASE	$\Rightarrow$	Computational AeroServoElasticity
$f_a(k)$	$\Rightarrow$	generalized aerodynamic force at time k
M	$\Rightarrow$	Mach number
N	$\Rightarrow$	Total number points needed for a training set
N_p	$\Rightarrow$	Number of points in a single mode's training set
na	$\Rightarrow$	number of past outputs required in ARMA model structure
nb	$\Rightarrow$	number of past inputs required in ARMA model structure
nr	$\Rightarrow$	number of eigenvectors used in the modal structural model
q	$\Rightarrow$	dynamic pressure
STARS	$\Rightarrow$	STructural Analysis RoutineS
SVD	$\Rightarrow$	Singular Value Decomposition
$x(k)$	$\Rightarrow$	generalized modal displacement at time k
η_{Total}	$\Rightarrow$	total benchmark cycles per second of a cluster
η_{Single}	$\Rightarrow$	total benchmark cycles per second of a cluster node
ρ	$\Rightarrow$	density

CHAPTER 1
INTRODUCTION

1.1 Background

Aeroelastic phenomena result from the combination of aerodynamic, inertial and elastic forces. The coupling of these forces can lead to destructive motions in aircraft, such as wing flutter. The identification of the points of static and dynamic divergence in an aeroelastic system has become a driving research goal in modern aircraft design.

Digital computing has allowed the modeling of complex aerodynamic and structural systems. The combination of these two systems leads to a simulation of the aeroelastic response of a flexible structure under aerodynamic loading. The aerodynamics model predicts force loading due to the shape and rate of motion of the structure. The structural model predicts the shape and rate of motion due to the aerodynamic forces applied. The results of one system are feed into the other as new inputs. The resulting aeroelastic simulation allows the complete modeling of an aircraft's aeroelastic characteristics. Figure 1.1 Shows a diagram of a computational aeroelastic simulation.

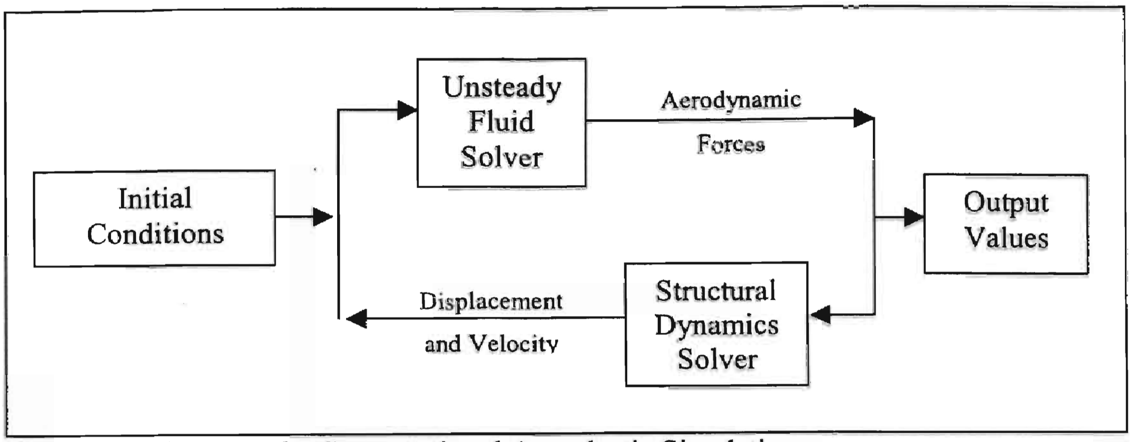


Figure 1.1 Diagram of a Computational Aeroelastic Simulation

The advancement of computation speed has allowed the use of CFD algorithms to simulate the aerodynamics forces. A Navier-Stokes or Euler flow solver coupled with a FEM or modal structural solver can produce highly accurate predictions of aeroelastic responses. Due to the computationally intense natural of CFD, the prediction of the aeroelastic behavior normally requires a time span on the order of days. Further worsening the problem is that a single simulation is not sufficient for a prediction. Several responses must be analyzed to bracket and estimate the instability. In essence, a sensitivity study is run to determine the effect of changes in dynamic pressure on the stability of the aeroelastic system. This process of running multiple simulations is both computationally and time intensive. Furthermore, the solution estimate must be refined to a certain resolution. The need for resolution requires search techniques that use the results of previous simulations in determining candidates for better resolution. This further slows down the dynamic pressure sweeps. This process can take weeks to months to finish; a time frame that is unacceptable for flight-testing purposes.

One method of accelerating the identification of divergence points is system identification of the aerodynamic forcing. This method allows the substitution of a linear model for the finite element CFD algorithm. This moves the time required for predictions from a matter of days to seconds. However, the new state-space model must be trained using results from the CFD solver. The generation of the training data for complex structural systems can require the same, or more, time as bracketing the instability with full simulations. This difficulty has lead to research into accelerating the processing for training data generation.

1.2 Motivation for the Study

As mentioned before the time required to find a flutter boundary is only practical for research purposes or when a design is finalized well before testing. There are a number of obstacles to accelerating the flutter prediction process, such as difficulty in numerical grid creation, structural eigenvector identification, and correctly transferring data from structural analysis to fluid flow solver. None of those problems can be corrected by speeding up the computations of either the structural model or the aerodynamics. In addition, all the tasks listed are completed quickly, compared the time of the flow solver. The primary area open to improvement is the speed at which aerodynamic flow solutions are generated.

As an example, one such test case that has time requirement that are prohibitive to the practical use of CFD based flutter prediction is the Aerostructures Test Wing, ATW. The ATW was a prototype wing flown on Dryden Flight Research Center's F-15B Research Testbed aircraft. The wing was flown until failure due to aeroelastic instability.

The ATW requires less than three seconds to analyze the FEM structural model and derive the modeshapes and natural frequencies from it. The interpolation of the modeshapes on the CFD FEM grid takes a second longer. The generation of the CFD grid is normally left for overnight generation, no more than twelve hours. The grid has 973,024 tetrahedral elements, as seen in Figure 1.2. The steady state solution of the flow field requires 8.43 CPU hours. The generation of the training data for system identification of the aerodynamic forces requires 66.22 CPU hours to identify the three structural modes of the ATW. That does not include any free response studies to check that the system identification is valid. The ATW would require a full workweek for an aeroelastic analyst to report back the flutter prediction findings, assuming everything worked on the first attempt.

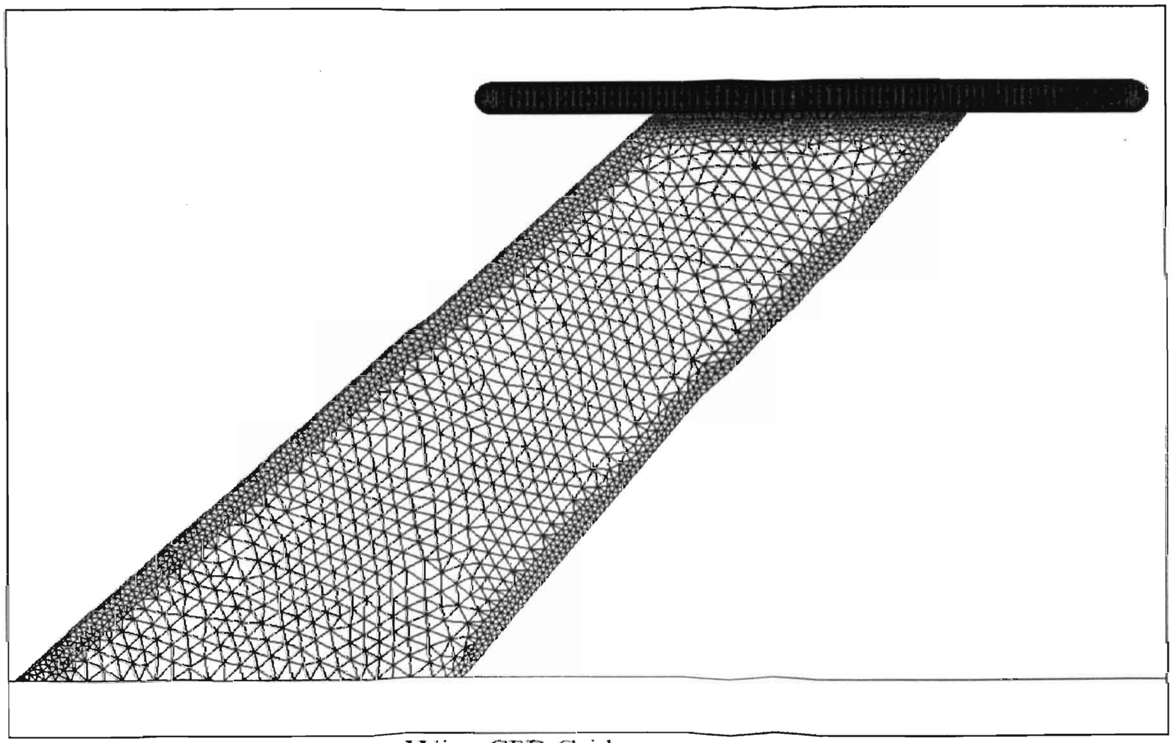


Figure 1.2 Aerostructures Test Wing CFD Grid

The large time requirement for the ATW is due in part to the fine grid mesh used to capture the motion of shock waves at the leading edge of the wing tip pod. If the ATW had more than just 3 modes, such as 15 modes, like many full aircraft simulations, then the training data would require 1557 CPU hours, two months of continuous computation, to complete. A dynamic pressure sweep would require only 662 CPU hours; assuming that only four dynamic pressures with 4000 time steps each were needed to estimate the pressure.

The STARS group at NASA Dryden has recently started to analyze test cases with over 2.5 million tetrahedral elements and up to 19 modes. The time to complete one time step is dependent on the complexity of the geometry. However, the more elements used in a simulation the more iterations must be used to resolve the time step. The simulation of large test cases can easily approach half a year.

Advances in computing speed help reduce the time, but the time is halved only every 18 months according to one interpretation of Moore's Law; which relates the number of transistors per square inch on a CPU chip and overall speed, to time. This advance rate does not significantly contribute the reduction of speed for recent test cases. Using the example of the ATW with 15 modeshapes, it will require 5 years before a processor will be able to finish the training data in one week, 9 years to until a processor can finish in one day. This relationship is shown in Figure 1.3. A new method of accelerating CFD based aeroelastic prediction the needs to be found.

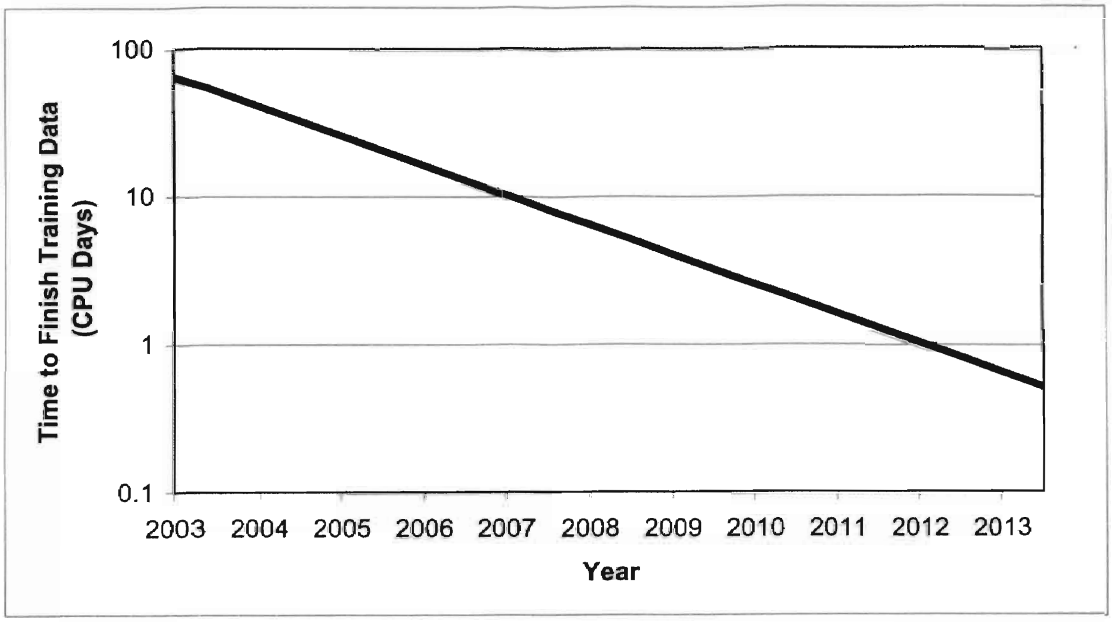


Figure 1.3 Projected Future Processor Performance

1.3 Problem Statement

Current techniques of free stream dynamic pressure sweeps to determine a flutter boundary take too much time for practical use in flight testing. Current techniques for system identification of a flow solution most often take less time, but still more than flight-test groups are willing to accept. New methods for reducing the time of flutter point identification need to be developed.

The solution to this problem must also meet one other requirement. It must be compatible with the STARS computer program suite developed at NASA Dryden Flight Research Center. In short, either a modified version of the STARS CFD flow solver or an additional integrated program must be produced that offers a method to accelerate both the generation of training data for system identification techniques and dynamic pressure studies.

1.4 Literature Review

This section covers the literature surveys and initial studies that lead from the problem statement above to the research objective in the following section. This covers the primary sources for this study.

1.4.1 Numerical Flow Solutions

One method for accelerating the speed of computational flow solvers is to change flow solvers. By there nature most flow models are only valid when certain assumptions hold. Four common flow models are discussed here. Euler solutions are analyzed in more detail in the next section.

One common fluid model is potential flow. This method uses the assumptions of incompressible and inviscid flow to determine the characteristics of the air around a defined shape. Potential flow solvers have relatively few computations, with a corresponding high speed. There exist several model alterations that account for the effects of compressibility, such as the Prandtl-Glauert rule [Katz, 2001]. This makes potential methods excellent for low speed flutter phenomenon, though this has limited applicability. However, the potential model cannot simulate supersonic or transonic flow, as it cannot model a shock of any type. It does introduce the concept of transpiration [Fisher, 1996], which removes the need for redefining the geometry of an aeroelastic system after structural motion. The inability to accurately predict aerodynamics in the transonic range makes these solvers useless as a general-purpose model.

The piston method is a previous attempt to accelerate simulation speeds [Hunter, 1997]. In this method, the unsteady wave equation is the basis for all simulations of perturbations about a steady state solution, usually determined by steady state Euler solvers. Although it has limits on accuracy, it provides reliable results for supersonic and hypersonic flows. The method does not model the motion of shock waves, such as those found in the transonic range. The solver fails to satisfactorily handle all flow speed regimes. However, a significant number of aeroelastic models operate in the supersonic range where the model is applicable. It does not meet the requirements of this study as it can be applied to all flow regimes.

The third type of solution is the Navier-Stokes numerical flow solver. This is considered the complete solution for an aerodynamic flow. Since it is based on the Navier-Stokes equation, the numerical model can simulate viscous effects. However, Navier-Stokes models require significantly addition time to complete an unsteady time step, as the viscous modeling needs computing. Experience has shown that in general viscous models are not necessary to model the aerodynamic forces on aircraft sized bodies. Furthermore, by attempting to model the viscosity of the fluid a new source of simulation error has been introduced. Neglecting viscosity can accelerate Navier-Stokes flow solvers.

The Navier-Stokes equation with the effect of viscosity neglected is the Euler equation. This model offers several advantages. The model holds at all speeds of interest for aeroelastic analysis. It accurately predicts the formation of shock waves and their motion in time, which is necessary for modeling the transonic flight regime were many aeroelastic phenomenon occur. It does have limitations. Since the Euler equation does not model the boundary layer of fluid flow around an object it does not handle the separation of the boundary layer. This is in general not a problem for aeroelastic stability determination. It can be efficiently discretized into finite difference or finite element solutions.

1.4.1.1 Euler3d Flow Solver

In 2003, Cowan [2003] showed that a noninertial reference frame could be used with FEM models based on the Euler Equation. The new program that study produced was *Euler3d*. This new flow solver was shown to out perform the STARS Euler solvers. It was user friendly, easily understood and more coherent in design philosophy that previous STARS flow solvers. The solver retained many useful features, such as the use of transpiration to model structural motion. It also contains an optional piston perturbation solver, which can be applied to models in the supersonic range.

1.4.2 System Identification

As defined, system identification is a process for obtaining a mathematical model of a dynamic system based on set of measured responses from that system [Ljung, 1987]. The time history of a dynamic systems response to a known input is used to fit a model with the least error to the response. For example consider the second order system of Equation 1.1.

$$\begin{aligned} \dot{x}_1(t) &= a \cdot x_1(t) + b \cdot x_2(t) \\ \dot{x}_2(t) &= c \cdot x_1(t) + d \cdot x_2(t) \end{aligned} \tag{1.1}$$

In this system *a*, *b*, *c*, and *d* are the unknown parameters. Using a set of responses from known inputs the unknown parameters can be determined [Kalaba and Spingarn, 1982]. The same technique can be applied to a discrete time dynamic system. The discrete version is shown in Equation 1.2, where a_j, b_j, c_j and d_j are the unknown parameters. The accuracy of identification model is dependent on the type of solution assumed and the method used to find the model parameters. Fortunately, the work of Cowan [1998] answers what approach to take.

$$\begin{aligned} x_{1,k+1} &= \sum_{j=1}^{j=k} a_j \cdot x_{1,j} + \sum_{j=1}^{j=k} b_j \cdot x_{2,j} \\ x_{2,k+1} &= \sum_{j=1}^{j=k} c_j \cdot x_{1,j} + \sum_{j=1}^{j=k} d_j \cdot x_{2,j} \end{aligned} \tag{1.2}$$

In 2003, Guezaine used an Eigensystem Realization Algorithm, (ERA) to identify the eigenvalue of entire aeroelastic system of an F-16 simulation. This method predicts the frequency and damping coefficient of the lowest torsional mode of the model. This approach is excellent for predicting the aeroelastic response at a single flight condition. However, it does not isolate the system response as a function of the dynamic pressure. In this regard it offers not advantage offer density sweeps in terms of accelerating predictions.

1.4.2.1 Autoregressive Moving Average Model

In 1998, Cowan applied an autoregressive moving average model to the identification of the aerodynamics of an aeroelastic system. This modeling technique assumes a statically nonlinear system with dynamically linear perturbations. Using a system model that predicts current forces based on past displacements and precious forces. This relationship is expressed numerically in Equation 1.3.

$$f(t)=\sum_{n=1}^{na} a_n \cdot f(t-n)+\sum_{m=0}^{nb} b_m \cdot x(t-m) \qquad (1.3)$$

This equation is actually a discrete version of Equation 1.4, which relates the motion of the structure to the unsteady aerodynamic forces.

$$\sum_i a_i \frac{\partial^i f}{\partial t^i}=\sum_i b_i \frac{\partial^i x}{\partial t^i} \qquad (1.4)$$

Using Equation 1.3, the parameters for system identification are the a_k and b_j values. Finding a set of values that closely model the training data produced by the CFD solver is goal of the system identification. The Cowan's ARMA model used Singular Value Decomposition, SVD. This method analyzes the data and determines the set of parameters that produces the least squared error with training data predictions. However, no automatic method exists to determine the value of *na* and *nb*, the number of previous forces and displacements. In order to determine the best value for each of these, a sensitivity study is run using a range of values for both.

It should be noted that this method only models the aerodynamics of the coupled aeroelastic system. This allows the same aerodynamic model to be used on multiple structural models. Also the system model is multistate. The motion of all modes is assumed to effect the forcing on all other modes. However, forces applied to one mode are not assumed to directly effect any other mode. In this way, a discrete time model is developed as in Equation 1.5. The matrix A_n is diagonal with zero in the off-diagonal terms. The B_m term handles the relationship from force to the motion of all the modes. The model also uses the current position from the structural model predictions.

$$f_a(t) = \sum_{n=1}^{na} [A_n] \cdot f_a(t-n) + \sum_{m=0}^{nb} [B_m] \cdot x(t-m) \tag{1.5}$$

It is important to note that the model assumes linear relationships between all mode displacements and forces. If this is not true, the system determined from the training data will poorly reflect the system. Also a sufficient number of data points must be contained within the training set to determine the system. The success or failure of system identification depends on the training data used.

1.4.2.2 Training Data Generation

In order to accurately model a system response, a system model must be trained with data points in the region of interest. When the system identification technique was developed by Cowan [1998] for application to the STARS unsteady flow solver, a multi-step on velocity was used. This method allowed the system model to relate velocity and displacement effects to force changes. However, the method made prediction of acceleration effects difficult as the training signal has either infinite acceleration or none. The multi-step training signal is shown in Figure 1.4. It is important to realize that this input signal is bypassing the structural dynamics entirely. The intention is gather data about the aerodynamic force response to displacement and motion. This training data is used to develop a model of the aerodynamic response. This new faster system model can then be coupled with the structural dynamics solver to model the linearly dynamic perturbations.

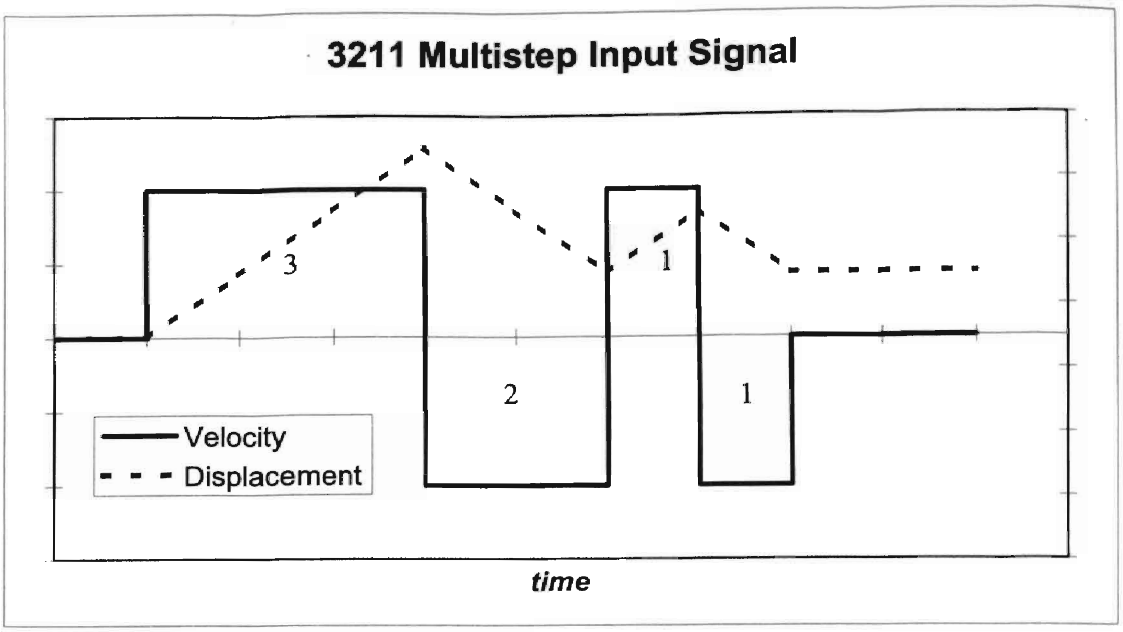


Figure 1.4 3211 Multistep Training Signal

In 2003, O'Neill developed an improved training signal. This new input signal was based on the chirp function in common use in system training data generation. The improved input signal allows more direct control of frequency range and magnitude. In addition, the new function allows the determination of effects related to the second and all higher derivatives, which the multistep input lacks. The new input signal can be seen in Figure 1.5. The signal has difficultly resolving low frequencies, as there is little power in them. The SVD algorithm also has problems determining the cause of forces, as both the velocity and displacement are symmetric about zero.

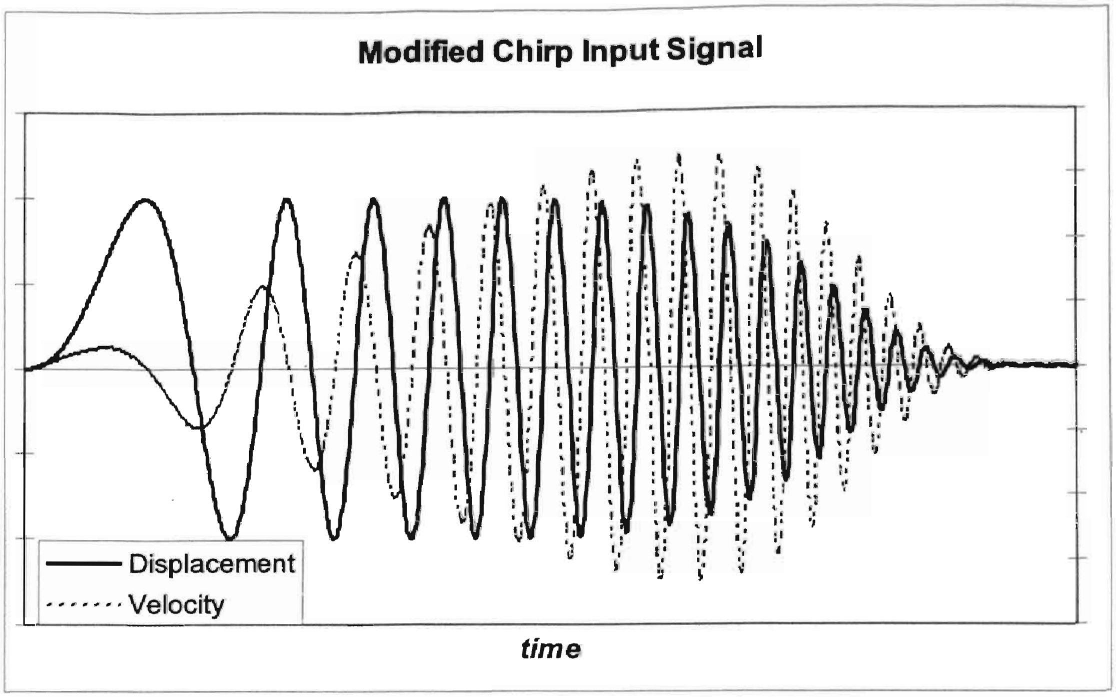


Figure 1.5 Modified Chirp Input Signal

To correct the problems with the Modified Chirp input signal an offset was added. The new signal the Offset Modified Chirp corrects the low frequency errors. It also helps the SVD algorithm define which terms are most important for system modeling. The Offset Modified Chirp Input Signal is shown in Figure 1.6.

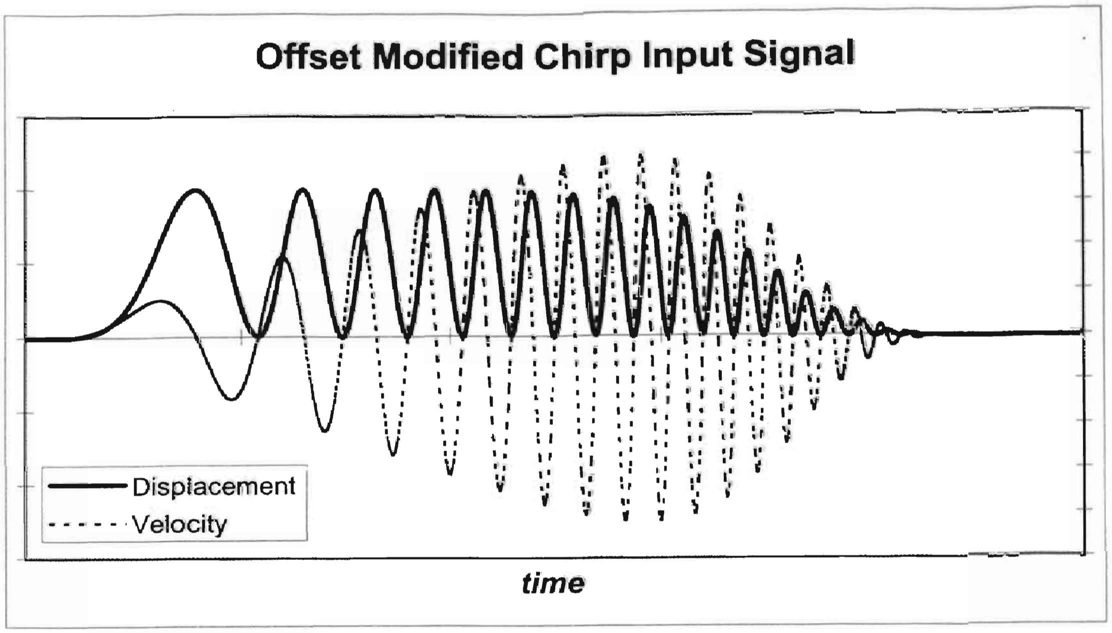


Figure 1.6 Offset Modified Chirp Input Signal

1.4.3 Parallel and Distributed Processing

By definition, parallel processing uses several independent Central Processing Units (CPU) to solve a single problem. In common usage this normally refers to the use of several processors to solve a single mathematical operation, such as matrix inversion. However, there are other methods that take advantage of the parallel processing concept.

One such method works by instigating multiple copies of a single simulation, each with a set of different initial conditions, often called distributed batch processing [Baker and Smith, 1996]. By distributing the simulations to several independent processors, the time to complete the task is reduced. This method has the advantage of no communication after the initial setup of the simulations. Since the simulations are not commutating, there is no need to handle time step matching, differences in speed of computation or bandwidth minimization. Generally, this method can be implemented quickly and with few or no changes to the algorithm of solution.

Another method for parallel processing, Domain decomposition as described in general by Gropp, [1999] and for CAE by Liu [2001], divides a physical problem into several physically smaller parts. This allows each processor to work on a portion of a much larger problem. However, domain decomposition does require time step matching and communication between the subdomains, often called zones. This stipulation requires that communication bandwidth not retard the speed of the independent computers. Domain decomposition also only works well on problem where each subdomain has an equal workload. If the problem does not have a uniform distribution of computation for each subdomain, many processors will set idle while waiting of others with a corresponding drop in efficiency. This method can usually be added into an existing solving routine, but only with detailed planning and will be solution specific. For unstructured CFD meshes, such as the type used by STARS, a sophisticated algorithm that divides the regions into equal computational zones, not equal physical size, must handle the decomposition.

Implicit distributed batch processing and domain decomposition are the primary methods for distributed processing of any large simulations, [Gropp, 1999]. Many other parallel and distributed models exist for data processing, but have only limited application to the simulation of complex systems.

One important side note is the definition of efficiency used in this work. Equation 1.6 states the relationship for efficiency. Baker [1996] has an interesting discussion about the methods for measuring performance of a parallel program. It this work, the time to complete a set of simulations is the bases of all efficiency calculations. The average time for a serial processor to finish one of the simulations in the set is used as a standard of comparison to the time for an entire set to finish in parallel. Also efficiency is not defined unless the number of simulations is equal to the number of processors used.

$$\eta = \frac{Time_{AverageSerial}}{Time_{Parallel}} \qquad (1.6)$$

1.4.4 Parallel and Distributed Processing in Aeroelasticity

It is interesting to note that no papers were found that explicitly study the effect of distributed batch processing of aeroelastic analysis. This many be a result of researchers not reporting the use of multiple machines. It many also be an artifact of the development of CFD based CAE. In the previous studies, researchers invested in a single high-speed computer to handle simulations. This type of equipment acquisition scheme does not lend itself to the development of distributed computing, as only one computer of significant speed is available. Several efforts have been made with success into the use of parallel processors on a single machine, such as the SGI Origin 3200 and IBM SP2 [Goodwin, 1999]. Although these computers are efficient and scalable, they are expensive and require extensive training for operation. No reports of the use of workstation clusters were found.

As a side note, all the studies surveyed that use some implementation of parallel processing employed the Message Passing Interface, MPI. One reported a use of MPI and Parallel Virtual Machine, PVM, and compared to results for each [Goodwin, 1999]. It is hoped that since MPI is highly portable studies comparing the use of modern inexpensive personal computer clusters and standard parallel supercomputers will soon be available in the literature.

In 1998, Byun and Guruswamy reported successful results for a multizone aeroservoelastic solver. Their solution used Navier-Stokes finite difference numerical methods. The multizone aspect they refer to is the domain decomposition of the flow volume into 8 zones, each simulated on a different processor within an IBM SP2 parallel supercomputer. The article was concerned with the simulation of response to control inputs, not flutter prediction. However it did introduces interesting methods. The parallel solver resolves the aerodynamics and structural response on different sets of parallel processors. The two simulations were matched at discrete time steps, but were otherwise independent. The ENSAERO codes used structured grids. This allows the operater to divide the flow volume into separate zones by simple inspection of the nearly rectangular grid. The ENSAERO code also used a moving mesh to simulate the effects of elastic deformations, a process that requires the regrinding of at least a small part of the computational mesh at every time step. Most importantly for use in this study, Byun and Guruswamy report that the parallel version of the ENSAERO code has near efficiency up to 16 processors. This last fact indicates that parallel processing will accelerate the prediction of flutter, even if only by decomposing the flow domain.

In 2003, Geuzaine developed enhancements to three-field methodology to model aeroelasticity. The three fields are aerodynamics, structure and mesh movement. The AERO-F, AERO-S, and MATCHER codes described the modeling of FEM based Navier-Stokes and structural analysis. The results of a free response simulation are examined using an Eigensystem Realization Algorithm, ERA. This method, as applied, will report the frequency and damping coefficient of the lowest torsional mode. This allows comparison to flight test data. However, this method is does not derive the Eigensystem as a function of flight conditions, such as density or Mach number, but reports the Eigensystem at the flight conditions input to the model. In order to locate a flutter boundary, multiple flight conditions must be simulated and analyzed. Like Byun and Guraswamy, Geuzaine used separate sets of processors to solve the fluid and structures response.

Neither Byun nor Geuzaine developed methods to accelerate control law development. Both have features that allow a control scheme to be tested in full flight simulation, but no method that allows for a quick systemic search of the several candidates to select the best option. In addition, all the parallel schemes surveyed, including Goodwin [1999] and Liu [2001], are full simulations that were only tested on shared memory parallel machines.

One problem with the literature on parallel processing based aeroelasticity is that very little of the literature is interested in locating flutter boundaries. Most are interested in replicating experimental results or responses at a set flight condition. Many, like Goodwin [1999], even use experimental results as the bases for determining what initial conditions to use. Few papers are interested in searching for unfavorable flight conditions for aeroelastic properties from scratch. One of the goals of this study was accelerating the flutter prediction of a test case where the user is unaware of experimental results, the same conditions that would be present in the design of new aircraft.

1.5 Feasibility

In order to test if the general technique of training a test case on multiple machines and then combining the resultant training data would work in practice, the AGARD 445.6 test case was used in an initial test.

The AGARD445.6 is a standard test case for aeroelasticity. The wing is slightly cambered with 45° backward sweep, the AGARD445.5 can be seen in Figure 3.1. For this study, the AGARD445.6 has two modes, first bending and first torsion. The serial training data is shown in Figure 1.7. The data took 30250 seconds or 8.40 hours.

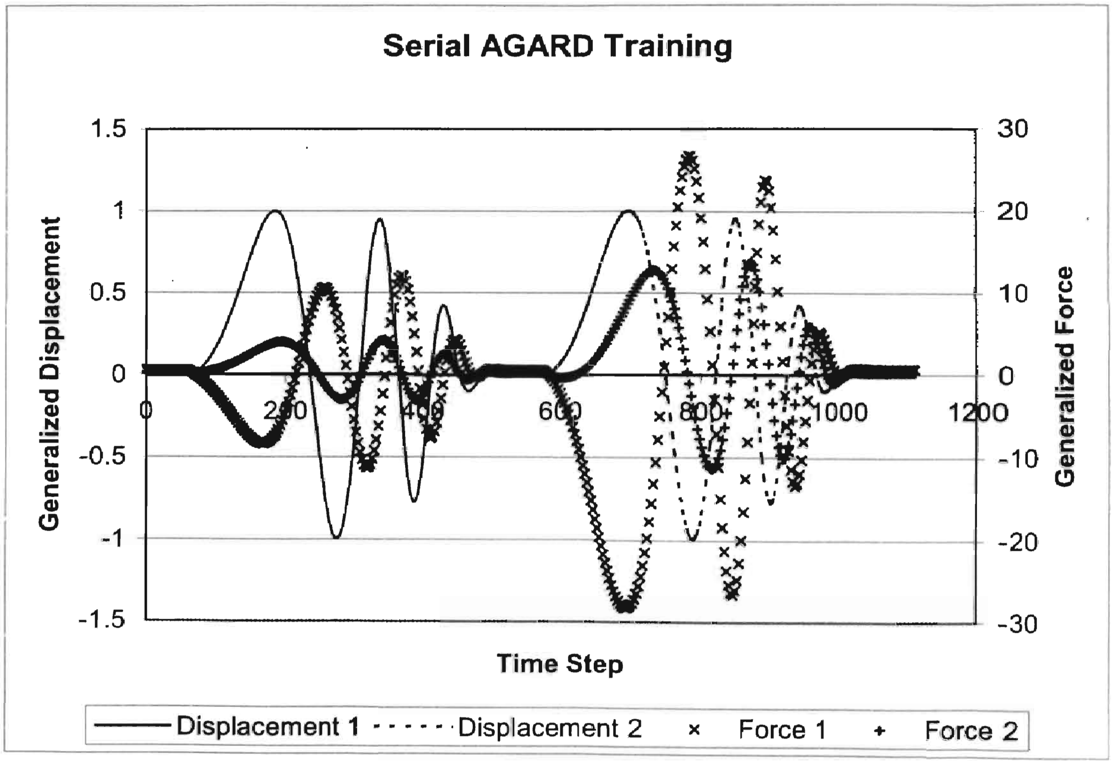


Figure 1.7 The AGARD445.6 training data generated in serial

This data was used to find a system model that would predict the flutter point of the combined aeroelastic system. The sensitivity study found that a *na, nb* of 4, 7 worked to predict the divergence of mode 1 at a pressure of 0.399 psi. The flutter point was determined by finding the first dynamic pressure that produced a system eigenvalue outside the unit circle for the complex plane. Since this is a discrete time system, any complex eigenvalue with an absolute value greater than one represents an unstable system. This can be seen in Figure 1.8. The graph is of the complex, or z, plane where the vertical axis is the imagery numbers and the horizontal is the real value. The unit circle is the boundary of stability. An eigenvalue the lays directly on the circle represents a dynamics system with no damping, a sine wave. Both modes 1 and 2 start very close to the unit circle, as neither have any structural damping and the low dynamic pressure has little effect on the structural response. This was used in comparison to the parallel training data.

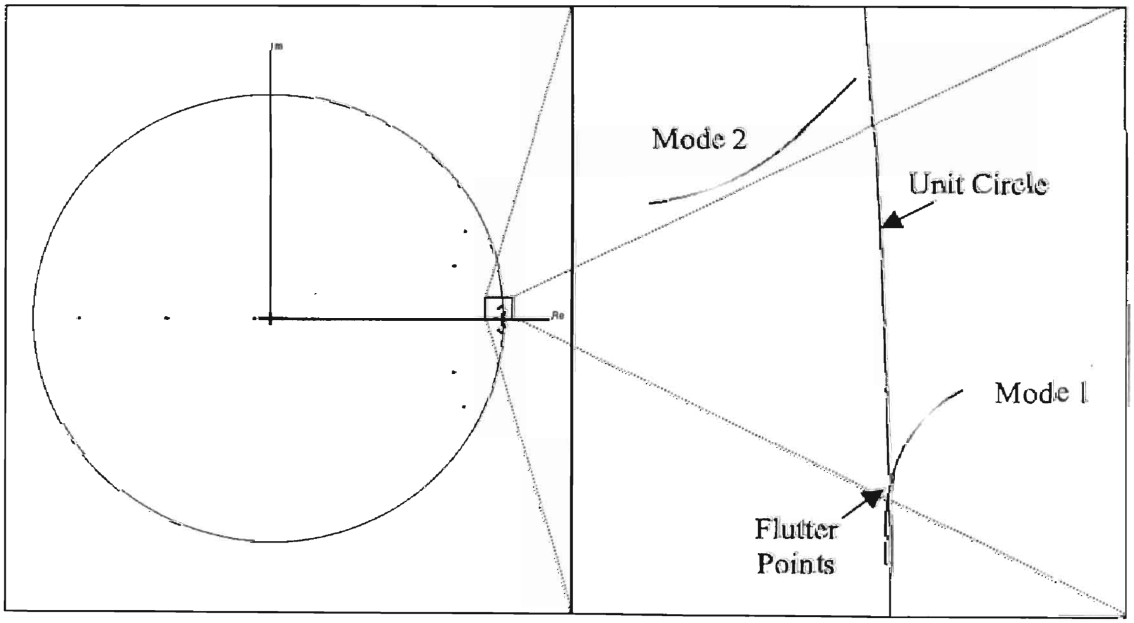


Figure 1.8 AGARD eigenvalues for q of 0.01 to 1.00 psi, (serial training)

The parallel data was run on two different computers, each at different speeds. The runs required 15480 and 38700 seconds. Although this took longer that the serial run, it was done on two computers, one significantly slower than the other. The computer that performed the serial run completed its required load in just over half the time the serial run used, 51.2%. This indicts that if two identical computers were used then the time reduction would be 48.8%. The parallel generated training data is shown in Figure 1.9.

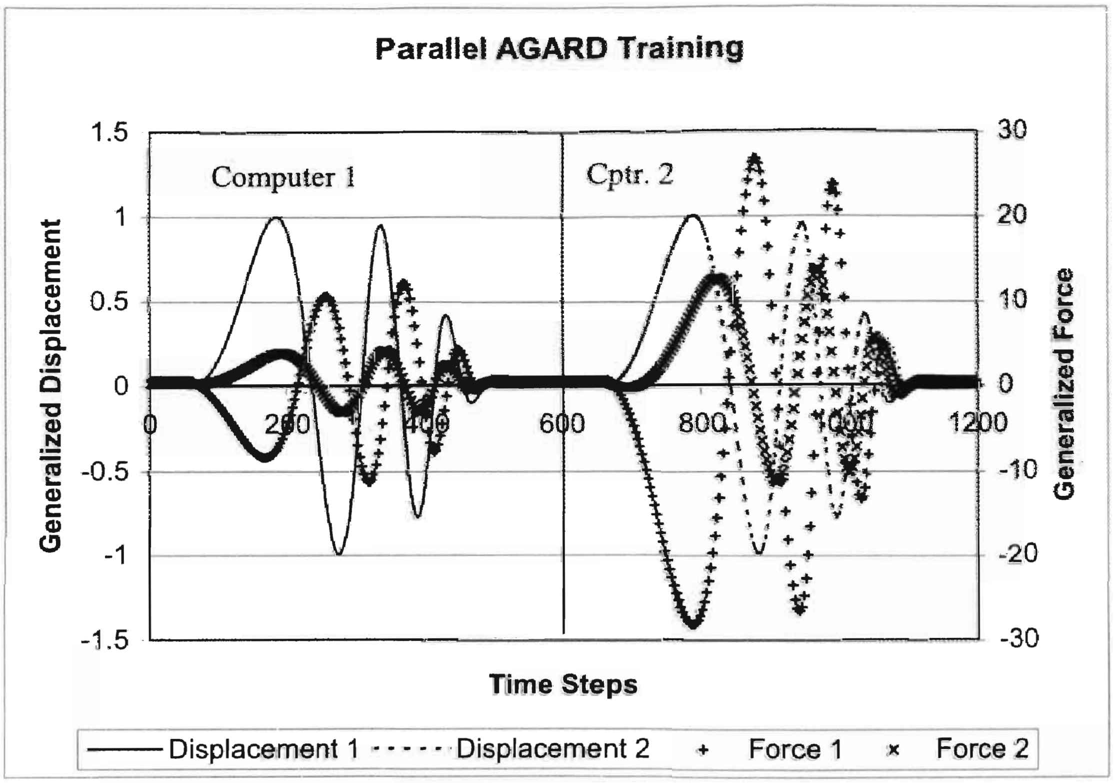


Figure 1.9 The AGARD445.6 training data generated in parallel

For the parallel data, the two time histories were simply pasted together one after the other. The same model order was used, 4–7. The system model predicted that the flutter point, with mode 1 divergence, was at 0.402 psi. This is a 0.75% difference from the serial training set, which is well within acceptable limits for aeroelastic instability predictions. This difference is most likely caused by the use of two different CPUs with slightly different floating accuracies. The graph of dynamic pressures is seen in Figure

1.10. This figure includes only the close up view of the flutter point cross over. It should be noted that the two modes have not only the same flutter point, but follow the same trend. Both have mode 1 moving close to neutral stability boundary then crossing over. Mode 2 starts near the unit circle and becomes more damped.

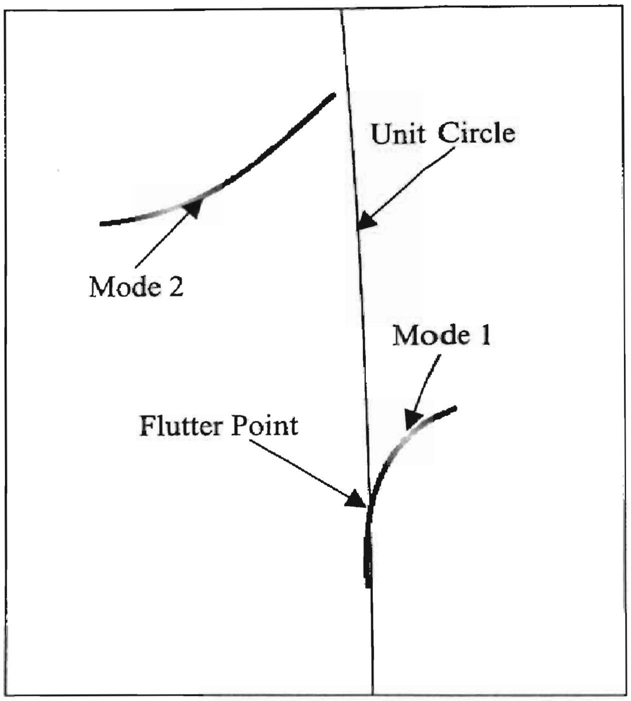


Figure 1.10 AGARD eigenvalues for q of 0.01 to 1.00 psi, (parallel training)

From this initial test, the parallel training of a test case can produce reliable results. Furthermore, the parallel data can predict the true experimental response of 0.425 psi, [Yates, 1987]. It also agrees with predicted values from Cowan [1998] and Gupta [1996].

1.6 Research Objective

This research focuses on the use of distributed batch processing to accelerate computational aeroelastic analysis. The first goal of the project was the modification of the *Euler3d* software package to automate the initiation and simulation of multiple free responses to varied initial conditions. This goal allows for the systemic sweep of dynamic pressure at a constant Mach number. The second goal was the addition of a new feature that allows the simultaneous training of multiple modes. The parallel training and response simulation will significantly reduce the time required to complete a flutter prediction.

CHAPTER 2

METHODOLOGY

2.1 Applications of Distributed Processing to Computational Aeroelasticity

Aeroelastic analysis is well suited to parallel processing. The large volume of data that must be generated for an instability prediction does not require sequential calculation, for either the dynamic pressure sweep or system identification approach. This allows an intelligent aeroelastic analyst to utilize all processors available to him. The two types of distributed parallel processing that concern this work are batch processing of multiple free responses at different initial conditions and training data generation.

2.1.1 Density Studies at Constant Mach Number

For dynamic aeroelastic systems the flutter prediction must be confirmed with full couple CFD structural dynamics system responses. Even with a system identification model predicting flutter it is best to confirm the prediction with a free response. Running responses to initial conditions both above and below the flutter boundary does this. These can be run in parallel, as the results of one have no influence on the others.

Dynamic pressure causes flutter. However, from Equation 2.1, the dynamic pressure is a function of Mach number, speed of sound and the air density. Since Mach and sonic velocity are held constant in a simulation, the change in dynamic pressure is directly related to change in density. This type of study will be referred as either a pressure or density sweep, usually density as this is the parameter that will be changed directly in the sweep.

$$q = \frac{1}{2} \cdot \rho \cdot (a_{\infty} \cdot M)^2 \qquad (2.1)$$

A density sweep is best described as a sensitivity study on density. The free response input conditions are set, such as the true sonic velocity, the Mach number, the global time step, the number of time steps, the iterations to be made on the solution between each global time step, and a few others. The method for initiating the vibration must also be chosen. The steady state solution and structural dynamics both remain constant for each free response in the study. The only thing that is changed in density sweep is the density, and through it the dynamic pressure.

For the new distributed *Euler3d* the density sweep will be automated. The control file will contain the same information as before, with the addition of a new parameter *delrho*. This value controls the increment of density for each processor in the distributed architecture. The equation relating the density, *rho*, on any processor *np* to the base density from the control parameter *rhoinf* is stated in Equation 2.2.

$$rho(np) = rhoinf + np \cdot delrho \qquad (2.2)$$

The density on any node is simply the base density plus the processor number multiplied by the increment *delrho*, noting that the first processor has an index, *np*, of zero.

An initial obstacle with a density approach is the starting point. Often an aeroelastician is given the information about structure, geometry and speed of interest, but little else. Since the system is unknown and possibly nonlinear, it is best to divide the flight envelope into equal increments. The sweep of density will reveal either that the structure does not flutter or determine a narrower range of density to investigate. This can be repeated until the required resolution is reached. With this system, the number of computers used determines the time until convergence on the proper resolution. It should be noted that a course density sweep could miss a flutter point entirely, as an aerostructure can move in and out of instability.

Once a free response has run out long enough for multiple cycles to be present, it can be analyzed. There are several approaches to this analysis.

The first method, and most obvious, is for the analyst to graph the time history of each structural mode's motion and look for instabilities. This has some drawbacks, as the free response may need to be run out to several cycles to show a clear damped or instable response. Although if a user is not able to clearly see a damping trend, most other methods will produce unreliable results.

A second method involves the use of data points to fit a free vibration model to the data. The curve is checked for either decay or divergence. Aeroelastic free responses, particularly those with several mode shapes, are prone to transitory motions as the modes shift to either a new static offset or to the frequency of motion with least energy for the flight conditions modeled. This motion may require that the free response be allowed to run out until clearly decoupled dynamic motion is visible. Equation 2.3 states the mathematical relationship for this approach, [Moretti, 2000]. The five last distinct peaks, or valleys, are excellent candidates for this approach to find the damping. If the rate is positive, the mode is divergent. Figure 2.1 Shows an example of four time histories at four different densities. The four histories have the densities of 1, 2, 3 and 4.

$$x(t) = C \cdot \exp(-\xi \cdot \omega_n \cdot t) \cdot \sin\left(\sqrt{1-\xi^2} \cdot \omega_n \cdot t + \phi\right) + k \qquad (2.3)$$

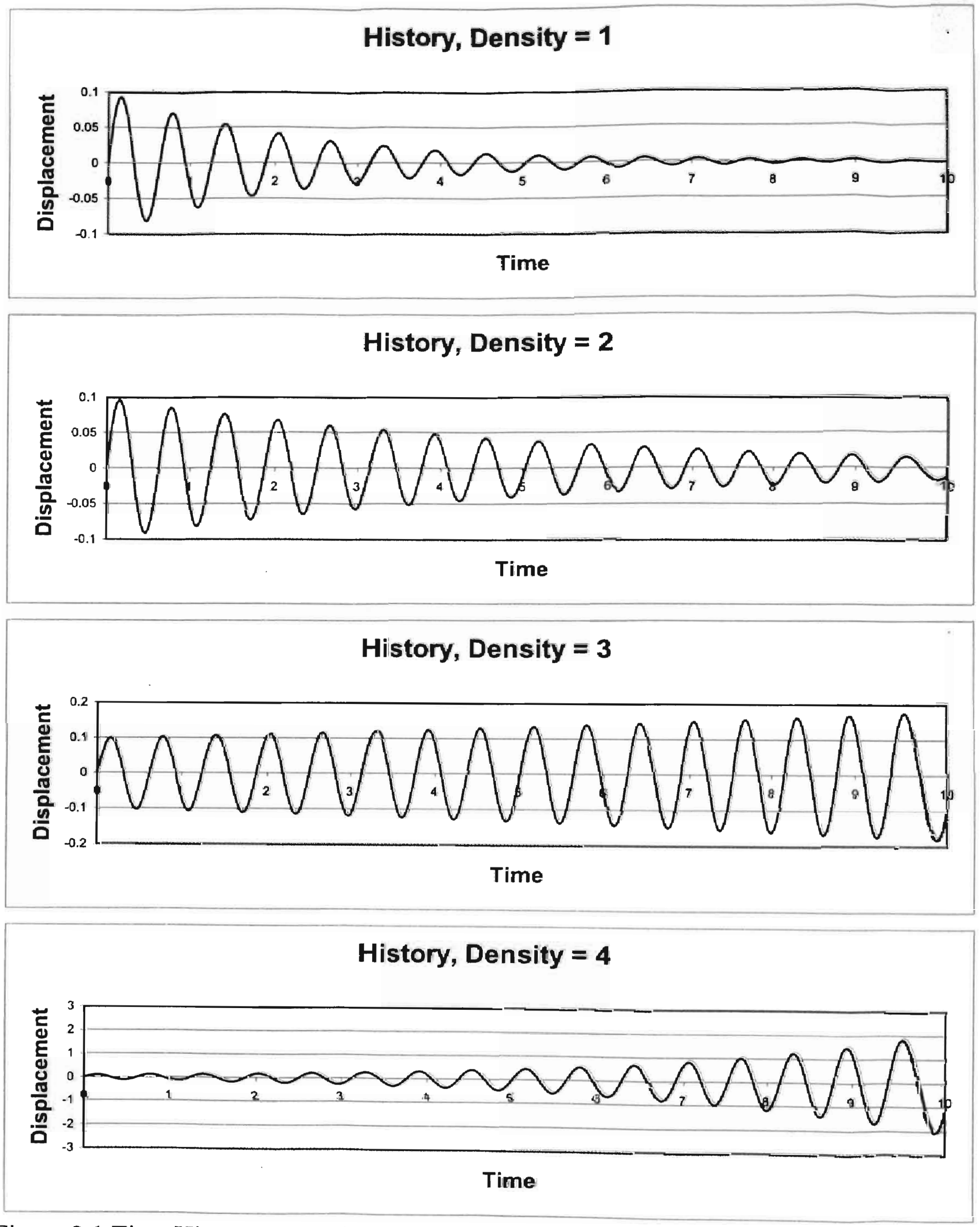


Figure 2.1 Time History Example

In actual practice, the points used are taken from the last few complete cycles of the time history. This helps avoid inaccurate estimates due the strong transients in the modes in the initial cycles of the time history. Also the full vibration description is not used. The last few peaks are identified and those are used to fit the relationship of Equation 2.2. Using the four data points, one for each time history, a cubic curve can be fit to estimate the neutral point, or critical damping, with respect to the density. Figure 2.2 shows the graph of the four damping values and the trend curve. Note that the true damping, from the equations used to generate the time histories, is plotted as well. Table 2.1 shows all four densities and the density that the curve estimates as the flutter, or neutral, point.

$$\xi = -\log\left(\frac{\frac{x_2}{x_1}}{\omega_n \cdot (t_2 - t_1)}\right) \qquad (2.2)$$

One problem with the use of Equation 2.2 is that it assumes the static offset about which the dynamic oscillation move is a constant zero. This is true in this example but not in general for aeroelastic test cases. This method is also very suspect when the frequency of motion has not stabilized; ω_n is changing. This problem is of particular concern as the only to combat it is the use of long term histories; which require large computational times. One advantage of the system identification technique is that is has less susceptibility to uncertainty.

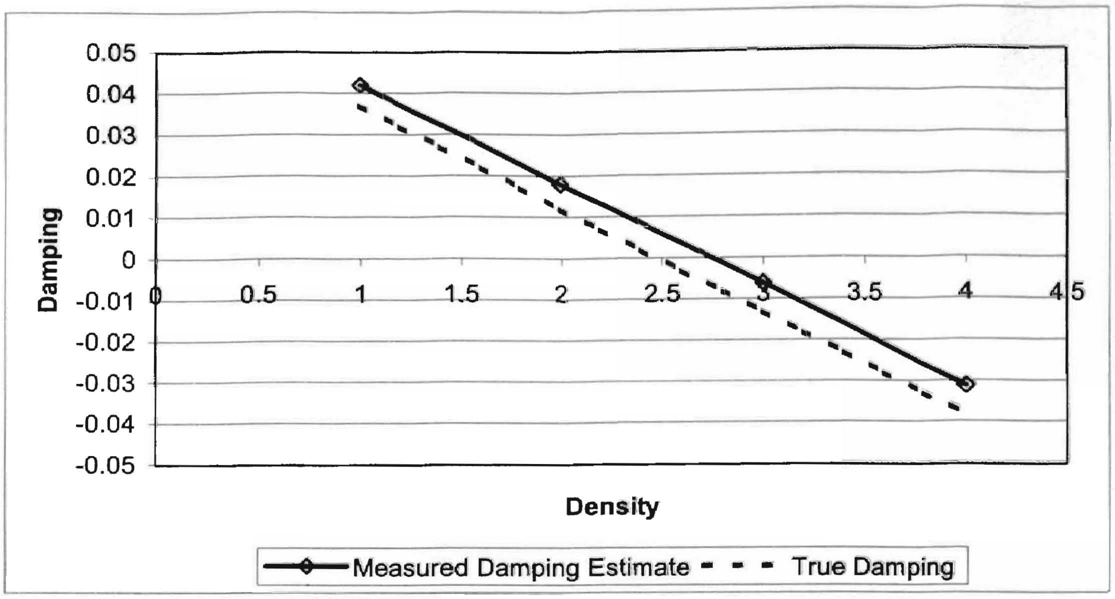


Figure 2.2 Graph of Damping versus Density

Density, ρ	Damping, ξ
1	0.042
2	0.018
2.73 (Estimate)	*0.000*
3	-0.006
4	-0.031

Table 2.1 Density Sweep Damping Estimates

2.1.2 Training Data Generation for System Identification

For linearly dynamic systems, it is possible to develop the training data for each mode in isolation from the other modes. This allows for the each mode to be trained at the same time on separate processors. This is the primary purpose of the new software package. The software will allow the selection of individual nodes and the choice of which input signal (Multistep, Modified Chirp, or Offset Modified Chirp) for the modes.

2.1.2.1 Time Savings Determination

From Cowan, 1998, the number of data points necessary to determine an ARMA model of the flow solver is *N(na,nb,nr)*. This shown in Equation 2.3, where *nr* is the number of modeshapes, *na* is the number of previous aerodynamic force values, and *nb* is the number of previous body displacements. This is the total number of data points that the training set must contain to explicitly solve for each parameter within the system model.

$$N(na,nb,nr) = nr^2 \cdot nb + nr \cdot na \tag{2.3}$$

The number of time steps is directly related to the time required to generate the data on *one* computer. The number of data points from Equation 2.3 is multiplied by the time to calculate a single time step, *dt*. Therefore the time to complete the training data is $N(na,nb,nr)\cdot dt$. Since batch processing allows that each modeshape be trained separately and in parallel, the number of model time steps needed to complete the training of one mode is Equation 2.4. This will contain the data for all the previous aerodynamic and displacement states for the single modeshape.

$$N_p(na,nb,nr) = \frac{nr^2 \cdot nb + nr \cdot na}{nr} = nr \cdot nb + na \tag{2.4}$$

The time required to finish a parallel simulation with computers of heterogeneous speed is the maximum value of the number of data points assigned to a processor multiplied by the time needed to calculate a single time step. Equation 2.5 states this relationship, where *i* is the index of the processor.

$$Time_{FINISH} = \max\left(N_i \cdot N_p \cdot dt_i\right) \tag{2.5}$$

For example, assume that 3 computer are available for parallel use and that the test case of interest has 10 modeshapes. The *na* is 3 and the *nb* is 8. The numbers of modes, total number of data points per computer, and time to complete a single model time step are listed in Table 2.2.

Computer Index, i	Number of modes, N_i	Total Data Points	Seconds per time step, dt_i	Time (Seconds)
1	4	332	0.50	166.00
2	3	249	0.75	186.75
3	3	249	1.00	249.00

Table 2.2 Heterogeneous Network Time Example

The time to complete the training set is 249 seconds. Computer 1, which has the most to calculate, finishes first since it takes less time to resolve a single time step. Computer 3 is actually the weakest performer. It would be faster to assign 5 modes to computer 1, taking 207.5 seconds to finish, and 2 modes to computer 3, requiring 166 seconds to finish. In that case the time to finish would be 207.5 seconds.

Now assume that the cluster of computers is homogenous. The variation in dt_i disappears. The new $Time_{FINISH}$ is Equation 2.6, where M is the next integer $\geq \frac{nr}{np}$, and np is the number of processors.

$$Time_{FINISH} = (M \cdot N_p \cdot dt) = \max(N_i \cdot N_p \cdot dt) \qquad (2.6)$$

Using the previous example, but setting the time required to complete a model time step to 0.50 seconds for all three computers, the $Time_{FINISH}$ is 166.00 seconds, the time computer 1 needs to finish 4 modes. Computers 2 and 3 finish 3 modes in 124.5 seconds each as shown in Table 2.3.

Computer Index, i	Number of modes, N_i	Total Data Points	Seconds per time step, dt_i	Time (Seconds)
1	4	332	0.50	166.00
2	3	249	0.50	124.50
3	3	249	0.50	124.50

Table 2.3 Homogeneous Network Time Example

The easiest time estimate is the case of *nr* homogenous computers. The $Time_{FINISH}$ becomes Equation 2.7.

$$Time_{FINISH} = N_p \cdot dt = (na + nb \cdot nr) \cdot dt \tag{2.7}$$

Since N_p is $\frac{N}{nr}$, the speed of training data generation is increased by a factor of *nr* over the serial generation. It is important to realize that artificially increasing the number of modes to finish the training data generation more quickly does *not* work. The total time to complete a training set is now a *linear* function of the number of modes, not a *quadric* as before. Since an aeroelastic system can only flutter if two or more modes are present, the distributed training set generation always reduces the time by at least one half as seen in the feasibility Section 1.5.

2.2 Software Development

This section details the software objectives, design and support programs.

2.2.1 Software Functionality

As possible methods of applying distributed processing to nonlinear aeroelastic analysis were developed, a list of objectives for the software was set. These included the primary functionality of the *Euler3d_dpp* software. This list was the starting point for creating the algorithm of the *Euler3d_dpp*. The objectives are listed here.

1. Maintain current interface and operation of *Euler3d*. The transition from the single process to the distributed version should be seamless. The input files for distributed *Euler3d_dpp* should work for single process *Euler3d*.
2. Automated sweep of densities within a flight envelope. This is one of distributed *Euler3d*'s primary purposes. This option speeds up the completion of nonlinear flutter boundary searches. It also reduces the bookkeeping aspect of a search.
3. Generation of training data in parallel. This is objective is the key goal of the project. The generation of sufficient training data to determine a system model within a time frame acceptable for flight-testing drove this much of this research.

2.2.2 Algorithm Description

Developing an algorithm can be a daunting task. Especially when it concerns a complex topic like parallel processing. In the case of *Euler3d_dpp*, the task has been simplified. The large amount of data can be divided into separate clearly defined and independent tasks that can then be sent to individual processors. In this program those simple tasks happen to be aeroelastic simulations. The trick will be to specifying what each node should simulate.

None of the algorithm objectives required any changes to the flow solver of Euler3d. This allowed algorithm design to focus on parallel software architecture. Since the slave nodes never communicate with each other, the simplest possible parallel layout was selected. Only the master, or central, node initiates communication with any other node. The slave, or computational, nodes only pass messages back to the master. This software architecture is shown in Figure 2.3.

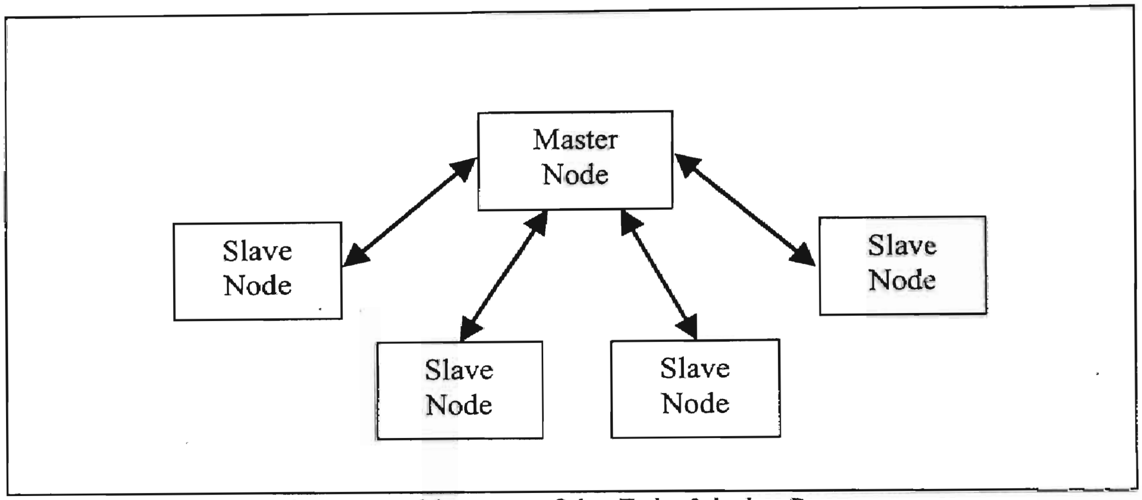


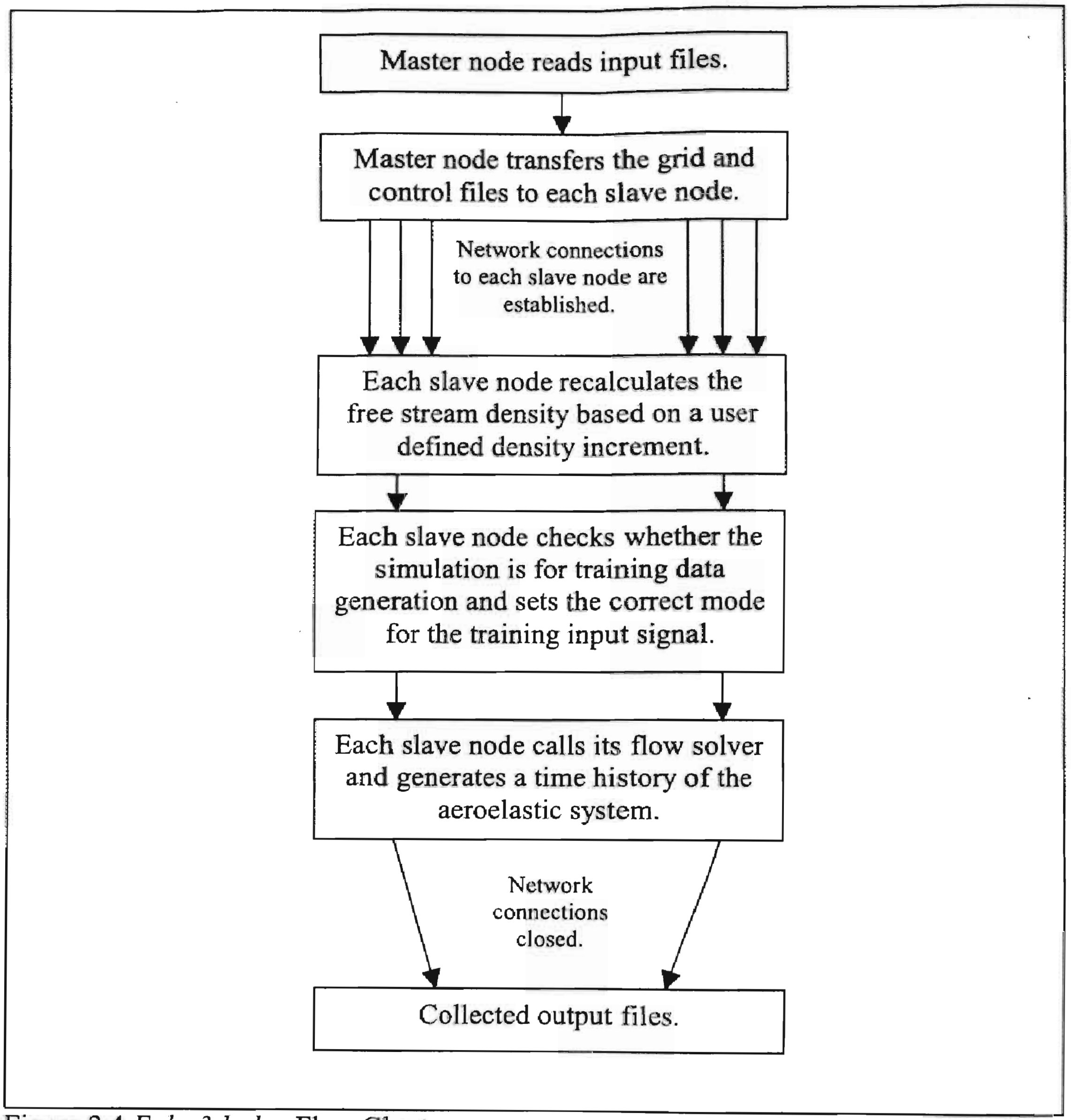
Figure 2.3 Parallel Software Architecture of the *Euler3d_dpp* Program

The distributed version of *Euler3d* needed a method for the master node, the central node, to contact the computing nodes, or slave nodes, and initiate processes on those nodes. This required that a networking interface be added into the original *Euler3d* source code. After survey of currently available message passing, the Message Passing Interface, MPI, was selected. The was chosen based on its reputation in the parallel processing community, availability of references, free software packages available on multiple internet servers, and its interface with Fortran, in which *Euler3d* is compiled. MPI also offers both a Microsoft Windows and Linux version. All parallel aeroelastic solvers in the literature used MPI. This decision also set how the program would operate.

Before the master node reads in any data, it contacts each slave node and checks that it ready for a processing request. The master node then reads in the control file, *case.con*, which contains the initial conditions (Mach, density, sonic speed, model time step, how to solve the problem and how long to run) and passes it to each slave node unaltered. The slave nodes then check that the initial conditions are correct for its assigned processor index, *np*. If a sweep of density is being done, then the slave node computes the correct density for its index. This is found from Equation 2.8, $\partial\rho$ is the increment of density for each index.

$$\rho_{np} = \rho_{base} + \partial\rho \cdot np \tag{2.8}$$

Once the density calculation is finished the master nodes reads in and sends the test case geometry, boundary conditions and the steady state values of the flow field to each slave node. The slave nodes accept these without change. The master node reads the modal deformations, and generalized mass, damping, and stiffness matrices. Each slave node receives the matrices and deformations. The slave node then checks if system identification training data is to be generated. If so the slave node determines the mode it has been selected to training by holding all modes constant with the *np+1* mode receiving a training input. The master node continues to read input files, such as the dynamic motion file for non-inertial reference frames and the forcing function file if required. A flow chart for the program is shown in Figure 2.4.


Master node reads input files.
Master node transfers the grid and control files to each slave node.
Network connections to each slave node are established.
Each slave node recalculates the free stream density based on a user defined density increment.
Each slave node checks whether the simulation is for training data generation and sets the correct mode for the training input signal.
Each slave node calls its flow solver and generates a time history of the aeroelastic system.
Network connections closed.
Collected output files.
Figure 2.4 *Euler3d_dpp* Flow Chart

2.2.3 PostProcessing Programs

The *Euler3d_dpp* program only generates the training data for the system modeling. It does not create the system models. *Cfdmdl3dsplice* is used to generate the system models. This program is an improvement of the original *cfdmdl*, which was used by Cowan [1998] to make system models from serially produced training sets. The program handles ordering the data set so that force and displacement histories are correctly listed. It performs singular value decomposition of the resulting data set. It assemblies the final parameters into formatted output files. Using this program is the second step in a system model flutter prediction.

The *cfdmdl3dsplice* program reads in the multiple *xn.dat#* files. It then reorders the data based on *na* and *nb* selected by the user. The data is ordered is as follows.

Output	Input
$f_1(k)$	$f_1(k-1),\cdots,f_1(k-na),x_1(k),\cdots,x_1(k-nb),\cdots,x_{nr}(k),\cdots,x_{nr}(k-nb)$
$\vdots$	$\vdots$
$f_{nr}(k)$	$f_{nr}(k-1),\cdots,f_{nr}(k-na),x_1(k),\cdots,x_1(k-nb),\cdots,x_{nr}(k),\cdots,x_{nr}(k-nb)$

This sets up *nr* functions for the SVD algorithm to resolve, one for each mode's forcing. Before the SVD algorithm is applied the static nonlinear values, or offsets, are removed from the forcing data. This removes a nonlinearity that the algorithm could not correctly identify. Those functions coefficients from SVD are recorded as a set of parameters in an output file. The program has an option to create more that one system model. For this a range of *na* and *nb* values are given and the program creates a model for each set.

In order to combine the training data from up to *nr* data files, the *xnmeld* program was developed. The aerodynamic forces are assumed to follow the principle of linear superposition. The nonlinear static offset is removed from each training file. The force values for each time step are summed. The static offset is added to the result. The output is the combined response of the CFD flow solver to an input on all modes. This data can then be used to assess the error in the system identification model.

When *cfdmdl3dsplice* is complete, it still has not made any predictions about the flutter. Using the formatted output of *cfdmdl3dsplice*, *asemdl3d* can search for dynamic instabilities in the combined structural and aerodynamic system models. This program can analyze the eigenvalues of the coupled system, determine error with a known system output, and run free responses of the system model. This program generates the predictions of instability and measures the error of the system model.

2.3 Cluster Design

With a complete and operating *Euler3d_dpp*, the requirements of a personal computer cluster to run the simulations became evident. The program only requires high bandwidth in the initial phase, where the master node distributes the input files to each slave node. Thereafter the MPI protocols only send small amounts of data back to the master node about the progress of each slave node. The network must also handle the steady stream of updates to the output files, such as *xn.dat#*. The initial communication, even on a low bandwidth networking medium should be an insignificant time loss compared to the time required to complete an simulation. This allowed the use of low cost off the shelf networking equipment.

The requirements other than those of the current distributed Euler3d were considered as well. In the future, it is expected that other groups investigating aeroelastic analysis will attempt to set up similar computer clusters. In an attempt to reduce as much confusion as possible, the cluster was designed with the idea of repeated replication. This design goal and the requirements of *Euler3d_dpp* lead to the current design and configuration of the cluster.

With the design goals decided, price to performance optimization was done on several configurations of cluster nodes. Early in the design, single processor machines proved to have lower price per benchmark ratings compared to dual, or multi, processor machines. The CASE lab has extensive data on the performance of the single processor Euler3d program. Since the communication medium does not affect the speed of the distributed Euler3d, the benchmarks still hold. A listing of the benchmarks available at the time of design is shown in Figure 2.5. The data from the Intel processors lies on a nearly linear trend line of CPU clock frequency versus the benchmark rating. The relationship is shown in Figure 2.6.

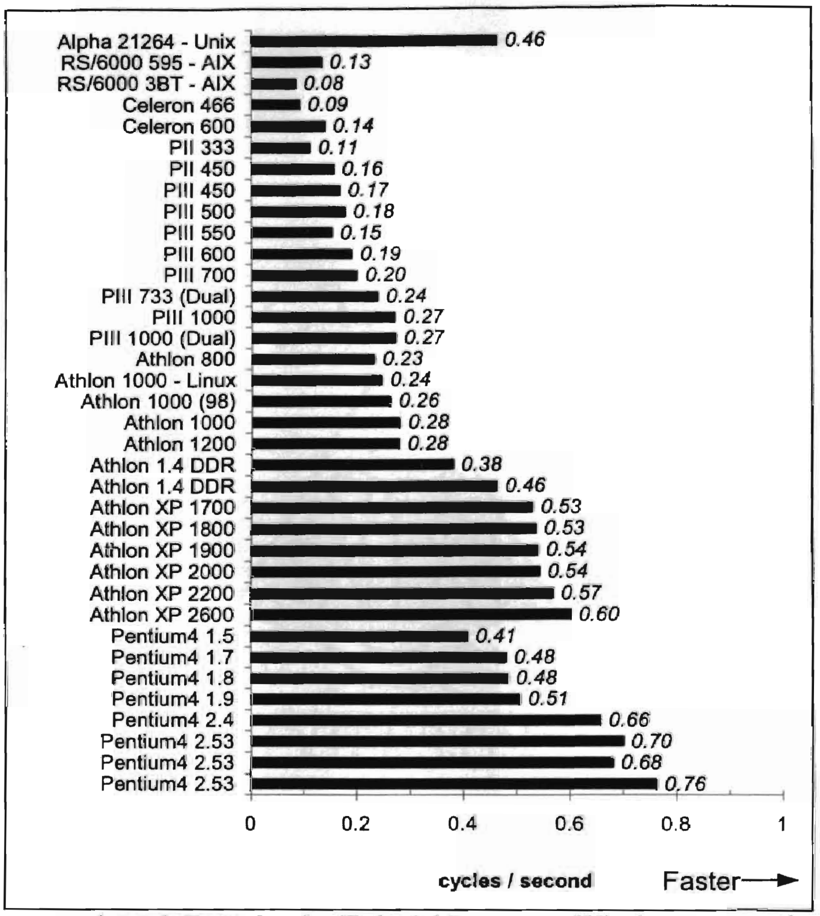


Figure 2.5 Benchmark Data for the Euler3d Program, Windows Operating System

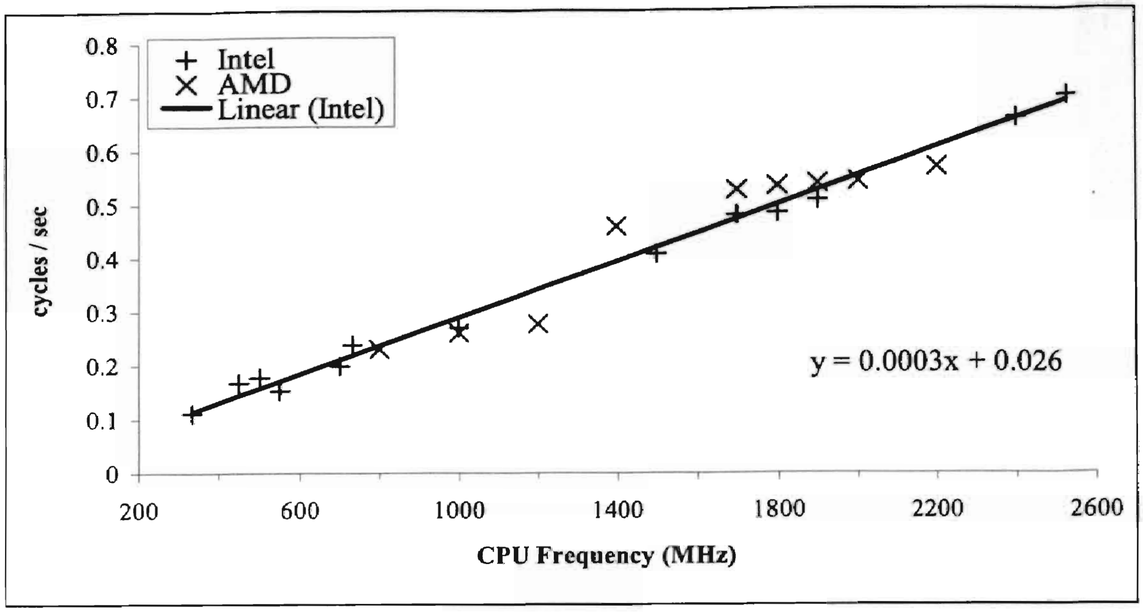


Figure 2.6 Benchmark Rating verses CPU Frequency

Using the trend line relationship, the benchmark values for the Intel Pentium 4 2.26, 2.4, 2.53, 2.66,and 2.8 GHz processors were found. They are listed in Table 2.4. The total processing power of a cluster of computers is expressed in Equation 2.9, where η_{Total} is the total cycles per second for the cluster, N is the number of cluster nodes, and η_{Single} is the cycles per second of a single processor. The total cost of the cluster is found Equation 2.10. The cost of the system components was estimated by averaging posted prices on Internet retailer WebPages. A search was performed to find the highest η_{Total} with a cost below the budget. Interestingly, the processor found to be optimal was the mid-range Pentium 4 2.53 GHz. This indicates that using the latest CPU will not necessarily produce the fastest cluster. By purchasing more mid range processors, by price and speed, more total calculations per second can be reached for the same cost. This is why many supercomputing clusters use mid range processors.

$$\eta_{Total} = \eta_{Single} \cdot N \qquad (2.9)$$

$$Cost_{Total} = N \cdot (Cost_{CPU} + Cost_{SYSTEM}) \qquad (2.10)$$

From those benchmarks and optimization analysis the best processor found was an Intel Pentium 4 2.53 GHz processor. In order to handle the large amount of data that many test cases use, one gigabyte of DDR333 RAM was specified for each computing node with the Intel 845ge chipset on the motherboard. This requirement set the value of $Cost_{SYSTEM}$, the cost of an assembled computer minus the processor. A complete description of the slave nodes is in Table B. 2.

Processor, Intel Rating	*Single Processor Benchmark*	*Number of Computers*	*Cluster Benchmark Estimate*
2.26	0.61	10	6.10
2.40	0.65	9	5.85
2.53	**0.76**	**9**	**6.84**
2.66	0.80	8	6.40
2.80	0.84	7	5.85

Table 2.4 Table of CPU Comparison for Cluster Performance

For the communication medium between the master and slave nodes, Fast Ethernet was selected. Ethernet has been established as the standard computer networking protocol and was available as an option on the motherboards considered optimal. The switch was selected was the Hewlett Packard Procurve 2124 fast Ethernet switch. This switch has 24 ports and can handle full duplex communication on all of them. This switch was selected for price, In addition to the inexpensive Ethernet, a KVM (*K*eyboard/*V*ideo/*M*ouse) switch was used to control all computers with only one monitor, mouse and keyboard, the layout for the KVM switch is in Figure 2.8. To simplify the network administration a commercial router and firewall were purchased. This allowed the entire cluster to be control and operated with only the requirement of the user copying the input files to the master node and initiating a distributed Euler3d_dpp simulation.

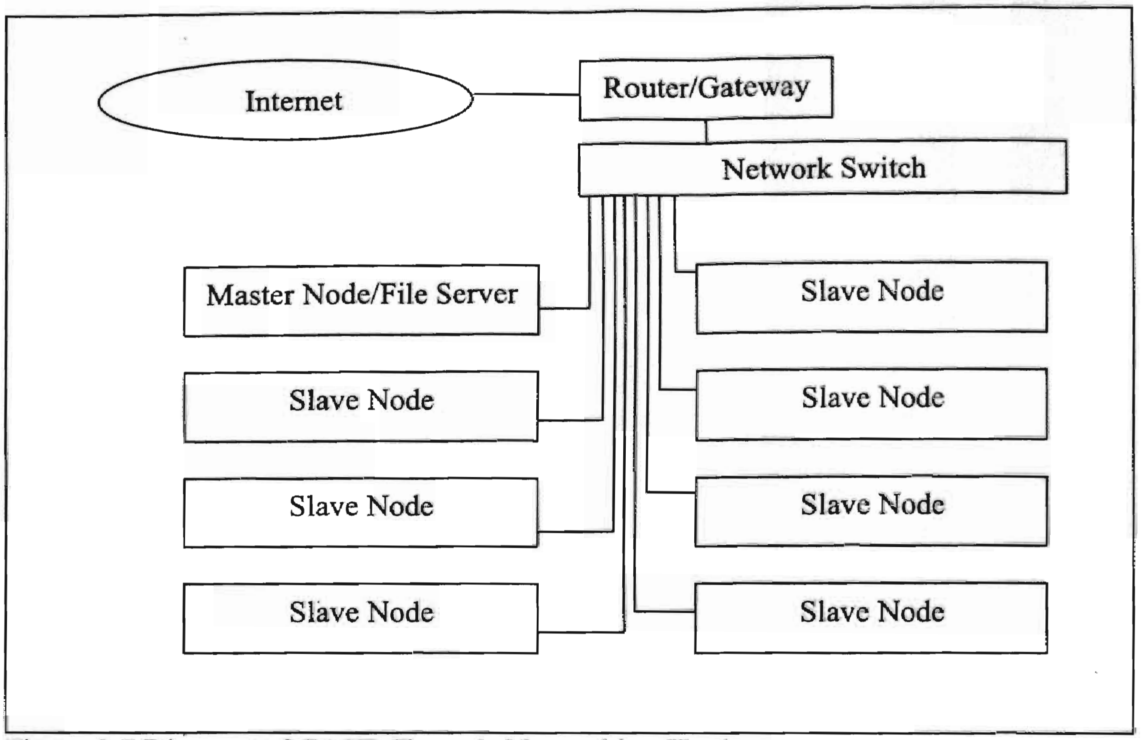


Figure 2.7 Diagram of CASE Cluster's Networking Hardware

The cluster was named CASE cluster, after the lab where it was designed and assembled. The assembled cluster is shown in Figure 2.9. The numerous cables and power cords are hidden behind the cluster. The control terminal for the KVM is shown in Figure 2.10. A more complete description of the cluster assembly is in Appendix B.

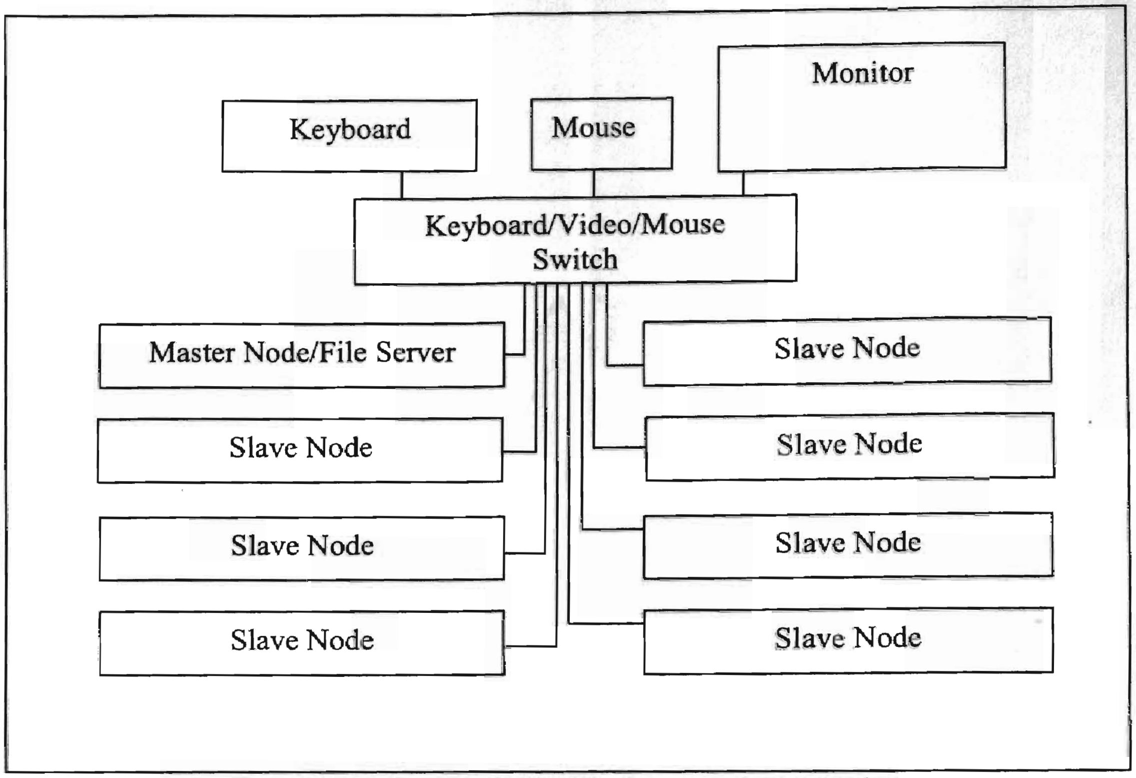


Figure 2.8 Diagram of CASE cluster's KVM Hardware and Connections

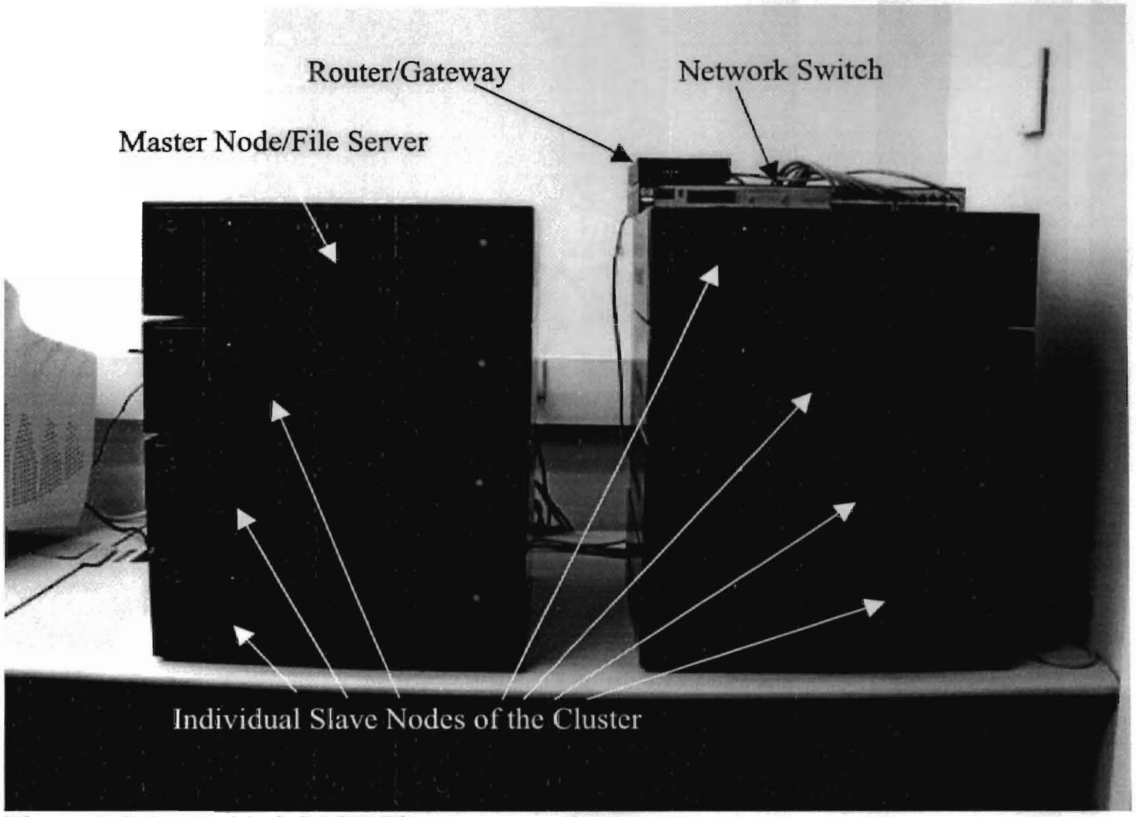


Figure 2.9 Assembled CASE Cluster

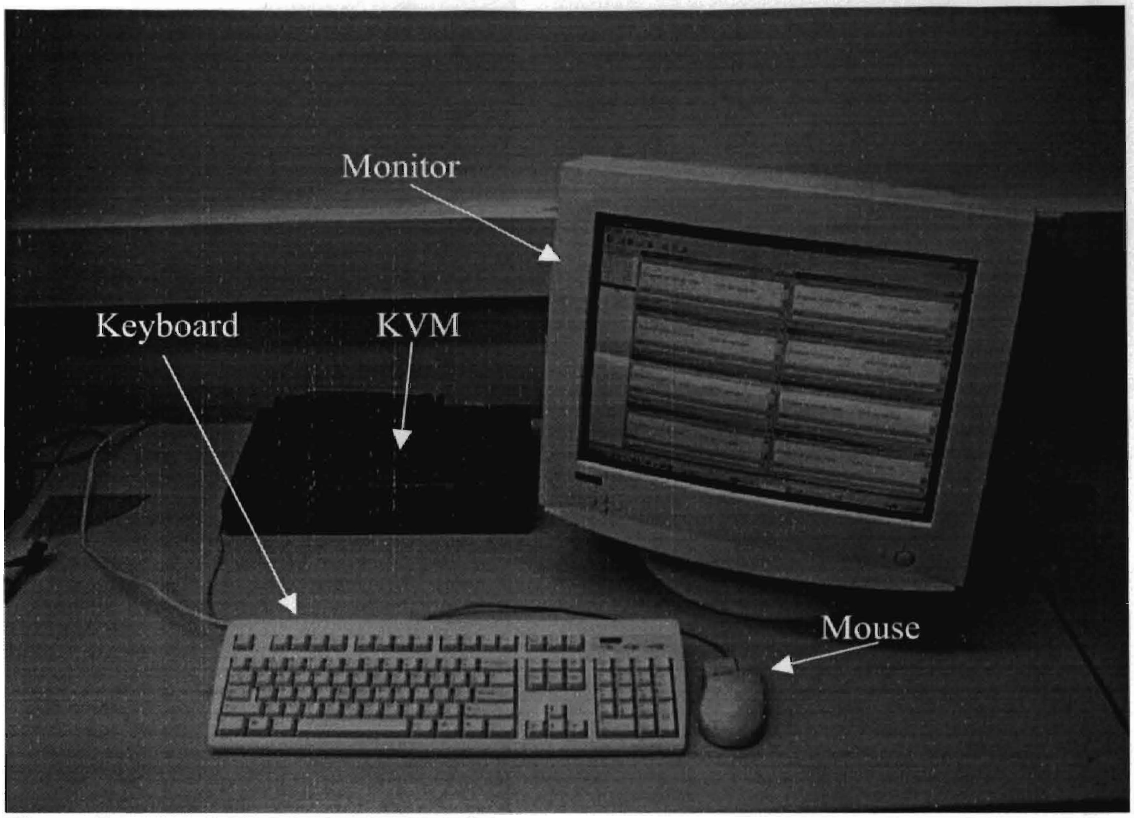


Figure 2.10 Control Terminal of CASE Cluster

CHAPTER 3

RESULTS

Using the methodology and program developed in chapter 2, the aeroelastic characteristics of several three dimensional test cases were investigated. The intent is to validate the distributed processing procedure for the STARS program suite and demonstrate how to implement the procedure on real test cases. All the examples shown here are commonplace with the aeroelastic literature and have already been analyzed by the STARS codes.

The distributed processing will be shown to save significant time in all three test cases over the serial training signal and manually initiated density sweeps. All computational work was performed on CASE cluster.

3.1 AGARD 445.6

The AGARD 445.6 wing configuration is a standard aeroelastic test case. It was investigated experimentally at NASA's Langley Research Center [Yates, 1987]. A view of the mesh is shown in Figure 3.1. The grid has 69,630 nodes and 373,798 tetrahedral elements.

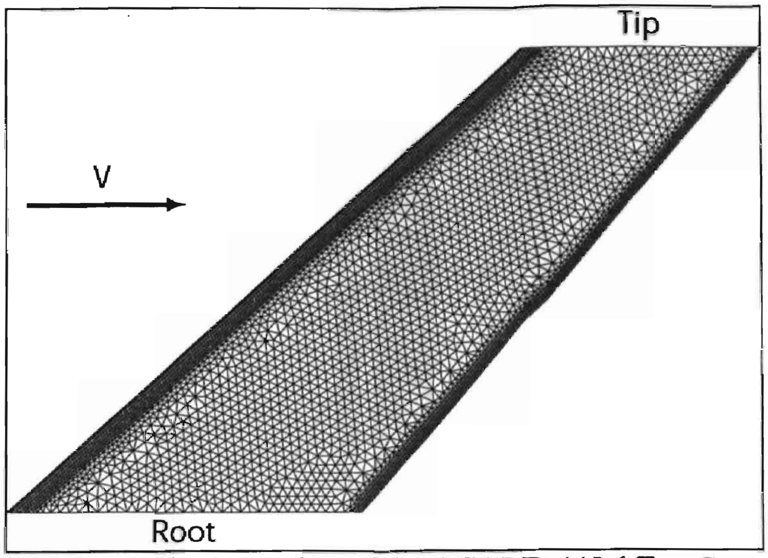


Figure 3.1 Planform View of the AGARD 445.6 Test Case

For this analysis only two mode shapes were used, the first two eigenvectors of the natural vibration analysis of the structure. Since the flutter boundary of this case is known, the first two modes will sufficiently model the instability. These two modes represent first wing bending and first wing torsion, and are shown in Figure 3.2 and Figure 3.3. The wing bending of Figure 3.2 includes the original undeformed mesh for reference. The natural frequencies of the two modes are 9.6 and 38.2 Hz respectively. The simulation will be run at Mach 0.96 with standard air and a time step of 2.5E-4 seconds. Since the model is in the transonic range, the Euler FEM model must be used to solve for the aerodynamics forces.

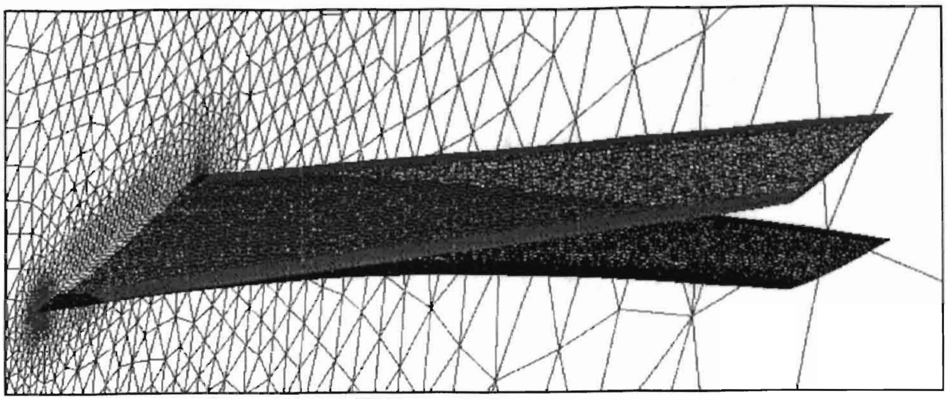


Figure 3.2 Mode 1, First Bending, of the AGARD445.6 Test Case at 9.6 Hz

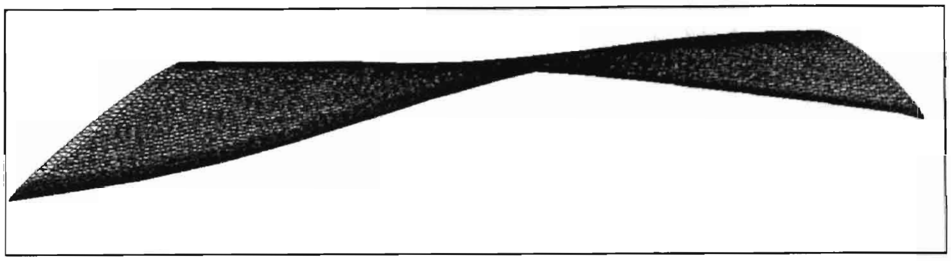


Figure 3.3 Mode 2, First Torsion, of the AGARD445.6 Test Case at 38.2 Hz

3.1.1 System Identification

From Appendix A, the values of *ratio* and *omega* were set to 512.9 and 1.015E-4 for this test case.

3.1.1.1 Training Data Generated in Serial

The serial training was run for 370 time steps. The training data from the serial generation is shown in Figure 3.4 and Figure 3.5. These two graphs show the input signal of in the general displacement of the mode and the forces results from those displacements. The two figures should be view together as force changes on mode 1 due to mode 2 motions are depicted on the graph of mode 1. The training required 8.4 hours on one node of CASE cluster.

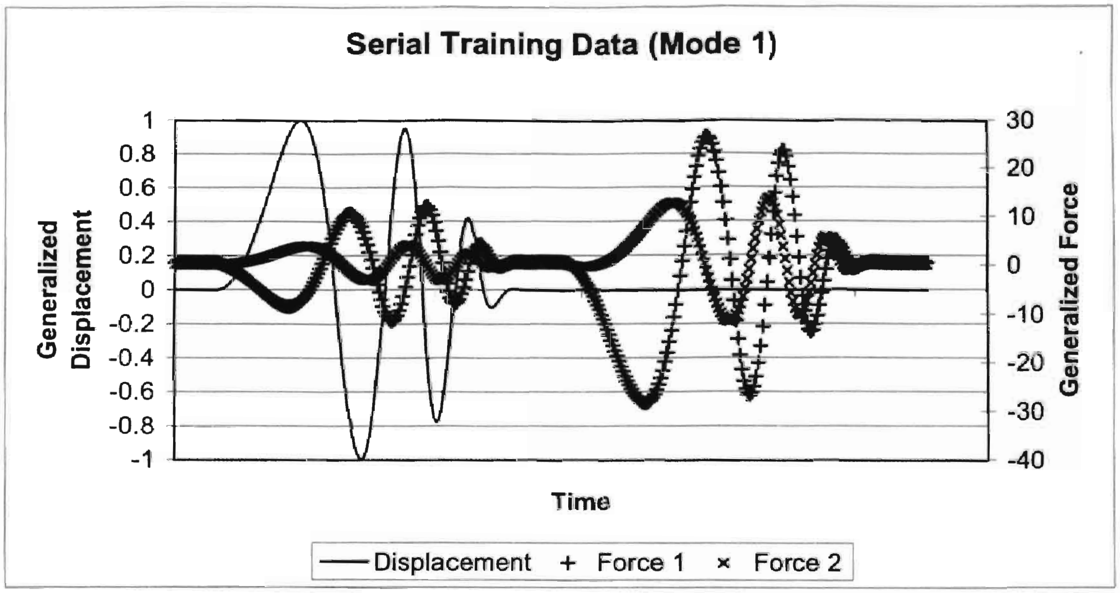


Figure 3.4 Training Data for Mode 1, First Bending, from Serial Generation

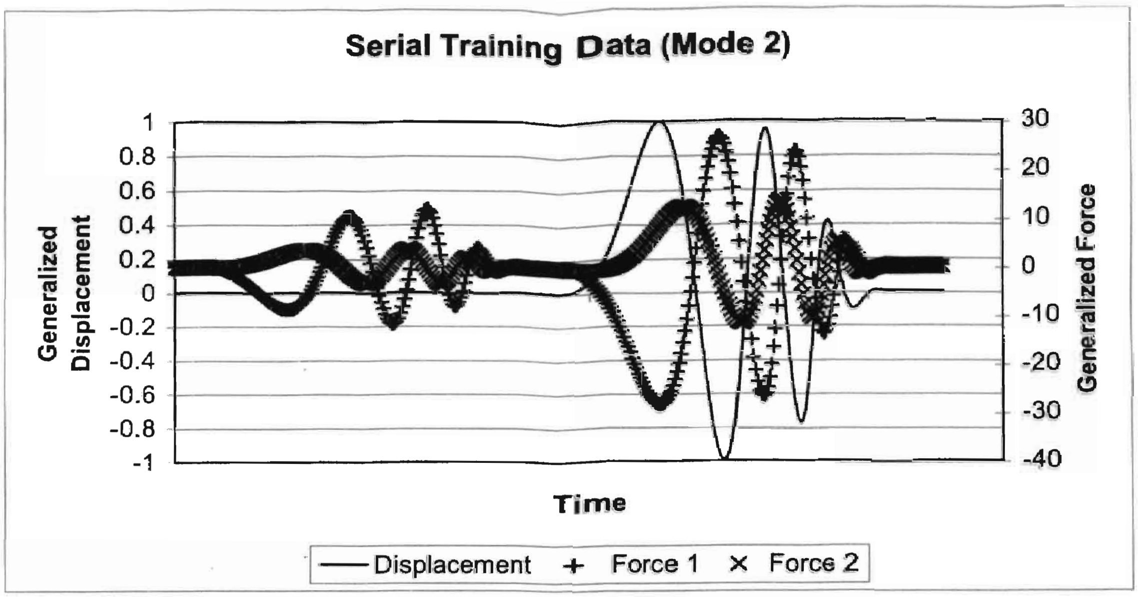


Figure 3.5 Training Data for Mode 2, First Torsion, from Serial Generation

The training data was used to generate a range of model with varying *na* and *nb* values. Using the training data as a standard, the error was found for each model. The model with *na* of 4 and *nb* of 7 was selected for its low error of 0.000952 and 0.00174 for mode one and two respectively. This RMS error is found by using the displacement and force data from the training set as input to the forcing model. The output of the model is compared to the actual value from the training set. The error is found by summing the square of all errors then dividing by the number of data points in the training set. The square root of the average squared error is normalized with respect to the largest force on the mode. The error is expressed in Equation 3.1, where f_T is the force value from training data, f_M is the force estimate of the model, f_{MAX} is the largest force value for the mode of interest from the training set, and n is the number of data points in the training set. The error measurement is effectively a statement of the average error at each point normalized by the largest force in the training set. So for mode 1 the average error is one thousandth, 0.000952, of the largest single force value on mode 1.

$$Error = \frac{\sqrt{\frac{\sum_{n}(f_T - f_M)^2}{n}}}{f_{MAX}} \tag{3.1}$$

The model predicted instability at 0.398 psi, with mode 1 diverging dynamically. This instability boundary is found by combining the aerodynamic forcing model with the structural dynamics in a state space formulation. This readily allows eigenanalysis of the total system. Sequentially larger densities are used to scale the forcing function until one value causes an unstable eigensystem. This prediction agrees with experimental and computational values from the literature.

3.1.1.2 Training Data Generated with Parallel Distribution

For the parallel training two nodes of CASE cluster were used. Simulations were run for 200 time steps each. The training data from the parallel generation are should in Figure 3.6 and Figure 3.7. The two CASE nodes required 4.3 hours each to complete the job.

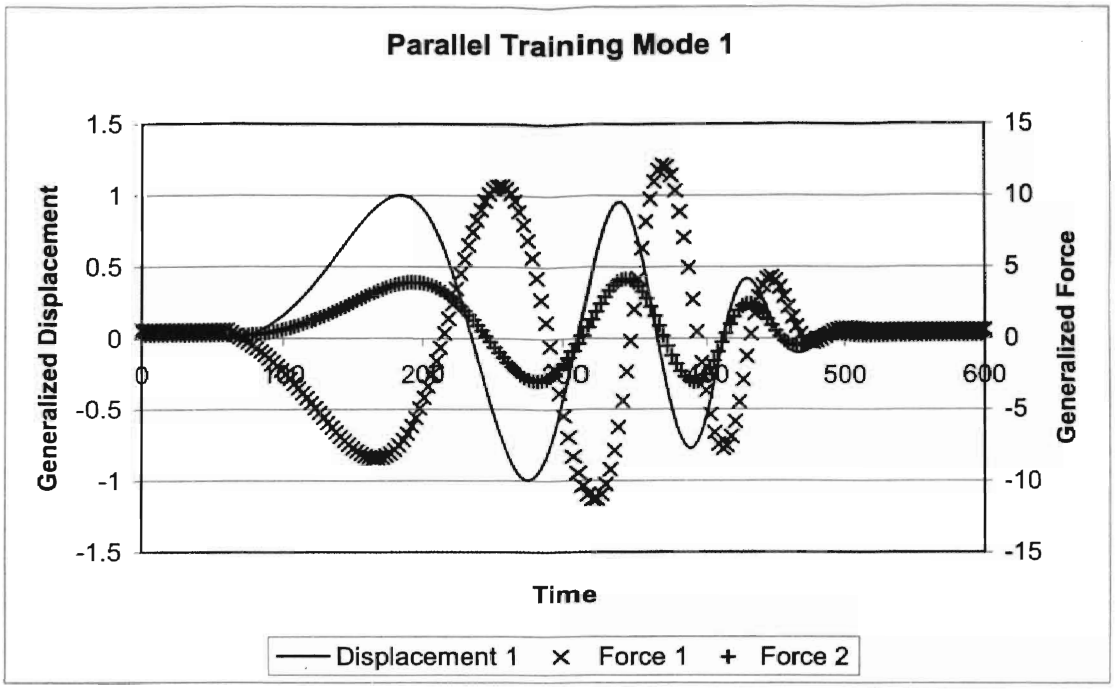


Figure 3.6 Training Data for Mode 1 of the Parallel Training Set

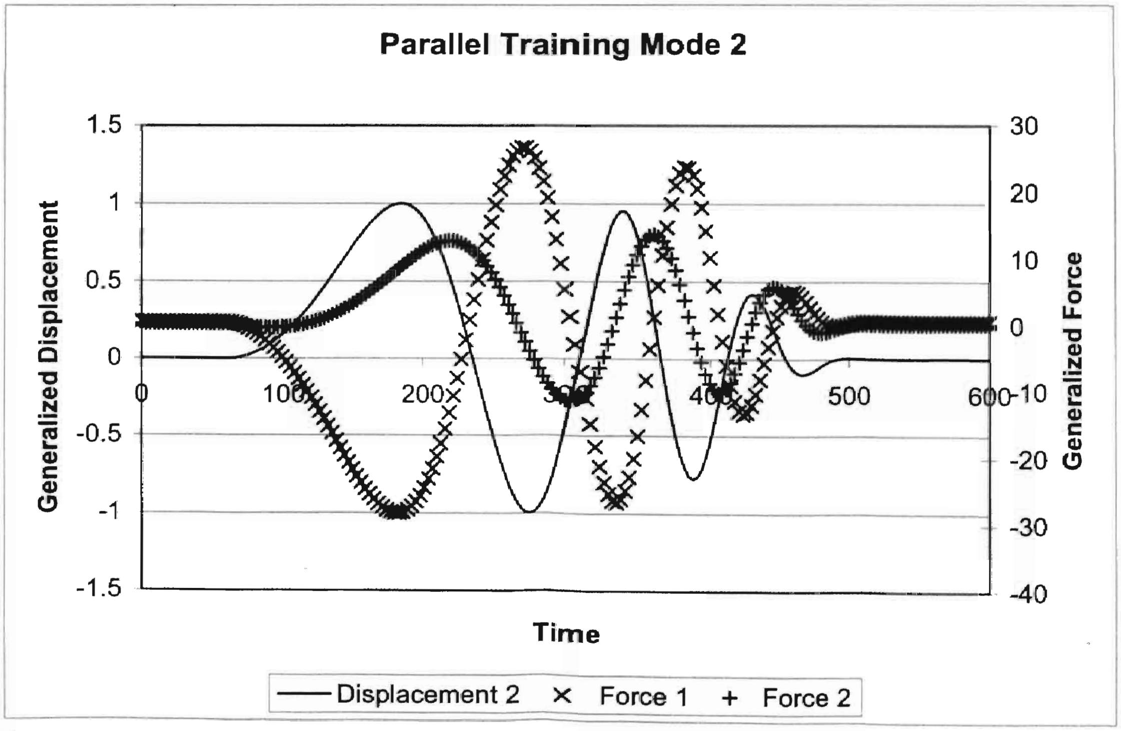


Figure 3.7 Training Data for Mode 2 of the Parallel Training Set

Again the training was used to determine a variety of system models. The error was found for each mode compared to the combined training set. The model with the lowest error was the model with *na* of 4 and *nb* of 8. The error for 4-7 model, the same model parameters used in the serial set, was slightly higher. In order to maintain consistency between the models, a 4-7 was used for the parallel training as well. The errors for that model were 0.000892 and 0.00241, with mode 1 dynamically diverging. The predicted flutter pressure is 0.402 psi. Which again agrees with the literature results. The comparison of time from serial and parallel is shown in Figure 3.8.

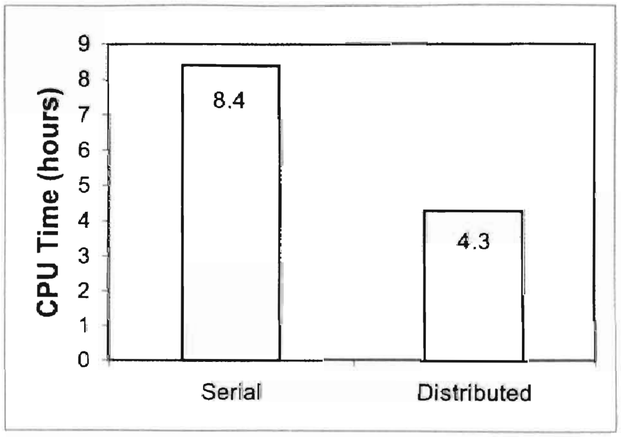


Figure 3.8 CPU Time to Generate Training Data for the AGARD445.6

Since the parallel and serial training data agree on flutter prediction, the instability can be confirmed with a density study around the prediction.

3.1.2 Density Sweep

To confirm the predictions from section 3.1.1, a density sweep from below the flutter prediction to above it was run. The prediction of 0.4 psi yields a density of

0.55E-10 $\frac{slinch}{in^3}$. The values for the simulations around the flutter boundary are found in Table 3.1. The free responses were run to 2000 time steps.

Test Condition	Formula	Density, ρ	Dynamic Pressure, q
Below Model Flutter Prediction	0.80 × Prediction	0.440E-10 $\frac{slinch}{in^3}$	0.32 psi
Below Model Flutter Prediction	0.90 × Prediction	0.495E-10 $\frac{slinch}{in^3}$	0.36 psi
Model Flutter Prediction	1.00 × Prediction	0.550E-10 $\frac{slinch}{in^3}$	0.40 psi
Above Model Flutter Prediction	1.10 × Prediction	0.605E-10 $\frac{slinch}{in^3}$	0.44 psi
Above Model Flutter Prediction	1.20 × Prediction	0.660E-10 $\frac{slinch}{in^3}$	0.48 psi

Table 3.1 The Initial Conditions for Density Sweep of AGARD445.6

All the free responses were started with the same initial condition, a small velocity on mode 1. Five nodes of CASE cluster were used to simulate the five free responses. The results of the five responses are listed in Table 3.2. Three simulations are compared in Figure 3.10 to Figure 3.12.

Dynamic Pressure, q	CPU (hours)	Damping on Mode 1, ξ
0.32 psi	22.6	0.01276
0.36 psi	22.4	0.01077
0.40 psi	22.5	0.00251
0.44 psi	22.6	-0.00141
0.48 psi	23.1	-0.01274

Table 3.2 Dynamic Pressures and the Damping for the AGARD445.6

Using the data compiled from Table 3.2, the damping trend was derived. It is shown in Figure 3.9. Normally only four points are used in the trend estimate; however, the five used here work well in a 4th order relationship. Notice that the curve fit for the damping trend is not valid outside the range of data points used to generate it.

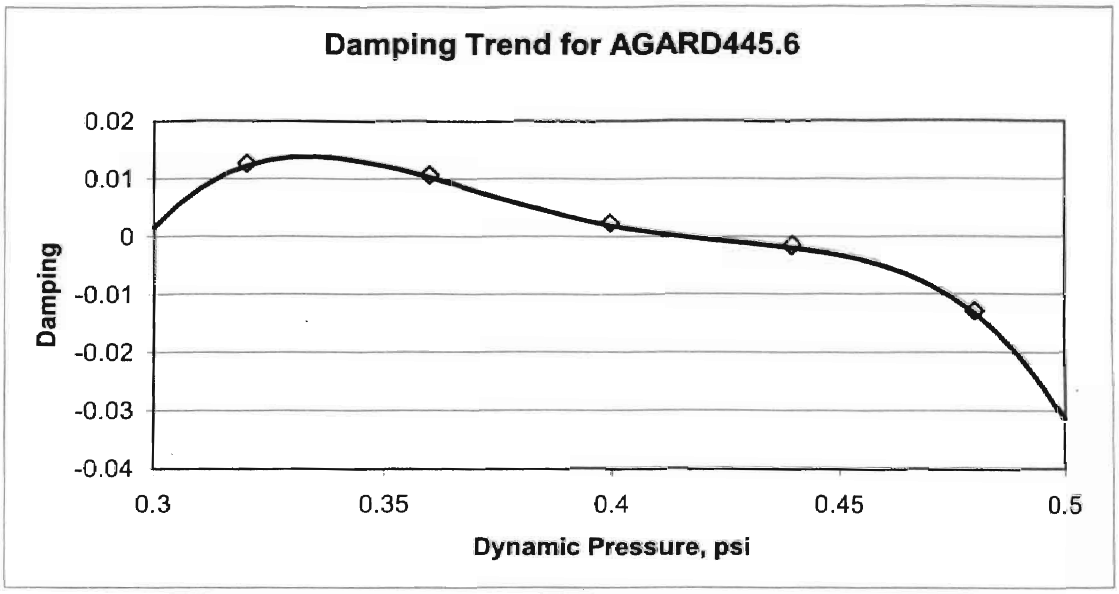


Figure 3.9 Damping Trend for the AGARD445.6

The damping trend predicts a flutter boundary at 0.422 psi. This matches very well with the values reported in [Yates, 1987]. It is a 5.5% difference from the system identification estimate. This is with the uncertainty of the model. Reducing the time of the model may reduce the difference between the system identification and the density sweep estimates.

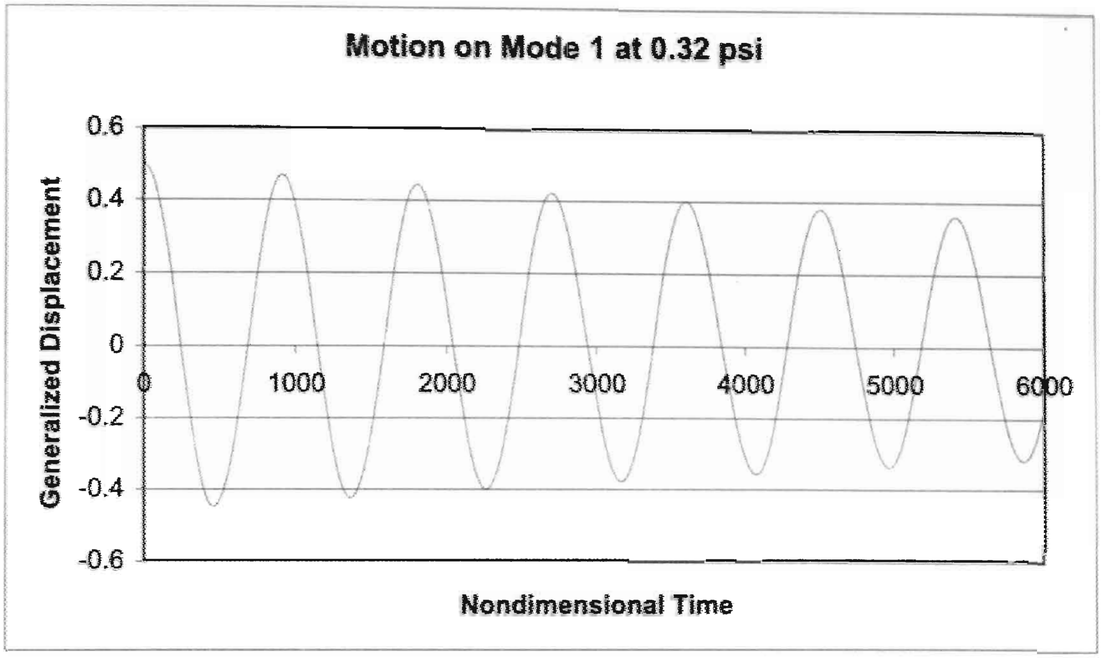


Figure 3.10 Free Response on Mode 2 at 0.32 psi for the AGARD445.6

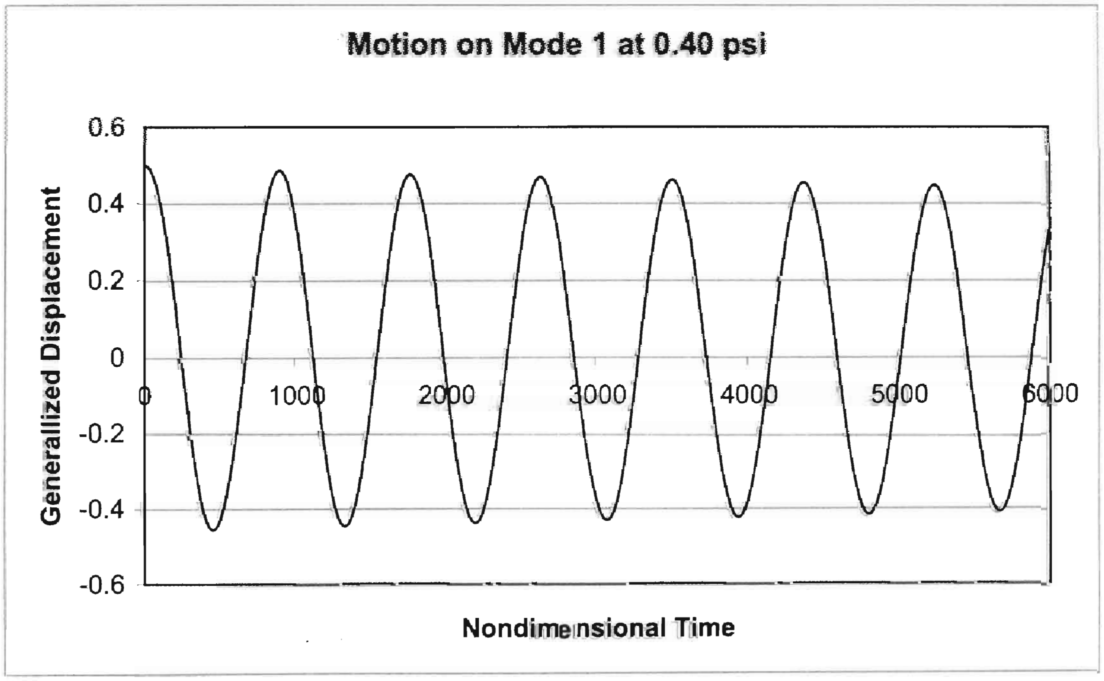


Figure 3.11 Free Response on Mode 2 at 0.40 psi for the AGARD445.6

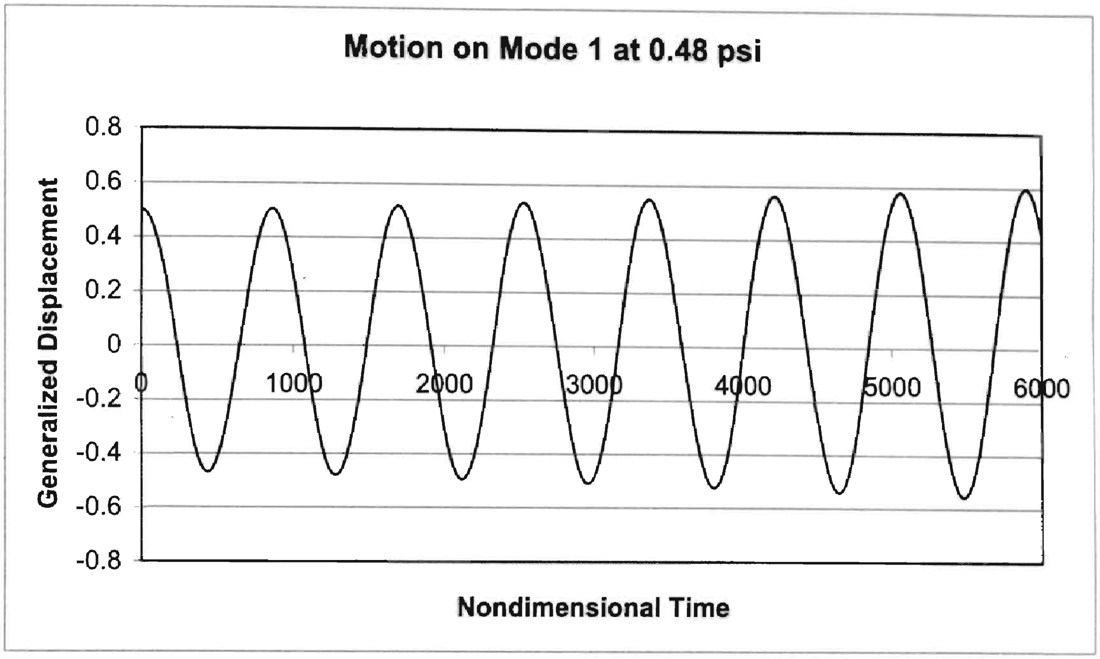


Figure 3.12 Free Response on Mode 2 at 0.48 psi for the AGARD445.6

3.1.3 Comparison of CPU time

The time required for each response is listed in Table 3.2. A sequential run of all five would require 113.2 CPU hours. The use of five nodes of CASE cluster reduced the time be 79.6 percent. That translates into a speedup of 4.9 in the density sweep. The graph in Figure 3.13 shows how the computation was accelerated.

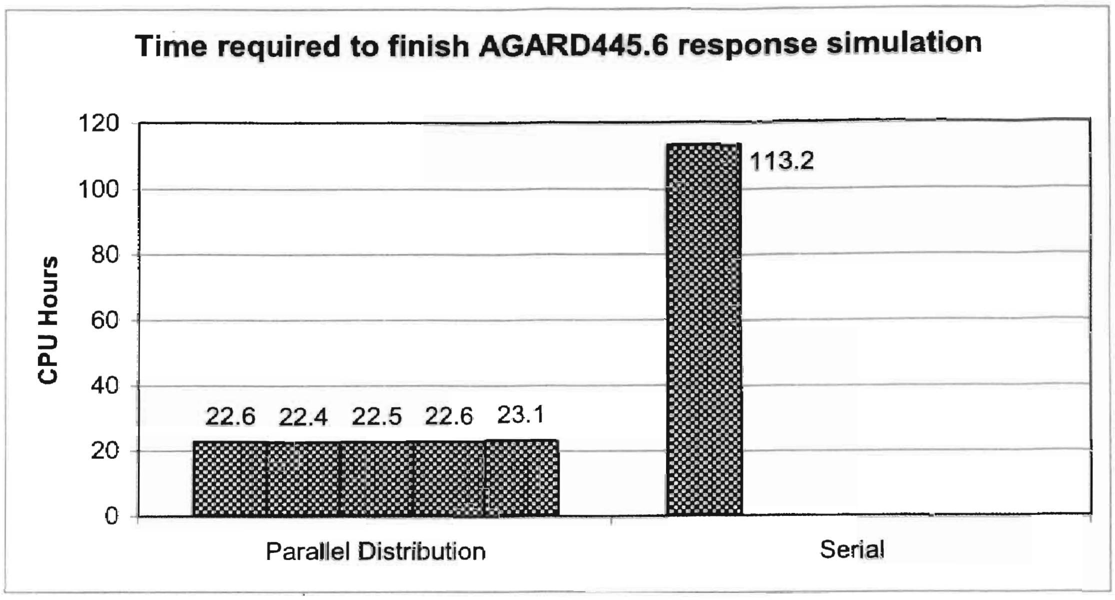


Figure 3.13 Time Required to Finish AGARD445.6 Response Simulation

The complete time required for both a system model prediction and confirming density sweep is shown in Figure 3.14. The difference between the serial method and the distributed parallel is 94.2 CPU hours. The parallel method is a 77% reduction in computing time. By using *Euler3d_dpp*, four days of computation were saved.

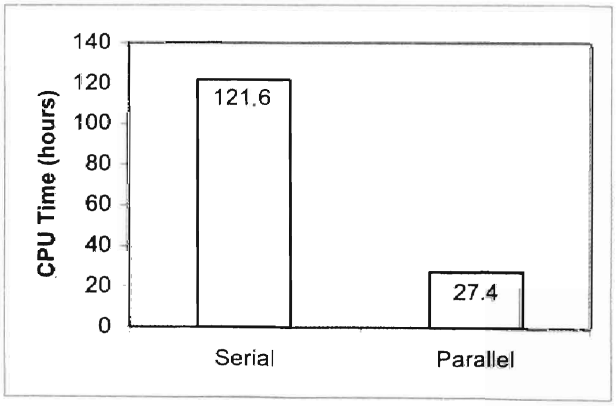


Figure 3.14 Combined Prediction Time for AGARD445.5

3.2 2×1 Plate

In this test case, a three dimensional plate is used with the piston perturbation method to identify a flutter point and confirm it with free responses. The aluminum plate in this model is one tenth of an inch thick, with the standard properties of aluminum. The CFD grid is shown in Figure 3.15.The structural FEM analysis found six modeshapes of interest. They are shown in Figure 3.16. The frequencies range from 589 to 1702 hertz. The fluid is standard air at Mach 2 with a time step of 4.565E-6 seconds.

Since the distributed methodology should work with any valid CFD solver, the full Euler FEM solver was replaced with the piston perturbation model. The airflow over the elastic plate is at Mach 2; which Hunter [1997] found was within the valid range for the piston perturbation method.

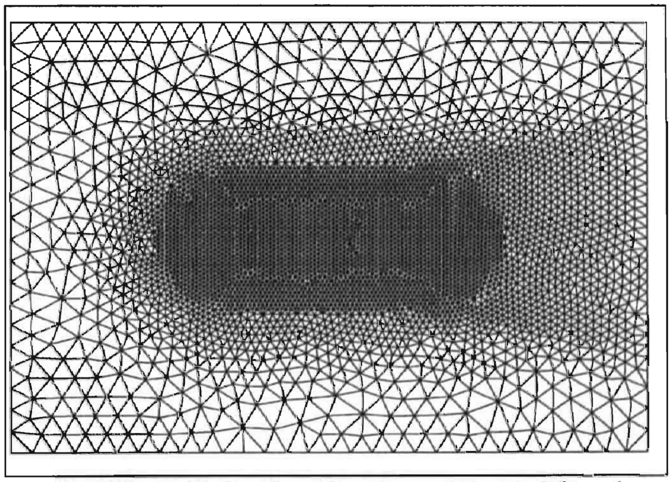


Figure 3.15 CFD Grid for the Aluminum Elastic Plate (top view)

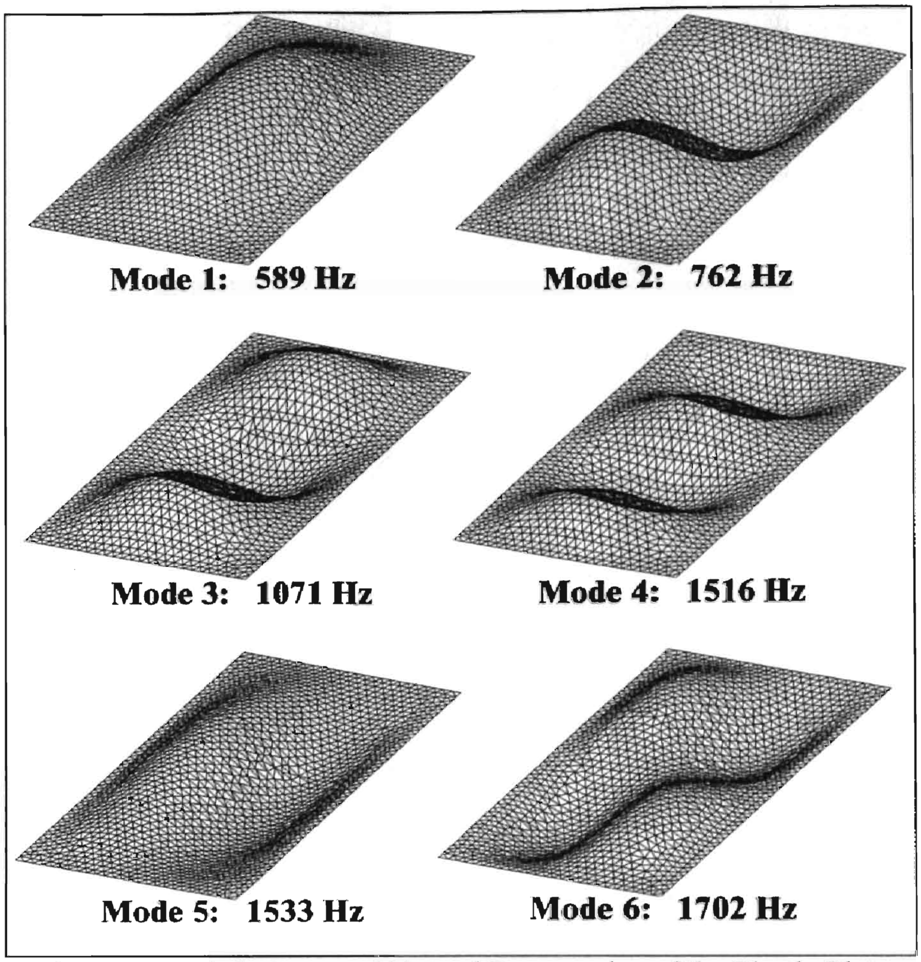


Figure 3.16 Modeshape and Natural Frequencies of the Elastic Plate

3.2.1 System Identification

Training data was generated using both parallel and serial techniques. For this test case the modified chirp was used. In order to sweep the correct frequency range with the necessary number of data points for an overdetermined system model, the *ratio* and *omega* values of the modified chirp were set to 4.75 and 8.535, respectively. The training input signals were run for a 1700 timesteps per mode.

3.2.1.1 Training Data in Serial

Using *cfdmdl3dsplice*, a system model was generated using *na* of 0 and *nb* of 11. Since the piston theory does not use the previous force values in its calculations, the zero value of *na* is expected. The serial training data predicted dynamic instability at 39180.3 psf. The dynamic divergence of mode 3 is shown in Figure 3.17. The training data required 45.91 seconds to complete on one node of the CASE cluster described in section 2.3.

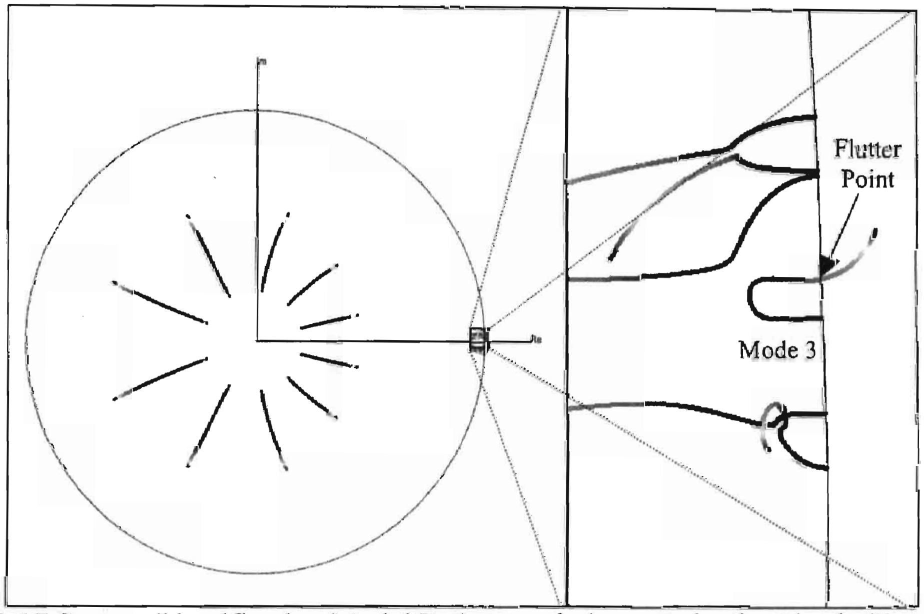


Figure 3.17 System Identification Model Estimate of Flutter Point for Elastic Plate

Using *asemdl3d*, the error was determined for mode. The error for the serial training was 0.0015366, 0.00077565, 0.000774, 0.000777, 0.000775, and 0.000775, for modes 1 to 6 respectively. So the mode with the greatest error in the serially trained model is 0.1% on mode 1. The training data and model prediction are compared in Figure 3.18. This graph only covers the effect of the input signal on mode 1 to the forces on mode 1. This is the mode with the most error. Notice that the error is not readily apparent.

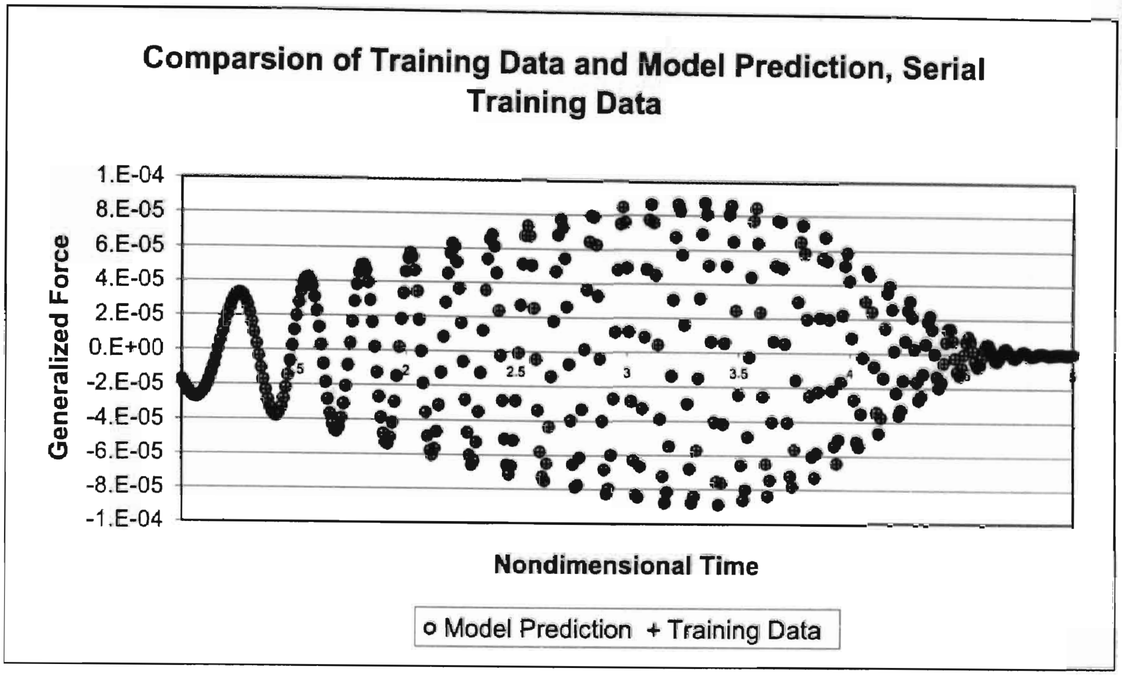


Figure 3.18 Comparison of Training Data and Model Prediction

3.2.1.2 Training Data in Parallel Distribution

The parallel training data was generated on six nodes of CASE cluster. The nodes required 8.06, 8.12, 8.19, 8.06, 8.08, and 8.17 seconds to finish. This is 17.8% of the time required for the serial simulation. The same model order from serial training was used to create a system model from the parallel trained data. This model predicted flutter at 39179.9 psf as well. Using xnmeld the parallel training data was combined into a single time history for error calculations. The errors of the parallel based model were 0.000775, 0.000775, 0.000775, 0.000775, 0.000775, and 0.000775. Interestingly, the error is that same on all six modes; however the prediction agrees with the serial training derived model. In the next section, it will be shown to agree with the density sweep results as well.

The training data developed in serial and parallel both predicted the same flutter point. This flutter point will be used to determine the initial densities of the density sweep.

3.2.2 Confirming Density Sweep

The flutter prediction of 39179.9 psf at Mach 2 yields an air density of 0.0162 slugs per cubic foot, or 6.8 times the density of air at sea level. In order to confirm the instability of mode 3 at that pressure, a series of free responses was simulated. Seven responses were used in a range from seventy percent of the system model predicted flutter boundary to thirty percent above the boundary. The seven densities and associated pressures are listed in Table 3.3. The seven responses were run on seven nodes of CASE cluster.

Test Condition	**Formula**	**Density, ρ**	**Dynamic Pressure, q**
Below Flutter Prediction	0.70 × Prediction	0.01133 $\frac{slug}{ft^3}$	27425 psf
Below Flutter Prediction	0.80 × Prediction	0.01295 $\frac{slug}{ft^3}$	31343 psf
Below Flutter Prediction	0.90 × Prediction	0.01457 $\frac{slug}{ft^3}$	35261 psf
Flutter Prediction	1.00 × Prediction	0.01620 $\frac{slug}{ft^3}$	39179 psf
Above Flutter Prediction	1.10 × Prediction	0.01780 $\frac{slug}{ft^3}$	43097 psf
Above Flutter Prediction	1.20 × Prediction	0.01942 $\frac{slug}{ft^3}$	47015 psf
Above Flutter Prediction	1.30 × Prediction	0.02104 $\frac{slug}{ft^3}$	50933 psf

Table 3.3 The Initial Conditions for Density Sweep of the Elastic Plate

Using seven nodes of CASE cluster the free response of the seven densities was studied. The results are shown in Table 3.4. The damping of mode 3 is used, as it is the mode that the system model predicts will diverge. Analysis of the other modes showed that only mode 1 would decay above the flutter boundary, so any mode's damping trend should find the same instability point.

Dynamic Pressure, *q*	**CPU seconds**	**Damping on Mode 3, ξ**
27425 psf	31.11	0.04514
31343 psf	30.94	0.05463
35261 psf	31.16	0.02596
39179 psf	30.78	0.00271
43097 psf	30.55	-0.00728
47015 psf	30.78	-0.00567
50933 psf	30.86	-0.00222

Table 3.4 Dynamic Pressures and Damping for the Elastic Plate

The data from Table 3.4 can be used to find a damping trend. The four points closest to the cross over point were used to fit a cubic function to the data. The results are shown in Figure 3.19. This trend line predicted a flutter boundary at 39879 psf. This trend is interesting in that it seems to indicate that the plate may return to a dynamically stable condition at higher pressures. This phenomenon is known to occur in aeroelasticity; however, this is most likely a result of projecting a curve fit outside its range of validity. The density sweep estimate is 2% off of the system identification predictions. This is within the uncertainty of a damping estimate.

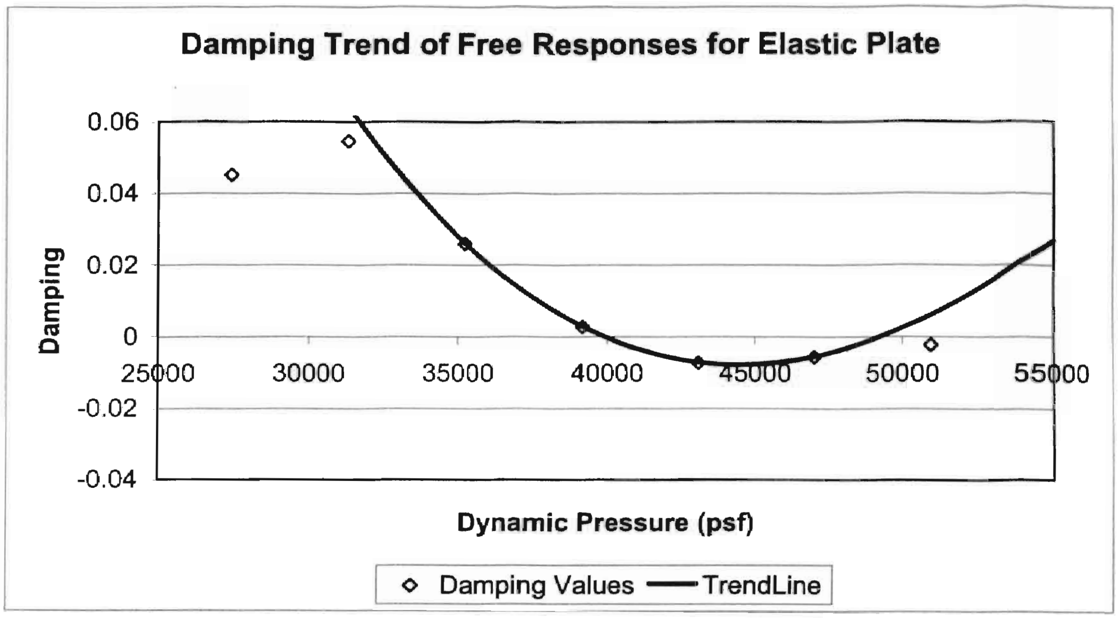


Figure 3.19 Damping Trend for the Elastic Plate

3.2.3 Time Comparisons

The elastic plate has six modes. From the time savings estimate in section 2.1.2.1; the parallel training should take 16.6% percent of the time a serial training set does. The serial training set required 45.19 seconds to finish. The parallel training took 8.19 seconds for the slowest processor. The parallel processing took 18.1% of the time the serial did, a speedup of 5.5 over the serial method. This is shown in Figure 3.20.

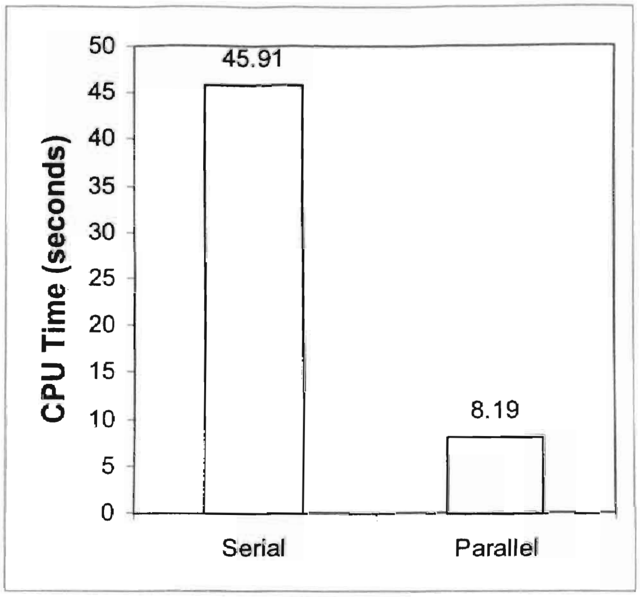


Figure 3.20 Comparison of Parallel and Serial Training Generation Times

This was an efficiency of only 92 percent for training signal generation. Density sweep results show a much better performance. The sweep required 31.16 seconds to finish. Using the fastest processor, 30.58 seconds, as the base line for the speed up, the speedup is 6.86. This is a 98% efficiency for the density sweep. This is shown in Figure 3.21.

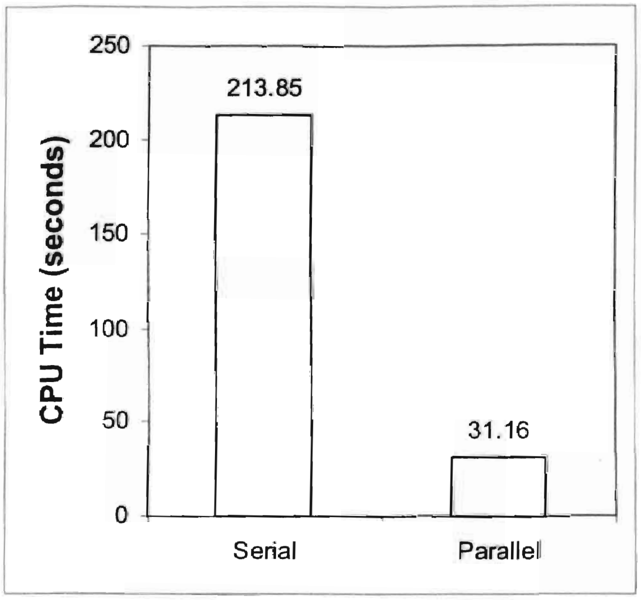


Figure 3.21 Parallel and Serial Free Response Set Simulation Times

For the combined system identification and free response, the serial required 259.76 seconds, while the parallel only needed 39.35 seconds. This is an 85 percent reduction of the time required to complete the analysis.

3.3 Generic Hypersonic Vehicle

The Generic Hypersonic Vehicle (GHV) consists of a typical hypersonic vehicle aerodynamic configuration with complicated structural modes. A long oblate fuselage with rear fins dominates the GHV. The fuselage base is blunt. Figure 3.22 shows the GHV geometry. The grid has 58,511 nodes and 321,755 tetrahedral elements.

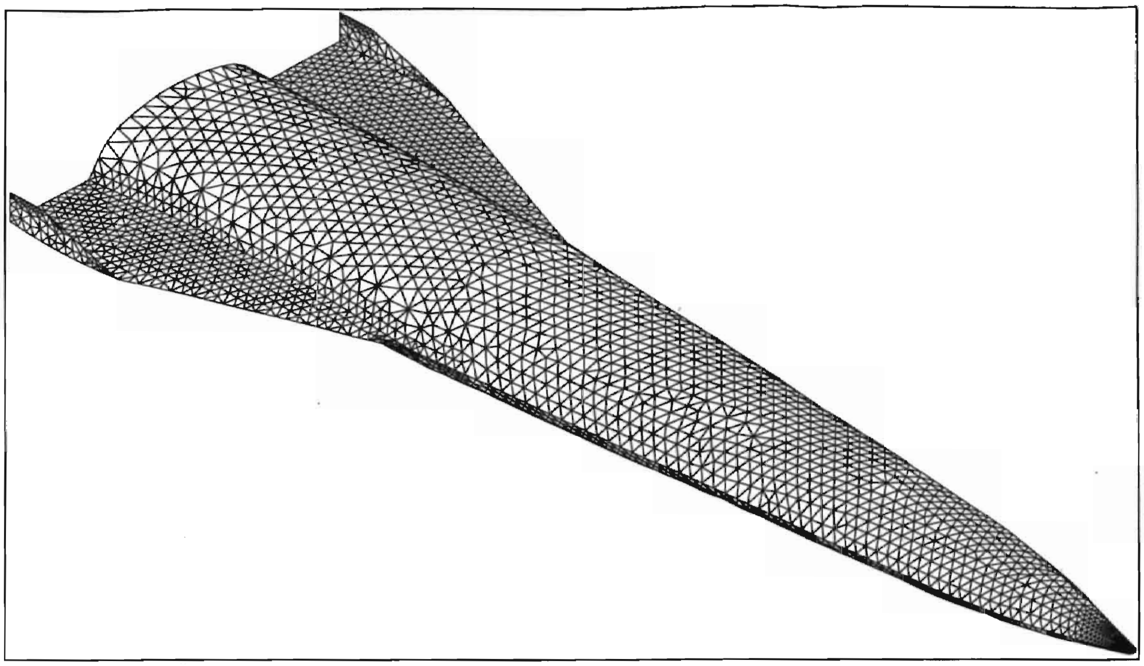


Figure 3.22 GHV Geometry

Nine structural modes were retained from the free vibration analysis with frequencies ranging up to 9.4 Hz. The simulation was run at Mach 2.2 using standard air and a time step of 5.3E-3 seconds in the Euler equation solver.

3.3.1 Density Sweep

The GHV posed an interesting possibility. This is the most structurally complex case tested in this work. Its nine modeshapes will need a large training set. It is possible that a straight density sweep could find the flutter boundary faster than a system identification model would. To test this, the density sweep was run before the system identification study.

The starting pressure was set at 103.2 psi. This is 90 percent of the value reported by Cowan [1998] from a system identification study. The pressure increment was arbitrarily set at 12.9 psi. Each time history was run to 320 time steps; this was enough for six cycles on the lowest frequency mode. The nine modes of the GHV result in nine damping values. To save confusion, and space, only the values for the second mode are listed in Table 3.5. In general, the mode of divergence is easily identified from visual inspection of the plotted time history.

Dynamic Pressure, q	**CPU (hours)**	**Damping on Mode 2, ξ**
103.2 psi	2.26	0.03392
116.1 psi	2.24	0.03746
129.0 psi	2.30	0.03696
141.9 psi	2.26	0.05131
154.8 psi	2.31	-0.6177E-04
167.7 psi	2.22	-0.04274
180.6 psi	2.28	-0.05290
193.5 psi	2.27	-0.07259

Table 3.5 Dynamic Pressures and the Damping of Their Response for the GHV

Looking at the damping trend, it appears that the dynamic pressure of 154.8 psi is the flutter point as its damping is near zero. To check that the damping values are reasonable the time history of the suspected flutter boundary and the pressure directly above and below are plotted in Figure 3.23, Figure 3.24, and Figure 3.25.

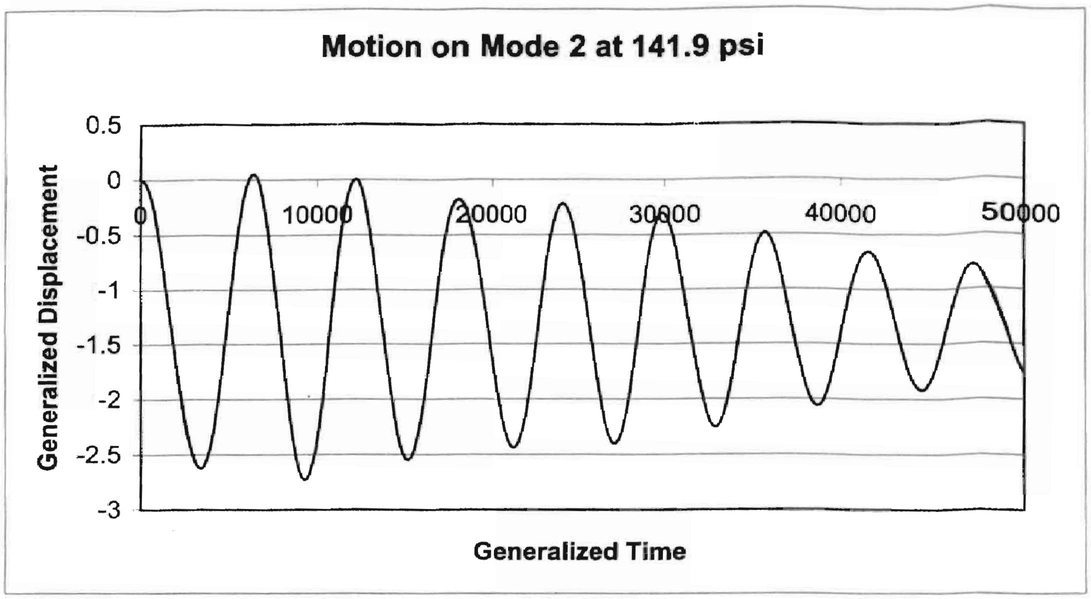


Figure 3.23 Response of Mode 2 at 141.9 psi for the GHV

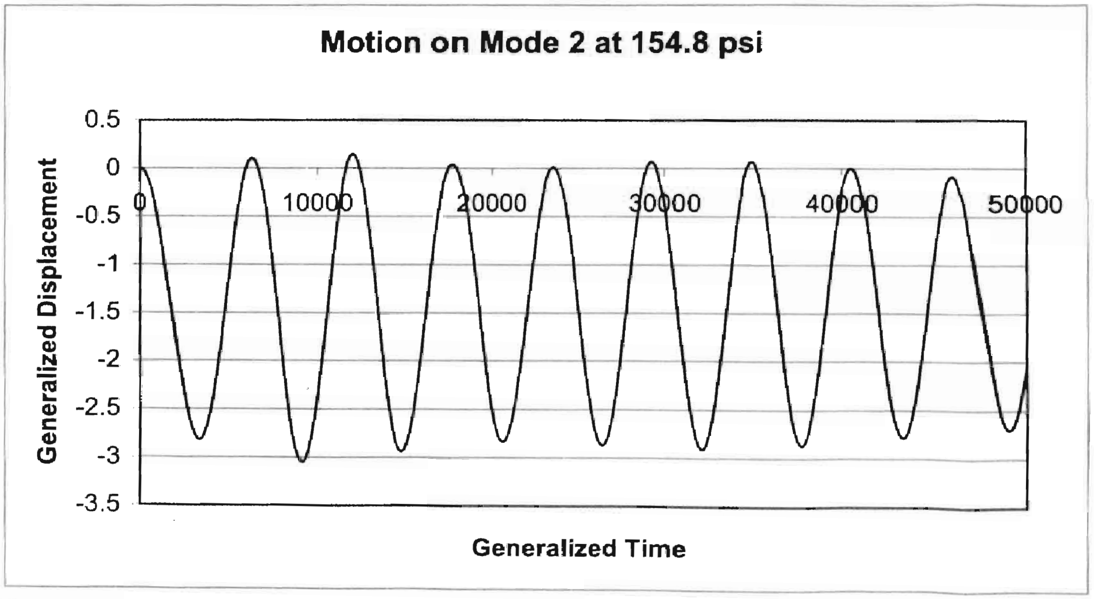


Figure 3.24 Response of Mode 2 at 154.8 psi for the GHV

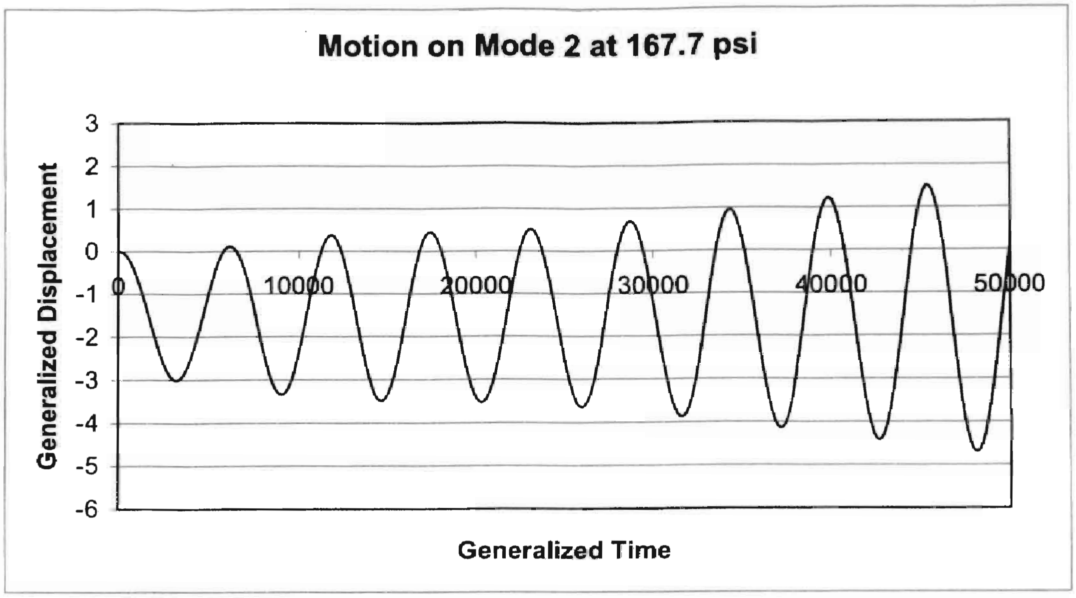


Figure 3.25 Response of Mode 2 at 167.7 psi for the GHV

It would appear that the 154.8 psi response is neutrally damped. In aeroelastic prediction this should be an acceptable value. For the purpose of this study, assume that the 141.9 to 167.7 psi range must be investigated for more resolution. Since the cluster finished this sweep in about 2.3 CPU hours, an aeroelastic analyst might decide to spend that second half of the day refining the prediction. A second sweep, dividing the pressure range into 8 new pressures was set up. The results of that pressure sweep are in Table 3.6.

Dynamic Pressure, q	CPU (hours)	Damping on Mode 2, ξ
141.9 psi	2.26	0.05131
144.8 psi	2.24	0.04624
147.7 psi	2.23	0.05485
150.6 psi	2.29	0.06916
153.5 psi	2.25	0.001982
154.8 psi	2.31	-0.6177E-04
156.3 psi	2.29	0.00090
159.2 psi	2.23	-0.01632
162.1 psi	2.26	-0.03071
165.0 psi	2.28	-0.02903
167.7 psi	2.22	-0.04274

Table 3.6 Refined Pressures and the Damping of Their Response for the GHV

From this refined study, the flutter point clearly lays between 153.5 and 159.2 psi. The flutter prediction can be set at 153.5 psi with a high degree of confidence. The time histories for the four pressures in the range are shown in the following figures.

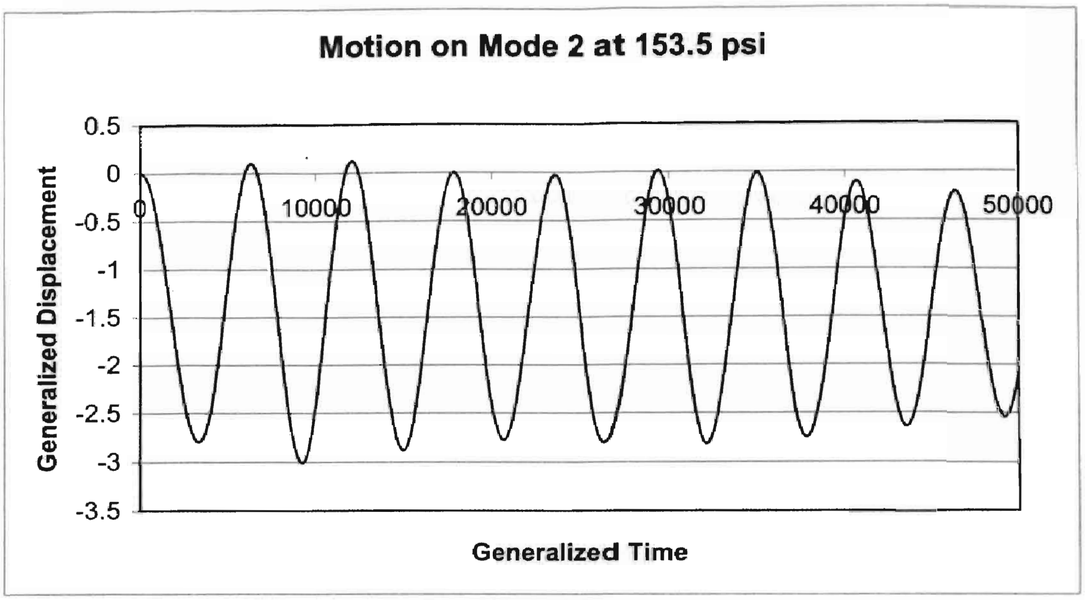


Figure 3.26 Response of Mode 2 at 153.5 psi for the GHV

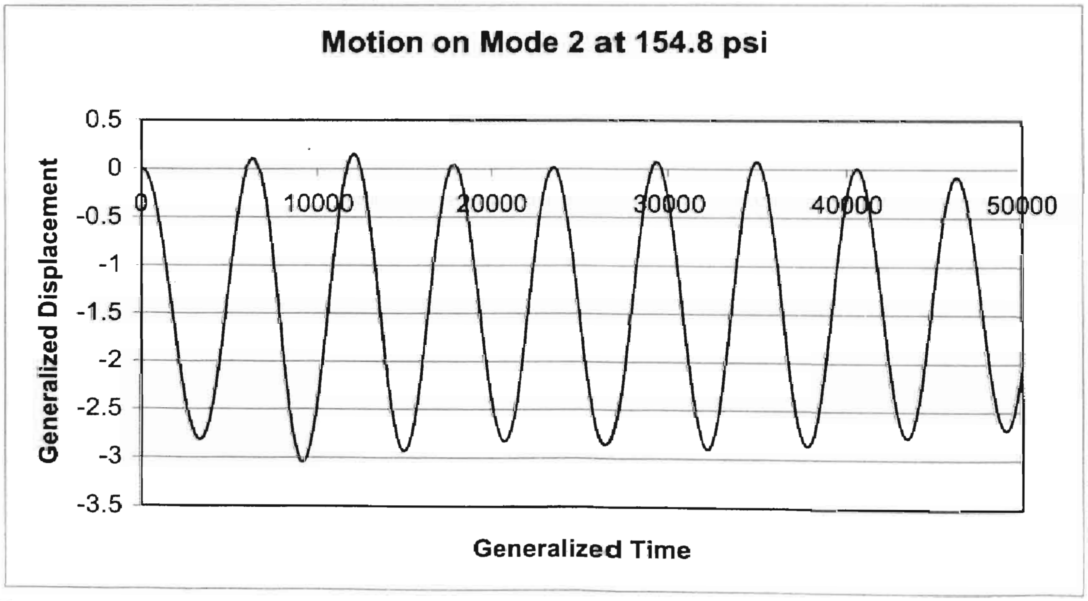


Figure 3.27 Response of Mode 2 at 154.8 psi for the GHV

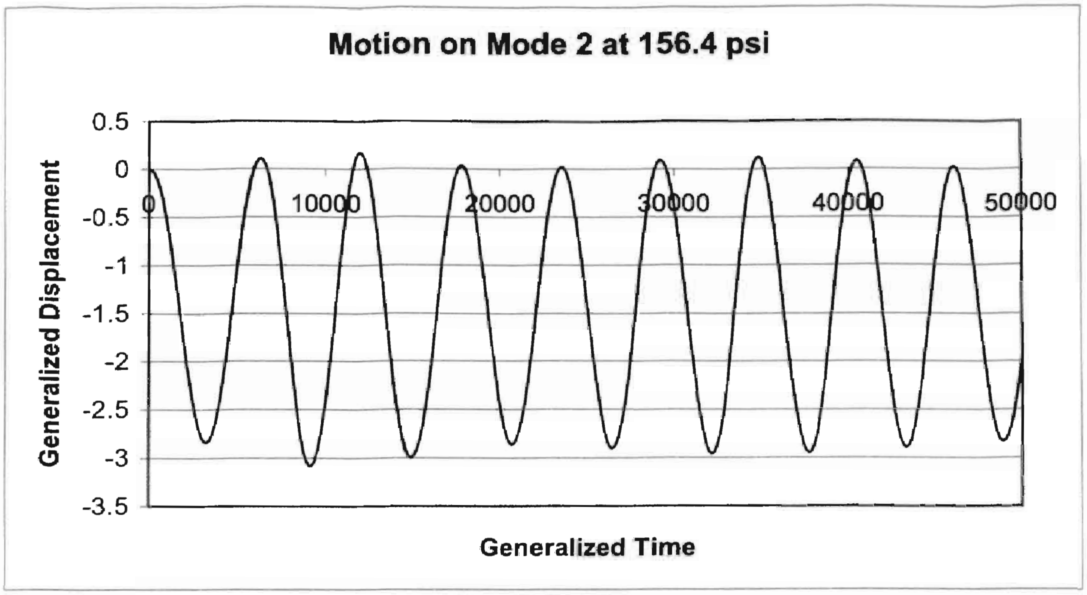


Figure 3.28 Response of Mode 2 at 156.4 psi for the GHV

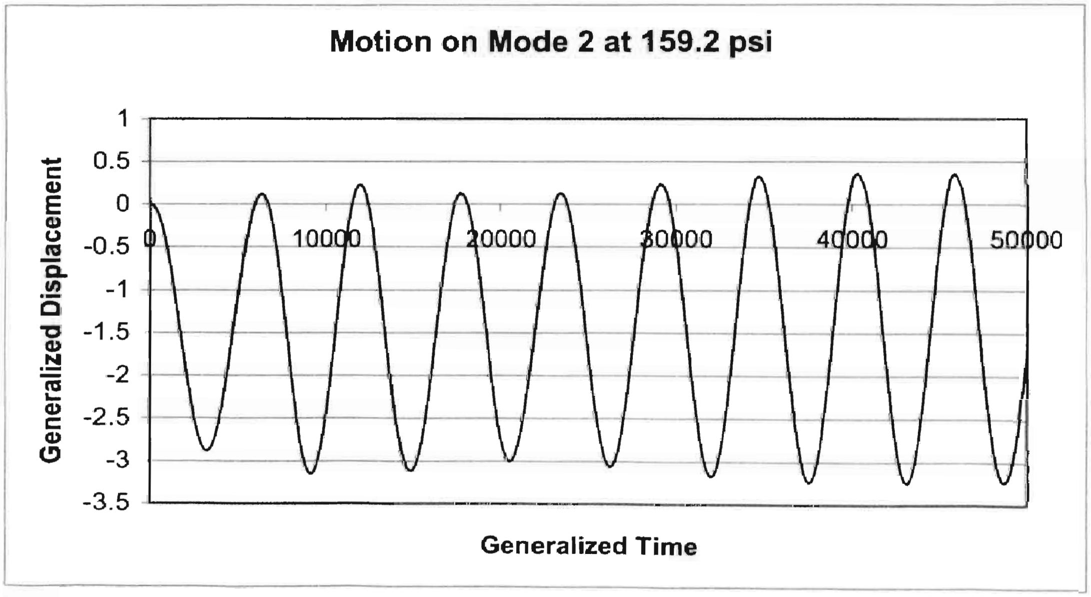


Figure 3.29 Response of Mode 2 at 159.2 psi for the GHV

The first sweep of pressures required 2.31 CPU hours to complete, the longest time of any node to finish. The second required 2.29 CPU hours. If all the response had been run in serial the total time would be 36.21 CPU hours, 18.14 in the first sweep and

18.07 in the second sweep. This is best represented in the bar graph of Figure 3.30. The distributed parallel processing completed the task in 12.7 percent of the time a serial job would require. Finding the speed, the inversion of time, that is a speedup of 7.87 over a single processor. The parallel processing with 8 processors achieved 98.4 % efficiency with respect to the average speed of the processors in the cluster. This imperfect efficiency is due to the parallel sweep waiting on the slowest processor to finish.

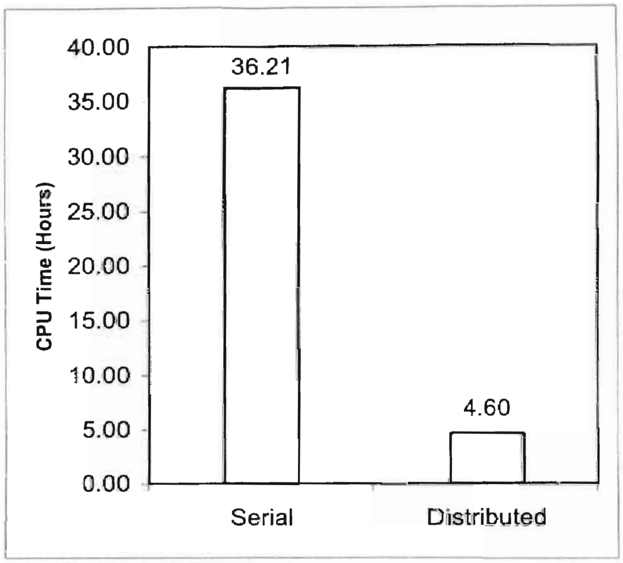


Figure 3.30 Time Required to Find a Prediction Using a Density Sweep

The density sweep required 2.31 CPU hours to find the flutter point. Since the GHV has nine modes it is a likely candidate for the possibility that the density sweep requires less time than the system identification technique.

3.3.2 System Identification

Using the same time step and flow conditions as the density sweep, the system identification training data was generated in parallel for the GHV. The *ratio* was set at

64413.191, the *omega* at 7.894E-8 and the *displ* at 0.10. The GHV was set to run on the eight cluster nodes. For this test a ninth identical computer was added to the cluster. The nodes required 3.24, 3.25, 3.30, 3.20, 3.33, 3.27, 3.22, 3.23, and 3.28 CPU hours to simulate the response to the input signals. The average was 3.26 hours. Even without accounting that the cluster should only have eight nodes, the system identification required more time than the initial density sweep. If the two shortest times for system identification are added to simulate the effect of trying to run nine simulations on a cluster with eight processors, then the time to complete the training data is 6.42 CPU hours. This is longer than the time needed to find the refined density sweep prediction. Figure 3.31 shows the relationship between the number of modes of a test case the time to develop the system identification training data. The time for the refined density sweep is marked as a comparison. This figure assumes that as many processors as needed are available.

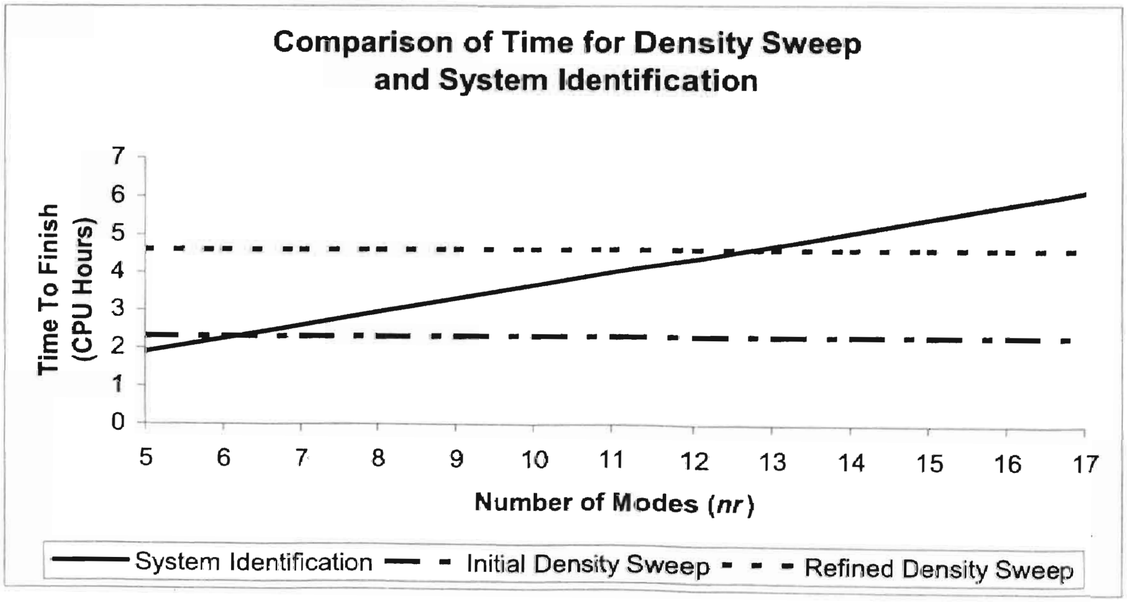


Figure 3.31 Time Comparison of Density Sweep and System Identification, GHV

Even though the system identification should take longer than the density sweep, it may be of interest for studying changes in the internal structure or of control scheme development. The asemdl3d module found that a model of 2-15 fit the data best. This model had a scaled RMS error of 0.242E-04, 0.202E-05, 0.974E-05, 0.571E-05, 0.531E-04, 0.394E-04, 0.579E-04, 0.397E-05, and 0.250E-05 on the respective mode. Since the density sweep indicted a flutter boundary around 150 psi, the stability of the combined system identification model and structural dynamics solver was checked from 0 psi to 200 psi. The sweep of densities is shown in Figure 3.32.

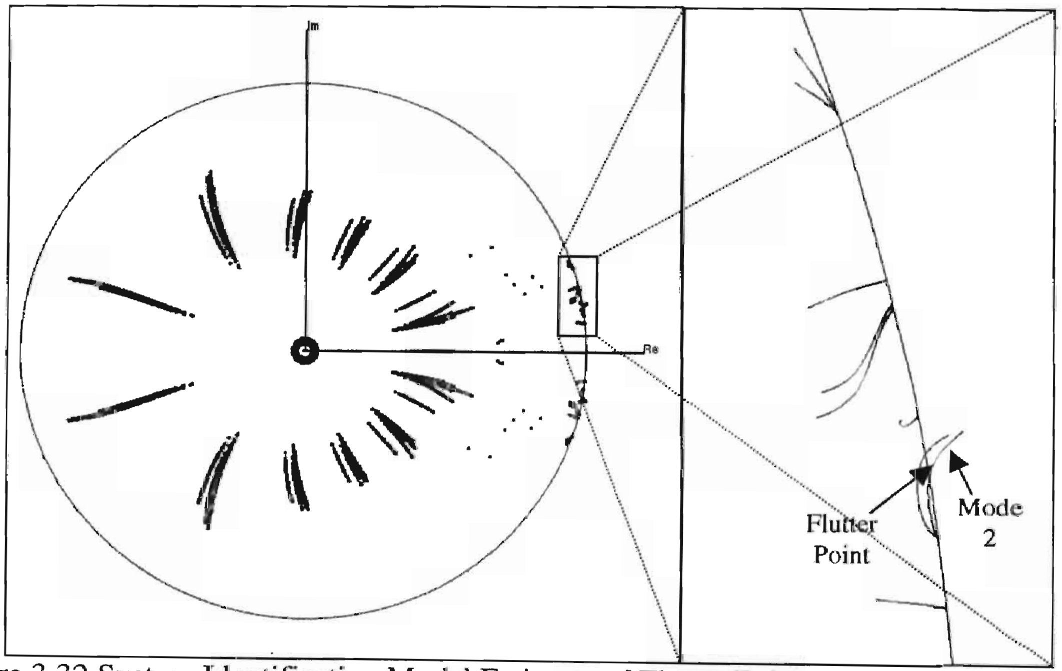


Figure 3.32 System Identification Model Estimate of Flutter Point

Since the GHV structural mode has no damping, the modes all start on the unit circle at zero dynamic pressure. Although it is difficult to see the second and third modes cross within the unit circle and proceed out to unstable eigenvalues. Just as in the density sweep, mode two diverges first. However the mode diverges at 133 psi in this model. This is a difference of 14 percent from the density sweep value. It is still below the flutter boundary and is a safe estimate. The difference is most likely related to the relatively course time step used for the GHV. It has been found that the more refined a time step used the closer a system model response is to the actual physical characteristics.

CHAPTER 4

CONCLUSIONS AND RECOMMENDATIONS

4.1 Conclusions

The distributed processing technique presented here has been shown to be an efficient method to accelerate the prediction of flutter boundaries with either system identification or dynamic pressure sweeps. Both transonic and supersonic test case were used and shown to agree with experimental and previous computational results. It was also shown that the principle of distributed processing holds regardless of CFD solver used.

Distributed parallel processing was chosen as the best method for this study for several reasons. First, it can be implemented without any changes to the flow solver. Secondly, it does not depend on which flow solver is used for improvement in turnaround time for flutter prediction. So, it can be applied to any flow regime or structural model. Finally, the only requirement for effective use of the distributed batch processing is multiple computers of comparable speeds.

The assembly of a computer cluster to help complete training data generation and free response simulation more quickly has a second advantage of allowing work to progress on more than one test case at the same time. In past studies, progress on secondary projects was limited to the use of legacy hardware deemed unsuitable for the primary task at hand. With properly configured computer clusters, any nodes not required for the priority job can be assigned to other tasks. This increases the "bandwidth" of an aeroelastic analysis group by allowing significant progress on multiple test cases.

Finally, the efficiencies reported here compare favorably to results listed in the literature to full parallel domain decomposition techniques. Geuzaine [2003] reported an efficiency of 91% for a Navier-Stokes code using 6 nodes to resolve the solution for one flight condition. The GHV test cases had 98.4% efficiency for the overall prediction of the flutter boundary.

4.2 Recommendations

Based on the results presented, several areas are recommended for further investigation. First, a more methodical approach to density sweep should be developed. At current, most literature uses known experimental values to determine starting values for density sweeps. This is not a practical approach to real world problems, as aeroelastic analysis has little value if the solution is already known. Some techniques for this could be a modified bisection search, or searches based on exponentially increasing densities.

One area of obvious interest is the division of the training input signal on a single mode into multiple parts. This would carry batch processing further by allowing every available computer to be used with no *nr* maximum. This is most obvious on test cases like the AGARD445.6. The test case has only two modes. CASE cluster has eight nodes for use. If the training input signal could be divided into four parts the training data could be finished in a fourth of the time of the current implementation.

Dynamic pressure is a function of both density and fluid velocity. Due to the relationship of density to pressure the density can be easily removed from the non-dimensional flutter predictions. The speed, or Mach number, cannot be isolated so easily. System identification of the aerodynamic forces due to Mach number is not as straightforward as relating force to density. Distributed batch processing offers a method to perform a study on the effect of Mach number with a reasonable time frame. Using the same method as the density sweep multiple Mach numbers can be studied. This would require software to automate the solution of steady state values for each Mach number and then apply initial conditions to generate the free response.

Finally, distributed processing offers the option of generating the training data for nonlinear system models with an acceptable time frame. With multiple simulations, the system model does not need to assume linear superposition of the mode shapes. Each node can run a different combination of modal motions to characterize the system.

BIBLIOGRAPHY

Baker, L. and Smith, B.L., *Parallel Programming*, McGraw-Hill, New York, 1996

Bisplinghoff, R.L., Ashley H., and Halfman, R.L., *Aeroelasticity*, Dover Publications, Inc., 1996

Byun, C. and Guruswamy, G. P., "Aeroelastic Computations on Wing-Body-Control Configurations on Parallel Computers," Journal of Aircraft, Vol. 35, No. 2, Mar-April 98, pp 288-294.

Cowan, T.J., "Efficient Aeroelastic CFD Predictions Using System Identification" Masters Thesis, Oklahoma State University, Department of Mechanical & Aerospace Engineering, Stillwater, OK, May 1998.

Cowan, T.J., "Finite Element CFD Analysis of Supermaneuvering and Spinning Structures" Dissertation, Oklahoma State University, Department of Mechanical & Aerospace Engineering, Stillwater, OK, July 2003.

Fisher, C.C. and Arena, A.S., "On The Transpiration Method For Efficient Aeroelastic Analysis Using An Euler Solver" *AIAA Paper 96-3436*, AIAA, Atmospheric Flight Mechanics Conference, San Diego, CA, July 29-31, 1996

Geuzaine, P., et al, "Aeroelastic Dynamic Analysis of a Full F-16 Configurtion for Various Flight Conditions," AIAA Journal, Vol. 41, No. 3, March 2003, pp 363-371

Geist, A., et al, *PVM: Parallel Virtual Machine, A User's Guide and Tutorial for Networked Parallel Computing*, The MIT Press, Cambridge, MA, 1994

Goodwin, S.A., et al., "Toward Cost-Effective Aeroelastic Analysis on Advanced Parallel Computing Systems," Journal of Aircraft, Vol. 36, No. 4, July-August 99, pp 710-715

Gropp, W., Lusk, E., and Skjellum, A., *Using MPI, Portable Parallel Programming with the Message-Passing Interface*, 2nd edition, The MIT Press, Cambridge, MA, 1999

Gupta, K.K., "STARS – An Integrated General-Purpose Finite Element Structural, Aeroelastic, and Aeroservoelastic Analysis Computer Program," *NASA TM-4795*, 1997

Hunter, J.P. and Arena, A.S., "An Efficient Method for Time-Marching Supersonic Flutter Prediction Using CFD," *AIAA-97-0733*, AIAA 35th Aerospace Sciences Meeting and Exhibit, January 6-10, 1997, Reno, NV

Kalaba, R., and Spingarn, K., *Control, Identification and Input Optimization*, Plenum Press, New York, 1982

Katz, J. and Plotkin, A., *Low-Speed Aerodynamics*, 2nd Edition, Cambridge University Press, 2001

Lee, M., "Development of a User-Friendly Molecular Dynamics (MD) Simulation System for Nanometric Cutting and Tribology," Masters Thesis, Oklahoma State University, Stillwater, OK, 2002.

Liu, F., et al., "Calculation of Wing Flutter by a Coupled Fluid-Structure Method," Journal of Aircraft, Vol. 38, No. 2, Mar-April 2001, pp 710-715

Ljung, L., *System Identification: Theory For The User*, Prentice Hall, Inc., New Jersey, 1987

Moretti, P., *Modern Vibration Primer*, CRC Press, Boca Raton, 2000

O'Neill, C.R., "Improved System Identification for Aeroelastic Prediction," Masters Thesis, Oklahoma State University, Department of Mechanical & Aerospace Engineering Stillwater, OK, July 2003

Snir, M., et al, *Message Passing Interface – The Complete Reference*, The MIT Press, Cambridge, MA, 1998

Umar, A., *Distributed Computing: A Practical Synthesis*, Prentice Hall, Inc., New Jersey, 1993

Worringen, J. and Scholtyssik, K., *MP-MPICH: User Documentation and Technical Notes*, RWTH Scalable Computing, 2002

Yates, E.C., Jr., "AGARD Standard Aeroelastic Configurations for Dynamic Response. Candidate Configuration I.-Wing 445.6," *NASA TM-100492*, 1987

APPENDICES

APPENDIX A

SOFTWARE OPERATION

A.1 Overview of Euler3d

Until recently, the STARS suite of programs lacked a flow solver capable of flow solutions in non-inertial frames of reference. In 2001, Cowan developed a new computational Euler based flow solver capable of handling dynamic rotations. In addition the addition of the non-inertial frame, the algorithm was improved and the input file formats simplify from previous STARS flow solvers. This appendix has a brief overview of the operation of *Euler3d*. This overview is intended to explain the difference between *Euler3d* and *Euler3d_dpp*.

A.1.1 Input Files

The new *Euler3d_dpp* was designed to work with existing *Euler3d* standard formats. As such the format of the input files has remained the same. The only changes have been to the control file, *case.con*. The new flags are listed here with there effects. The following three values control the generation of training data for system identification. The method for determining the correct values for the three parameters is covered in A.2.

- `omega` – This value determines the rate of frequency (ω) sweep.
- `displ` – This value determines the magnitude of **`displ`**acement.

- `ratio` – This value determines the length of the training signal.

There is only one parameter to adjust when running a sweep of several densities to confirm a system identification prediction.

- `delrho` – This is the increment of density. The value of the density on any given node α is `delrho` $\times$ α + `rhoinf`.

The following two parameters are logical, true or false, controls.

- `iwrite` – Determines the recording of full solution files from nodes other than `0`.
- `irsds` – This option is disabled in Euler3d_dpp. It controls the residual study command. If an unexplained error occurs during the first step of euler3d_dpp, this value may be set to *true*. If so change to *false* and this should correct the error. NOTE: It is possible to perform a residual study in Euler3d_dpp, but use **one** and only **one** node.

A more detailed discussion of how to find the value for parallel parameters is given in section A.2. In addition to these new flags, new options have been added for `IBXN` in the vector file, *case.vec*. In particletre, the values 6 and 7 which induce a modified and offset modified chirp signal.

A.2 System Identification Parameters

This section was copied from a MathCAD worksheet, which is used to find omega and ratio for the chirp-training signal.

Begin by selecting the largest ARMA model to investigate and set number of modes.

na = number of aerodynamic terms

nb = number of body displacement terms

nr = number of elastic modes for the model

overdetermined% (ε) = the percentage of data points to use to over determine the system, i.e. 400 percent is 4 data points for every parameter. Then find the number of data points needed by using Equation A.1.

$$datapts = \left(nr^2 \cdot nb + nr \cdot na\right) \cdot \frac{\varepsilon}{100}$$

Example:

na := 6

nb := 3C

nr := 2

overdetermined% := 100(

$$\text{NumPtsNeeded} := \left(\text{nr}^2 \cdot \text{nb} + \text{na} \cdot \text{nr}\right) \cdot \frac{\text{overdetermined\%}}{100}$$

NumPtsNeeded = 1320

Next, input the highest structural frequency and minimum number of points per cycle. Using these values, find the non-dimensional time step for the ARMA model.

fmax = highest natural frequency of structural mode (in Hertz)

minpts = number of points per cycle of the **fmax** mode

M = Mach number

ainf = Speed of sound (in inches per second)

refdim = reference dimension (usually set to 1 inch)

Using Equation A.2, find the time step.

$$dt = \frac{1}{f_{MAX} \cdot mnpts}\left(\frac{M \cdot a_{\inf}}{ref_{\dim}}\right)$$

Continued Example:

$f_{max} := 25$

$minpts := 10($

$M := 0.8$

$a_{inf} := 1291($

$refdim := 1$

$$dt_{real} := \frac{1}{f_{max} \cdot minpts}$$

$dt_{real} = 4 \times 10^{-4}$ in seconds

$$dtstar := dt_{real} \cdot \frac{M \cdot a_{inf}}{refdim}$$

$dt = 2.4$ *dt* of the *case.con* file; *dt* can be smaller, but not larger

The number of points in a training signal can be determined by dividing the number of points needed for an overdetermined model by the number of mode shapes.

$$MinPtsInChirp := \frac{NumPtsNeeded}{nr}$$

$MinPtsInChirp = 660$

Now, **ratio** can be found by multiplying the number of data points in a training signal by the time step.

$$\mathrm{ratio} := \mathrm{MinPtsInChirp} \cdot dt$$

$$\mathrm{ratio} = 1584$$

This is **ratio** for the *case.con* file.

Find **omega**, the rate of frequency sweep, by setting the $\mathrm{multi}_{\mathrm{high}}$ as a faction of the **minpts** per highest cycle, this prevents the chirp going beyond the Nyquist frequency.

$$\mathrm{multi}_{\mathrm{high}} := \frac{\mathrm{minpts}}{10}$$

$$\mathrm{multi}_{\mathrm{high}} = 10$$

Find the Nyquist frequency angular value:

$$\omega_{nq} := \frac{1}{2} \cdot \frac{2\pi}{dt}$$

$$\omega_{nq} = 1.309$$

This formula finds the rate of the frequency so that the frequency sweep ends a frequency equal to the Nyquist divided by the value of $\mathrm{multi}_{\mathrm{high}}$. In this case it is a tenth of the Nyquist.

$$\omega := \omega_{nq} \cdot \frac{1}{\mathrm{multi}_{\mathrm{high}}} \cdot \frac{1}{2} \cdot \frac{1}{\mathrm{ratio}}$$

$$\omega = 4.132 \times 10^{-5}$$

This is **omega** for the *case.con* file.

The number of time steps to run the training signals for both serial and parallel generation.

Total points of the training signal:

$$\text{TotalPoints} := \frac{20 \cdot dt + (nr + 0.1) \cdot \text{ratio}}{dt}$$

$\text{TotalPoints} = 1406$

For a standard chirp without parallel processing this is **`nstp`**.

$$\text{TotalPoints\_Parallel} := \frac{20 \cdot dt + (1.05) \cdot \text{ratio}}{dt}$$

$\text{TotalPoints\_Parallel} = 713$

For a Parallel processed chirp, this is **`nstp`**.

Note: Both omega and ratio should be entered into the control file as double precision values.

Example:

`omega` = 0.1203E-6,

`ratio` = 2107.0d0,

A copy of this document in both MathCAD and Abode Acrobat formats is available at www.caselab.okstate.edu.

A.3 Using Euler3d_dpp

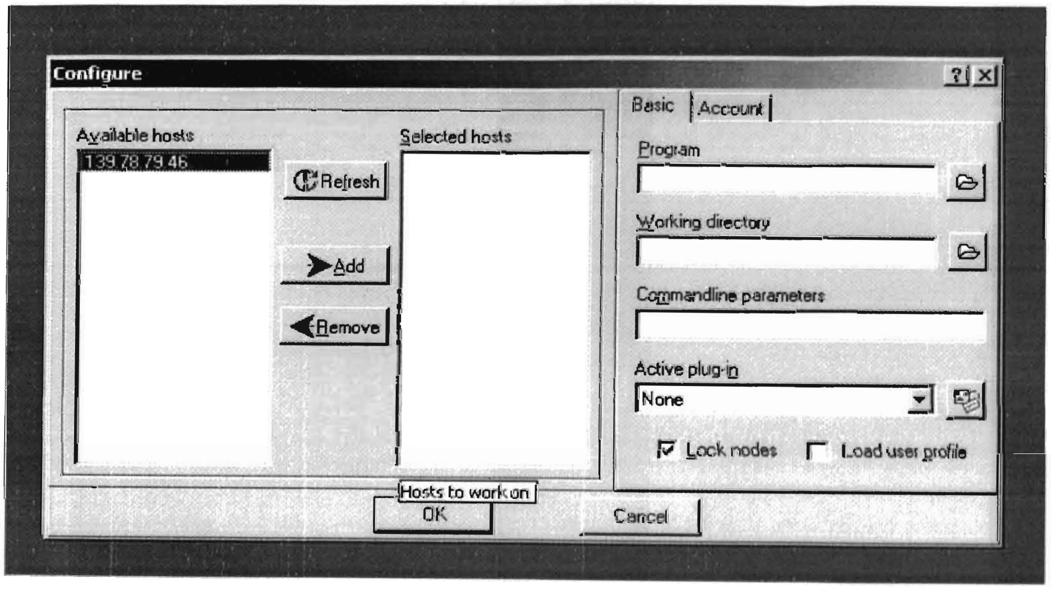
This section handles the details of starting Euler3d_dpp. This section assumes that the steady state solution, grid file and mode shape displacements are already available and correct. It also assumes that NT-MPICH has been installed on all computers with in the cluster and is working properly. See section B.2 for instructions on setting up the NT-MPICH services on a Windows machine.

`Rexecshell.exe` – This is the graphical interface to the MPI software.

1. Start the `Rexecshell.exe` program.

2. If the dialog pops up a box asking to use **`machines.txt`**, answer it. This usually means that the computers, or **`machines`**, you will be using have been listed in a convenient file. If you don't use this file, the program will automatically search the connected networks for machines that support the MPI protocol. If the machine is connected to a large network, like those found at most universities, it will take some time to finish.

3. Open the configure dialog box from **File → Configure...**



Available hosts: 139.78.79.46

Selected hosts:

Program:

Working directory:

Commandline parameters:

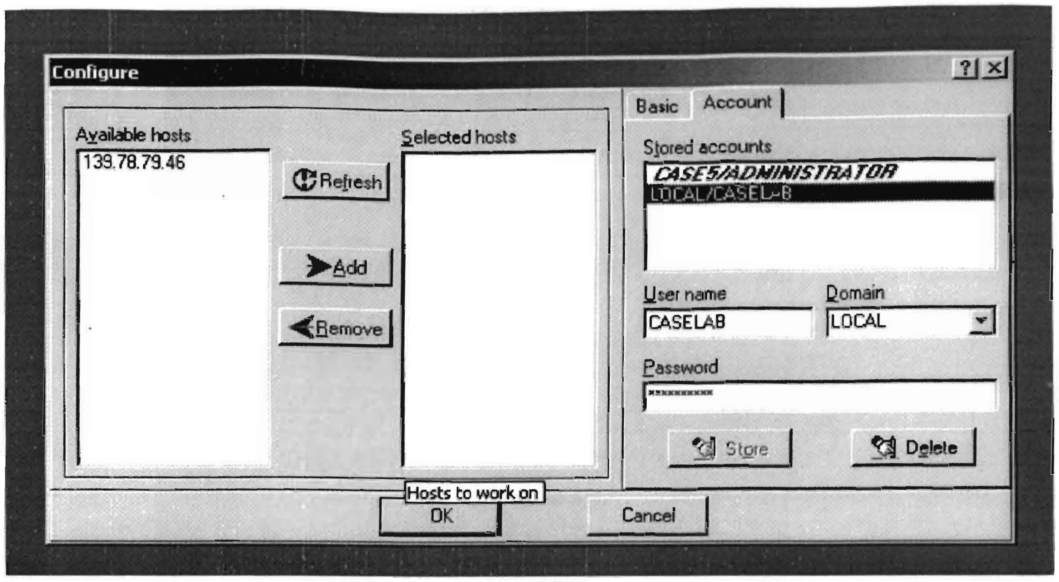
Active plug-in: None

☑ Lock nodes ☐ Load user profile

4. Select the machines you wish to use form the list under *Available hosts*. A computer can be selected multiple times. Try to find a balance so that all the machines will finish at about the same time.

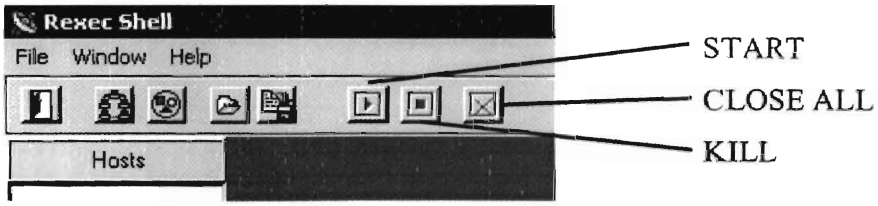
5. Under the ***Basic*** tab, select **ch_wsock** from the drop down menu for *Active plug-in*. On the same tab, type in the path to the `euler3d_dpp.exe` program in the *program* field. In the *working directory* field type the path to the directory with the test case. Alternatively, these can be selected from the drop down menus.

6. On the ***account*** tab, type in the username, password and domain for the user account that will be used to run the distributed test case. The domain is most offer *local*.



Stored accounts: CASE5/ADMINISTRATOR, LOCAL/CASEL-B
User name: CASELAB
Domain: LOCAL
Password: **********
Available hosts: 139.78.79.46

7. Select OK in the dialog box. Then click the start button.



8. In the window for node number 0, the *master* node, the node will be waiting for the name of the test case. Select the window and type in the name. Press <Enter>. All the nodes should run without need for user input from this point.

9. When all nodes report that the simulations are complete, press the kill button to release the nodes. The shell can be used to initiate a new set of simulations.

A.4 Euler3d_dpp Output Files

This section covers the output files from Euler3d_dpp. There are three sets of files that can be generated with distributed Euler. Note that the *case#.lds* and *case#.un##* files are only generated if **`iwrite`** is set to **`true`**.

xn.dat#	These files contain the displacement, velocity and accelerations on each mode shape from each node used by the cluster software. The # indicates which node generated the file.
case#.lds	These files contain the aerodynamic loads on the solid walls on the test case. The # indicates which node generated the file.
case#.un##	These files contain the nodal values for the primitive flow variables for each node in the computational grid. The # indicates which node generated the file. The ## indicates the sequence of the solution files.

A.5 Euler3d_dpp PostPrecessors

The operation of `glplot3d.exe` is the same was with regular `Euler3d`. However, in order to view the solutions the input files *case.con*, *case.vec* and *case.un##* must be rename to *case#.con*, *case#.vec*, and *case#.un##*. This allows glplot3d to know from which node it should get the solution.

The following section was originally written as instructions on determining the best ARMA model for a training set.

1. Copy each *xn.dat#* to *train.dat#*. Note that the file *xn.dat0* should be renumbered to *nn*, where *nn* is the number of nodes used in for Euler3d_dpp. Don't rename the files; copy them to the new file name. They will be used again later.

2. Run `cfdmdl3dsplice.exe`

   a.) Select option 4 `= Generate MULTIPLE spliced aerodynamics models`

   b.) Input the range of `na` and `nb` that was used to determine the chirp signal length

   c.) When finished note the number of models produced. This is listed at the very end of the run

3. Run `xnmeld.exe`. Input the number of *xn.dat#* files and `nr` as prompted. This will produce an new file called *xn.dat*.

4. Run `asemdl3d.exe`

   a.) Select option 5 `= RMS Error Study of Force Response`

   b.) Input the number of models from 2c

5. Using *EXCEL*, open the *RMS.dat* file. The last two columns of this data are both measures of how well a model matches the training signal. Find the model that has the lowest value for each measure. It is possible that the same model is the lowest in both, but not necessarily true. The models are mostly likely similar. (NOTE: the quickest way to sort the data file is use the "Sort Ascending" command in EXCEL. This is usually on the toolbar as a down arrow next to an A above Z.)

6. Run `cfdmdl3dsplice.exe`

   a.) Select option 1 `= Generate Single aerodynamics models` b.) Input the `na` and `nb` of one of the models found in step 5

7. Run `asemdl3d.exe`, to find the flutter boundary of this model

a.) Select option `1 = Compute eigenvalues of coupled system`

b.) Input a dynamic pressure well below the flutter boundary (usually 0 is selected)

c.) Input a course resolution for the dynamic pressure increment

d.) Input a number of increments that will test a dynamic pressure well above where the fluttery boundary is believed to be

e.) The program will output the first dynamic pressure that results in an unstable system

f.) If the program did not find an unstable system, increase the number of increments until one is found

g.) Rerun `asemdl3d.exe` with better resolution between the unstable pressure and the next lowest pressure in the increment set

h.) Repeat part g until the fluttery boundary is determined to the required resolution

8. Confirm that the structural mode shapes cause the flutter boundary, not instability in the aerodynamic or body motion models. This is best done by running `asemdl3d.exe`, option `1` for the selected model from a dynamic pressure of zero to beyond the flutter boundary. Using `gleigplot.exe`, look at the eigenvalues (*case.eig*). The structural modes should move from close to the unit circle (right on it if the structure has no damping) to outside of the circle (not necessarily by a straight line, they may move in then out). If any other type of eigenvalue is the cause of the instability, then this is not a

valid model of the coupled aerodynamic/structural system. Try another model that matched the forced response well.

9. Run `asemdl3d.exe`

   a.) Select option 3 `= Model sensitivity study`

   b.) Input a range of dynamic pressure with the required resolution around the flutter point that was found in 6

   c.) Input the number of models from 2c

   d.) The program tests every model with dynamic pressures in the specified range to find the fluttery boundary, so it may require a several minutes to a few hours

10. Open *sensitivity.dat*. The file contains the results from 7. Using the data from *RMS.dat*, the models with low error should all have nearly identical flutter boundaries, within ~25 percent. This group of valid models should be readily apparent.

11. Confirm the flutter boundary with free response in `Euler3d_dpp.exe`. Run a case of free response above the flutter boundary and one below to confirm that the flutter prediction is valid. These cases should be within the acceptable range of error for the flutter boundary.

APPENDIX B

CLUSTER DESIGN AND ASSEMBLY

The impetus to design and build a cluster of inexpensive personal computers to solve large-scale nonlinear aeroelastic problems came from discussion about maximizing computational performance within a set monetary budget. That discussion lead to the development of a mathematical model that found the best computer configuration for the most processing within the set budget. This appendix details how the computers were selected and how the computers were assembled into a cluster.

B.1 Component Selection

The original purpose of the software was to reduce the time required to generate training data for complex structural models. Knowing how the software would operate, the speed of the cluster could be evaluated using existing *Euler3d* benchmarking software. Using the benchmark predictions, the current cost of a machine, and estimate for the cost of cluster networking and the known budget, a systematic search was done to find the best cluster design. Table B.1 contains the results of the search for five different Intel Pentium 4 processors. The benchmark results are in time steps per second, the higher the better. The *computers* column represents the number of computers that be purchased within the budget.

Processor	*Number of Computers*	*Benchmark Estimate*
2.26	10	6.10
2.40	9	5.85
2.53	**9**	**6.84**
2.66	8	6.40
2.80	7	5.85

Table B. 1

The search identified the Intel Pentium 4 2.53 GHz processor as the best option for creating a cluster of personal computers. The motherboard, memory, hard disk and chassis/power supply selections were made to reduce the cost further. It is important to note that the selection of motherboard and memory was made to remain in budget and to get the best possible memory assess speed. Table B.2 lists the components selected. The motherboard should also include video and fast Ethernet ports.

Processor	Intel Pentium 4 2.53 GHz
Memory	2 × DDR333 512MB
Motherboard	Asus P4GE-V/L
Hard Disk	Maxtor 30.0 GB 7200RPM ATA133
Computer Case	Avance Corp's BL6004

Table B. 2

Node selection is a straightforward attempt to optimize benchmark versus price of the cluster. The other major component in a cluster is the networking medium. In general for parallel batch processing, like Euler3d_dpp, the speed of the medium has only a small effect on the performance of the cluster. So the least expensive form of networking several computers together should be selected. However it is important to consider that batch processing is usually a first step toward parallel processing of a single simulation. For parallel schemes where several processors are working on the same simulation, communication speed is important. The network switch should be able to handle high volume traffic quickly, and allow expansion of the cluster. For the cluster, the HP procurve 2124 fast Ethernet switch was selected. The cluster will start with only 8 nodes, but the switch has 24 ports allowing expansion in the future and temporary addition of extra machines.

In order to speed up assembly and reduce administration of the cluster, a commercial DHCP server and firewall was purchased, the Netgear FR114P. A KVM, keyboard/Video/Mouse, switch was used to allow users to control each node individually. This effectively completed the components of the cluster. However, Uninterruptible Power Supplies, UPSs, were used to handle power spikes and power loss.

B.2 Cluster Assembly

Once the components for the nodes arrived, assembly preceded one machine at a time. Each part was visually inspected. Then the pieces were assembled. Each motherboard was removed from its protective packaging. A CPU was opened and mounted on the motherboard. Two sticks of memory were installed. The motherboard was attached the computer chassis and screwed down.

With two people working assembly only required four hours. As each computer was finished, it was powered up and allowed to run it's self-test. For this cluster, CD-

ROM drives were used to install Microsoft Windows 2000 Professional on each machine. However, Windows 2000 Server has an optional service that allows the installation of operating systems over a local area network. It is recommendation that this service be used if possible. As the operating system is install, make sure that a user account is created that will allow the master node to start programs on all slave nodes.

Once Windows was installed all cluster nodes, NT-MPICH was installed on each node. The software package was unzipped on a network drive. The service was installed on each node individually by running `rcluma-install.bat` from the network drive. The dynamic link libraries from the `lib` directory of the unzipped NT-MPICH were copied on to each computer. Alternatively, the `PATH` system variable could be changed to include the network drive containing them. The cluster was ready to process MPI based programs.

#2

VITA

Anthony Andrew Boeckman

Candidate for the Degree of

Master of Science

Thesis: ACCELARATING COMPUTATIONAL FLUID DYNAMICS BASED AEROELASTIC ANALYSIS USING DISTRIBUTED PROCESSING

Major Field: Mechanical Engineering

Biographical:

Personal Data: Born in Tulsa, Oklahoma on September 22, 1978, the son of Anthony P. and M. Jean Boeckman.

Education: Graduated from the Oklahoma School and Science and Mathematics in Oklahoma City, Oklahoma in June 1997; Received Bachelor of Science in Aerospace Engineering with a minor in History from Oklahoma State University, Stillwater, Oklahoma in May 2001; Completed requirements for Master of Science with major in Mechanical Engineering at Oklahoma State University in May 2003.

Experience: Employed by Site Specific Technology Development Group as a Computer Graphics Programmer from 2000 to 2001; employed by Oklahoma State University as a graduate research and teaching assistant in the Department of Mechanical and Aerospace Engineering from 2001 to 2003.

Professional Membership: American Institute of Aeronautics and Astronautics