AEROELASTICITY OF MORPHING WINGS USING NEURAL NETWORKS

Anand Natarajan

A Dissertation Submitted to the Faculty of
Virginia Polytechnic Institute and State University
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in

Aerospace Engineering

Rakesh K. Kapania, Chairman

Eric R. Johnson

William H. Mason

Daniel J. Inman

Liviu Librescu

July 03, 2002

Blacksburg, Virginia

Keywords: Morphing Wings, Neural Networks, Aeroelasticity, Adaptable Bump,
Flutter, Time Varying Systems, Variable Stiffness Wing

Aeroelasticity of Morphing Wings Using Neural Networks

Anand Natarajan

Abstract

In this dissertation, neural networks are designed to effectively model static non-linear aeroelastic problems in adaptive structures and linear dynamic aeroelastic systems with time varying stiffness. The use of adaptive materials in aircraft wings allows for the change of the contour or the configuration of a wing (morphing) in flight. The use of smart materials, to accomplish these deformations, can imply that the stiffness of the wing with a morphing contour changes as the contour changes. For a rapidly oscillating body in a fluid field, continuously adapting structural parameters may render the wing to behave as a time variant system. Even the internal spars/ribs of the aircraft wing which define the wing stiffness can be made adaptive, that is, their stiffness can be made to vary with time. The immediate effect on the structural dynamics of the wing, is that, the wing motion is governed by a differential equation with time varying coefficients. The study of this concept of a time varying torsional stiffness, made possible by the use of active materials and adaptive spars, in the dynamic aeroelastic behavior of an adaptable airfoil is performed here.

A time marching technique is developed for solving linear structural dynamic problems with time-varying parameters. This time-marching technique borrows from the concept of Time-Finite Elements in the sense that for each time interval considered in the time-marching, an analytical solution is obtained. The analytical solution for each time interval is in the form of a matrix exponential and hence this technique is termed as Matrix Exponential time marching. Using this time marching technique, Artificial Neural Networks can be trained to represent the dynamic behavior of any

linearly time varying system. In order to extend this methodology to dynamic aeroelasticity, it is also necessary to model the unsteady aerodynamic loads over an airfoil. Accordingly, an unsteady aerodynamic panel method is developed using a distributed set of doublet panels over the surface of the airfoil and along its wake. When the aerodynamic loads predicted by this panel method are made available to the Matrix Exponential time marching scheme for every time interval, a dynamic aeroelastic solver for a time varying aeroelastic system is obtained. This solver is now used to train an array of neural networks to represent the response of this two dimensional aeroelastic system with a time varying torsional stiffness. These neural networks are developed into a control system for flutter suppression.

Another type of aeroelastic problem of an adaptive structure that is investigated here, is the shape control of an adaptive bump situated on the leading edge of an airfoil. Such a bump is useful in achieving flow separation control for lateral directional maneuverability of the aircraft. Since actuators are being used to create this bump on the wing surface, the energy required to do so needs to be minimized. The adverse pressure drag as a result of this bump needs to be controlled so that the loss in lift over the wing is made minimal. The design of such a ”spoiler bump” on the surface of the airfoil is an optimization problem of maximizing pressure drag due to flow separation while minimizing the loss in lift and energy required to deform the bump. One neural network is trained using the CFD code FLUENT to represent the aerodynamic loading over the bump. A second neural network is trained for calculating the actuator loads, bump displacement and lift, drag forces over the airfoil using the finite element solver, ANSYS and the previously trained neural network. This non-linear aeroelastic model of the deforming bump on an airfoil surface

using neural networks can serve as a fore-runner for other non-linear aeroelastic problems.

This work enhances the traditional aeroelastic modeling by introducing time varying parameters in the differential equations of motion. It investigates the calculation of non-conservative aerodynamic loads on morphing contours and the resulting structural deformation for non-linear aeroelastic problems through the use of neural networks. Geometric modeling of morphing contours is also addressed.

Acknowledgment

First and foremost, this work was made possible due to the grace of the Supreme who provided me with the strength and faith that is required. This dissertation was also made possible because of the help and advise of many people. I am extremely grateful to my advisor Professor Rakesh K. Kapania for his guidance and encouragement through out the tenure of this work. I am really indebted to him for always being available for advise and corrections during the many stages of this Ph.D. research. I am very thankful to Professor Daniel J. Inman of the Dept. of Mechanical Engineering for his help and support during the progress of this research. Many thanks go to Professor William H. Mason for his advise with all the fluid dynamic problems. I thank Professor Eric Johnson of the Dept. of Aerospace Engineering and Professor Liviu Librescu of the Dept. of Engineering Science and Mechanics for all their technical help and advise. I would like to express my gratitude to Dr. Erwin Sulaeman for guidance with unsteady aerodynamic codes and to Yong-Yook Kim for solutions to many software problems. I thank the members of the UCAV team, Professor Robertshaw of the Dept. of Mechanical Engg., Greg Pettit and Dr. Frank Gern for their help and advise. The financial support of AFOSR and DARPA is gratefully acknowledged.

I thank my parents for their constant encouragement and support during both the good and bad times of this research period. Finally, I would like to express my deep gratitude for my wife for her ever-ready help, patience and sacrifice.

Contents

List of Figures

List of Tables

Nomenclature

ANN	Artificial Neural Network
AR	Aspect Ratio
b	Semi wing span, Semi airfoil chord
C_d	Coefficient of drag
C_l	Coefficient of lift
C_n	Coefficient of yaw moment
C_p	Coefficient of pressure
E	Modulus of elasticity
e	Internal Thermodynamic Energy
f	External Force
G	Penalty multiplier

H Hessian matrix

h Airfoil plunge displacement

I Mass moment of inertia

J Jacobian matrix

K Thermal Conductivity

k Stiffness

$k-\epsilon$ Turbulence model using turbulent kinetic energy and its dissipation

L Lift

M Pitching moment

m Mass

p Aerodynamic pressure

s First Moment of inertia

t Time

u Local Flow velocity in the x direction

U_∞ Free stream velocity in x direction

v Local Flow velocity in the y direction

W	Neural Network weight
w	Local Flow Velocity in the z direction
x	Displacement
y	Displacement
α	Airfoil pitch angle
κ	Strain Energy
μ	Doublet potential
ω	Natural Frequency
Φ	Wagner function
ϕ	Velocity Potential
ρ	Fluid Density
τ	Shear Stress
ξ	Non-dimensional coefficient of damping

Chapter 1

Introduction

The use of smart materials in aircraft wings for actively changing wing shapes in flight will have an enormous impact on the design of advanced civil and military airplanes. Specifically, the use of adaptable wings will bring significant changes in the aeroelastic behavior of the wing as described by Inman, et. al.[1]. Adaptive materials in the wing allow changing the shape of wing sections and wing configurations in flight. Wings which are adaptable to the fluid flow around them, both geometrically and structurally, thereby changing the wing surface contours and/or wing structural parameters so as to provide the best design under any flight conditions can be called as morphing wings. Theoretically, this means that there is no need for control surfaces as adaptive camber can perform the function of a deflected flap. Morphing wings have a smooth contour thereby not allowing abrupt shape changes in the direction of the air-flow. These wings also use a distributed set of non-hydraulics based actuators. These actuators can be piezoelectric devices such as THUNDER[2] or electro-active polymers[3] and shape memory alloys[4]. Replacing current heavy hydraulic systems

by these light weight actuators can increase payloads and flight range[5]. The internal structural components of a wing, namely the spars and ribs can also be adaptive, i.e., their structural properties can be made to vary[6]. This goal of a fully adaptable all wing aircraft requires the knowledge of the behavior of a highly flexible structure subjected to changing aerodynamic forces. *The objective of this dissertation is to address this important goal of analyzing some of the effects of fluid structure interactions on adaptive wing aircraft.*

The term ”smart materials”, sometimes also called intelligent materials, describes a group of material compounds with unique properties. These unique properties usually relate to a large strain deformation when the smart material is subjected to electrical/thermal/magnetic fields. Smart materials which deform under an electric field are termed piezoelectric or electrostrictive. Piezoelectric materials can deform under compression or elongation whereas electrostrictive materials usually deform one way, that is either in elongation or in compression. Smart materials which respond to a thermal field are called Shape Memory Alloys (SMA). These materials exhibit large deformations and phase transformations. However there response is quite slow compared to piezoelectric materials. Smart materials which respond to magnetic fields are termed magnetostrictive materials. Just like electrostrictive materials, magnetostrictive materials can also possess anisotropic behavior. Also, there are a few shape memory alloys which respond to magnetic fields. These are termed as Ferromagnetic Shape Memory Alloys (FSMA). The detailed behavior of all types of active materials can be found in Ref. 7.

The use of such smart actuators and adaptive wing spars allows the development of a variable stiffness wing. While analyzing, the dynamics of such a variable stiffness wing, the differential

equations of motion will possess time varying coefficients. Hence it is required to first study the efficient solution techniques used in studying the response of time-variant systems. Herein is considered the analysis of linear time varying systems and its application in dynamic aeroelasticity. Since the morphing wing can possess an arbitrarily changing wing surface, it is important to predict the deformation of a surface of a wing under applied actuator loads and non-conservative aerodynamic loads. Such a two dimensional aeroelastic analysis of a bump on a wing surface is performed here. In order to proceed with these two novel aeroelastic problems namely the dynamic aeroelastic behavior with time varying stiffness and the aeroelasticity of a morphing surface, the fundamental tools in these aeroelastic computations are introduced.

1.1 Fluid Dynamic Models for Flow Over Adaptive Wings

The laws of physics which govern fluid flow remain the same whether the wing is adaptive or non-adaptive. What does change between the fluid dynamic model of a fixed wing aircraft and a morphing wing aircraft are the boundary conditions and the nature of any simplifying approximations. Usually aeroelastic solvers do not use the fundamental governing equation of fluid flow, that is the Navier-Stokes equations. This is because of the essential non-linear nature of these equations. Many of the conventional aeroelastic solvers use a technique known as the ”Doublet Lattice Method” to model the unsteady aerodynamics. The doublet lattice method is basically derived based on the assumption of irrotational flow and the governing mathematics allows the use of the acceleration potential of a doublet as a basis for calculating the unsteady pressure over an

oscillating wing[8]. Recently there have been many improvements made on the original Doublet Lattice method[9] which may enable it to be used even under transonic conditions. On the other hand, a morphing wing may require the use of the Navier-Stokes equations. This is because, flow separation and vortex shedding may become a factor to consider when formulating the aeroelastic model. Also the Doublet Lattice method only models the camber of the wing. For a Morphing wing, it is required to capture the effect of thickness changes. The Navier-Stokes equations are solved numerically using finite difference approximations to the derivatives. Computational Fluid Dynamics (CFD) approximates the governing partial differential equations of motion by difference equations. This basically means that the partial differential equations are now replaced by "modified" equations consisting of truncated Taylor's series representations of the derivatives in the original equation. Therefore all CFD calculations involve some level of truncation and round-off errors. In obtaining the difference equations, one replaces the original fluid continuum by a discrete mesh, with the values of the parameters in the difference equations calculated at the nodes of the mesh. A theorem given by Peter Lax[10] states that the CFD model of the continuum converges to the exact solution if the numerical scheme is stable and the truncation error approaches zero as the finite difference step size approaches zero. The governing equations of fluid dynamics in 2-D Cartesian co-ordinates can be represented as

$$\frac{\partial \mathbf{Q}}{\partial \mathbf{t}} + \frac{\partial \mathbf{F}}{\partial \mathbf{x}} + \frac{\partial \mathbf{G}}{\partial \mathbf{y}} = \frac{\partial \mathbf{J_x}}{\partial \mathbf{x}} + \frac{\partial \mathbf{J_y}}{\partial \mathbf{y}} \tag{1.1}$$

$$\text{where } \mathbf{Q} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho e \end{pmatrix}, \mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho eu + pu \end{pmatrix}, \mathbf{G} = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho ve + pv \end{pmatrix}, \mathbf{J_x} = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} + K_x \dfrac{\partial T}{\partial x} \end{pmatrix},$$

$$\mathbf{J_y} = \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ K_y \dfrac{\partial T}{\partial y} + u\tau_{xy} + v\tau_{yy} \end{pmatrix}$$

Equation (1.1) is in conservative form and it is nonlinear. Since there is no analytical solution for Eq.(1.1), numerical approximations to the derivatives are made which result in the finite difference equations. Another approach to solving Eq. (1.1) is to use the finite volume method. In this approach, Eq. (1.1) is converted into an integral equation using the Gauss Divergence Theorem. A description of the numerical techniques used to approximate the Navier-Stokes equations, Eq. (1.1) by finite difference equations is given in Ref. 10.

Figure 1.1 displays a computational mesh around an airfoil. Each element of the mesh has nodes placed at the vertices of a triangle. The boundary conditions are imposed at the airfoil wall and at the far field of the mesh. The mesh density determines the accuracy of the solution in as much as it reduces the truncation error. Mesh adaptation at the airfoil wall allows proper representation of the boundary layer. One of the major issues in computational fluid dynamics is turbulence modeling since the requirement is for a turbulence model that represents the turbulent behavior and at the same time is computationally efficient. There are many turbulence models existing

today[11]. The simpler turbulence models such as the one equation Spallart-Allmaras model or the two equation $k-\epsilon$ model are restricted to certain types of flows and the advanced turbulence models such as the Reynold's 5 equation model are computationally very expensive[11]. This implies that certain compromises regarding the accuracy of the results will have to be made when a highly turbulent flow is modeled using CFD. The use of CFD calculations is required only in the case of

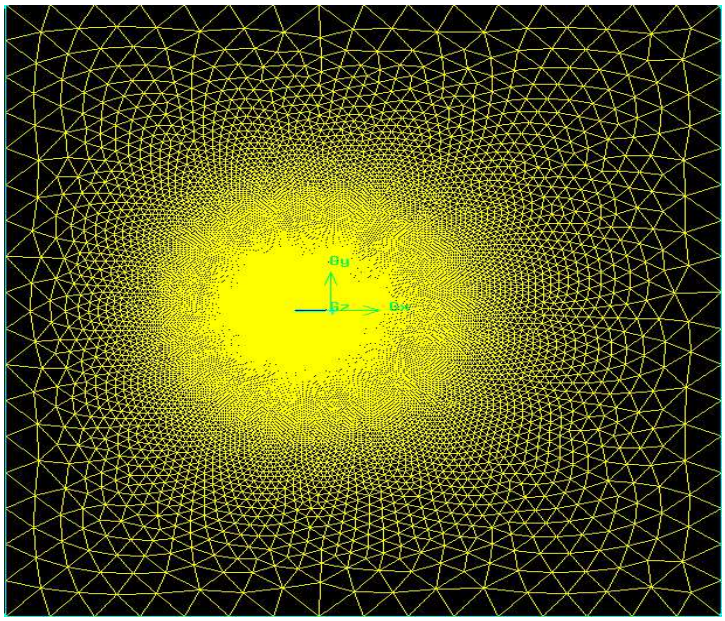


Figure 1.1: **Computational Grid :- Triangular Grid generated using the software GAMBIT showing the mesh surrounding an airfoil with Pressure Far Field Boundary Conditions at 10 times chord distance**

compressible flow or when viscous forces play a predominant role. For incompressible inviscid flow, the use of panel methods provides a rapid estimate of the aerodynamic forces. Panel methods consist of distributing singularities such as sources, vortices and doublets on the surface of an airfoil or a wing. The potential function of such singularities satisfies the governing equation of

incompressible inviscid aerodynamics. The flow boundary conditions on the wing surface or airfoil then determine the strengths of these singularities. Once the singularity strengths are known, it is possible to calculate the unsteady pressure distribution at each point over the wing or the airfoil. The details of the implementation of panel methods for morphing airfoils is presented in Chapter 5. Using panels of doublets over a morphing airfoil, the incompressible inviscid aerodynamics is modeled. The boundary conditions that are imposed over the airfoil are the non-penetration condition and the Kutta-condition. The non-penetration condition requires that the normal velocity of the fluid at the airfoil wall should be equal to the airfoil surface velocity. The Kutta condition at the trailing edge forces the vorticity of flow at the trailing edge point to be zero for steady flows. For unsteady flows the velocity at the trailing edge should be finite. More details on the unsteady Kutta condition can be found in Ref. 12. The position and size of the panel at the trailing edge needs to be carefully chosen to obtain finite velocities at the trailing edge. Using this approach of distributed doublets over an airfoil, the unsteady aerodynamic lift and moment over the airfoil can be captured.

Unsteady aerodynamics can be treated either as a ”quasi -steady” phenomena or a ”fully unsteady” phenomena. In a quasi -steady approach, the circulation around an airfoil is assumed to be a function of the space co-ordinate only. If the flow is unsteady, then the circulation will be a function of both space and time co-ordinates. It is the time-varying circulation that makes the calculation of unsteady aerodynamic forces a formidable task. Theodorsen[13] showed that the effect of the unsteady circulation on the airfoil can be represented as a ratio of Hankel functions when the airfoil oscillatory behavior approaches flutter. This ratio of Hankel functions was termed the Theodorsen

function, popularly represented by the symbol $C(k)$ where k is the reduced frequency. The reduced frequency in turn is expressed as $k = \frac{\omega\, b}{U_\infty}$ where ω is the frequency of oscillation of the airfoil with b as the semi-chord and U_∞ as the free stream velocity. Physically, the Theodorsen function is described as a lift deficiency factor since its effect is to reduce the net quasi-steady lift.

The unsteady lift on an airfoil subjected to an impulsive change in the angle of attack in incompressible flow is analyzed using the Wagner function approach. The Wagner function[13,14] is the Fourier Transform of the Theodorsen function and it provides a measure of the circulation growth around the airfoil when the airfoil is impulsively given an angle of attack with respect to the free stream. This provides the solution for the sub-critical oscillations of the airfoil in irrotational flow. By sub-critical flow is meant that the regime of oscillations of the airfoil when the velocity of the free stream is below the flutter speed.

1.2 Aeroelasticity

Aeroelasticity deals with the science that studies the mutual interaction between aerodynamic forces and elastic forces for an aerospace vehicle. One of the major research areas in the field of aeroelasticity is flutter control. Flutter is a dynamic instability of a body subjected to non-conservative forces wherein its oscillations increase without bounds. Hence this is a catastrophic phenomena and its prevention forms a critical role in wing design. However, most materials have some non-linearities associated with their behavior such as hardening or internal damping. Structures subject to axial compressive forces also possess geometric non-linearities associated with

buckling. Moreover, even the fluid dynamics can be non-linear due to separation of flow or due to the presence of a shock. These conditions of non-linearity, both structural and aerodynamic are usually beneficial with regard to the control of flutter. They provide for a new phenomena called limit cycle oscillations (LCO). Limit cycle oscillations are non-decaying bounded oscillations of a non-linear system. They can occur even in the presence of damping for a non-linear system. A classic example of limit cycle oscillations is the Van-der-Pol oscillator[15]. Non-linear behavior also prevents analytical solutions in most cases, especially when the non-linearity is related to the aerodynamics. Hence to obtain the aerodynamic forces for a compressible viscous flow, CFD analysis has to be performed in almost all cases.

The fluid dynamic forces and control forces form the right hand side of the aeroelastic equations of motion. For a two degree of freedom airfoil as shown in Fig. (1.2), the aeroelastic equations of motion are

$$\mathbf{m}\ddot{\mathbf{h}} + \mathbf{s}_\alpha \ddot{\alpha} + \mathbf{k_h h} = -\mathbf{L} \tag{1.2}$$

$$\mathbf{I}\ddot{\alpha} + \mathbf{s}_\alpha \ddot{\mathbf{h}} + \mathbf{k}_\alpha \alpha = \mathbf{M}$$

In Eq. (2), h is the vertical displacement of the elastic axis as given in Fig. (1.2), α is the rotation of the airfoil about the elastic axis, m is the mass of the airfoil, I_α is the mass moment of inertia about the elastic axis, s_α is the first moment of the area about the elastic axis, k_h and k_α are the stiffnesses in plunge and pitch respectively. For an adaptive wing, changes in the geometric twist of the wing at various sections play a major role in design considerations. The variable twist wing has attracted attention recently[16]. Varying the twist by using adaptive material, requires change of torsional rigidity. As mentioned earlier, the torsional rigidity can also be altered by using adaptive

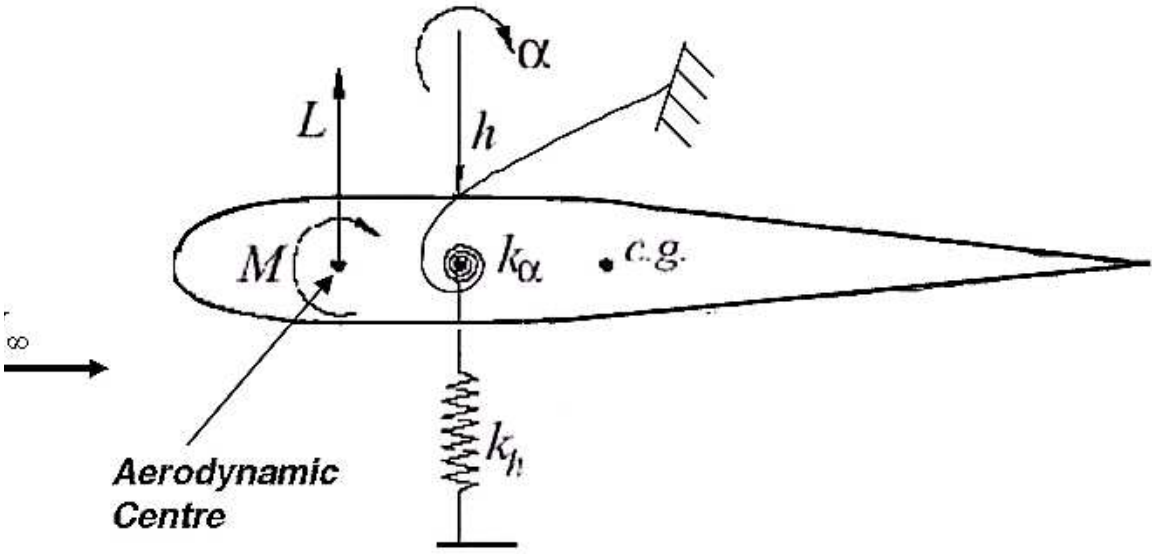


Figure 1.2: **A two degree of freedom airfoil with pitch and plunge motion capability situated in a uniform free stream**

spars. For quasi steady systems, such variations play a small role in the dynamics of the aircraft. However for a rapidly oscillating system, such as the one governed by Eq. (1.2), this form of variation in the twist makes the differential equations of motion have variable coefficients, i.e. makes them a time-variant system.

1.3 Aeroelasticity of a Morphing Surface

The analysis of the flight of birds has attracted a lot of research. Birds possess a light weight skeletal structure covered by a skin. Adaptive wing aircraft can be similar to birds in their construction. The inner spars and ribs need to be stiff and strong enough to support various maneuvering loads

on the aircraft and the outer skin needs to be flexible enough to facilitate shape changes. The effect of the wings and the tail of a bird on flight have been studied by different authors[17,18]. Reference 17 shows that birds use their tail very effectively for yaw and roll control. Considering that modern unmanned aircraft and fighter aircraft wish to avoid tails to decrease their radar signature, all-wing adaptable aircraft need to have effective lateral directional control. All-wing aircraft such as the XB-35 use split flaps on the wing tips to produce yawing moments created by unsymmetrical drag forces on the wings. Aircrafts such as the YB-49 also use drag rudders for roll control and to eliminate adverse yaw[19]. Many modern fighter aircraft are all-wing aircraft or aircraft without vertical surfaces. The lateral directional stability of such aircraft needs to be analyzed. Conventional tailless aircraft use split flaps on wing surfaces by using separated flow on one wing as a means of generating yawing moment. The equivalent device for an adaptable wing aircraft are ”spoiler bumps”. This means that the adaptive nature of the wing is used to deform small sections of the wing to form bumps which separate the fluid flow locally. This separated flow causes localized increase in drag over one wing. This unsymmetrical drag force can produce a significant yaw moment. However, such a procedure results in loss of lift over the wing on which the flow is separated. Since the wing is adaptable, a cambering of the wing section is also possible in flight. A selective local cambering of wing sections enables the aircraft to increase both its lift and provide for roll control. In fact, recently it has been shown[20] that the variable camber trailing edge can prevent the roll reversal phenomenon from occurring even at flight speeds at which the traditional ailerons are no longer effective. Cambering of the trailing edge as opposed to a deflected flap also eliminates sharp pressure jumps near the flap hinge. This is because, unlike the sharp discontinuity in

curvature that the deflected flap causes, variable cambering of the trailing edge ensures curvature continuity.

Adaptive materials are used to form bumps on the wing surface for a variety of reasons such as reduction of wave drag under transonic flow[21] and boundary layer control[22]. Adaptive bumps on wings can also be used for lateral directional control. Cole[23] describes flutter control of a rectangular wing with spoilers placed at 10% of the chord. This was done for a wind tunnel model. This is an action that can be made feasible in flight by using morphing wings which have adaptable leading edge bumps. Hence the aeroelastic behavior of these bumps contribute to studying the flutter control of adaptable wing aircraft. Adaptive shape control of the airfoil can control buffeting[24], another dynamic aeroelastic phenomena. Buffeting is the structural response to flow separation. Control of buffeting by adaptive shape changes tells us that other aeroelastic effects can be controlled by changing the aerodynamics. The use of adaptive devices in wings for flow control has also received wide attention. For example, periodic excitation at the leading edge has been used to control flow separation[25]. This is because the velocity of the fluid normal to the oscillating surface is unsteady. This unsteady term in the governing boundary layer equations allows for a reduced adverse pressure gradient which helps prevent flow separation. Sinha[26] has elaborated on the use of the Active Flexible Wall (AFW) on aircraft wings to control boundary layer separation. Here, a small transducer located at the leading edge of the wing provides for an oscillating wall which enables boundary layer control. Modifying the curvature at the trailing edge can increase circulation due to Coanda effect[27]. The Coanda effect mainly states that the fluid flowing along a curved surface will follow the curvature of that surface. Thus if the trailing edge

of the airfoil is given an acute curvature downwards, then the circulation over the airfoil is greatly increased, thus increasing the lift.

Considering that the adaptive nature of the wing is used for different purposes, it is beneficial to study the formation of such bumps. These bumps are formed by actuator forces acting on elastic material under aerodynamic forces. Hence the formation of such bumps involves study of aeroelastic behavior. This sort of aeroelastic effects on a wing is different from classical aeroelastic problems such as flutter, divergence, buffeting. The aeroelasticity of a bump on the wing, that is caused by deforming localized portions of the wing skin, may be non-linear in terms of both the structural response and the aerodynamic response. The structural behavior is non-linear because the deformations are large in comparison to the size of the wing element that is undergoing that deformation. The fluid dynamics is non-linear due to flow separation or due to the presence of a shock in transonic flow.

Investigation of such non-linear aeroelastic systems is very beneficial as it would describe a technique that can be used for other non-linear aeroelastic problems and also provide insights for the aeroelastic analysis of a general morphing wing. Currently, the investigation of fluid dynamic non-linearities in the aeroelastic response is through the use of CFD data. Since CFD calculations are expensive and time consuming, many researchers have adopted certain reduced order models to capture the CFD analysis effectively for large time using short runs of CFD calculations. Silva[28] describes the approach of a two term Volterra series model to study fluid dynamic non-linearity. This form of mathematical reduced order models have been used by many researchers[29]. However, in this work, a neural network approach is taken to represent the data obtained from computational

fluid dynamics. These details underline the need to study the aeroelastic behavior of morphing wing surfaces in a manner that is different from conventional aeroelastic approach, but still rapid in its analysis.

1.4 Time Variant Structural Dynamics

The structural dynamics of rotary wings, such as those in helicopters involve differential equations with time-varying coefficients, the variation basically being periodic. These type of differential equations are termed Mathieu-Hill differential equations and they are of the form

$$\ddot{\mathbf{x}} + (\mathbf{a} - \mathbf{q}\cos(\lambda \mathbf{t}))\mathbf{x} = \mathbf{0} \tag{1.3}$$

where a, q, λ are constants. Linear time-variant equations such as the Mathieu-Hill equation occur in many applications such as dynamic buckling of structures and wave propagation in periodic media. Here the time-varying coefficients are periodic. This type of variation can be analyzed by the application of Floquet theory[30]. Periodic variation in system parameters has been investigated in non-linear systems also using an extension of the Floquet theory. Sinha[31] describes the method of obtaining the state transition matrix of periodic systems in terms of Chebyshev polynomials of the first kind. By using this solution in a Liapunov-Floquet transformation, the stability of the equation of motion can be determined. Another type of structure which can possess a time varying stiffness under rapid oscillations is a structure with a propagating crack. One of the most catastrophic failures is failure due to fatigue crack growth. This crack growth is gradual initially, but after a certain critical length, the speed of the crack growth can reach the Rayleigh wave speed

in that material. Hence it is extremely important that the crack length is below critical length. The flutter calculations for a structure which possesses a gradually propagating crack should be based on differential equations such as Eq. (1.2), but with time varying stiffness. This would allow to predict the decrement of flutter speed with crack length.

As mentioned in Ref. 6, it is possible to provide for a variable torsional stiffness to the aircraft wing by using adaptive wing spars. While the details of the type of stiffness variation are not known, it is known that a continuous stiffness variation is possible and that the stiffness needs to be monotonically increasing or decreasing following some power law. This stiffness variation can be modeled based on other active stiffness variations. For example, the active control of the stiffness of vehicle suspensions[32] is in accordance with the time constant of the controller. Thus the stiffness takes a form $k_{new} = k_{old} + (1 - e^{-t/\Delta t})\Delta k$. Therefore the variation in torsional rigidity can also approach such a variation, i.e. an exponential change. Hence it is required to choose methods other than those used for periodic systems in order to solve the differential equation. One method would be to use a stable time integration procedure such as the Wilson - θ method[30]. However, for a rapidly time varying system, numerical time integration may result in erroneous results since the natural frequency ω_n being time varying, may cause the time step chosen initially for integration to be too small or too large. The former would lead to an increase in the computational cost, while the latter to an increased error. The presence of a closed-form analytical solution would enable us to avoid these numerical difficulties. Closed form solutions to the state transition matrix are obtained only for certain cases[30,33,34].

Herein, a scheme is used wherein, the state transition matrix that is obtained as an exponential

of a matrix, can be applied to any linear time-variant system. Therefore the analytical solutions can be used for time-variant systems without setting any bounds on the type of variation of the parameters, except for the fact that the response of the system will be linear. The novelty of the present scheme is that although we borrow the basic approach from time invariant systems, we still retain an analytical formulation without averaging the time varying parameters such as in Ref. 35, so that the response of time-variant systems is continuous in time and not a numerical solution available at only some discrete points. A detailed analysis of Linear Time-varying Systems (LTV) for single degree of freedom systems is presented in Chapter 2.

1.5 Geometric Modeling for Adaptable Surfaces

The aeroelastic behavior of adaptable airfoils and wings requires modeling the geometry of the airfoil first. Standard airfoils are usually modeled geometrically by means of power series for the camber and thickness. This sort of a representation is not ideal for an arbitrarily varying airfoil. Secondly, it is required that localized variations in shape be captured without affecting the airfoil geometry at other locations. The approach of conformal mapping as elucidated by Jones[36] is a powerful tool for airfoil modeling. However it cannot support arbitrary localized variations in shape. Herein, the method of B-splines is used to model airfoils. B-splines are parametric splines based on an iterative calculation. They are defined using a set of control points, a knot vector and the degree of the curve. The knot vector is a vector that discretizes the parametric representation of the B-spline curve. The control points are locations in the modeling space that are used to

define the shape of the object and hence also to alter the shape in any manner. The Cox-de-Boor[37] iterative formula is used for evaluating the spline points. A comprehensive description on B-splines can be found in the book by Rogers and Adams[38] and also in Farin[39]. Another advantage of the use of B-spline representation is in optimization. The degree of the B-spline is independent of the number of control points chosen. Therefore a cubic polynomial could be chosen and the number of control polygons can be optimized for a proper representation of the airfoil. Figure 1.3 describes a Karman-Trefftz[36] airfoil modeled by B-splines.

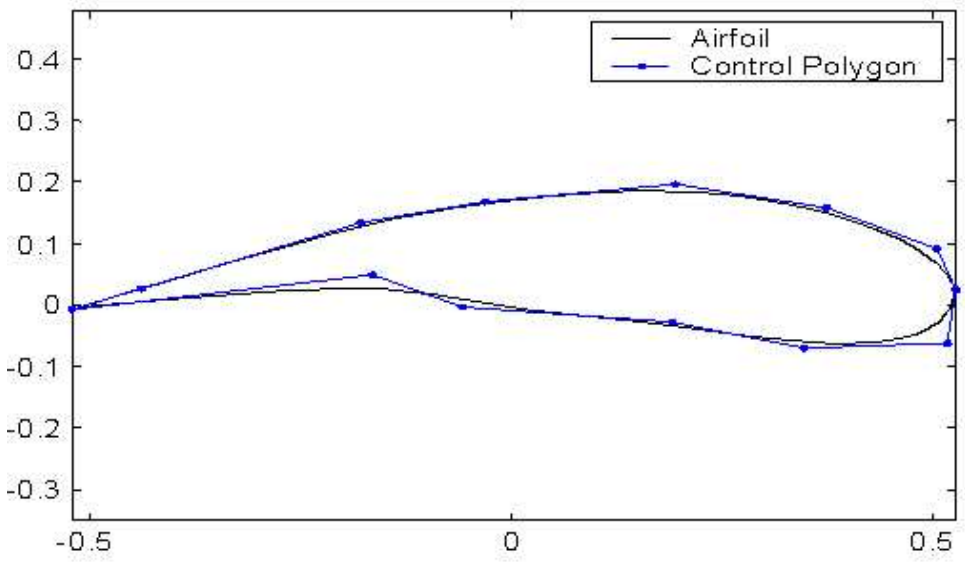


Figure 1.3: **A B-spline model of an airfoil with a finite angle trailing edge showing the airfoil and its control polygon**

Figure 1.3 shows an airfoil constructed with one fourth degree B-spline curve with 13 control points and with a non-uniform knot vector. The non-uniform knot vector is required to construct the pointed trailing edge. By repeating the parametric value of the B-spline curve in the knot

vector, one can reduce the degree of continuity of the curve at that parametric value[38]. Hence this pointed trailing edge which will be very difficult to obtain with a cubic spline, is easy to obtain using a B-spline by repeating the parametric value of the curve at the trailing edge $n-1$ times in the knot vector, where n is the degree of the curve.

1.6 Aeroelastic Analysis Using Neural Networks

Response surface methodology and neural networks have been adopted by certain authors like Rai and Madavan[40]. Response surfaces are usually polynomial series approximations to the referred data. Neural networks on the other hand are more sophisticated response surfaces as they include non-linear functions operating on polynomial expansions. However, once the model has been obtained, neural networks can take more floating point operations in the simulation than response surfaces. Both types of models have been used in aerodynamic optimization [40]. Since neural networks can model a wide variety of system behavior, both linear and non-linear, they are very versatile in modeling non-linear aeroelastic response. Further, sophisticated toolboxes exist in readily available commercial packages such as MATLAB for creating a variety of neural networks. The concept of neural networks is an attempt at an artificial simulation of the functioning of the human brain[41]. Design of a neural network consists of arranging "neurons" in different layers, deciding on their connectivity, defining the inputs and outputs of the system and determining the strength of each connection in between the neurons. The output of each layer of neurons is fed to the next layer of neurons through a selected function. The strength of the neuron connections is

based on training a neural network on a series of data sets. Once a neural network has been trained using sufficient data, then the network is capable of providing the desired output given a new set of input data. Training of the neural network is a very important task and care should be taken that the neural network is able to reproduce the target data without over-fitting. This means that the error margin of the neural network away from the target data should not be significantly greater than near the target data. The neural network toolbox in MATLAB is used for the training purposes. Two popular methods of training, namely the Levenberg-Marquardt scheme and the scheme of "Resilient Back-propagation" are used.

There have not been significant publications in using neural networks as a rapid aeroelastic solver that can determine the aeroelastic response by representing the actual fluid-structure coupling. Most applications of neural networks in aeroelasticity have been in system parameter estimation for aircraft control [42] or in flutter speed prediction and flutter control from experimental data [43,44]. The present dissertation looks at flutter suppression as only one aspect of a broader goal of representing the fluid-structure coupling in adaptive wing aircraft using neural networks. Since the coupling between the aerodynamics and the structural response is the most important factor that has to be taken into account in computational aeroelasticity, the present work looks at an integrated fluid-structure interaction tool using neural networks, that is capable of calculating the static and dynamic aeroelastic response of adaptive structures without performing iterations due to fluid-structure coupling.

The training of the neural network is accomplished using aerodynamic data from a CFD solver or from a panel method and structural data from a finite element solver. The finite element analysis

could be spatial for a static aeroelastic problem such as the study of a deforming bump on a wing surface or Time-Finite element based, such as required for a structural dynamic's problem. Given a set of input data that describes the configuration of an airfoil and the type of flow, to the trained neural network, it is capable of predicting the aeroelastic behavior of that airfoil. In the case of an oscillating airfoil, the input data of the airfoil should be at each time instant considered in the oscillation. In many cases, it is simpler to consider many trained neural networks communicating with each other for representing the aeroelastic behavior instead of a single neural network. For dynamic aeroelastic behavior of time-varying systems, the method developed herein for analysis of such systems can provide a building block for training artificial neural networks also.

These two approaches to calculate the aerodynamic forces over an airfoil, i.e. CFD for viscous flows and panel methods for irrotational flows, enable the training of neural networks for rapid aeroelastic analysis of morphing airfoils. These methods indicate the limitations of analytical closed form solutions such as the Theodorsen solution. Though the Theodorsen model and later models by Wagner and Kussner provide good estimates of irrotational unsteady aerodynamics, their usage in non-linear aeroelasticity and aeroservoelasticity may not provide the best computational model. Particularly with adaptable wings, the main goal is to broaden the flight envelope with the best design in each flight condition. This requires accurate aeroelastic computations and a means for high fidelity control over fluid-structure interactions. In order to provide for an active control of the aeroelastic response, a neural network modeling of the aeroelastic system may provide a rapid fluid-structure response solver.

Therefore a complete representation of the two dimensional aeroelasticity of adaptable wings is

made. This means the modeling of the aeroelastic system is made with adaptive geometry, computational fluid dynamic models, numerical aerodynamics, expert systems such as neural networks and an efficient computational tool for time-variant structural dynamic systems. The use of modern computational resources is made use of to the fullest extent in determining a rapid and accurate aeroelastic system model. Chapter 2 details the method of solving linear time-variant systems. The method is based on time-finite elements in the sense that the time domain of interest is broken down into small intervals and an analytical solution is used for each time interval. The analytical solution within a time interval is based on a matrix exponential that is the solution to the equations of motion of the time variant system, when formulated as an exact differential equation.

Various examples of time-varying systems are solved to prove the efficiency of the scheme. Chapter 3 introduces the technique of artificial neural networks to analyze static and dynamic problems. Different types of deterministic neural networks are described and their ability to model static and dynamic problems is discussed. Chapter 4 uses trained neural networks to solve a non-linear aeroelastic problem, it being, the behavior of an adaptable bump on the surface of an airfoil in causing flow separation with maximum pressure drag with minimum loss in lift and minimum strain energy of bump deformation. This is a novel example of a static non-linear aeroelastic problem and it requires the use of CFD and non-linear finite element analysis in order to train the neural networks to represent the aeroelastic response. Chapter 5 extends the concept of neural networks to dynamic aeroelasticity with time varying stiffness. The unsteady aerodynamics is described by calculating the effect of doublet panels distributed on the surface of an airfoil immersed in a free stream. The feasibility of using artificial neural networks to represent periodic airfoil oscillations

is investigated. The neural network is trained using unsteady aerodynamic data obtained from the doublet panel code and the matrix exponential time marching used for solving the dynamics of time-variant systems. Chapter 6 concludes this dissertation and discusses the effectiveness of using neural networks for representing the aerodynamics and the aeroelastic response of a morphing airfoil.

Chapter 2

Linear Time Variant Systems

The presence of time varying coefficients in the differential equation for a linear dynamic system generally prevents calculation of an analytic solution. In general if the mass of the system is not rapidly varying with time, single degree of freedom (SDOF) linear time-variant systems are represented by the following equation:

$$\ddot{\mathbf{x}} + \frac{\mathbf{c(t)}}{\mathbf{m}}\dot{\mathbf{x}} + \omega_{\mathbf{n}}{}^{\mathbf{2}}\mathbf{(t)x} = \frac{\mathbf{f(t)}}{\mathbf{m}} \tag{2.1}$$

where the terms c, ω_n, m, f refer to the damping, natural frequency, mass and external force respectively.

Linear time-variant equations such as the Mathieu-Hill equation occur in many applications such as dynamic buckling of structures and wave propagation in periodic media. The history of linear time-variant dynamic systems dates back more than a century. Mathieu-Hill equations have the

form

$$\ddot{\mathbf{x}} + \mathbf{a}(\mathbf{1} - \lambda\,\mathbf{cos}(\mathbf{qt})) = \mathbf{0} \tag{2.2}$$

Here a, λ, q are constants depending on the nature of the problems to be solved. These type of equations occur in many physical applications such as the dynamic buckling of a beam, stability of side rods in locomotives, vibration of helicopter blades, etc. An application of periodic variation in parameters in mechanical devices is in the meshing of spur gears. The damping can also be varying with time, it being usually proportional to the stiffness. This type of variation can be analyzed by the application of Floquet theory[30,32].

The use of smart materials in wings allows the active control of the deflections of the wing structure. These adaptive structures are preliminary models of biological structures. One of the properties of biological structures is that they can possess time varying stiffness while executing movements[45]. In Ref. 45, the human elbow is explained to possess variable stiffness while executing movements. The stiffness could be exponential varying such as in the case of tactile sensors for end-effectors of robots as mentioned in Ref. 46. Another common example wherein one might encounter exponential variation in the mass or stiffness terms is the time varying modulus of rigidity in a structure with internal damping that depends on the strain rate[47]. These type of equations involve sophisticated mathematics even for a one dimensional problem. To avoid this, numerical techniques are often used[48]. Analytical techniques for single degree of freedom (SDOF) time-variant systems under free vibration can be obtained by using Bessel functions[49]. Using specialized functions for solving the differential equation of motion for free vibration of SDOF systems could mean that the solution procedure to arrive at an analytic representation could

be different when considering periodic and aperiodic variations in the coefficients. Therefore the approach herein is to derive a consistent analytical approach that can be used for any type of variation in the system parameters. A conditionally stable time integration procedure such as the Wilson - θ method for a rapidly time varying system, may result in erroneous results since the natural frequency ω_n being time varying, after a passage of time, the time step chosen in the integration may be too small or too large. The former would lead to increase in computational cost and the latter to an increased error. Therefore this chapter endevours to explain the formulation of a near -exact analytical solution to any linear SDOF time-variant system under free or forced oscillation.

Consider the governing differential equation of a linear time-variant system written in state-space form

$$\{\dot{\mathbf{x}}\} + [\mathbf{C(t)}]\{\mathbf{x}\} = \mathbf{0} \tag{2.3}$$

This can be expressed as an exact differential equation

$$\frac{\mathbf{d}}{\mathbf{dt}}(\mathbf{e}^{\int_0^{\mathbf{t}}[\mathbf{C(t)}]\mathbf{dt}}\{\mathbf{x}\}) = \mathbf{0} \tag{2.4}$$

if the exponential of the matrix $e^{\int_0^t [C(t)]dt}$ and $[C(t)]$ commute, that is

$$\mathbf{e}^{\int_0^{\mathbf{t}}[\mathbf{C(t)}]\mathbf{dt}}\,[\mathbf{C(t)}] = [\mathbf{C(t)}]\,\mathbf{e}^{\int_0^{\mathbf{t}}[\mathbf{C(t)}]\mathbf{dt}}$$

Here $C(t)$ is a matrix of time varying coefficients. Usually, the only times when the matrices $C(t)$ and $e^{\int_0^t [C(t)]dt}$ commute, are when [C] is diagonal or when [C] is a matrix of constants. None of these is the case here. However as we shall show, Eq. (2.3) can still be used to describe linear differential equations with time varying parameters, but within a certain time duration; the duration being dependent on the rate of variation of parameters. Appendix (A) gives a mathematical explanation for defining the state transition matrix as in Eq. (2.3), even though the matrices involved do

not commute. We investigate the exactness of the solution to the governing Eq. (2.2), expressed by Eq. (2.3), in terms of the rate of variation of the system parameters. Using the solution for this time duration, we can march in time to obtain the response during the entire duration of interest. Thus we shall arrive at a technique which is applicable to any linearly time-varying system.

2.1 The Exponential of a Matrix

Equation (2.4) requires the evaluation of an exponential of a matrix. This can be done in a number of ways. Assuming distinct eigenvalues, the following methods are feasible.

1. Taking the Laplace transform :- For a time-invariant system, this method is very efficient as in

   $e^{[C].t} = \mathcal{L}^{-1}[sI - C]^{-1}$

   However, when the matrix $[C]$ is time varying, the method is not so straight forward. This is because, the Laplace transform of $e^{[C(t)]}$ may possess high degree polynomials in the denominator which implies that the inverse Laplace transform may be difficult to evaluate, requiring a large number of partial fraction expansions.

2. Using the eigenvectors for diagonalization:-

   $e^{[C(t)]} = P\, e^{\left[P^{-1}\, C\, P\right]}\, P^{-1}$.

   where P is the matrix of eigenvectors of C. This method is also computationally highly expensive, since it requires the determination of the eigenvectors of an arbitrarily varying matrix.

3. Using Sylvester's theorem :- A closed form expression can be found[34] as

$$\mathbf{e}^{[\mathbf{C(t)}]} = \beta \begin{pmatrix} \frac{c_{11}-c_{22}}{2} + \frac{\triangle}{tanh(\triangle)} & c_{12} \\ c_{21} & \frac{c_{22}-c_{11}}{2} + \frac{\triangle}{tanh(\triangle)} \end{pmatrix} \tag{2.5}$$

where $2\triangle = \sqrt{(c_{11}-c_{22})^2 + 4\,c_{12}c_{21}}$ and
$\beta = e^{\frac{c_{11}+c_{22}}{2}} \frac{sinh(\triangle)}{\triangle}$

In the present chapter, Eq. (2.5) will be used for the evaluation of the exponential of a matrix since it is in a closed form and hence computationally, it is easier than methods (1) or (2). Nineteen different methods for evaluating the exponential of a matrix are commented upon by Moler [50].

2.2 Response of the Mathieu Equation

The classical problem of the buckling of a slender beam under periodic loads is governed by the Mathieu-Hill equation

$$\ddot{\mathbf{x}} + \mathbf{2}\,\xi\dot{\mathbf{x}} + \mathbf{\Omega^2}(\mathbf{a} - \mathbf{q}\mathbf{cos}(\lambda\mathbf{t}))\mathbf{x} = \mathbf{0} \tag{2.6}$$

where $\Omega^2 = \frac{1}{m}\,(\frac{n\pi}{l})^2$, $a = EI(\frac{n\pi}{l})^2$, q is the amplitude of the periodic external force with a frequency λ applied to the system. Here ξ is a measure of the system damping, m is the mass of the beam and n is the buckling mode.

Conventional methods of solving this equation make use of the periodicity of the loading which allows for the application of the Floquet theory. Accordingly, Eq. (2.6) is first transformed into an undamped Mathieu-Hill equation by the transformation $y(t) = exp(\xi t)\,x(t)$. This reduces Eq.

(2.6) to the form

$\ddot{y} + \Omega^2(b - qcos(\lambda t))y = 0$ where $b = a - \dfrac{\xi^2}{\Omega^2}$.

Using Floquet theory, the solution to the above equation can be written in the form of

$$\begin{aligned} \boldsymbol{y_1(t)} &= \boldsymbol{exp[\mu\, t]\ \phi(t)} \\ \boldsymbol{\phi(t)} &= \boldsymbol{\sum_{j=0}^{m} q^m\ A_m\ exp(i\ m\ t)} \\ \boldsymbol{y_2(t)} &= \boldsymbol{y_1(t)} \\ \boldsymbol{y(t)} &= \boldsymbol{y_1(t) + y_2(t)} \end{aligned} \tag{2.7}$$

In Eq. (2.7) , $i = \sqrt{-1}$, μ is a complex constant dependent on the constant coefficients in Eq. (2.6). The complex conjugate of $y_1(t)$ is denoted by $y_1^*(t)$. Further details of the Floquet method may be found in several texts (See Richards[30], for instance).

To illustrate the simplicity and accuracy of the present method, Eq.(2.6) is written in the state space form and the matrix [C] in Eq. (2.3) is given as:

$$C = \begin{pmatrix} 0 & -1 \\ \Omega^2(a - qcos(\lambda t)) & \xi \end{pmatrix}$$

Let us assume $\xi = 0.05$, $a = 1$, $q = 0.1$, $\Omega = 1$, $\lambda = 0.25$

We use Eq. (2.3) to obtain the solution. After performing the integration, $\int_0^t [C(t)]dt$

$$\int_0^t [C(t)]dt = \begin{pmatrix} 0 & -t \\ t - \frac{q}{\lambda}\, sin(\lambda t)) & \xi t \end{pmatrix}$$

the exponential of a matrix is evaluated by means of Eq. (2.4), and assuming initial conditions as $x(0) = 1$, $\dot{x(0)} = 0$, the solution can be obtained from Eq. (2.3). Plotting the response obtained by the above solution and comparing it with a numerical solution of the Mathieu equation generated by Mathematica, Fig. (2.1), we find the solution to be near exact. However, the erroneous behavior of the response can be observed as the present solution overestimates the actual response. The response is very accurate for times close to the initial conditions. Making use of this observation, a modified approach is used.

The state transition matrix has been assumed to be the exponential of a matrix, Eq. (2.3), and the plot of the solution in Fig. 2.1 shows that this is valid for small time instants close to the initial time. In order to minimize the error with an increase in time, we divide the time duration of interest into intervals of 0.25 seconds and use the response at the end of each 0.25 second interval as the initial conditions for the next 0.25 second interval. In other words, the response of the system is governed by the following procedure.

$$\begin{pmatrix} x_{i+1} \\ \\ \dot{x}_{i+1} \end{pmatrix} = \mathbf{\Phi(t)} \, . \begin{pmatrix} x_i \\ \\ \dot{x}_i \end{pmatrix} \tag{2.8}$$

where $\Phi(t)$ is the state transition matrix, described previously in Eqs.(2.3) and (2.4). The result of this procedure is shown in Fig. 2.2. This result is compared with the numerical solution that uses an implicit Adams method and is generated by Mathematica. Figure 2.2 shows that the solution to the Mathieu-Hill equation as described by Eqs. (2.3) and (2.6) is valid. Since the implicit scheme generated by Mathematica gives a very accurate solution, it is taken as the exact solution for com-

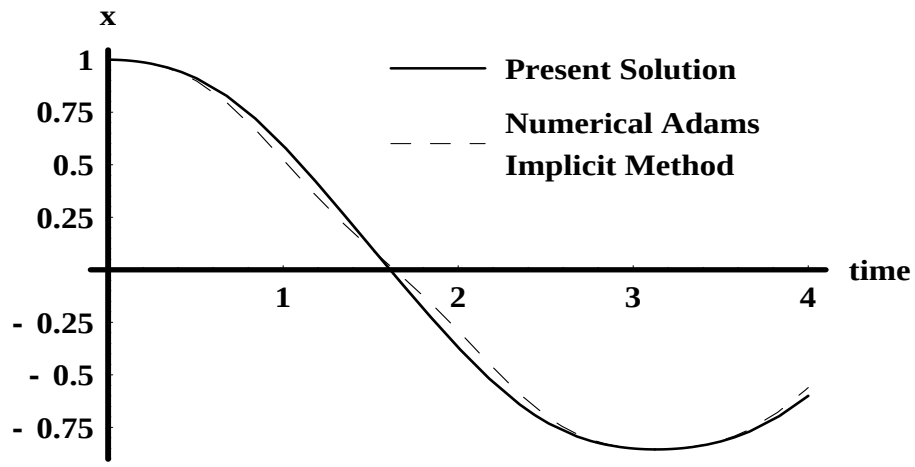


Figure 2.1: **Response of the Mathieu Equation** $\ddot{x}+0.1\dot{x}+(1-0.1cos(0.25t))x=0$ **by the present method without time marching and compared with the solution generated by Mathematica using an accurate implicit numerical scheme**

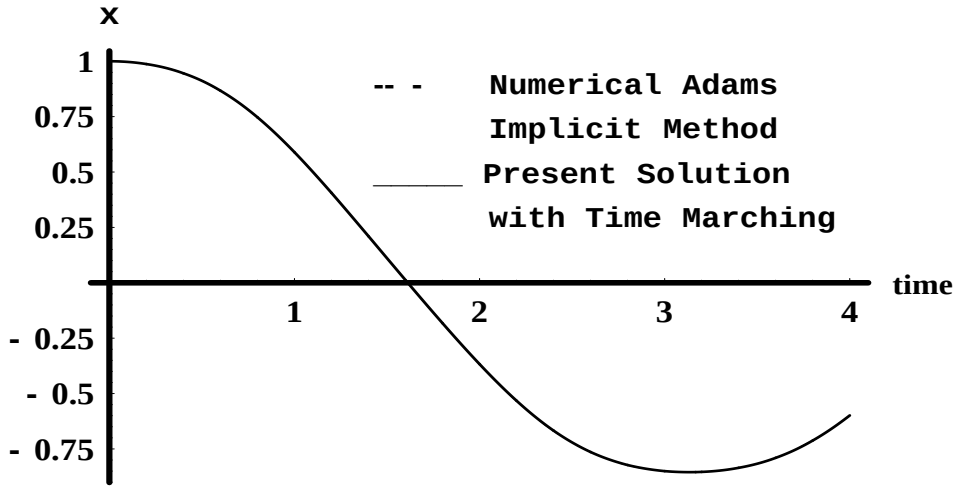


Figure 2.2: **Response of the Mathieu Equation** $\ddot{x}+0.1\dot{x}+(1-0.1cos(0.25t))x=0$ **by Matrix Exponential Time Marching with a time step of 0.25 seconds**

parison purposes. The order of the implicit Adams method that is being used by the Mathematica package is not known exactly. However, the implicit Adams method was seen to match with the solution generated using a Runge-Kutta 4th order iteration.

It can be seen that the results obtained by the modified approach are identical to the reference results as seen in Fig. 2.2. Thus for the time period considered, $\Delta t = 0.25sec$, the exponential of the matrix as given in Eq. (2.3) is the state transition matrix. Therefore the advantage of this modified method is that a closed form analytical solution needs to be calculated only once and it is applied to small intervals of time with each interval borrowing its initial conditions from the previous interval.

2.2.1 Response Sensitivity to Selected Time Step

The Mathieu-Hill equation i.e. Eq. (2.6), can be made to possess a more rapid variation by either increasing the value of q or by increasing the frequency of the harmonic term, $\cos(\lambda t)$. An estimate of the time step required is obtained by setting the L_2 norm of the error between the present solution and the reference solution to a prescribed value. A measure of the error in the response computed by the present solution can be obtained by noting the result of the non-commutativity of the matrices, that is

$$\mathbf{e}^{\int_{\mathbf{t_0}}^{\mathbf{t}}[\mathbf{C(t)}]\mathbf{dt}}\,[\mathbf{C(t)}] - [\mathbf{C(t)}]\,\mathbf{e}^{\int_{\mathbf{t_0}}^{\mathbf{t}}[\mathbf{C(t)}]\mathbf{dt}} = \epsilon(\mathbf{t}) \tag{2.9}$$

Therefore even if the exact solution to the problem is unknown, an estimate of the error is obtained as a function of time. This can be used to estimate the time step required . In the present example,

Eq. (2.6), the main sensitivity parameter is not q or λ but $\frac{q}{\lambda}$. It was determined using Eq. (2.9) that the time step chosen, i.e. 0.25 second is appropriate for all $\frac{q}{\lambda} \leq 0.4$ when the L_2 norm of the error is fixed at 0.001. Similarly one can analyze the time steps required for other values of $\frac{q}{\lambda}$.

2.3 Rapidly Time-Variant System

The method of obtaining the response of a linear time-variant system by using Eq. (2.3) is now applied to a rapidly varying time-variant system. The system has an exponentially decreasing stiffness. This type of a system is chosen since 1) The variation in its system properties with respect to time is very high, thus ensuring that assumptions of slowly varying systems cannot be applied, 2) The system is unstable. Hence if the current methodology is applicable here, it can be used in other circumstances where an instability needs to be predicted. The system is governed by the equation

$$\ddot{\mathbf{x}} + \omega_{\mathbf{n}}{}^{\mathbf{2}}(\mathbf{t})\mathbf{x} = \mathbf{0} \tag{2.10}$$

where the time varying natural frequency is given by $\omega_n{}^2(t) = 40\, e^{-t}$. The variation in the natural frequency chosen above is similar to the one used by Li[49]. The matrix, $\int_{t_0}^{t} C(t)dt$ in Eq. (2.3) is then obtained as :

$$\int_{t_0}^{t} -C(t)dt = \begin{pmatrix} 0 & t - t_0 \\ 40e^{-t} - 40e^{-t_0} & 0 \end{pmatrix}$$

where t_0 is the initial time. The exact solution for this type of a differential equation, is given by a summation of Bessel functions[51]. Plotting the result and comparing with the reference solution, Fig. 2.3, the present solution obtained using Eq. (2.3) is clearly erroneous. Considering that we

are modeling a time-variant system by an exponential of a matrix, this error is not large. However we can reduce the error and reach an almost exact fit to the reference solution. We divide the time duration of interest into steps of 0.2 second duration each and use Eq. (2.8), with the state transition matrix being given by Eq. (2.5). Thus using one analytical result given by equation (2.4), but with different initial conditions for each time interval chosen, one can provide a near-exact solution for all subsequent time. Figure 2.4 shows that the present solution matches the numerical solution to the system with an implicit Adams numerical scheme as generated by Mathematica.

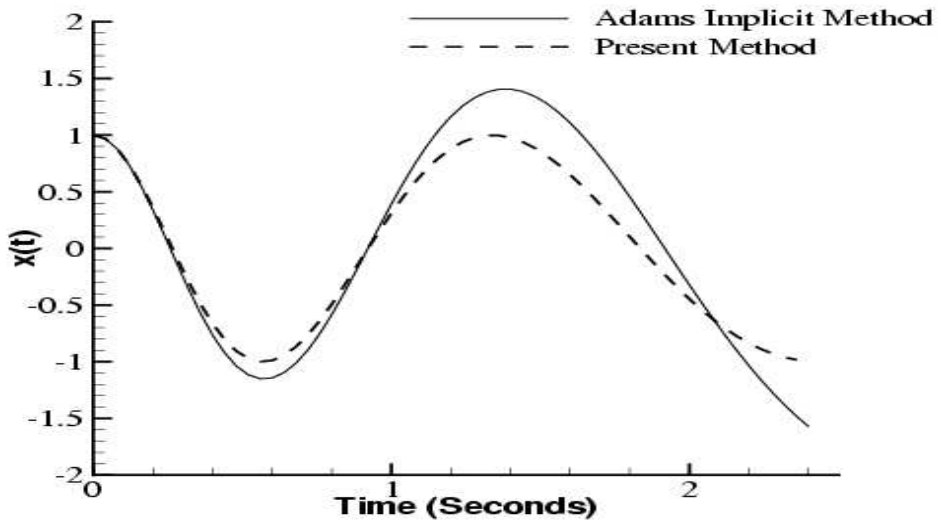


Figure 2.3: **Response of the rapidly varying system,** $\ddot{x} + 40\ Exp[-t]x = 0$ **using the present method but without resorting to a time marching procedure**

Figure 2.5 shows the same result upto 40 seconds. The present method captures the reference solution very well indeed. The results also tally with a fourth order Runge-Kutta method. However, the Runge-Kutta method requires a time step which is an order of magnitude less than the present method. We now compare our scheme with the widely used, Newmark-β and Wilson-θ

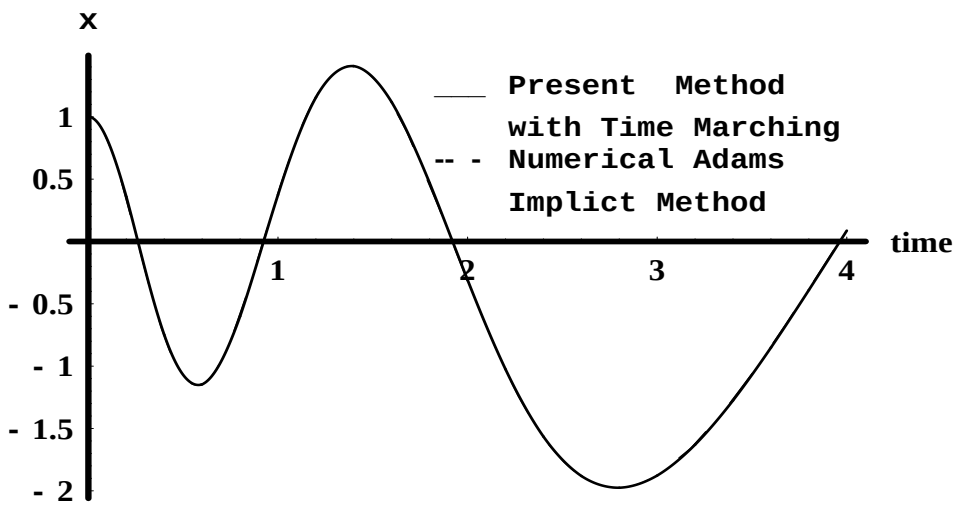


Figure 2.4: **Response of the rapidly varying system,** $\ddot{x} + 40\ Exp[-t]x = 0$ **using Matrix Exponential time marching with a time step of 0.2 seconds.**

methods. Among the explicit schemes, the fourth-order Runge-Kutta is seen to perform well. Figure 2.6 shows the response of the same system given by Eq. (2.10), but calculated by using the Newmark-β method and Wilson-θ method with a time step of 0.04 seconds. These are compared to the fourth-order Runge-Kutta scheme which gives an accurate solution. The present scheme is seen to match the Runge-Kutta method even though a time step one order higher in magnitude is used whereas the Newmark-β method and Wilson-θ method show erroneous results. The explicit methods use discrete time steps since it is a numerical iteration for obtaining solutions to the governing differential equation Eq. (2.8). The present method provides a continuous solution in each time interval of 0.2 second chosen in the present example.

Table 1 gives the error involved in using these numerical schemes and compares these schemes with the present methodology. The Newmark-β and Wilson-θ methods are extremely sensitive to

the time steps chosen. To obtain results given in Table 1, a time step of 0.04 second was used as an example. The time marching method with the exponential of a matrix was also performed at intervals of 0.04 second to facilitate comparison with the other methods. One can see that the

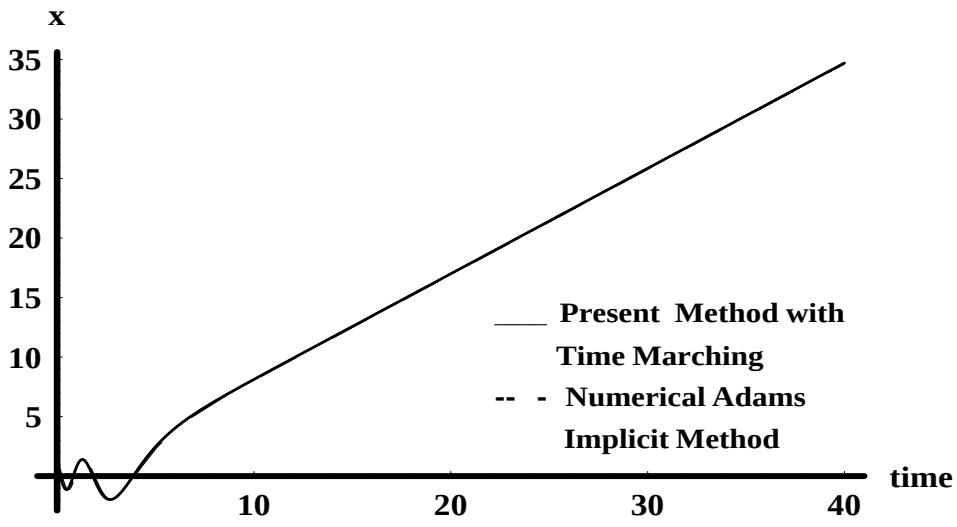


Figure 2.5: **Response of the rapidly varying System** $\ddot{x} + 40\ Exp[-t]x = 0$ **as computed using Matrix Exponential Time Marching for large time with a time step of 0.2 seconds**

present method can give excellent results even for a rapidly time-variant system. To approach the accuracy of the present method, the time steps needed for the Newmark-β and Wilson-θ methods has to be of the order of 0.002 second . Whereas, as shown in Fig. 2.5, using the time marching scheme with the exponential of a matrix, an accurate result is obtained even with a time step of 0.2 second, two orders of magnitude better than the two widely used methods.

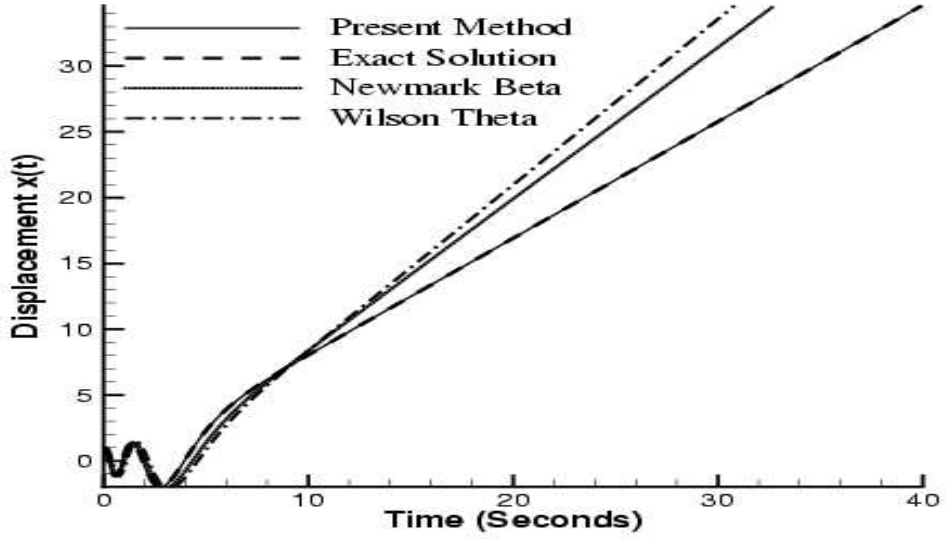


Figure 2.6: **Response of the rapidly varying System** $\ddot{x} + 40\ Exp[-t]x = 0$ **using explicit numerical schemes with a time step of 0.04 seconds and compared with the present method**

Table 2.1: Results for Free Vibration using a 0.04 Second Time Step

Displacement of mass after 40 seconds		
Method	x(t)-Free Vibration	Percentage Error from Exact
Newmark-β	38.9785	12.57%
Wilson-θ	40.8566	18.005%
Present Method	34.6580	0.102%

2.4 Forced Oscillations of a Time-Variant System

All of the examples given previously dealt with free vibrations only. Now we consider a forcing function. The exact differential equation formulation, Eq. 2.3 will be modified as

$$\frac{\mathbf{d}}{\mathbf{dt}}(\mathbf{e}^{\int_{\mathbf{t_0}}^{\mathbf{t}}[\mathbf{C(t)}]\mathbf{dt}}\{\mathbf{x}\}) = \mathbf{e}^{\int_{\mathbf{t_0}}^{\mathbf{t}}[\mathbf{C(t)}]}\,\mathbf{Q} \tag{2.11}$$

where Q is the vector containing the forcing function. The exact solution of the problem will be

$$\mathbf{x(t)} = \mathbf{x_h(t)} + \mathbf{x_p(t)} \tag{2.12}$$

$x_h(t)$ and $x_p(t)$ are respectively, the homogenous solution and the particular integral. The homogenous solution is the solution to the free vibration problem which has been examined in a great detail previously. The particular integral is given by

$$\mathbf{x_p} = \mathbf{e}^{\int_{\mathbf{t_0}}^{\mathbf{t}}[-\mathbf{C(t)}]\mathbf{dt}} \int_{\mathbf{t_0}}^{\mathbf{t}} \left(\mathbf{e}^{\int_{\mathbf{t_0}}^{\mathbf{t}}[\mathbf{C}(\tau)]\mathbf{d}\tau}\,\mathbf{Q}\right)\,\mathbf{dt} \tag{2.13}$$

Depending on the type of time variation present in the exponential of a matrix in the above equation, Eq. (2.11), the integral may be difficult to evaluate, but in most cases it can be simplified. To illustrate this procedure we consider the same example of Eq.(2.8), but with a forcing function. The governing equation of the system is :

$$\ddot{\mathbf{x}} + {\omega_\mathbf{n}}^\mathbf{2}\mathbf{(t)x} = \mathbf{2}\,\mathbf{cos(0.5t)} \tag{2.14}$$

where the time varying natural frequency is given by ${\omega_n}^2(t) = 40\ e^{-t}$. For simplifying the particular solution given by Eq. 2.11, we make the approximations that

$\lim_{z\to 0}\cosh(z) = 1$ and $\lim_{z\to 0}\sinh(z) -> z$

This does not cause any discrepancy for the time interval sizes chosen in this analysis that is of the order of 0.1 second. Having done this, the particular solution can be evaluated analytically in a closed form. Using equations (2.11) and (2.12), the exact analytical solution for a small time

period and the initial conditions for the subsequent time period can be calculated. Therefore the procedure for obtaining the response of a time-variant system under forced vibrations is written as

$$\begin{pmatrix} x_{i+1} \\ \dot{x}_{i+1} \end{pmatrix} = \mathbf{\Phi(t)} \, . \begin{pmatrix} x_i \\ \dot{x}_i \end{pmatrix} + \begin{pmatrix} x_{p_i} \\ \dot{x}_{p_i} \end{pmatrix} \tag{2.15}$$

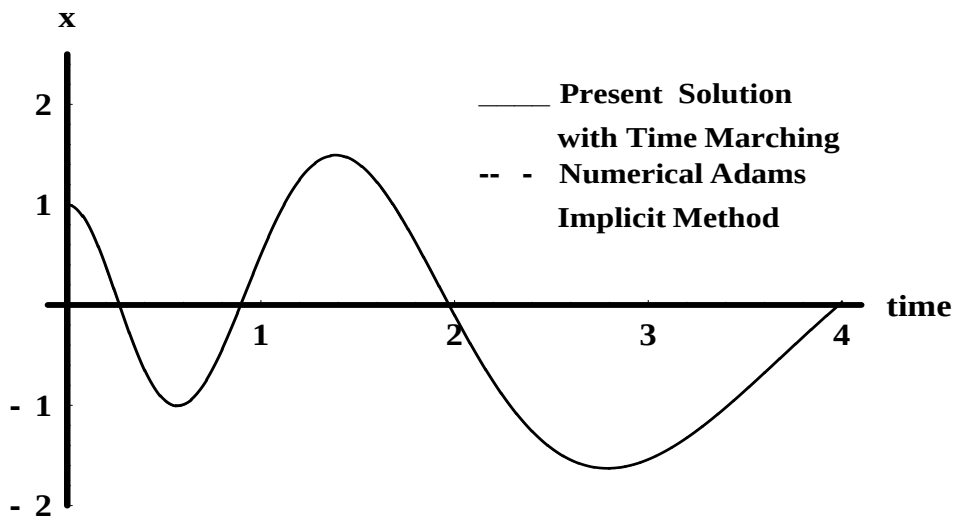


Figure 2.7: **Response of the Forced System** $\ddot{x} + 40\ Exp[-t]x\ =\ 2\ cos(0.5t)$ **using Matrix Exponential Time Marching with a time step of 0.08 seconds**

Figures (2.7) and (2.8) show the response of the forced damped system. The error between the solution using Adams implicit scheme and the present method is seen to be negligible. It can be verified from this that the present method can indeed predict the behavior of the system for all times. Again this method was compared with Newmark-β and Wilson-θ methods. Table (2) shows the comparison between the results obtained using the various methods. The present method is shown to be much more accurate than the other methods. This is because an analytical solution

was first determined. Hence this solution will be extremely accurate provided the time step selected for time marching is small enough such that the non-commutativity of the matrices involved in the exact differential formulation, Eq. (2.11), will not cause a discrepancy. For the present example, we have chosen a time step of 0.08 second. Depending on the nature of the variation, a suitable

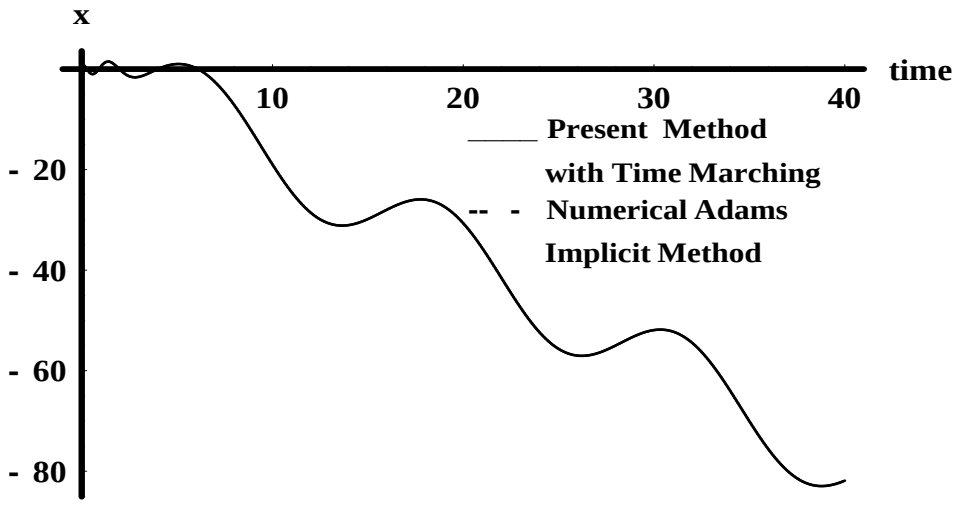


Figure 2.8: **Response of the forced system** $\ddot{x} + 40\ Exp[-t]x = 2\ cos(0.5t)$ **for large time using Matrix Exponential Time Marching with a time step of 0.08 seconds**

time step will need to be chosen. However, it is emphasized that the current methodology can use time steps one order larger than that required for methods such as the Wilson-θ and the Newmark-β procedures and still get much better results as shown in Table 2. Wilson-θ and Newmark-β methods were also run at a time step of 0.08 second. The value of the parameter θ in the Wilson method was chosen as 1.4 as this ensures unconditional stability. Both these schemes show a lot of sensitivity to the time step as shown in Fig. 2.9 wherein it is seen that the response of the system after a time of 40 seconds moves away from the exact value of -81.7 as the time step is increased

from 0.02 second to 0.08 second. The present scheme is quite insensitive to small variations in the time step.

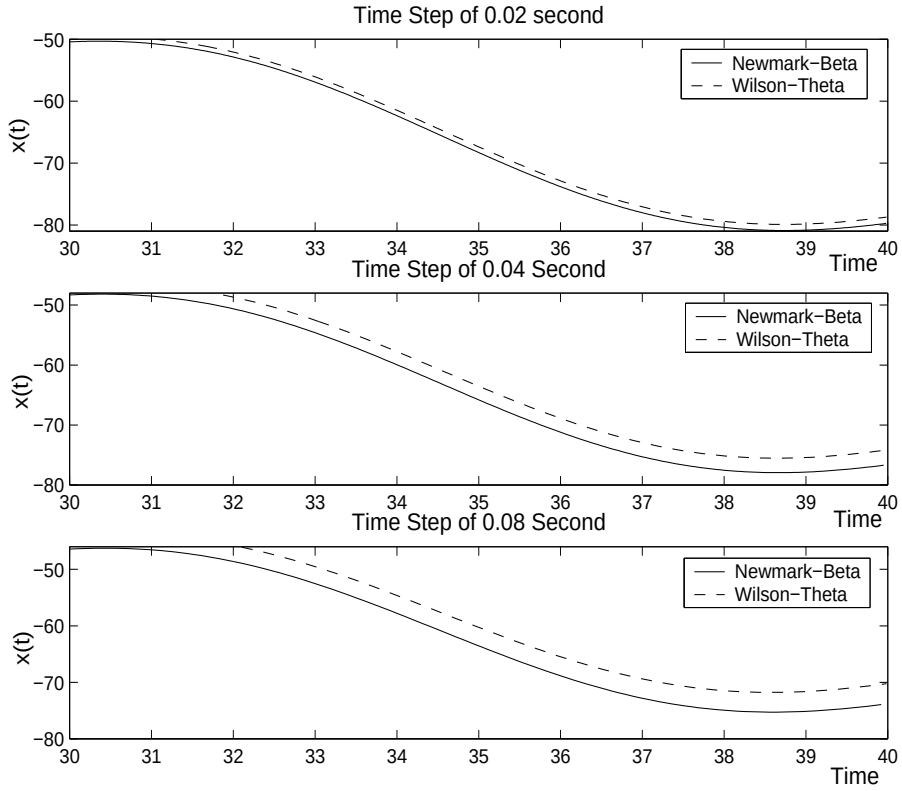


Figure 2.9: **Response of the forced system** $\ddot{x}+40\,Exp[-t]x = 2cos(0.5t)$ **obtained by Newmark -**β **and Wilson -** θ **methods showing their sensitivity to the time step**

2.4.1 Forced Vibration of the Mathieu-Hill Equation

Shahruz[52] gives an example of the Mathieu-Hill equation such as Eq. (2.5), but with a forcing function on the right hand side,

$\ddot{x} + 1.8\dot{x} + (1 - 0.8\,cos(0.25t))x = cos(0.5t)$

The authors [52] used a "time freezing" technique at particular instants to evaluate the solution and

Table 2.2: Results for Forced Vibration using a 0.08 Second Time Step

Displacement of mass after 40 seconds		
Method	x(t)-Forced Vibration	Percentage Error from Exact
Newmark-β	-73.935	9.65%
Wilson-θ	-70.236	14.16%
Present Method	-81.886	0.071%

this did not provide satisfactory results. Using the technique described here, of successive approximation of the state transition matrix and using an updated initial conditions (Eq. 2.8) we obtain a near exact result as shown in Fig. 2.10:

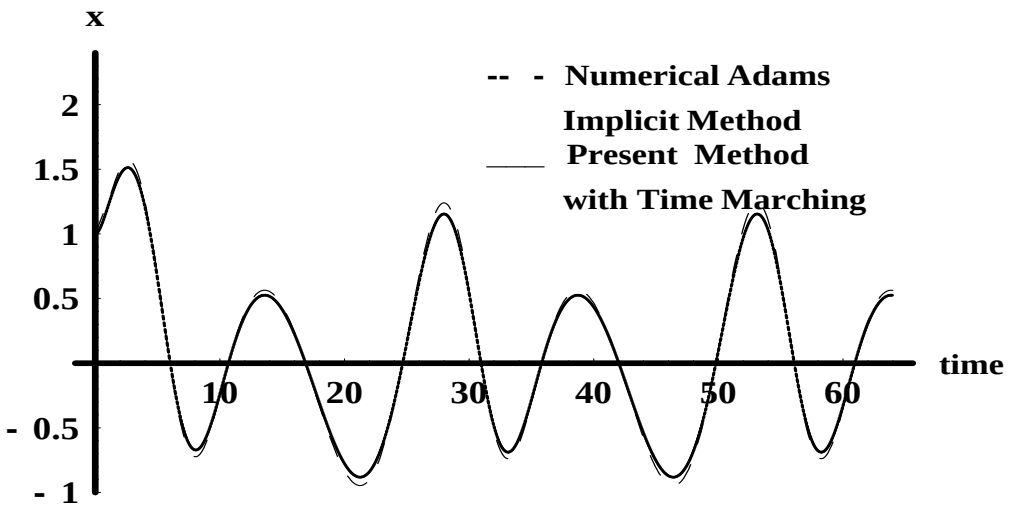


Figure 2.10: **Response of the Mathieu-Hill Equation** $\ddot{x}+1.8\dot{x}+(1-0.8cos(0.25t))x = cos(0.5t)$ **using Matrix Exponential Time Marching with a time step of 0.08 seconds**

2.5 Stability Analysis

In the previous sections it was shown that the scheme of matrix exponential time marching is very accurate and can be applied to any arbitrarily varying SDOF system. The question now arises as to whether the stability of an arbitrarily varying system can be predicted by this method. This would be an extremely advantageous result as there has been no single scheme that can be readily used for any type of time varying coefficients. The present method of Matrix Exponential time marching presents a simple technique of knowing the stability of any linear time varying system.

Consider Eq. (2.8) for the solution to an arbitrarily varying system. This is a time marching scheme where the solution for the time interval $i+1$ depends on the state transition matrix for that time interval and the initial conditions for that time interval. However the initial conditions for this time interval depends on the state transition matrix for the previous interval. This explanation can be extended backwards to the start of the time marching. Hence the stability of the scheme depends on the state transition matrix. This is calculated using Eq. (2.5). The coefficient multiplying the matrix in Eq. (2.5) is the term $\mathbf{e}^{\frac{\mathbf{c_{11}+c_{22}}}{\mathbf{2}}}\ \frac{\mathbf{sinh(\Delta)}}{\mathbf{\Delta}}$. This is the multiplying factor through out the time marching scheme. Hence for stability of the scheme

$$|\mathbf{e}^{\frac{\mathbf{c_{11}+c_{22}}}{\mathbf{2}}}\ \frac{\mathbf{sinh(\Delta)}}{\mathbf{\Delta}}| < \mathbf{1}\ \forall \mathbf{t} > \mathbf{0} \tag{2.16}$$

Using Eq (2.9), the time interval for the validity of the matrix exponential marching scheme can be determined. Once this has been done, the use of Eq. (2.16) reveals the stability of the system. For example, consider Eq. 2.6. This is a stable system. The plot of $\mathbf{e}^{\frac{\mathbf{c_{11}+c_{22}}}{\mathbf{2}}}\ \frac{\mathbf{sinh(\Delta)}}{\mathbf{\Delta}}$ versus time

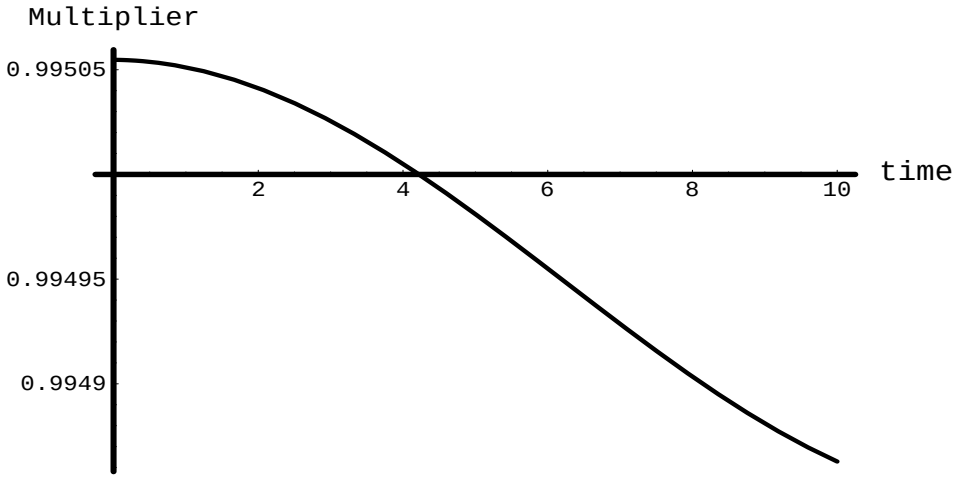


Figure 2.11: **Variation of the multiplier $e^{\frac{c_{11}+c_{22}}{2}}\ \frac{sinh(\Delta)}{\Delta}$ used in the computation of the Matrix Exponential for the Mathieu Equation governed by Eq. 2.6**

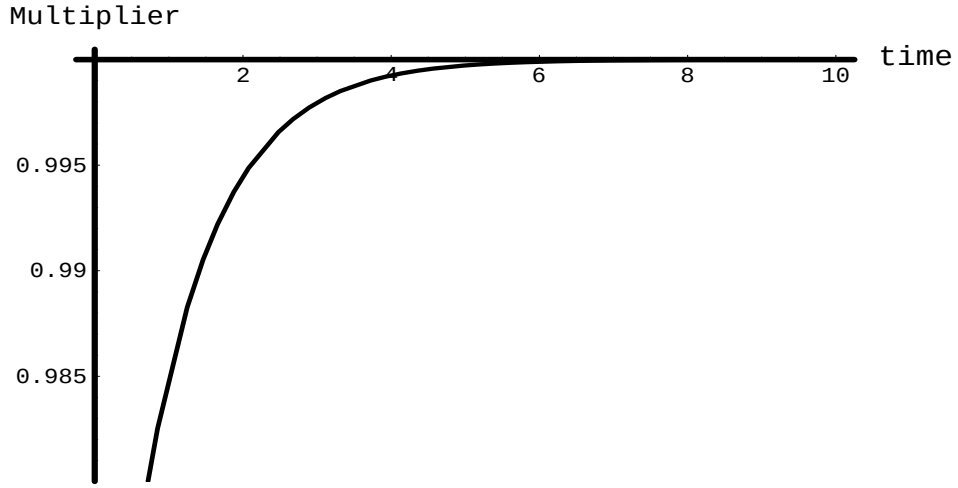


Figure 2.12: **Variation of the multiplier $e^{\frac{c_{11}+c_{22}}{2}}\ \frac{sinh(\Delta)}{\Delta}$ used in the computation of the Matrix Exponential for the Equation governed by Eq. 2.10**

as shown in Fig. 2.11 shows a monotonically decreasing function with a value below 1. Hence the method as described by Eq. (2.16) immediately verifies that this is a stable system. On the other hand, consider the system governed by equation (2.10). The plot of $\frac{sinh(\Delta)}{\Delta}$ versus time as shown in Fig. 2.12 shows a monotonically increasing function saturating at a value of one. Hence by using Eq. (2.16) we conclude that this is an unstable system. This was verified to be true from the response shown in Fig. 2.8. Thus the stability of any linear time variant system can be predicted without having to solve the respective differential equation.

2.6 Critically Damped Time-Variant System

Damping was considered in studying the response of the Mathieu equation. However, the damping was not time varying. The understanding of system damping is still a research area. Many authors like Amabili [53] consider proportional damping to a time-variant stiffness. Here, we define a critically damped system possessing time variance in stiffness and damping. A critically damped system is required whenever the time for the system to return to equilibrium position should be a minimum. An example of such an application is in shock absorbers for mounting motors or other equipment where vibration of the assembly is not acceptable. Critical damping is also used in control systems to stop oscillations of the system about the equilibrium value. For a time-invariant single degree of freedom (SDOF) system, it is a standard practice to define the coefficient of damping as $\zeta = \frac{c}{2\sqrt{km}}$ and classify the damping depending on whether ζ is greater, less than or equal to unity.

When we consider the parameters to be varying with time, such a classification can still be used if the system parameters are slowly time varying. For a time-invariant system, a critically damped case would mean identical roots to the characteristic equation of the differential equation of motion. This means that the eigenvalues are not distinct. The assumption was made in the previous sections involving calculation of the exponential of a matrix, that the equation shall possess distinct eigenvalues. This can also be similarly evaluated for non-distinct eigenvalues, but for the critically damped system, we shall make use of a different approach. Consider the system governed by Eq. 2.1 with no external forcing. The assumption is now made that the rate of change of the natural frequency is always negligible in comparison with the natural frequency itself. If this is the case, then the system with a damping coefficient that does not vary significantly away from unity can be modeled as

$$\mathbf{\frac{d}{dt}\left(\frac{1}{\omega_n(t)}\frac{d}{dt}\left(x\, e^{\int_0^t \omega_n(t)dt}\right)\right) = 0} \tag{2.17}$$

This equation when expanded will result in:

$$\mathbf{\ddot{x} + (2\,\omega_n(t) - \frac{\dot{\omega}_n(t)}{\omega_n(t)})\dot{x} + \omega_n^2(t)x = 0} \tag{2.18}$$

Clearly if $\omega_n(t)$ is slowly time varying, then $\frac{\dot{\omega}_n(t)}{\omega_n(t)}$ is small in comparison with $(2\,\omega_n(t))$. If this assumption is made, then this time-variant system given by Eq (2.14) will always be critically damped.

The natural frequency is assumed to vary in an exponential fashion as

$$\mathbf{\omega_n(t) = 10 - \frac{1}{(0.5\,exp[-t] + 2)}} \tag{2.19}$$

This variation in natural frequency is similar to the bending stiffness variation in viscoelastic ma-

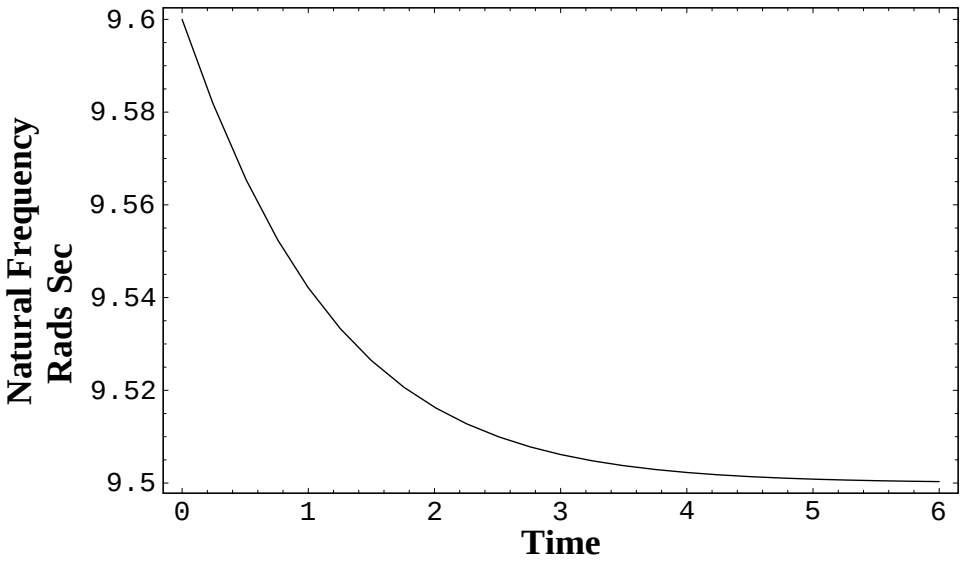


Figure 2.13: **Variation of the natural frequency of the critically damped system** $\ddot{x} + (2\,\omega_n(t) - \frac{\dot{\omega}_n(t)}{\omega_n(t)})\dot{x} + \omega_n^2(t)x = 0$

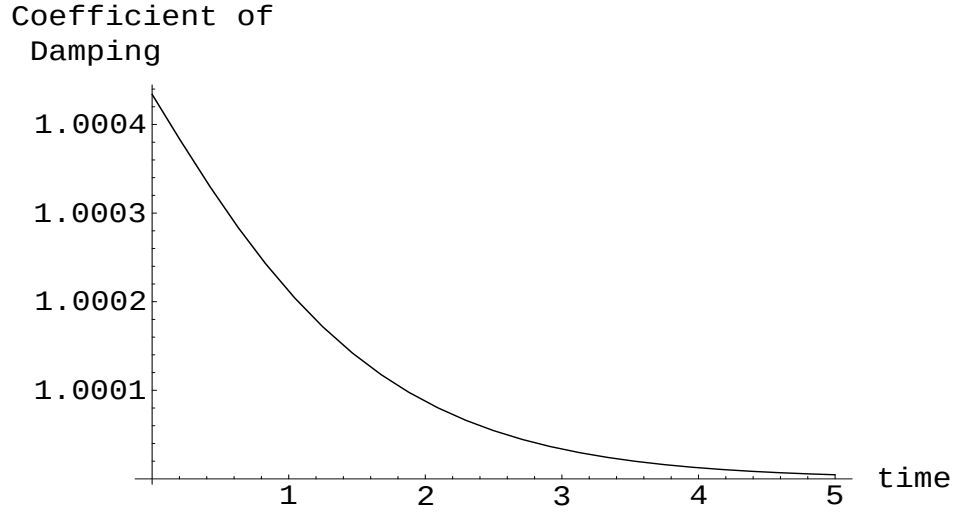


Figure 2.14: **Variation of non-dimensional coefficient of damping of the critically damped system** $\ddot{x} + (2\,\omega_n(t) - \frac{\dot{\omega}_n(t)}{\omega_n(t)})\dot{x} + \omega_n^2(t)x = 0$

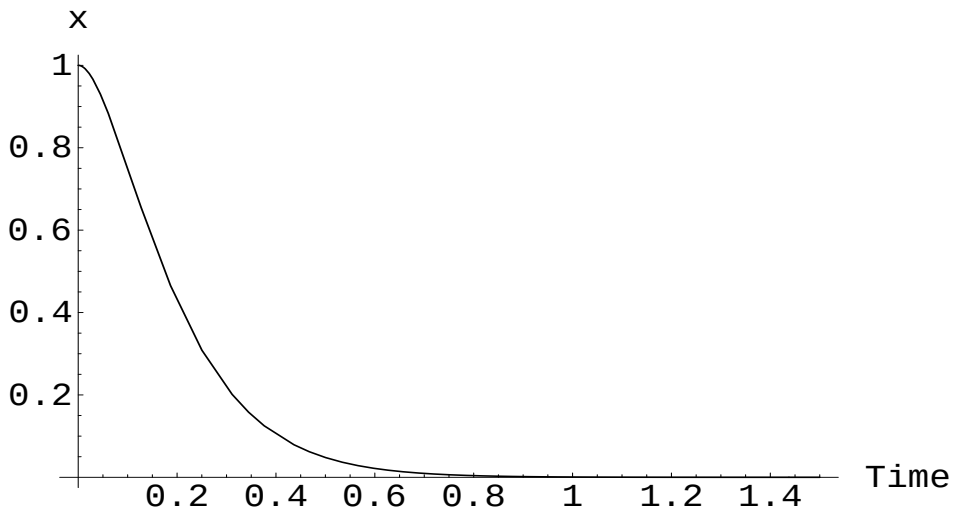


Figure 2.15: **Response of the critically damped system** $\ddot{x} + (2\,\omega_n(t) - \frac{\dot{\omega}_n(t)}{\omega_n(t)})\dot{x} + \omega_n^2(t)x = 0$

terials as assumed by Ackleh[47]. The variation of this function is shown in Fig. 2.13.

From Eq. (2.15), we can see that the coefficient of damping would be given as

$$\zeta(\mathbf{t}) = \mathbf{1} - \frac{\dot{\omega}_\mathbf{n}(\mathbf{t})}{\mathbf{2}\omega_\mathbf{n}^\mathbf{2}(\mathbf{t})} \tag{2.20}$$

The variation of the coefficient of damping is shown in Fig. 2.14 and it shows that the damping coefficient is almost uniformly equal to unity. The response of the system from Eq. (2.15) is $x(t) = e^{-\alpha}\,(c_1\,\alpha + c_2)$ where $\alpha = 10t - 0.5\,ln(0.25 + e^t)$

The plot of the response is shown in Fig. 2.15. This response is the exact response for the critically damped system.

Thus the concept of critical damping has been described for a SDOF system with slowly time varying natural frequency and an analytical procedure for determining its response for all times has been described. The method of using the exponential of a matrix can also be used here. However, when the system is subjected to the restriction that the rate of change of the natural frequency is

always much less than the natural frequency itself and is critically damped, the above procedure of writing the equation of motion in the form of Eq. (2.15) is more lucid in its delineation of the system behavior.

2.7 Comparison with Other Similar Techniques

Reference 35 recommends averaging the time varying parameters over each subdivided time period. Thus for each time interval, the governing differential equation becomes a constant coefficient differential equation, the constant coefficients being the average of the time varying parameters over that interval. To compare the present methodology with the method proposed in Ref. 35, the Mathieu differential equation in Eq. (2.6) is solved with the constants taken as $a = 0.9975$, $\lambda = 0.25$ and $q = -1.9$. It should be noted that the choice of such constants in Eq. (2.6) renders the system to be unstable. The differential equation , Eq. (2.6) is solved for each period with the time varying parameter replaced by its average for that period. Using this approach, Eq. (2.6) was solved using the same times step as before, that is 0.1 second. One can see from Fig. 2.16 that the results are not accurate. However, the present method of matrix exponential time marching with a time step of 0.1 second gives very accurate results as seen in Fig. 2.17. Thus it can be concluded that the time varying parameters should not be averaged over the time period, if such large time steps are desired. Instead, use Eq. (2.8) with the state transition matrix for each time period being given by Eq. (2.5).

Figure 2.18 compares the result obtained using the method provided in Ref. 35 wherein the time

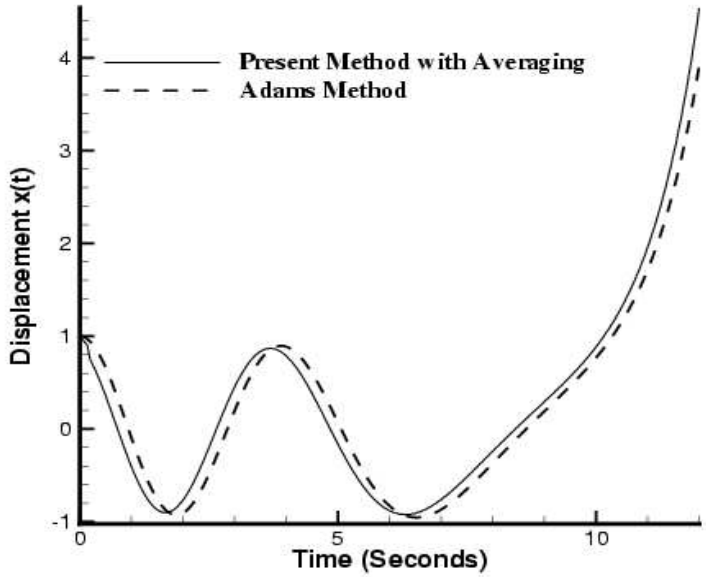


Figure 2.16: **Comparison of the response of the Mathieu Equation** $\ddot{x} + 0.1\dot{x} + (0.9975 + 1.9cos(0.25t))x = 0$ **by the Matrix Exponent Time marching method with Averaging the time varying parameters over each time period with the Implicit Adams Method**

varying parameters are averaged over each time period, with the exact solution for the system with an exponentially varying stiffness that is governed by Eq. (2.8). It can be seen that the solution obtained thus is erroneous. The present method with a time step of 0.08 seconds predicts very good matching with the theoretical result as seen in Fig. 2.19. From these illustrations it can be concluded that the present method for solving arbitrary time varying systems is very accurate and efficient for single degree of freedom systems. The method is also extremely stable and hence can be used to predict instabilities.

Since the method is based on integration over a time domain, it is similar to the concept of time

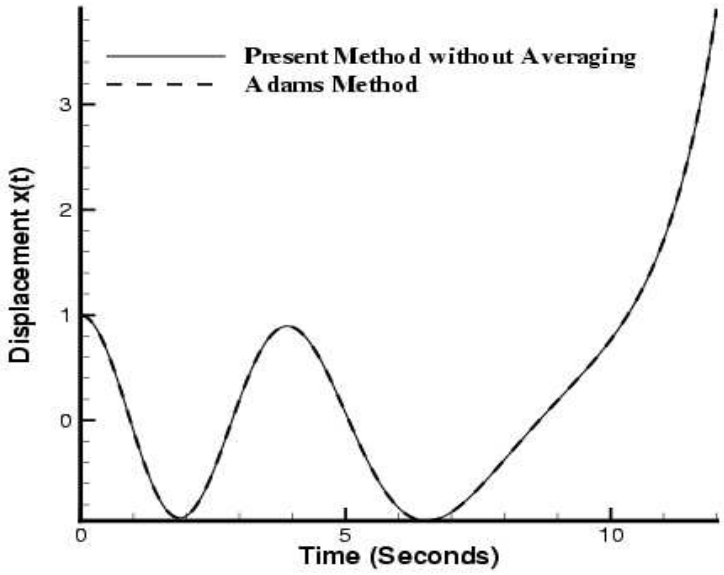


Figure 2.17: **Comparison of the response of the Mathieu Equation** $\ddot{x} + 0.1\dot{x} + (0.9975 + 1.9cos(0.25t)x = 0$ **by the Matrix Exponent Time Marching method without averaging the time varying parameters with the Implicit Adams Method**

finite elements wherein the governing equation is integrated over a period of time

$$\int_{\mathbf{0}}^{\mathbf{t}} (\mathbf{m\ddot{x}} + \mathbf{c\dot{x}} + \mathbf{kx} - \mathbf{f(t)})\xi(\mathbf{t}) = \mathbf{0} \tag{2.21}$$

Here, m, c, k represent the mass, damping and stiffness respectively and $f(t), \xi(t)$ represent the forcing function and a weighting function respectively. In time finite elements, the solution is taken as

$$\{\mathbf{x(t)} = \sum_{\mathbf{i=0}}^{\mathbf{n}} \mathbf{C_i}\chi_{\mathbf{i}}(\mathbf{t}) \tag{2.22}$$

where C_i are nodal displacements and $\chi_i(t)$ is the trial function which could be a Legendre polynomials or Chebyschev polynomials. Equation 2.18 is now substituted in Eq. (2.17). The weighting function, $\xi(t)$ is usually taken as the trial function itself. An advantage of this, is the fact that many

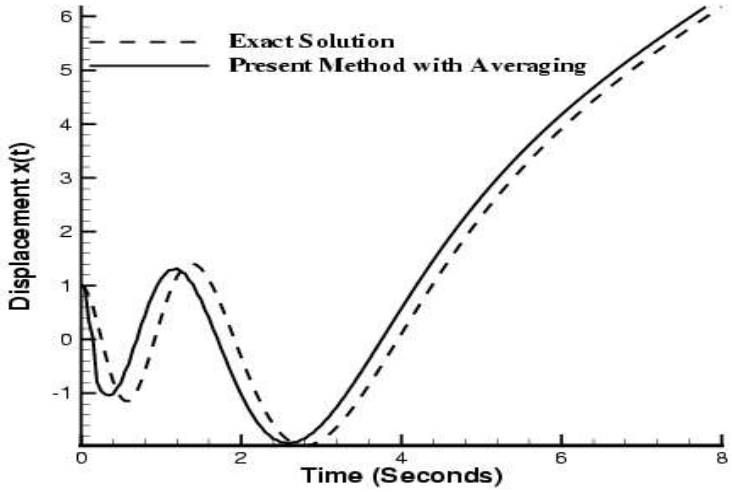


Figure 2.18: **Comparison of the response of the rapidly varying system,** $\ddot{x} + 40\ exp[-t]x = 0$ **using Matrix Exponent time Marching with averaging the time varying parameters over each time step with the Implicit Adams Method**

trial functions are orthogonal with respect to themselves or orthogonal with respect to the product of the trial function with another variable. This greatly simplifies the integral in Eq. (2.20). However in this type of representation given by Eq. (2.20) and (2.21), the initial conditions have to be enforced in the form of constraints.

In the method presented in this chapter, the time domain was divided into sub-intervals with an analytical solution for each sub interval. The matrix exponential of the coefficient matrix was used as the trial function. In this sense, this method is similar to the method of Time-finite elements. However, the initial conditions are directly formulated into the problem and are not imposed in the form of constraints. As can be seen from the various comparisons, Figs 2.1-2.19, this method of matrix exponential time marching is highly accurate and stable.

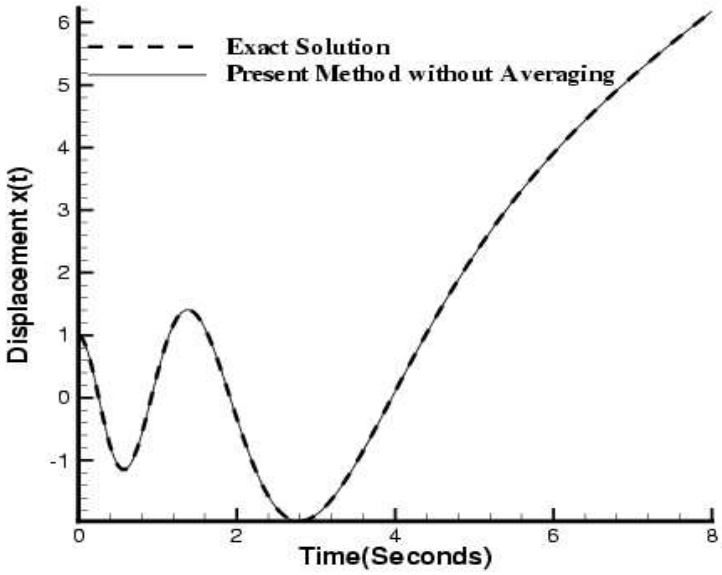


Figure 2.19: **Comparison of the response of the rapidly varying system,** $\ddot{x} + 40\ exp[-t]x = 0$ **using Matrix Exponent time marching without averaging the time varying parameters with the Implicit Adams Method**

Chapter 3

Artificial Neural Networks

As mentioned in the Introduction, artificial neural networks attempt to imitate the function of the human nervous system. In this dissertation, the application of neural networks to track the response of fluid-structure interactions is considered. Figure 3.1 describes the general architecture of a neural network. The main constituents of a neural network are the weights which multiply each input and the functional operation performed to each such weighted input. For example if the functional operation, $f(x)$ which happens to be the transfer function of the first layer, is a linear operator, then the output of the first layer is a weighted sum of the inputs, i.e., the output of the first layer is

$$y = \sum_{i=0}^{n} W_i x_i$$

On the other hand, if the operator $f(x)$ happened to be a non-linear function, such as a Sigmoidal function, then the output of the first layer could be

$$y = \frac{1}{1 + e^{\sum_{i=0}^{n} W_i x_i}}$$

Thus depending on the layer transfer function, a rich blend of system representations can be made. Further, as seen in Fig. (3.1), the output from each layer is used as the input to another layer for more functional operations. Thus the underlying mathematics of the neural networks can become quite complex. Neural networks are classified based on whether they represent deterministic

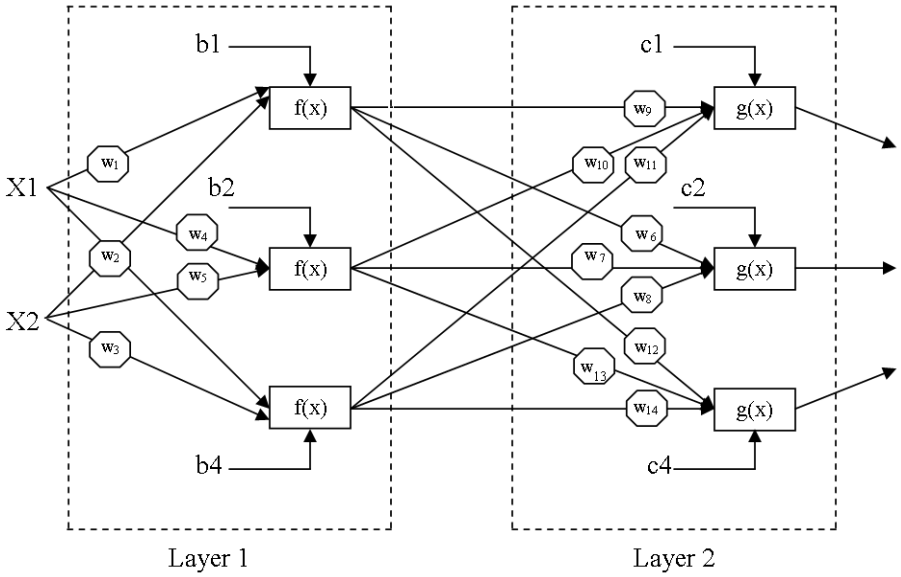


Figure 3.1: **Schematic diagram depicting two layers of a neural network, with input weights w_i, layer transfer functions, $f(x)$ and $g(x)$, Neuron Biases, b and c**

systems or stochastic systems. Stochastic neural networks have a probabilistic state associated with each neuron and they are usually trained based on thermodynamic principles. In this work, only deterministic neural networks will be considered. These can be classified according to the organization of neurons as

1. Feed Forward Networks: In this type of neural network architecture, the flow of data is

always uni-directional, that is, the input of one layer of the neural networks is obtained only from the previous layer. However, during the training process, the neural networks require back propagation techniques, that is, the weights of a layer are adjusted depending on the output of the next layer. If an input vector, $x = [x_0, x_1, ...x_n]$ and an output vector $y = [y_0, y_1, ...y_n]$ are defined, then feed-forward networks provide a mapping, $\nu(W, x)$, parameterized by the weights W to map the defined inputs given to the first layer with the defined outputs that emerge from the last layer. Feed forward neural networks are generally used for input-output mapping where the mapping is not time dependent. However dynamical systems can also be modeled using feed forward networks if appropriate training algorithms are used.

2. Recurrent Neural networks: Recurrent neural networks have the output of the network fed back to previous layers to form a new input of the network. Hence recurrent neural networks are able to represent time delays and transient response. These neural networks can represent dynamical systems. Some of the popular recurrent neural networks are the Hopfield, Elman and Jordan networks. The names are based on the authors of the networks. Hopfield[54] networks are capable of converging to a stable attractor of the dynamical system given an initial conditions set. Hence they can be used to locate stable fixed points of the dynamical system. For obtaining the transient response of the system, Elman or Jordan networks are used. These networks have the output of the previous time step as an input to the hidden layers[55]. Jordan and Elman neural networks differ only by the layer from which the feedback occurs. In Jordan type networks the output of one time state is fed-back to the input layer

for the next time state. In Elman neural network, the output is fed-back from the first hidden layer to the input layer. The number of layers is not important provided at least one hidden layer exists. The design of the layers needs to be carefully selected so that the number of layers used is minimized.

3. Self Organizing Maps: The basic Self-Organizing Map (SOM) is a sheet-like neural-network array of neurons which arrange themselves according to some fixed pattern. A random output vector is chosen from the given data set. The weights of the neuron are chosen such that the weights that best map the input to the output get trained best. Now another output vector is chosen for which the process repeated. Hence this results in a pattern of neurons, each pattern capable of reproducing one type of output. As is evident, this type of neural networks are usually used in image processing or speech recognition. The design of these type of networks was first implemented by Kohonen[56] in 1982.

3.1 Aeroelastic Applications

Aeroelastic calculations involve fluid-structure response coupling. This means that every displacement of the structure under fluid dynamic loads predicted by a structural solver will in-turn alter the fluid dynamic response. This coupling between the aerodynamics and structural behavior can lead to very large computational costs for aeroelastic analysis that requires use of computational fluid dynamics/computational structural mechanics. Hence efficient alternative methods that can provide rapid aeroelastic response solutions need to be investigated. In the present dissertation,

both static and dynamic aeroelastic problems will be investigated through the use of neural networks. Neural networks must be able to represent the aeroelastic response of a structure given the system parameters. In a static problem, the boundary conditions will be specified and in a dynamic problem, the initial conditions are specified. The type of neural networks required is largely dependent on the inputs/outputs of the network.

The static aeroelastic problem that will be handled here, different from the classical divergence problem, is the deformation of an adaptive bump on the surface of an airfoil. This bump is deformed based on elasticity principles under the action of actuator forces. However, since the bump is situated on an airfoil in a free stream, there are also non-conservative aerodynamic forces which change significantly as the bump deforms. Such a problem requires a large number of data sets that can capture the bump shape and the fluid dynamic pressure over its surface. However, the amount of data sets for the network training could be minimized by fitting suitable functions to the data and using the coefficients of such fits as the inputs/outputs of the neural network. The deformation can also be represented incrementally with respect to the increase/decrease of actuator loading. From the above classification of neural networks, the best representation would be a feed-forward neural network.

For dynamic aeroelastic applications such as flutter control or gust alleviation, a time stepping aerodynamic scheme and a time stepping structural dynamic scheme are required. Using conventional finite difference schemes for the numerical integrations of the equations of aeroelasticity (Eq. 1.2), requires very small time steps of the order of 10^{-3}. Similarly, training of neural networks based on these numerical results for dynamic aeroelastic problems requires that the time

step is small. Otherwise, noise generation could be a problem. For such time stepping schemes, the use of recurrent neural networks is advantageous. Here, the neural networks are trained for a small sample of the output, which could be the pitching angle of an airfoil, the coefficient of lift etc. This sample output which is confined to a certain time interval will be one of the inputs to the next sample of the succeeding time interval. Figure 3.2 shows the result of using a recurrent neural network for training a time variant dynamical system. As can be seen, the initial response

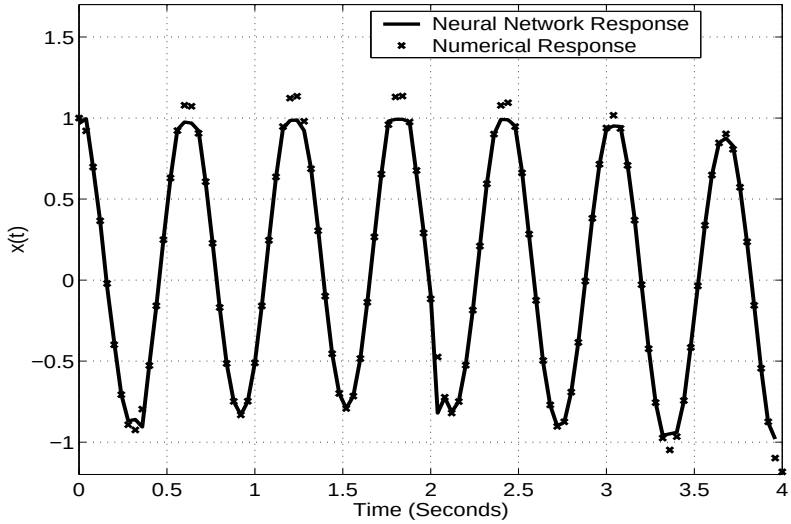


Figure 3.2: **Response of the system** $\ddot{x} + (100 + 10sin(t))x = 20Sin(t)$ **as computed using an Elman recurrent neural network with 0.04 second interval of time used in the time marching**

is erroneous in the amplitude and also possesses some noise. However, as time progresses, there is an improvement in the quality of the neural network simulation when compared with the theoretical result. Feed-forward neural networks can also be used for dynamic aeroelastic problems, provided a suitable training algorithm is developed. In Chapter 5, a suitable representation of dynamic aeroelastic response through feed-forward neural networks is presented.

3.2 Neural Network Training

The training of the neural network is the most important process in the representation of any system using neural networks. Most neural network training routines are gradient based, that is they require the calculation of the derivatives of the output with respect to some parameters. Accordingly, the methods of training that are used can be classified as steepest descent methods, conjugate gradient methods or Gauss-Newton methods. The steepest descent and conjugate gradient methods are first order methods that require the computation of the Jacobian of the estimated error in the output with some parameters. These methods can be written in the following form

$$\mathbf{x_{k+1}} = \mathbf{x_k} - \delta_\mathbf{k}\, \mathbf{J_k} \tag{3.1}$$

Here, the output at iteration $k+1$ is dependent on a learning rate δ_k and the Jacobian matrix J_k constitutes the first derivatives of the error in the neural network output with respect to the neuron weights. The learning rate is a constant for the steepest method. Conjugate gradient methods choose the direction of the gradient based on certain conditions. Two popular Conjugate gradient schemes are the Fletcher-Reeves and the Polak-Ribiere method. More information on these schemes can be found in Ref. 57.

A second order gradient based scheme is the Gauss-Newton method, a popular version of which is the Levenberg-Marquardt method. Here the first and second derivatives of the error in the neural network output needs to be estimated. Thus the Jacobian and Hessian matrices needs to be evaluated. The computation of the matrix of the second derivatives of the error with respect to the neuron weights constitutes the Hessian matrix. It is clear that the Gauss-Newton second-order methods

are computationally expensive and need a significant amount of system memory. However they are very reliable numerical schemes and have a faster rate of convergence than the conjugate gradient/steepest descent methods. The Levenberg-Marquardt method can be written mathematically as

$$\mathbf{x_{k+1}} = \mathbf{x_k} - (\mathbf{H} + \mathbf{I}\,\delta)^{-\mathbf{1}}\mathbf{J^T}\,\mathbf{e} \tag{3.2}$$

where H is the Hessian matrix, δ is a constant diagonal multiplier and e is the error in the output. The Levenberg-Marquardt algorithm provides for a diagonally dominant matrix by means of the added constant δ. Many of the neural networks trained in the aeroelastic analysis that are described in Chapters 4 and 5 are based on the Levenberg-Marquardt method.

The MATLAB software package has a versatile neural network toolbox and in this dissertation, the training of neural networks is done using this toolbox. Figure 3.3 depicts the training process using the MATLAB routine, "trainrp" or training using the method of resilient back-propagation. This is a steepest descent method with scaled partial derivatives. One can see that, using this method, the error in the network output for the given data set is far from the desired tolerance of 10^{-8}. Hence the data set, the network architecture or the training routine must be changed. If a different training routine performs to satisfaction, then the data set and the network need not be altered. The same network is now trained using the Levenberg-Marquardt method.

Figure 3.4 shows the convergence obtained between the Neural network output and the desired target using the Levenberg-Marquardt method. One can see that within a short duration of training, the desired accuracy was obtained. Hence for the designed network with the given data sets, the Levenberg-Marquardt method should be used for training. Figure 3.5 describes the goodness of

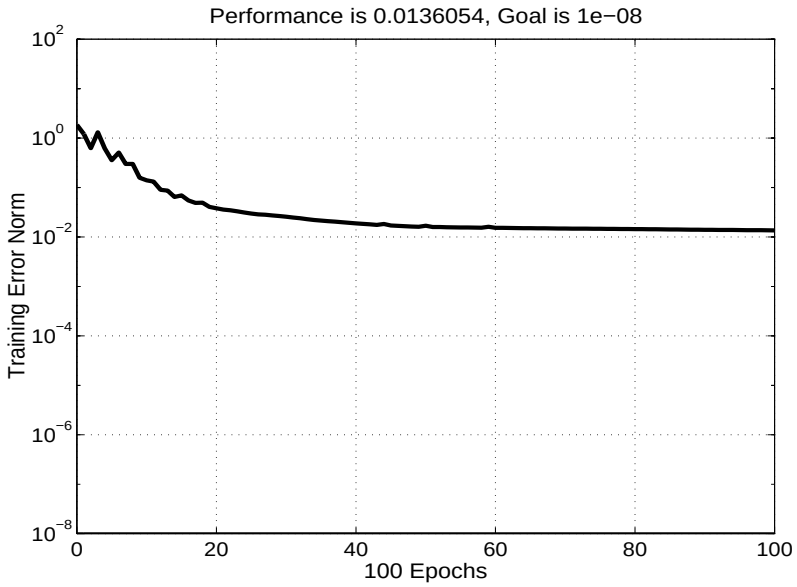


Figure 3.3: **Convergence of the Neural Network to the desired target using the Resilient Back-Propagation training routine**

fit between the neural network simulation of the target data and the actual target. For a perfect fit, the plot of the simulated output versus the original output should be a straight line with a unit slope. Figure 3.5 shows that this is nearly achieved in this training process. However, as mentioned earlier, care must be taken that the network behaves appropriately when interpolating between the original design points that were used in training.

The time duration of one training session depends on the size of the data set and the number of neurons in the neural network. Figure (3.3) and (3.4) show that the neural network was trained for 100 ”epochs”. Each ”epoch” is the time taken for sampling a different input pattern, i.e. each time during the training that the weights of the neurons in one layer are adjusted so as to change the input to the next layer in order to reach the target. The duration of each epoch depends on the input data

size and the time taken for each iteration of the gradient based training scheme. For the networks used in this dissertation, the average epoch training time was around 15 seconds. Therefore 100 epochs of the training take approximately 25 minutes. Some of the networks like the ones used in Chapter 5 for dynamic aeroelasticity may take slightly less time. Some networks used in Chapter 4 for static aeroelastic calculations may take slightly longer, but the average time is around 25 minutes. Sometimes, this training may not be satisfactory like the routine depicted in Fig. (3.3) and hence alterations have to be made in the neural network or in the training routine and a new training sequence has to be initiated. In Fig. (3.4) a new training routine, the Levenberg-Marquardt method was used which was shown to converge to the desired accuracy level.

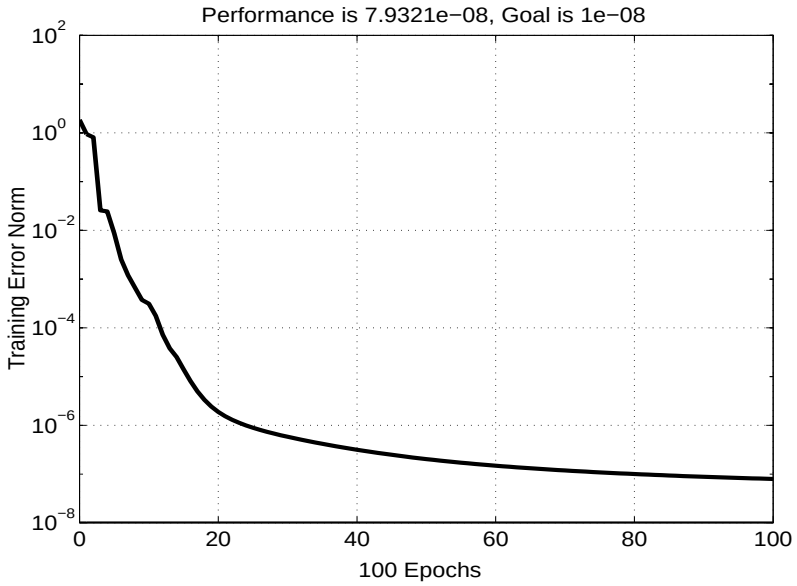


Figure 3.4: **Convergence of the Neural Network to the desired target using the Levenberg-Marquardt training routine**

For dynamical systems, the input-output relationship must capture the state of the system. Haykin[41] defines a state of a system as ”A set of quantities that summarizes all the information about the

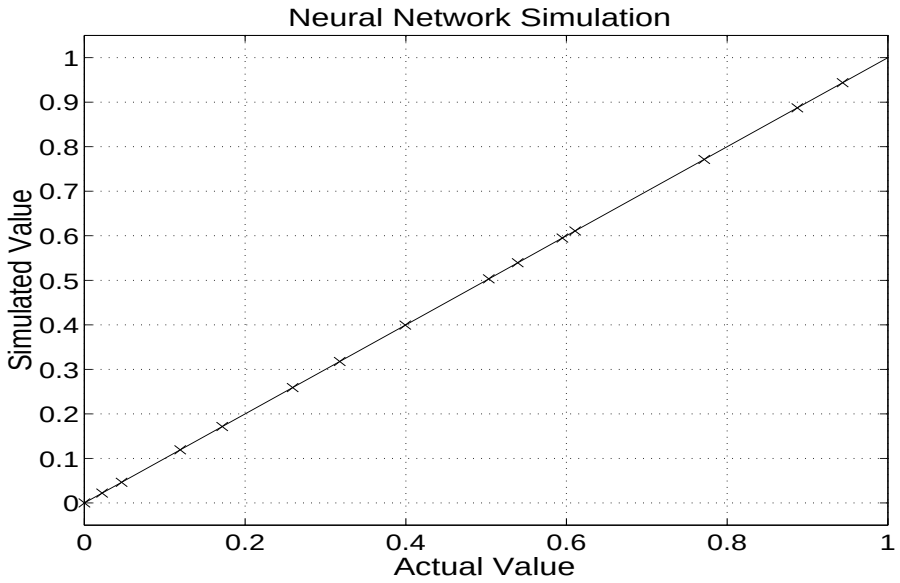


Figure 3.5: **The result of an accurate training of a neural network, showing the output of the neural network fitting the target data very closely**

past behavior of the system that is needed to uniquely describe its future behavior, except for the purely external effects arising from the applied output" Hence for dynamical systems, first a state space model must be developed before developing the neural network model. The next two sections describe the method of developing neural network models and their application to non-linear systems.

3.2.1 Algorithm For Selection and Training of Neural Networks

Here the procedure for designing a neural network to represent the desired input-output mapping is outlined.

1. Decide on the type of system output, whether it is of stochastic nature or deterministic. In

the present work, only deterministic networks are described.

2. Decide the type of neural network connectivity, i.e., is it feed-forward, recurrent or something else. This is a critical factor especially when dealing with dynamical systems or with very large data quantities.

3. Decide on the layer transfer functions to be used depending on whether the system that is being represented behaves non-linearly or linearly. Typically the common layer transfer functions are a) Linear Ramp, b) a threshold type function, c) Hyperbolic Tangent and d) Sigmoidal or Logistic functions. The details of these transfer functions can be obtained from Ref. 41.

4. Arrange the neurons in different layers for optimal performance. The input layer typically has one neuron for every input item that has to be weighted. The output layer has as many neurons as output data items. In between the input layer and the output layer, are present the hidden layers whose neural structure has to be carefully designed.

5. Choose the training routine. As was mentioned earlier, gradient based schemes can be first order or higher order based convergent schemes. There are also non-gradient based schemes for training neural networks. One such method is the use of Genetic Algorithms. Genetic Algorithm is a rule based scheme which is usually used to select global optima for a particular objective. However, they can be tuned to minimize the error in the neural network representation of the target data. One training routine may not be sufficient. The trained network may need to be adapted suitably when different input data sets are presented to it.

6. Decide on the input and output data set. Scale the data sets to a constant order of magnitude. For non-linear problems, the neural networks will have to be properly initialized since the convergence of the training routine can depend on the initial conditions.

7. Test the trained neural network with different sets of data to check for consistent performance. The neural network simulation should not deteriorate drastically in the domain defined by the input data sets.

8. If the performance is consistent, then decide on the boundaries of operation of the neural network with regard to the input data. If the performance is below tolerance, then change the data sets, the training routine or the initial conditions. If no clear training routine can be selected even after repeated trials for correct input and output data sets, then re-start the procedure from item 2.

3.3 Representation of Non-linear Systems

The output of the neural network is a weighted summation of the inputs operated upon by a non-linear function. This type of representation is a parametric representation and it is ideal for simulating non-linear behavior. The Weiner-Hammerstein model for parametric representation of non-linear systems allows for any non-linear dynamical system to be decomposed into

a) A zero memory non-linearity operated upon by a linear dynamical model.

b) A linear dynamical system operated upon by a zero memory non-linearity.

These methods are termed to be parametric because the given non-linear system is represented by

assuming a fixed model. What remains to be decided is the estimation of the design parameters of this fixed model. Hence a neural network is a parametric representation. On the other hand, non-parametric representations of non-linear systems also exist. Here the model structure has to be decided first, before deciding on the design. A popular non-parametric representation of non-linear dynamical systems is the Volterra series. In the Volterra series representation, the response of the non-linear system is evaluated as an infinite summation of higher order impulse response functions.

$$\mathbf{y(r) = h_0 + \sum_{i=0}^{n} h_1(i)\, x(r-i) + \sum_{i=0}^{n}\sum_{j=0}^{n} h_2(i,j)\, x(r-i)\, x(r-j) + ...} \tag{3.3}$$

Here h_i are the impulse response for a sudden change in the function x. Higher order impulse responses are the evaluation of the system response to the sudden inputs x given at different time instants. As can be seen such a representation can be very complicated. Nonlinear Aeroelastic representations using Volterra series has been used by Silva[13] and Librescu[58] Another type of parametric representation of non-linear systems are the NARMA models. Nonlinear Autoregressive Moving Average Model(NARMA) is a discrete time representation of a non-linear system by state values at N previous time instants.

$$\mathbf{y(k) = F(y(k-1), ..., y(k-N), u(k-1), ..., u(k-N))} \tag{3.4}$$

Here F is a non-linear function operating on the response of the system $y(k-i)$ and the system input $u(k-i)$. For a linear system, the functional operation F would be only a weighted summation. This is called an Autoregressive Moving Average Model (ARMA). From the details of recurrent neural networks, the method used is a type of ARMA or NARMA model, since the neural network

is based on functions operating on past network outputs and current inputs. Further details on NARMA can be found in Ref. 59.

As can be seen, neural networks that represent dynamical systems must be able to remember past states. Recurrent neural networks possess this capacity. Recurrent neural network representations of dynamical systems implicitly possess the impulse function response of the system to the given input. Hence theoretically once the dynamical system has been represented by the right recurrent neural network architecture, the neural network should be able to emulate the dynamics of that particular system under any loading. Recurrent neural networks can be looked as ”folded” feed-forward networks. There is one folded layer for every time instant of operation. This folded layer has its output fed back to the input of the previous layer for the next time step. The feedback causes a time delay and also gives the network the capability to possess memory about its past states. A recurrent neural network can be ”unfolded” by removing this time delay and duplicating the previous layers as feed-forward connections. This implies that the new feed-forward network obtained from the recurrent network is much more massively inter-connected and hence cannot be computationally efficient. However, many training algorithms for recurrent neural network training implement this ”unfolding” procedure to train the network using back-propagation algorithms.

Apart from recurrent networks, feed-forward networks can also be used to represent dynamical systems, if the network has the capacity to emulate the state of the system for every time instant of interest in the domain of computation. The main factor to be taken into account while training neural networks to represent dynamical systems is that a sufficiently large time step for transient response simulation must be made possible. Otherwise, very large data sets will be necessary to

account for the state variables at every time step. Large data sets in the training routine can lead to improper data fitting. Neural networks are a more advanced version of response surfaces and as such each layer of the neural network is a type of ”response surface” which is operated upon by further layers of the neural network.

In the following chapters is described the analysis of aeroelastic response through neural networks. Chapter 4 discusses the static aeroelastic response calculations for an adaptable bump on the surface of an airfoil. In Chapter 5, a feed-forward neural network is designed for dynamic aeroelastic response calculations. Hence both static and dynamic aeroelastic computations can be performed through neural networks. This allows for a computationally rapid calculation of the fluid-structure interaction. Conventional aeroelastic computations used to treat the airfoil as a flat plate and inviscid aerodynamics were used. With developments in the aerospace industry, this representation was no longer accurate. Accurate fluid dynamics and accurate geometry of the airfoil or wing surface are required. Such aeroelastic analysis could also possess non-linear aerodynamic and/or non-linear structural dynamic calculations. The coupling between the aerodynamic and structural responses for the aeroelastic analysis also provides tremendous computational challenges. Hence a lot of interest has been generated[60,61] in rapid computational schemes for aeroelastic analysis. Neural network representations of fluid-structure coupling are suitable means to implement such a rapid aeroelastic solver for both linear and non-linear aeroelasticity.

Chapter 4

Aeroelastic Analysis of a Deforming Surface using Neural Networks

The use of Computational Fluid Dynamics (CFD) for obtaining gradient information for optimization methods is a very daunting task. It can be extremely difficult to extract gradients of aerodynamic coefficients with respect to airfoil surface parameters using CFD alone. However for the computational analysis of flows wherein there is separation, there is no alternative to using computational fluid dynamics. Hence numerical techniques such as the continuous adjoint method[62] have been devised to extract the sensitivity information of the flow parameters to various independent variables. To extract gradient data regarding the sensitivity of the flow to airfoil surface contours, it is easier to use expert systems such as neural networks which are trained with aerodynamic data from prior CFD runs. This chapter focuses on the aeroelastic analysis of a deforming bump on an airfoil surface. The design of this bump results in sufficient drag for lateral directional control

of the aircraft while at the same time minimizing loss in lift due to fluid separation. The idea is borrowed from the concept of split flaps or spoilers that are used in tailless aircraft such as the B-2 bomber. The deflected spoilers use flow separation on one wing to cause an unsymmetrical drag force resulting in a significant yawing moment. Here a bump is used to separate the flow over the airfoil. Higher drag increases are obtained for higher curvature changes in the bump shape. However, the larger the curvature change, the higher is the energy required for the bump deformation. The energy available for deformation is limited due to the use of non-hydraulic actuators. Also, a bump situated at the leading edge can separate the flow for a much smaller bump size than a bump situated downstream. But the flow is very sensitive to leading edge perturbations resulting in drastic reduction in lift. Hence this is an optimization problem of designing a deforming surface that provides maximum drag for maneuverability with minimum loss in lift and minimum strain energy of deformation.

The aerodynamic analysis is performed using the CFD software Fluent. The structural analysis of the airfoil bump is done using the ANSYS software. Since the structural model that is used involves buckling of an elastica under various loads, a non-linear finite element calculation is performed using frame elements. The structural analysis is performed in an incremental fashion with the aerodynamic pressures before deformation of the bump being corrected using the trained neural network after deformation of the bump. The bump deforms under the actuator loads and aerodynamic loads. The effect of this deformation is to change the aerodynamic loads. The changed aerodynamic static pressure is calculated using the neural network that had been trained using FLUENT data. The ANSYS analysis is again performed using this changed aerodynamic pres-

sure. This procedure is repeated until an equilibrium position of the bump is achieved. A series of such ANSYS runs is performed using a variety of actuator loading. The results obtained are used in training a Neural Network for the structural model.

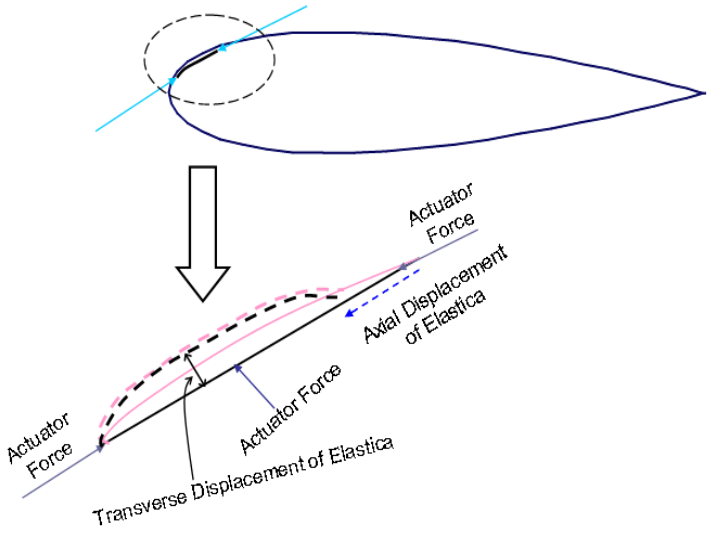


Figure 4.1: **Actuation mechanism for creating an adaptable bump at the leading edge of an airfoil**

4.1 Yaw Moment Generation

The change in drag over the wing due to a leading edge bump will lead to a yaw moment. If 2D aerodynamics is used to calculate the drag, then the yawing moment coefficient is defined by

$$\mathbf{C_n} = \frac{\mathbf{1}}{\mathbf{4b}}\,\mathbf{AR}\int_{\beta}^{\mathbf{1}} \mathbf{\Delta C_d}\,\mathbf{c}(\eta)\,\eta \mathbf{d}\eta \tag{4.1}$$

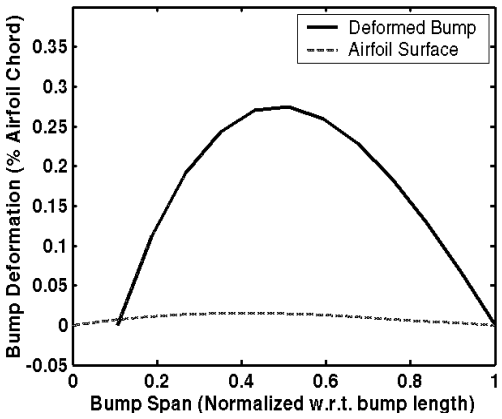
In Eq. (4.1), η is the non-dimensional span and $c(\eta)$ is the chord variation across the span. For the drag force to be effective, the bump must be located on the outboard wing; from the wing tip to a certain span distance β. Since the leading edge governs the formation of the boundary layer, the bump must be situated near the leading edge. The flow on the bottom side of the airfoil is quite insensitive to the presence of bumps. The size, shape and location of this bump is critical to the determination of the drag rise. Bauer[63] describes the yaw moments generated due to the presence of micro drag generators (MDGs) on the top surface of the wing. Raney[64] details the effect of shape change arrays on an adaptable wing in producing yaw. These accounts indicate that a yaw moment coefficient of the order of 10^{-3} is adequate. Incorporating the wing geometry in Eq. (4.1) would then reveal the change in drag force required to obtain the desired yaw moment coefficient. The usual practice for control of aircraft lateral dynamics is to use deflected spoilers. This is extremely dependent on the rate of spoiler deflection[65]. Moreover, spoilers separate flow by providing an obstacle to the boundary layer. Whereas, in the present work, the adaptive bumps separate flow by providing for an adverse pressure gradient. Therefore the size of the bumps at the leading edge are much smaller compared to the width of a spoiler. In contrast, if the bump is provided at the trailing edge as an imitation of a spoiler, the size of the bump required to generate sufficient yaw moment would be too high and hence the actuation energy needed to create such a bump would be too high to be realistic.

4.2 Structural Model

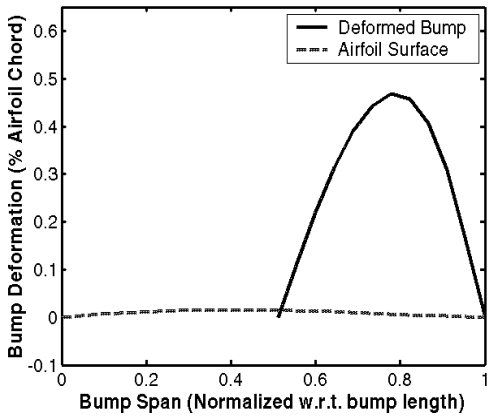
Here it is proposed to analyze the structural deformation of the bump using a beam model. The bump is deformed using applied axial and transverse actuator loads under the non-conservative aerodynamic loads. Since one of our requirements is to minimize the energy required for deformation, we look at the minimum energy curve. The minimum energy curve is the curve described by a pinned-pinned elastica. This curve is obtained by applying horizontal forces greater than the Euler critical load to the ends of the elastica.

Figure 4.1 describes a portion of the leading edge of the airfoil. Under the airfoil skin is shown a curved elastica. Using the actuator forces as a means for buckling the elastica element, a bump is created on the airfoil surface. Vertical actuator forces to change the shape of the bump are also required. The elastica shape is the minimum energy bump on the airfoil surface for a specified bump height and size. However, this bump will not give highest drag. Thus the bump shape is optimized such that sufficient drag for the production of a yawing moment is obtained, while minimizing the actuation energy required. The behavior of the elastica after bending, is non-linear with respect to the elastica slope. The aerodynamic forces on the bump also continously change with the bump shape and size. Therefore, we have a non-linear system subject to non-conservative forces. The solution to such a problem, even in 2-D is not straightforward. Since the aerodynamic forces are also non-uniform over the bump, a finite element model is used to solve the problem. These aerodynamic forces are evaluated from the CFD model. The effect of these changing aerodynamic forces on the actuators is significant in the sense that maintaining

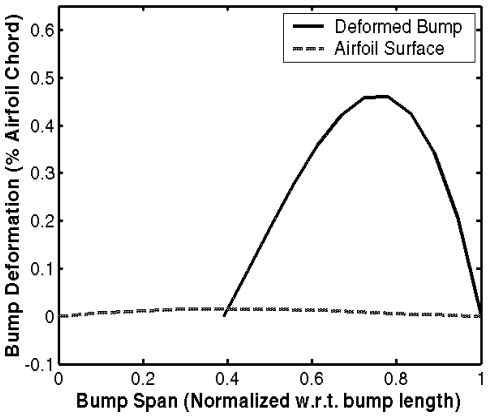
a state of constant deformation is difficult. In order to modify this elastica shape into a curve which produces more drag, additional actuator forces will need to be applied normal to the airfoil skin. Thus a distributed array of actuators is required depending on the extent of shape change necessary. This analysis of the deformation of the bump under a set of actuator forces and fluid dynamic pressure is done using ANSYS. A frame finite element is used with 2 nodes per element. The frame finite element has 3 degrees of freedom per node that is 2 translations and an in plane rotation. Non-Linear analysis is performed in ANSYS using incremental loading and specifying the number of steps for each load to be applied in full. For each load step the displacement is determined using a Newton-Raphson iteration.



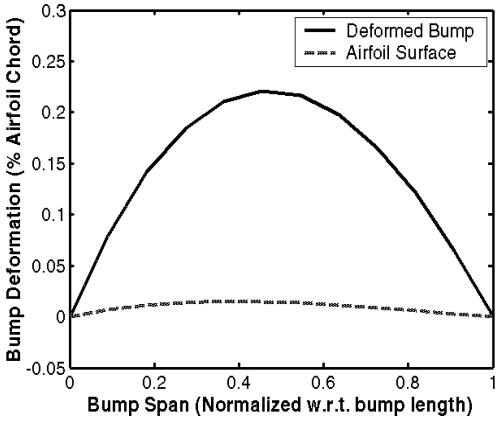
Bump A



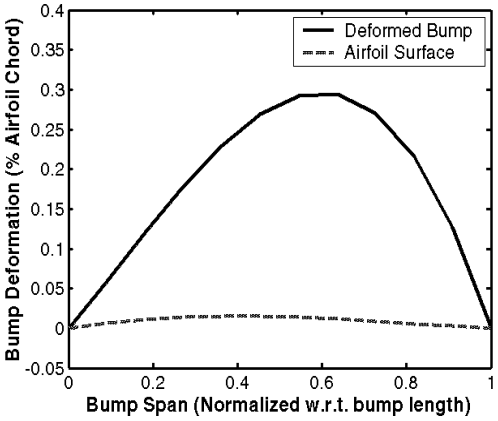
Bump B



Bump C



Bump D



Bump E

Figure 4.2: **Various bump shapes that were used for training neural networks to represent the aerodynamic load over the bump**

4.3 Aerodynamic Model

Figure 1.1 shows the computational grid, made around an 8% thick symmetric airfoil. The free stream is at an angle of attack of 5^0. The free stream Mach number is set at one of the two values of 0.3 or 0.7. This is to investigate effectiveness of the bump shape optimization at low speed flow and moderately high speed flow. Since drag forces varies as the square of the velocity, the bump deformation for a higher Mach number, may not require as large a deformation as the bump used in the lower flow velocity. However this remains to be investigated.

Using pressure far field boundary conditions, a $k-\epsilon$ turbulence model and an implicit coupled Navier-Stokes solver with upwind finite differencing, the fluid flow is modeled in the CFD software Fluent. Better turbulence models such as Reynold's averaged stress 5 equation model could not be used due to the computational time involved. The $k-\epsilon$ model uses the Boussinesq hypothesis wherein the turbulent stresses are expressed as the product of the isotropic eddy viscosity and the mean velocity strain rate. A variation of the standard $k-\epsilon$ termed as "Realizable $k-\epsilon$" model is provided by FLUENT and this is used herein. The Reynolds numbers of the current study is close to 10^7. Since this is a high Reynold's number region, we use the Realizable $k-\epsilon$ model as suggested by Shih[66]. For separated flow, an unsteady flow solver is used. The convergence limit is based on the convergence of the aerodynamic forces. The airfoil is modeled using B-splines. The mesh representation of the airfoil must reflect the true leading edge shape and the bump shape. Hence the grid is extremely fine near the airfoil surface. To capture the effect of flow separation and possible subsequent re-attachment, a grid adaptation is done at the airfoil surface. The aerodynamic

loads on the bump surface can be obtained at every incremental step in the process of increasing the bump size. These aerodynamic forces are made available to the structural model. The effect of the bump on the flow can be seen even before flow separation, by noting the boundary layer thickness. Once a vortex shedding pattern emerges at the bump, the flow solver is switched from the steady solver to an unsteady implicit solver. Though the separated flow is unsteady and turbulent, it is not chaotic. There is repeatability in the analysis with regard to the aerodynamic forces.

4.4 Neural Network Implementation

Neural Networks are a very versatile tool to predict a system response given prior data about the system behavior. Optimization procedures based on neural networks has been used for analyzing wing structures using equivalent skin analysis [67]. In the present work, an aeroelastic optimization tool is developed using trained neural networks. The neural network package available in the MATLAB software is used. The data required for training the neural network is :

1. The bump parameters : Size, shape and location.

2. The aerodynamic forces : Lift, drag, static pressure over the bump.

3. The actuation energy required, the actuator loads.

The parameterization of the bump is done using cubic splines. Though other representations of the bump such as Hicks-Henne bump functions were tried, it was found that a single cubic spline fit is satisfactory. Then the equation of the splines is given as

$y(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$

where, the bump is defined by $y(x)$ and the coefficients $a_0 - a_3$ need to be determined for different bump shapes. The coefficients of the cubic spline which represent the bump are used as inputs to the neural network. The x span values of the bump are normalized so that they range between $0-1$. The Neural network is trained using the Levenberg-Marquardt algorithm. This is a variation of the Gauss-Newton's method involving a second order Taylor's series expansion of the minimization function about an initial value. The advantage of the Levenberg-Marquardt scheme is that the Jacobian matrices that are involved in the Taylor's series expansion of the minimization function are forced to be diagonally dominant for each iteration in the minimization process. This ensures faster convergence than the Gauss-Newton scheme for non-linear data fitting. The disadvantage of the Levenberg-Marquardt scheme is that there is a limit to the amount of data that it can handle due to memory constraints.

Two neural networks are used for the aeroelastic analysis.

1. The neural network NN-CFD is trained based on the data from FLUENT. The output of this neural network is the aerodynamic pressure acting on the bump. Its input is the bump shape parameters in the form of the coefficients of the cubic spline that describes the bump. The aerodynamic pressure over the bump as output by NN-CFD is used by the ANSYS package to calculate the deformation of the bump. The result of the ANSYS analysis, in the form of new bump shape parameters is fed back to this neural network to calculate the changed aerodynamic pressure. Sufficient amount of ANSYS runs are performed on different bump shapes in order to estimate the required actuator loads and the strain energy.

2. Using the structural data, namely the strain energy, actuator loads, a second neural network, NN-STR is trained. This neural network has as its output the strain energy of forming the bump, the actuator loads, the lift and drag coefficients. The inputs for NN-STR are the bump shape parameters in the form of the coefficients of the cubic spline that describes the bump.

The present two neural networks, NN-STR and NN-CFD use three inter connected layers. The output function of the layers and the number of neurons per layer are given below

1. Layer 1: Hyperbolic Tangent Sigmoid, 20 neurons
2. Layer 2: Log Sigmoid, 12 neurons
3. Layer 3: PureLin linear ramp function. Number of neurons depends on the number of outputs of this neural network which was 4 in case of NN-CFD, representing the static pressure over the bump and 7 in the case of NN-STR, signifying the actuator loads, their location and the lift and drag coefficients over the airfoil.

The number of neurons in each layer are determined by trial and error during the training process. The training routine in MATLAB which is used, assigns appropriate neuron weights and biases. The neural network is trained on input bump shapes such as those shown in Fig. 4.2. These two neural networks replace the analysis performed using FLUENT and ANSYS for aerodynamics and structures respectively.

Software Validations

As described in the earlier sections, the CFD software Fluent is used to analyze the effect of the bump on the flow field over the airfoil. The bump deformation due to actuator loads and aerodynamic loads is analyzed using ANSYS. Before starting the actual analysis, the ANSYS and Fluent softwares were tested for procedures providing desired results. The documentation accompanying the Fluent 5 software describes the appropriate boundary conditions and grid generation for analysis of flow of an airfoil. Accordingly, 'Pressure far-field' boundary conditions were applied at the far stream which were situated at 10 chord distance from the airfoil surface. This means that the far field pressure, air density and flow velocity are specified along with any turbulence intensity levels at the four far-field boundaries of the mesh surrounding the airfoil. The grid structure is shown in Fig. 1.1. The computational equations that are used are the coupled Navier-Stokes equations using second order upwind finite differencing. As mentioned earlier, the realizable k-ϵ model as suggested by Shih is used. Fluent 5 provides three turbulence models, a one equation model, three variants of the the two equation k-ϵ model and the five equation Reynold's averaged model(RSM). Though the RSM model is the best amongst these options, it is extremely time consuming. Using the RSM model, an analysis of flow separation as caused by a leading edge bump could take more than 6 hours on an SGI origin machine while using 4 processors. Considering that a large database has to be provided for training and testing the neural networks, it is not feasible to use the RSM model. The next best model amongst the available choices is the realizable k-ϵ model and this was what was used.

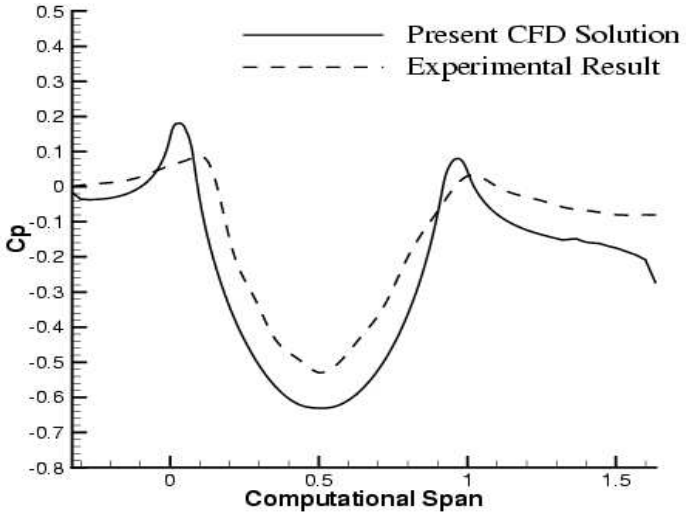


Figure 4.3: **Comparison of the present CFD pressure distribution with the experimental result obtained by Webster**[68]

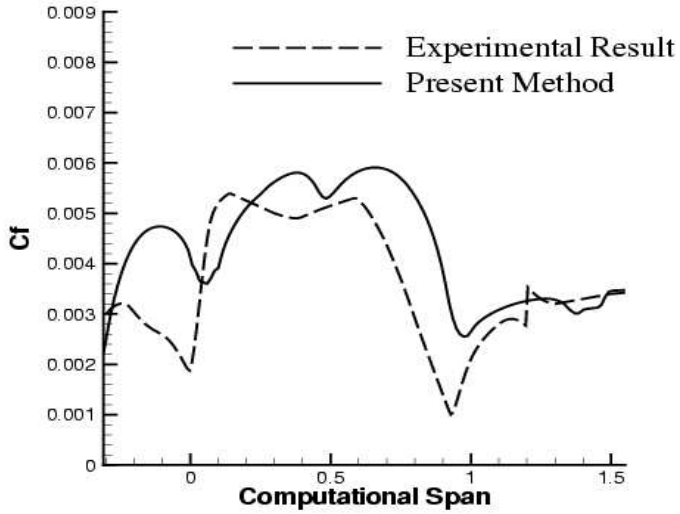


Figure 4.4: **Comparison of the present CFD skin friction coefficient with the experimental result obtained by Webster**[68]

To test the flow solver that was being used, an appropriate problem from literature was used. This was the two dimensional study of the turbulence of the flow over a bump situated in a wind tunnel. This problem is especially appropriate as this experimental result had also been modeled using CFD with various turbulence representations. Having a CFD model for a wind tunnel test is advantageous as it provides a picture of the computational domain which the original wind tunnel test does not provide. Accordingly the results of Webster[68] and Wu[69] are used. Here the bump is described by one convex circular arc and two concave arcs at both the ends of the convex arc. The wind tunnel allows the development of the boundary layer leading to the bump. The free stream flow velocity is at 16.6 m/s with a 0.2% turbulence intensity. The exit stream has a slight drop in pressure. The experimental results are mentioned to have an uncertainty of 5% for the normal stresses and 10% for the shear stresses.

The same unstructured triangulated grid that is used to model the airfoil with the bump is used in this analysis also. The realizable k-ϵ model and second order upwind differencing with a coupled Navier-Stokes solver are used. It was not possible to specify an exact boundary layer height at the start of the bump in the Fluent model as was mentioned in the wind tunnel testing. Apart from this, the computational domain of this wind tunnel model was achieved based on the CFD analysis of Wu[69]. Figure 4.3 describes the coefficient of pressure as calculated by the present CFD model in Fluent and the experimental results. As can be seen, the CFD result behaves similar to the experimental result but possesses some differences especially at the inlet and at the bump peak(minimum pressure location). A comparison of the CFD analysis by Wu[69] on the same bump shows that the RANS model also predicts a higher C_p at the start of the bump and a lower C_p at the bump

peak, unlike the experimental results. This is the trend observed in the present calculations also. Also considering that a $k-\epsilon$ model is being used, there will be some deviation between this CFD result and the actual results. Figure 4.4 describes the variation in skin friction coefficient. This parameter is also experimentally difficult to obtain. The RANS model predicts a higher C_f than the experiment at the bump peak[69]. The present model also predicts a higher C_f through the bump span. However the trends in the C_f variation are same between the present CFD solution using the $k-\epsilon$ model and the experimental model. In both Fig. 4.3 and Fig. 4.4, the computational span is the domain of the CFD analysis that includes the bump along with the upstream and downstream regions. The domain of the bump is confined between the locations 0-1 as marked in Fig. 4.3 and Fig. 4.4.

These results show that the computational solution consisting of the unstructured triangular mesh with the Navier stokes solver using second upwind differencing and the realizable $k-\epsilon$ turbulence model is acceptable for the analysis that is being attempted here, the main purpose of which is to train neural networks to represent the aerodynamic loads over the airfoil.

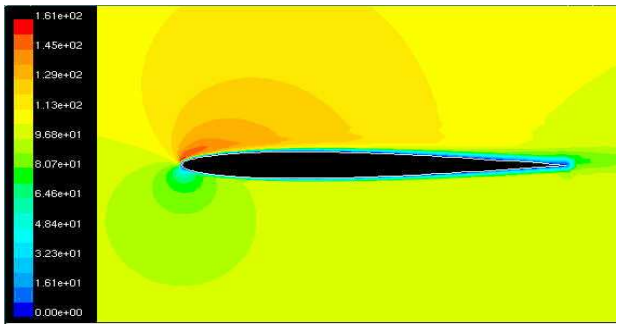
The structural package ANSYS is used to analyze the beam model of the bump which is treated as an elastica. Under axial compression, a non-linear finite element analysis is required. The ANSYS package provides the verification for a straight beam elastica under axial compressive loading. This verification has an example program which when run shows the good match between the ANSYS results and the theoretical results on the load-deflection diagram. Since this example has been verified by ANSYS, herein the same model is followed to analyze the bump, the only difference being, the addition of non-conservative non-uniform transverse forces along with the axial compressive

force. However, since the non-linear finite element solver for the elastica example verification used by ANSYS already uses an incremental load step procedure, the same solver is used for the bump analysis also. At each incremental load step, along with the updation of the axial load, the non-conservative aerodynamic loads are also updated in accordance with the bump deformation. The "NLGEOM" function in ANSYS is used to account for the geometric non-linearity. The BEAM3 finite element is used as referenced in the ANSYS documentation for the elastica problem. This has 2 nodes per element and 3 degrees of freedom per node, they being, two translational degrees of freedom and one rotational degree of freedom.

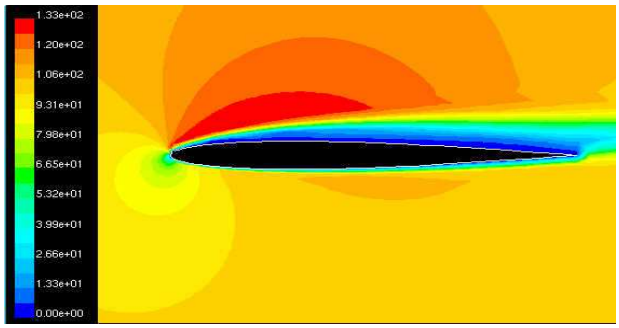
4.5 Low Speed Aerodynamics

The FLUENT CFD software was used to analyze the fluid flow over the selected airfoil using the grid shown in Fig. 1.1. For the low speed analysis the free stream velocity is assumed to be at Mach 0.3. The far-field pressure boundary conditions are specified on all the four faces of the boundary. The bump is located just after the leading edge at 0.4% of the airfoil chord on the upper surface of the airfoil. The size of the bump is blown up from a y co-ordinate value of 0.2% of the airfoil chord to a value of 0.4% of the airfoil chord. The effect of the variation of the bump size on the lift and drag is studied. Figure 4.5 shows the velocity magnitude contours. It can be seen that beyond a bump size on the order of 0.35% of the chord, there is a total separation of the flow.

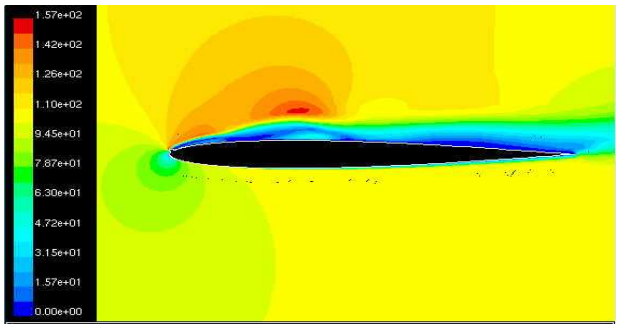
However as the bump height is further increased, as seen in the last portrait in Fig. 4.5 there is a re-attachment of the flow after an initial separation at the bump. This re-attached flow is not stable and separates after a zone of attached turbulent flow. Both from the point of view of actuation



a) Leading Edge Bump Height 0.2% Chord



b) Leading Edge Bump Height 0.35% Chord



c) Leading Edge Bump Height 0.4% Chord

Figure 4.5: **Velocity Contours generated using the FLUENT software showing the flow separation due to a leading edge bump located at 0.4% of the airfoil chord at a Free Stream Velocity of Mach 0.3**

energy and of aircraft control, it is necessary to confine the bump size to the region of predictable

separation. This means that the required bump height is of the order of 0.35% of the airfoil chord. Any further increase in the bump height produces an unstable region before a full separation of the flow over the top surface of the airfoil takes place. Since the purpose of having this bump is to provide a yaw control mechanism, it needs to be determined whether the drag produced by the specified bump size is sufficient. At the same time, the lift generated by the airfoil should not be seriously reduced. Figure 4.6 describes the variation of the pressure drag coefficient and the

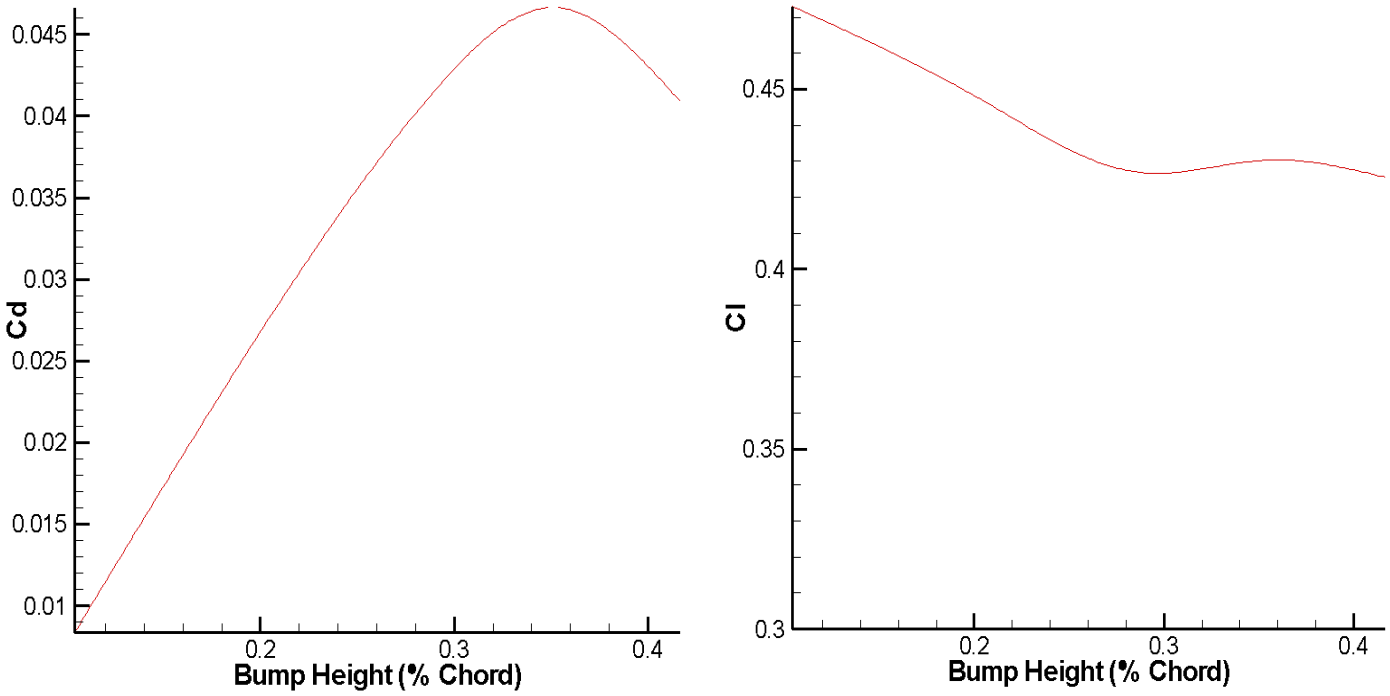


Figure 4.6: **Effect of the size of the Leading Edge Bump located at 0.4% of the airfoil chord on the Drag and Lift at a Free Stream Velocity of Mach 0.3**

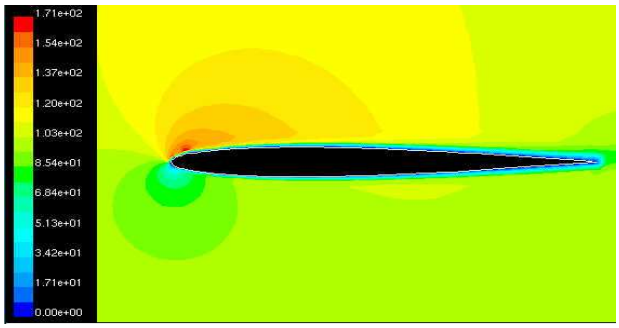
coefficient of lift with respect to the bump height. The lift appears not to be seriously affected. At the position of maximum pressure drag, the drop in the lift is about 20%. Above the bump height

of 0.085" or 0.35% of the chord, the drag and lift forces are oscillatory with respect to further increase in bump height. This is because of the re-attachment phenomena and further separation after re-attachment. This flow regime is also highly unsteady due to vortex shedding. It is seen that a pressure drag coefficient on the order of 0.04 is attainable. Using Eq. (4.1) and wing properties as

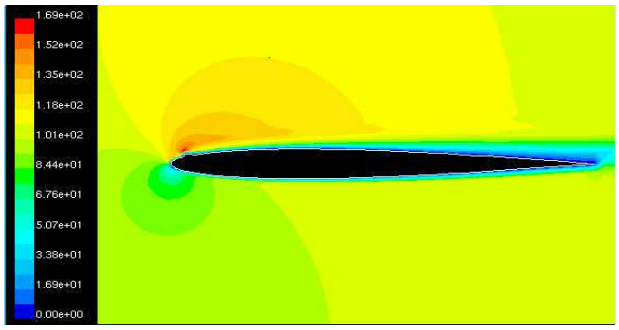
Span = 20m, Root Chord = 8m, Tip Chord = 2m, Aspect ratio = 4.0 and Wing material = Aluminum, maximum bump height = 0.35% chord, the yaw moment coefficient can be estimated as $C_n = 0.002$.

As mentioned earlier, this C_n is comparable to those estimated in Refs. 63 and 64. Figure 4.7 displays the effect of the same type of bump as in Fig. 4.3 but at a location further down the airfoil. One can notice from the two figures that the height of the bump required to cause separation is greater than the earlier version. In Fig 4.3, the bump was at a location of 0.4% of the airfoil chord, while in Fig 4.5 , the bump is at a location, 3.8% of the airfoil chord. Therefore, the boundary layer has thickened and consequently a larger bump is required for separation.

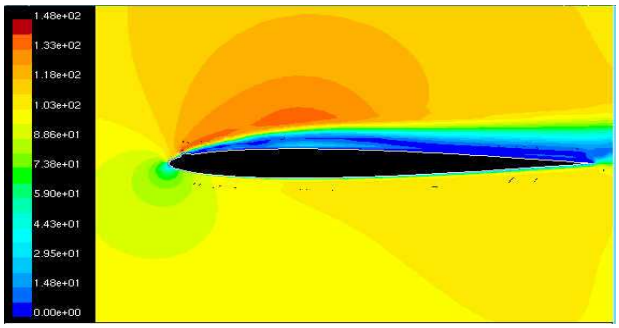
Figure 4.8 displays the variation in the lift coefficient and the drag coefficient with bump size. Again, after a certain bump height, oscillatory patterns are observed in the coefficient of lift. The main detail that emerges from these figures is that sufficient drag for yaw moment can be achieved without serious deterioration in the lift. The decrement in lift is "not substantial" since we are dealing with morphing wings, wings for which any loss in lift can be overcome with adaptive cambering of wing sections or changes in the wing twist[20] at locations where the flow does not separate. The rate of loss in lift as seen in Fig. 4.8 is smaller than that seen in Fig 4.6.



a) Turbulent Boundary layer



b) Onset of Boundary layer Separation



c) Fully Separated Flow

Figure 4.7: **Velocity contours generated using the FLUENT software showing the fluid separation over the airfoil at a free stream velocity of Mach 0.3 from a bump located at 3.8% of the airfoil chord**

This suggests that large changes in the fluid flow occur with small changes in the bump size and shape when the bump is placed at the leading edge than away from the leading edge.

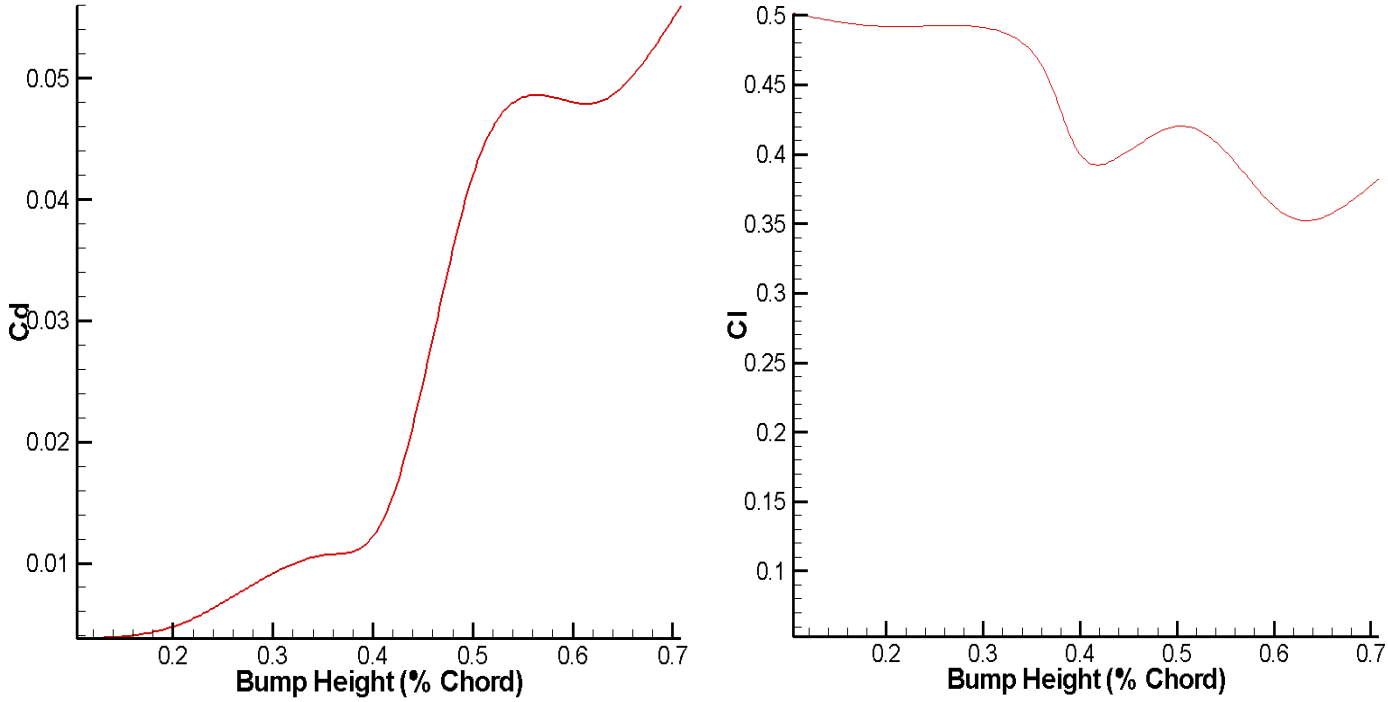


Figure 4.8: **Effect of the Size of a Bump, located at 3.8% of Chord on the Drag and Lift of the airfoil. (Free Stream Velocity of Mach Number = 0.3)**

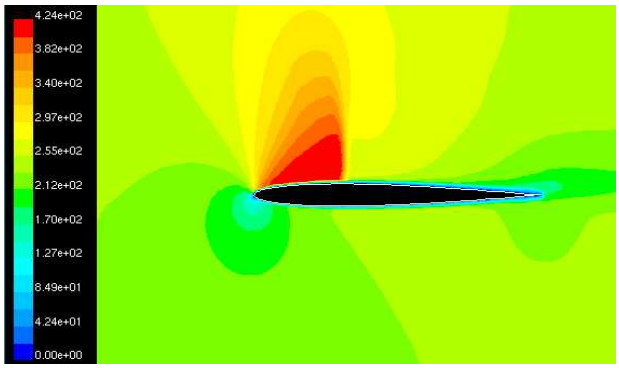
4.6 High Speed Aerodynamics

The previous section dealt with analysis of the effect of bump height on an airfoil with fluid flow at Mach 0.3. Typically the adaptable bumps that have been the subject of widespread investigation as mentioned earlier[21] have been to study wave drag reduction. The flow is transonic over the airfoil. Therefore, we investigate a flow speed of Mach 0.7 over the same 8% thick airfoil. Figure 4.9 displays the velocity magnitude contours over the airfoil. The first portrait shows the airfoil with no leading edge bump. A strong shock is seen over the airfoil. This naturally leads to a substantial wave drag. Now an adaptive bump is grown at a location of $\frac{x}{c} = 0.4\%$. The next two portraits

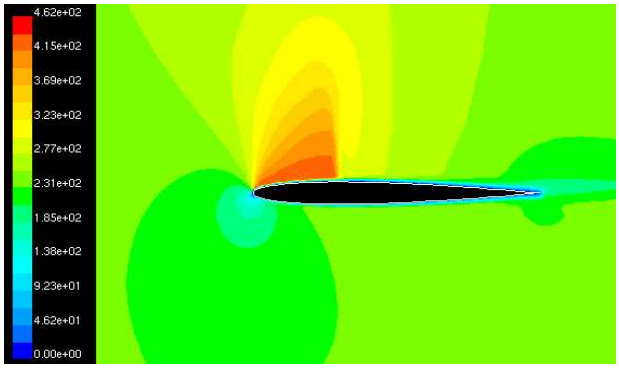
in Fig. 4.7 display the velocity contours at bump heights at 0.1%, 0.28% of the airfoil chord. It can readily be observed that the shock is weakened significantly. At a bump height of 0.2% of airfoil chord, there is boundary layer separation. On further increase of the bump height to 0.28%, the shock is completely lost and we get total boundary layer separation. From the point of view of actuation energy required to produce this bump, it can be seen that a very small bump size is enough to nullify the shock. In contrast, typical wave drag minimization bumps have been situated near the edge of the shock, towards the trailing edge with a maximum bump size of 0.5% of the airfoil chord[21]. This limit on the bump size to 0.5% of the chord is to prevent boundary layer separation.

4.7 Boundary Layer Separation Control

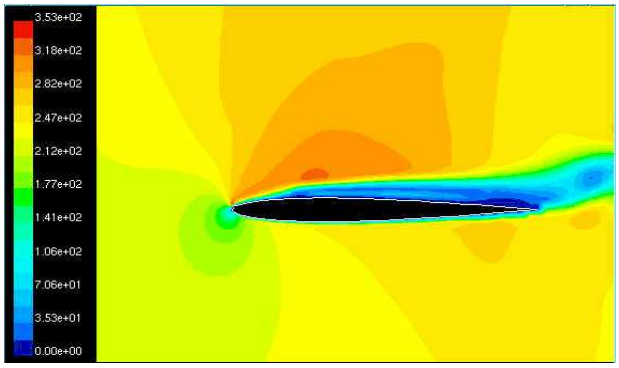
The basic mechanism for drag creation used in this chapter is a controlled disturbance to the boundary layer at its initiation. This disturbance to the boundary layer comes in the form of a shape change at the leading edge. There have been other mechanisms that have been used for boundary layer control at the leading edge. For example, periodic excitation at the leading edge has been used to control flow separation. A wide variety of techniques for providing for this perturbation are available. A substantial discussion on these techniques can be found in Ref. 27. All these techniques involve the presence of an external agency such as air blowing, air suction or a periodically vibrating media. The present method employs internal actuators to morph the shape of an adaptive material. This therefore provides for a direct method for boundary layer control at



a) Clean airfoil



b) Leading Edge Bump Height 0.1% Chord



c) Leading Edge Bump Height 0.28% Chord

Figure 4.9: **Velocity Contours generated using the FLUENT software showing the effect of the bump size on the shock and the boundary layer (Free Stream Mach No. 0.7)**

the leading edge.

Reference 2 describes the boundary layer control through an adaptive leading edge bump for causing delay of the onset of flow separation at negative angles of attack. Here the leading edge has an adaptive bump which is deflected using the THUNDER actuator. However, the authors[2] mention that the control of the position and shape of this bump is a difficult task without knowing before hand the change in aerodynamic forces with bump displacement. It is therefore necessary to perform an aeroelastic analysis, to estimate the effect of the actuator forces on the bump shape and the resulting changed aerodynamic forces. Neural networks trained from CFD analysis and structural analysis can be used to predict the best shape for controlled flow separation. The following sections describe the training of such a neural network.

4.8 Optimization Variables and Constraints

As mentioned earlier, the bump is defined by a single cubic spline. Hence this curve has four undetermined coefficients. Since axial displacement is allowed, the location of the curve end becomes another variable. Thus there are five geometric design variables. A geometric constraint that is imposed is that the bump maintains C_0 continuity between its ends and the airfoil surface throughout the optimization procedure.

Structurally, the bump is modeled as a pinned-pinned beam as shown in Fig. 4.10. This figure describes the actuator loads as an unknown axial compressive load and an unknown transverse load. The axial load is unknown in magnitude. The transverse load is unknown in magnitude, direction and location. The optimum bump size determines these loads. Hence structurally there

are four unknown variables.

The design goal is to maximize pressure drag while minimizing losses in lift and minimizing the strain energy of deformation. Hence the design constraint is minimal loss in lift and minimum strain energy of deformation while the objective is to increase pressure drag so as to generate sufficient yawing moment. The constraints will be included in the objective function as penalty functions. In order to facilitate the constraint of minimum strain energy, the structural deformation of the bump is modeled as an elastica as the elastica curve is the minimum energy curve.

4.9 Low Speed Aeroelastic Optimization

Using the aerodynamic results from FLUENT, the neural network, NN-CFD capable of predicting the fluid static pressure at four points over the bump was trained. Since the bump itself was only 1.3% of the airfoil chord, using 4 points over the bump to evaluate the pressure variation was found to be sufficient. A variety of bump shapes similar to those depicted in Fig. 4.2 are used in the training. The static pressures calculated by the neural network are inputs to the ANSYS structural analysis of the bump. Using ANSYS, both the actuator loads and the strain energy required for forming the bump are obtained and are used as targets for the second neural network, NN-STR.

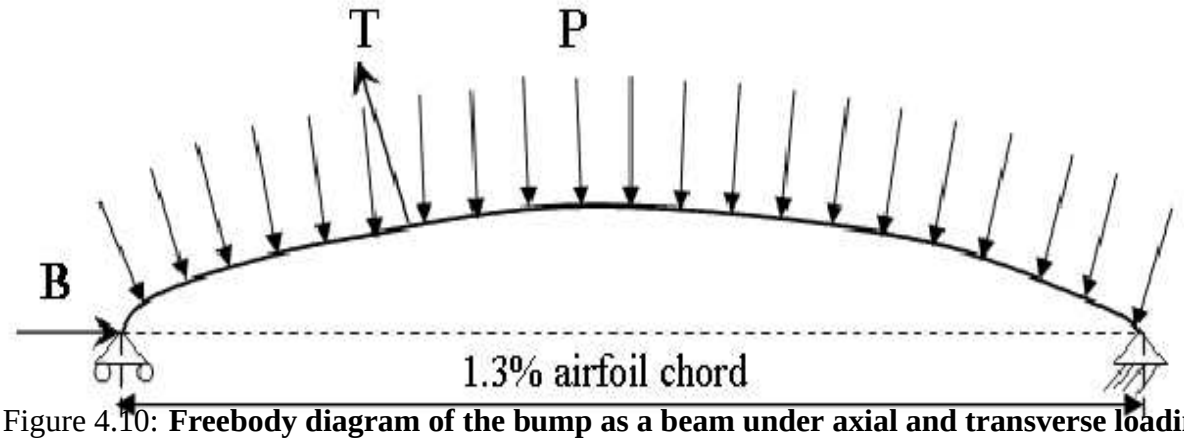


Figure 4.10: **Freebody diagram of the bump as a beam under axial and transverse loading.**

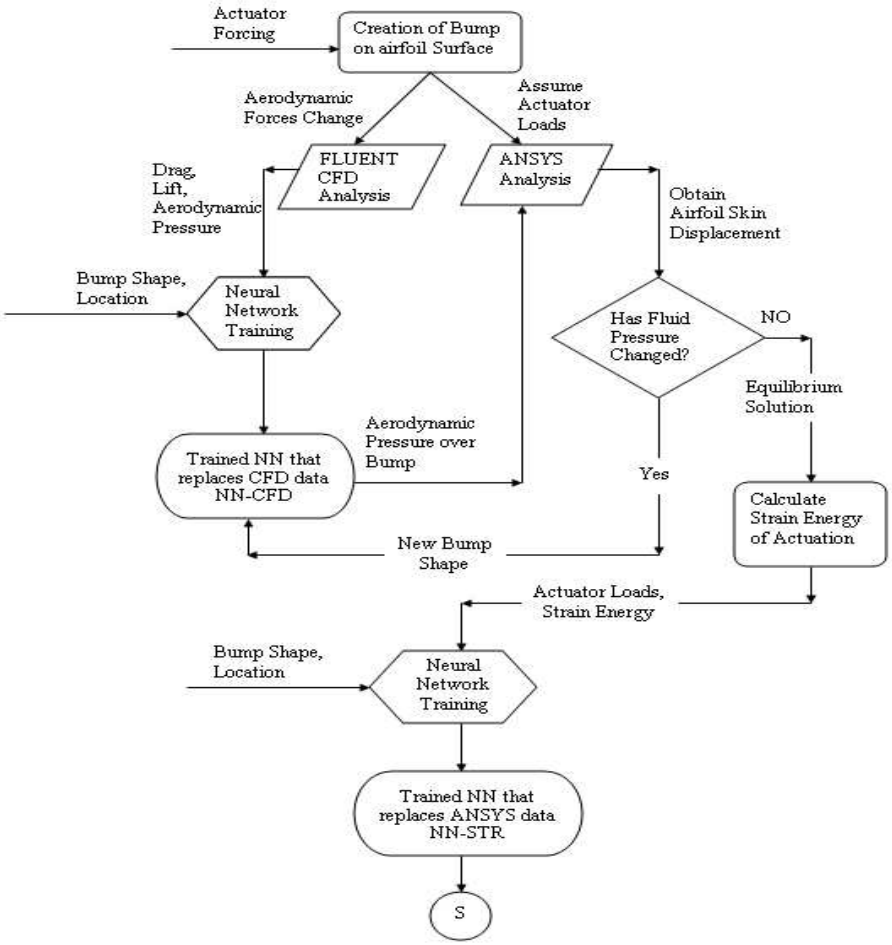


Figure 4.11: **Process of evaluating the aeroelastic response of the bump by training two Neural Networks, one for predicting the aerodynamic loads over the bump and the other for the structural response of the beam model and actuator loads that deform it**

This neural network also uses the drag and lift coefficients over the airfoil as targets. Using these targets the neural network is trained to predict strain energy, actuator loads, lift and drag over the airfoil, given the coefficients of a cubic spline fit to the bump as its input. Once the relevant data has been obtained from FLUENT and ANSYS, the training of the neural networks in MATLAB is completed in a quick time of about 20 minutes. The trained neural networks are then stored in binary files. These trained neural networks are used for optimizing the bump shape.

The bump is subjected to fluid dynamic pressure, a compressive axial load and an arbitrary vertical load. The axial and vertical loads are the actuator loads. Figure 4.10 describes the free body diagram of the bump as a curved beam subjected to an axial compressive load, B, aerodynamic pressure, P and an arbitrary load T. The load T is arbitrary in the sense that its direction and magnitude as well as its location needs to be fixed by the optimization process. The beam length is small, being 1.3% of the chord in the case of low speed flow and 3.5% of the chord in case of high speed flow. Figure 4.11 depicts a flow chart which describes the process of training the neural networks for obtaining the structural loads and energy for deformation of the bump and the fluid dynamic pressure over the bump along with the aerodynamic lift and drag. We formulate the optimization problem as

Minimize:

$$\frac{1}{{C_d}^2} + G * (C_l - 0.47)^2 + G_2 * \kappa^2$$

Subject to: C_0 continuity between bump and the airfoil (4.2)

Here, C_l, C_d represent the coefficients of lift and drag respectively. κ represents the scaled strain energy, scaled with respect to the the largest deformation strain energy, G and G_2, two penalty

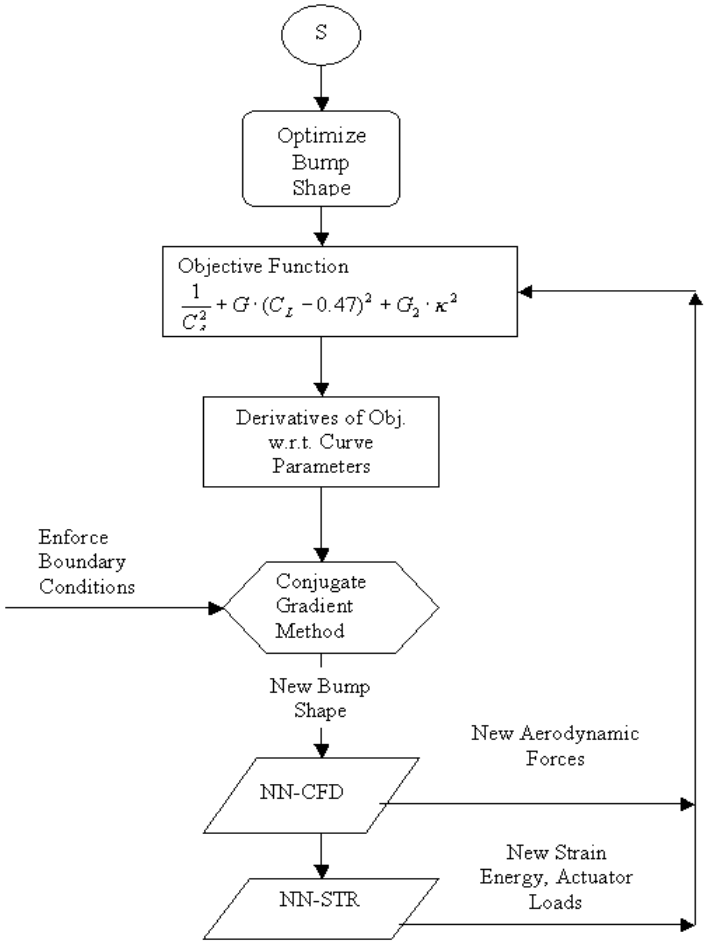


Figure 4.12: **Process of optimizing the bump shape using the trained neural networks**

weights, selected by trial and error depending on the rate of minimization of the objective function and also the C_0 continuity between the bump and the airfoil. This problem is schematically represented in the flow chart shown in Fig 4.12.

The optimization is performed using the conjugate gradient method. Since this problem is nonlinear, the number of iterations for convergence for the conjugate gradient method is not fixed. However, it is still better than the method of steepest descent. The direction of the gradient is de-

termined using Newton-Raphson's method. The C_0 boundary conditions for the bump are fixed by imposing a penalty on the objective function to be minimized. Since the conjugate gradient method requires the calculation of the Jacobian and Hessian matrices, a finite difference approximation for the derivatives at each point in the iteration is calculated. At each step in the iteration, the neural network NN-STR simulates the new actuator loads, strain energy and resulting drag and lift over the airfoil. However, the problem is non-linear. Hence, the optimum value for the bump shape that is obtained from the neural network may be a local optimum. The optimization is repeated for a set of bumps with initial aerodynamic and structural parameters; the aerodynamic parameters being the drag and lift, the structural parameters being the actuator loads and the strain energy. Amongst this calculated set of optimal solutions, the best solution is that which approaches the design lift constraint of 0.47 and still provides a drag coefficient of at least 10^{-2}. The calculated bump shapes which minimize the objective function is shown in Fig. 4.13. The neural network was trained based on bump shapes similar to those shown in Fig 4.2. One can conclude that, of all bump shapes that are within the range of those used in training the neural network, the bump shapes shown in Fig. 4.13 minimize the strain energy and retain maximum drag and minimum loss in lift. The latter criterion, i.e. minimum loss in lift is significant. The created airfoil surface bump, not only provides for drag sufficient for yaw control, but also allows for a minimum loss in lift such that it can be compensated by slight cambering of a different wing section. Since the neural network NN-STR is capable of simulating the aeroelastic interactions of the bump, we have a rapid 2-D static aeroelastic optimization tool.

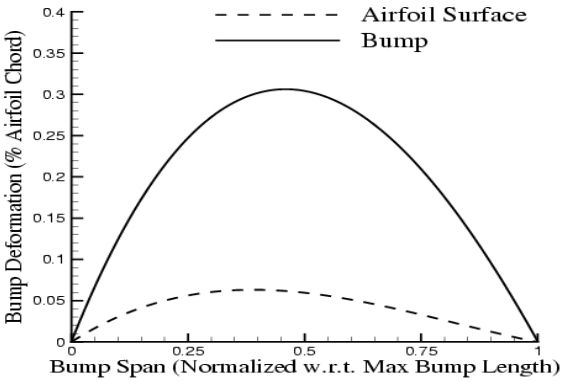
The three bump shapes shown in Fig. 4.13 detail the optimum shapes for minimal loss in lift.

As given by Eq. (4.2), we require that the lift should not be drastically reduced below a value of 0.47. The drag and lift coefficient values are given along with the figures. Due to the limitations of the neural network training, the predicted values of drag and lift using the neural network are about 10% away from the corresponding CFD results. The predicted strain energy using the trained neural network is approximately 16% away from the corresponding strain energy as predicted by ANSYS.

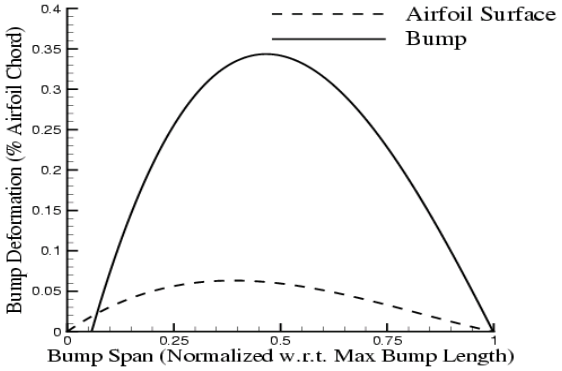
1. Figures 4.13 (a) shows a bump that was formed without any axial load. The bump has only a vertical actuator load at its center. As shown in the figure, there is a moderate drag increase without substantial loss in lift.

2. Both Fig 4.13(b) and 13(c) show bumps formed with axial and tranverse actuator loads. A moderate drag increase without serious loss in lift is obtained from the bumps.

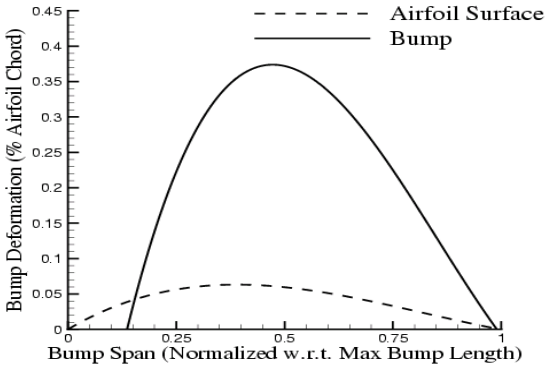
From the obtained drag, lift and strain energy for the bumps shown in Fig. 4.13, it can be said that Fig. 4.13(b) describes the optimum bump shape amongst the set of bumps that can be generated from the beam model shown in Fig. 4.10. Table 4.1 compares the optimized results with other bumps used for training the neural networks. The bumps shown in Fig. 4.2 are taken as reference for comparing the data for the Optimized bump shapes. Each column of the data in Table 4.1 for the non-optimized bumps refers to Fig. 4.2. Thus a drag coefficient of the order of 0.02 is attainable without serious loss in lift and by minimizing the strain energy required for forming the bump. The clean airfoil obtains a lift coefficient of 0.52 at 5 degrees angle of attack.



a) C_d=0.0227, C_l=0.4543, Strain Energy=115.7 J/cm width



b) C_d=0.0245, C_l=0.456, Strain Energy=109.6 J/cm width



c) C_d=0.0222, C_l=0.4393, Strain Energy=128.4 J/cm width

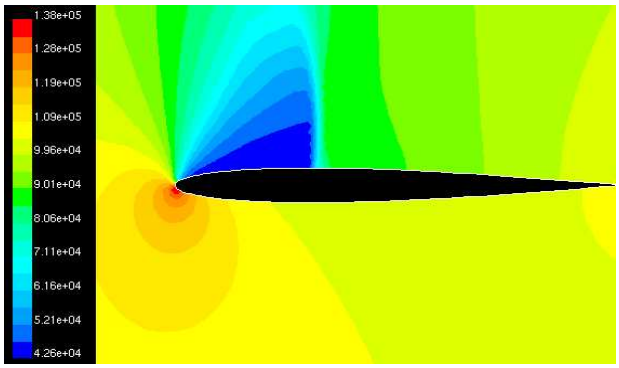
Figure 4.13: **Optimized bump shapes for an adaptable bump on the surface of the airfoil when the airfoil is placed in a free stream with a velocity of Mach 0.3**

Therefore there are lift losses of the order of 12%. However, since we are dealing with an adaptable wing, the wing can possess adaptive cambering at a section different from wing section that has a bump. A 1% camber increase at a location of 70% of the chord of an airfoil section with attached flow is enough to offset the decrement in lift due to the bump.

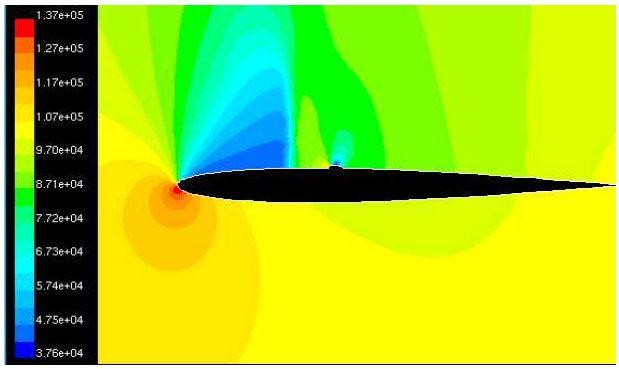
4.10 High Speed Aeroelastic Optimization

For the free stream flow velocity of Mach 0.7, Fig. 4.9 describes the effect of the leading edge bump on the shock and also the boundary layer. The same Realizable $k-\epsilon$ model was used, as was done in the low speed case. This turbulence model may not be adequate for such high speed flows, but due to limits in computational time, a more advanced model such as the 5 equation Reynold's stress model could not be used. This can be a topic of future research work, i.e. significance of turbulence models on aeroelastic analysis. As described earlier, a very small perturbation (of the order of 0.1% of the chord) decreases the wave drag. Further increase in the bump size leads to flow separation. From the actuator point of view, the main concern here is the rapidly changing aerodynamic pressure over the bump with small perturbations in the bump height. The leading edge bump could not be used for high speed flow as a steady state bump equilibrium is difficult to achieve due to the rapid fluctuation in the aerodynamic load over the bump with even minute bump shape changes.

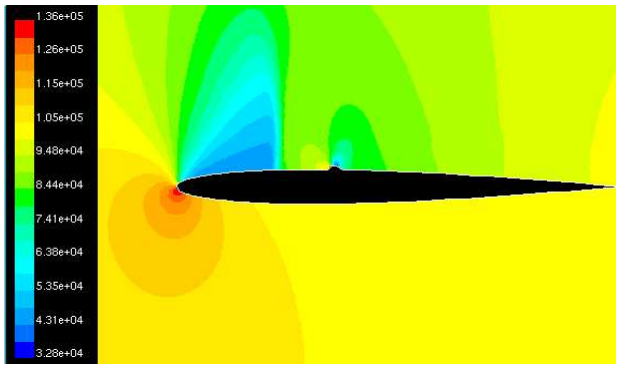
Also the drastic changes in pressure over the bump with minute changes in bump shape could not be captured by the neural network.



a) Clean Airfoil, C_d=0.0433 C_l = 0.717



b) C_d = 0.0482, C_l = 0.632



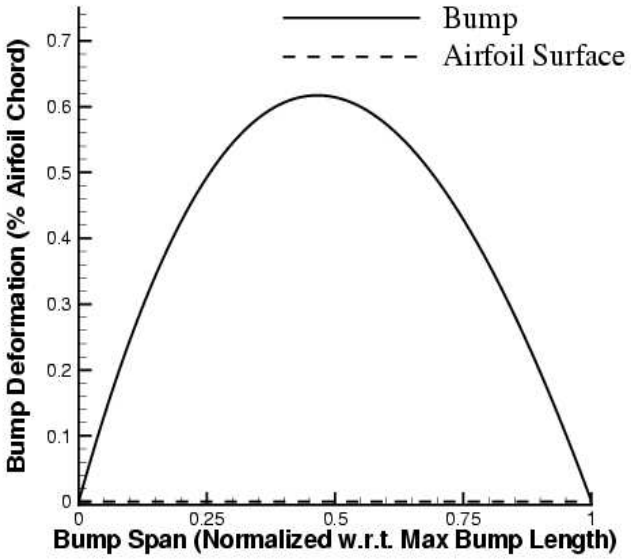
c) C_d = 0.0536, C_l = 0.613

Figure 4.14: **Static pressure contours over the airfoil as generated using the FLUENT software showing the effect of the bump at a Free Stream Velocity of Mach 0.7**

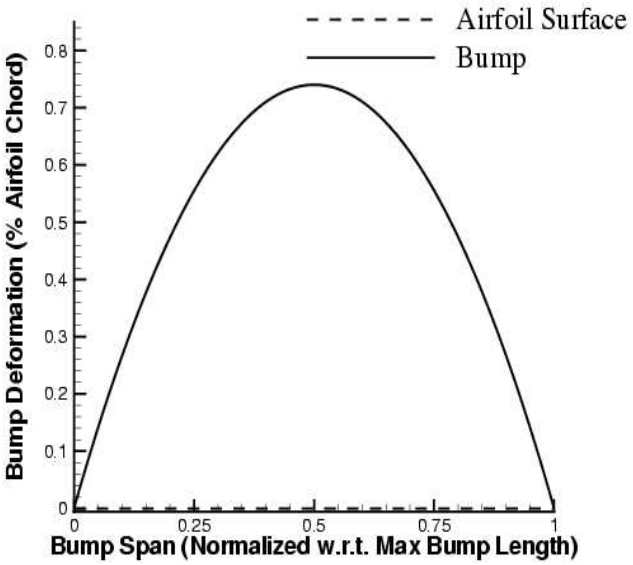
Since this effort is not concerned with shock control, the bump is placed downstream of the shock.

This allows for choosing a bump shape that is not drastically affected by changing aerodynamic pressure. Even then, the low pressure over the bump, does not allow for axial compression, unlike the low speed model. This is because the bump is now susceptible to buckling at very low axial loads. Hence a pinned-pinned beam is considered subjected to an arbitrarily oriented load at any arbitrary position.

As was done for the case of low speed, a total of 14 to 15 runs were performed in FLUENT and ANSYS with different bump shapes to collect sufficient data to train a neural network. The trained neural network is then used to optimize the bump shape that provides maximum drag with a minimum loss in lift and minimum strain energy required to form the bump. Figure 4.14 shows the effect of bump on the shock and the boundary layer. It can be seen that the amount of separation generated is lesser than the low speed case. This means that the amount of drag increase is also less as can be seen in Fig. 4.14. The effect of the bump on the shock is also seen. Compared to Fig. 4.9, Fig. 4.14 shows that a much larger bump size is required to cause a full weakening of the shock. The height of the bump in Fig. 4.14 (c) which shows the weakened shock is of the order of 0.92% of the airfoil chord. Many such bump shapes were used in training the neural network. Again the Levenberg-Marquardt training algorithm is used. The trained neural network is used to operate a conjugate gradient scheme to select the optimum bump shape for high drag with low loss in lift and low actuation energy. Fig. 4.14 shows that the low pressure exists over the bump even at large bump heights as can be seen from Fig 4.14(c).



a) C_d=0.0498 C_l = 0.681, Strain Energy = 104 J/unit cm Thickness



b) C_d = 0.0519, C_l = 0.653, Strain Energy = 198.5 J/unit cm Thickness

Figure 4.15: **Optimized Bump Shapes for an adaptable bump on the airfoil surface when the airfoil is placed in a free stream with a velocity of Mach 0.7**

The objective function is the same as was used for the low speed case, i.e. Eq. (4.2), but the lift coefficient was fixed at 0.63, not 0.47 as in Eq. (4.2). The bump shapes that minimize this objective function are shown in Fig 4.15. Again, many initial conditions were tried to select the best optima. The two optimized bump shapes are shown in Fig. 4.15: The optimized bump shapes from the neural network predicted the drag and lift coefficients within 10% of the corresponding results as predicted by the Fluent software. As in the case of low speed analysis the strain energy of deformation as predicted by the trained neural networks was within $12\% - 16\%$ of the ANSYS result.

1. A moderate pressure drag coefficient increase of 0.005 but with a low loss in the lift and low strain energy required to form the bump as shown in Fig. 4.15(a).

2. Fig. 4.15 (b) shows the bump with a significant drag increase and with a low loss in the lift and moderate strain energy required to form the bump.

These two bump shapes can be selected depending on the amount of drag that is required. Note that the strain energy reported is per unit cm thickness of the beam.

The bumps used in high speed optimization were larger than those used in the low speed case. This was due to the fact that the bump for the high speed case was situated downstream of the shock. However the rise in the coefficient of drag for the high speed case due to the effect of the bump was lesser than the coefficient of drag rise for the low speed case. Since the drag force varies as the square of the flow velocity, this lower coefficient of drag rise in high speed is acceptable. Another feature in high speed flow, is the fact that initial bump deformations reduce the net drag over the

airfoil due to the fact that the wave drag is reduced as the shock weakens.

4.11 Effectiveness of Neural Networks for Aeroelastic Optimization

The first advantage of the use of neural networks is the saving in time. A typical CFD run using the Fluent software during the training process took around 2 hours on an SGI origin machine using 4 processors. The ANSYS runs were much shorter in time. A convergence solution with one set of actuator loads and aerodynamic loads takes about 10 minutes. However, the aerodynamic loads change after this convergence and hence the ANSYS iterations have to be re-initiated. On the contrary using the two neural networks NN-STR and NN-CFD, one need not perform any iteration between the structural and fluid dynamic solver for an aeroelastic analysis. However, iterations have to be performed for aeroelastic optimization. Each aeroelastic simulation using the neural networks takes only a few seconds using 1 processor on the SGI origin machine. Hence one optimization routine starting with an initial bump shape can be completed in a few minutes. However the result of the optimization may not be a global optima and hence many trial runs need to be performed with various bump shapes. The aerodynamic load output of the trained neural networks was within 10% of the corresponding CFD results. The structural loads and strain energy that were predicted by the neural network was about 16% away from the corresponding ANSYS result.

The adaptive bump at low speed being close to the leading edge attains much higher coefficient of drag rise than the bump at high speed. At high speeds, the low pressure over the bump can be used as an advantage in deforming the bump. Hence the bigger bump sizes that were used in the high speed flow analysis are achievable, but they still provide less coefficient of drag rise. This is however compensated by the fact that the drag force depends on the square of the flow velocity. Hence at transonic speeds also the bumps can be effective.

Neural Networks were able to capture the non-linear aeroelastic effects of the deforming bump on the aerodynamic behavior of an airfoil. However, it was found that the neural networks respond accurately only for a certain range of bump shapes. This is due to the procedure used in training the networks and also the type of data that were used in the training. The neural networks also showed erroneous behavior if extreme fluctuations in the static pressure were present for very small changes in bump shape. Merely increasing the data quantity does not improve the accuracy. A better training algorithm may also have to be used. The neural networks used here showed promising behavior.

This facilitates a rapid calculation of aeroelastic behavior of the airfoil bump. However, the sensitivity of the response to the shape of the structure is still confined to a certain domain of shapes, since the neural networks are trained using gradient based schemes. Improvements in the neural network performance may be possible if non-gradient based training schemes such as Genetic Algorithms or Simulated Annealing are used. A clear representation of the aeroelastic behavior of an adaptive wing through neural networks speaks of a high potential in the design of an active aeroservoelastic flight control system.

Table 4.1: Comparison of Optimized Results with Other Results

Bump Shapes used in Training	A	B	C	D	E	Optimized Shapes		
Clean Airfoil: $C_d = 0.005$, $C_l = 0.52$								
Strain Energy-J/cm Thickness	17.24	91.536	89.43	25.71	133.09	115.704	128.628	109.64
C_d	0.0174	0.039	0.0274	0.0078	0.018	0.0227	0.0222	0.0245
C_l	0.453	0.3689	0.414	0.4771	0.4686	0.4559	0.4393	0.4476
Axial Load - N	4000	9012	4000	0	0	0	1566	1086
X-Load - N	-5000	0	3000	0	-30000	0	-28110	-3582
Y -Load - N	8000	0	20000	19000	60000	50010	64950	60990
Position of X-Y Load % Span	18%	0	70%	45%	73%	45%	19.94%	47.92%
Objective Function	3529.1	10879	4468.3	16487	3089.2	2140.1	2972.3	2168.3

Chapter 5

Dynamic Aeroelastic Analysis of an Adaptive Airfoil using Neural Networks

An airfoil exhibiting two degrees of freedom, that is, motion in pitch and plunge displacements is considered as depicted in Fig. (1.2). The torsional stiffness of this system described in Fig. (1.2) is allowed to be time varying. The solution procedure for a time-varying system is described in Chapter 2 and the same procedure is adopted herein to train artificial neural networks to compute the structural dynamic response of the 2-D system. The unsteady aerodynamic loads over the airfoil are calculated using distributed doublet panels. Artificial neural networks are trained using the panel code to represent the unsteady aerodynamics. Thus the dynamic aeroelastic response of the system with a time varying torsional stiffness can be represented using neural networks. Having done so, a flutter suppression system based on this concept of a variable stiffness is designed using the trained neural networks.

For incompressible inviscid flow, panel methods provide a more rapid estimate of the unsteady fluid dynamic forces as compared to CFD. Panel methods consist of distributing singularities such as sources, vortices and doublets on the surface of an airfoil or a wing. The boundary conditions on the wing or airfoil then determines the strengths of these singularities. Once the strengths of the singularities are known, it is possible to calculate the unsteady pressure distribution at each point over the wing or airfoil. A descriptive detail of various panel methods can be found in Ref. 70.

The classical aerodynamic analysis was framed in the frequency domain wherein Fourier Transforms were used to describe the unsteady aerodynamics. This analysis resulted in analytical solutions for the unsteady lift and pitching moment in the form of complex Bessel functions. However such analysis was confined to represent the aerodynamics over a flat plate. Using panel methods on the other hand allows the shape of the airfoil to be considered in the analysis. Furthermore, the presence of a time-varying stiffness changes the aeroelastic solution from the classical solution. Hence we model the unsteady aerodynamics numerically in a time marching scheme.

In the present chapter, we are considering only two dimensional aerodynamics. Using distributed doublet panel elements over the surface of a morphing airfoil, the incompressible inviscid aerodynamics is modeled. The boundary conditions that are imposed over the airfoil are the non-penetration condition and the Kutta condition. The non-penetration condition requires that the normal velocity of the fluid at the airfoil wall should be equal to the airfoil surface velocity. The summation of the normal velocity of the fluid at the airfoil surface due to the motion of the airfoil, the free stream and the doublet perturbation must be zero. The Kutta condition at the trailing edge requires that the flow velocity at the trailing edge point to vanish. Numerically this is difficult to

achieve, hence the position and size of the panel at the trailing edge needs to be carefully chosen.

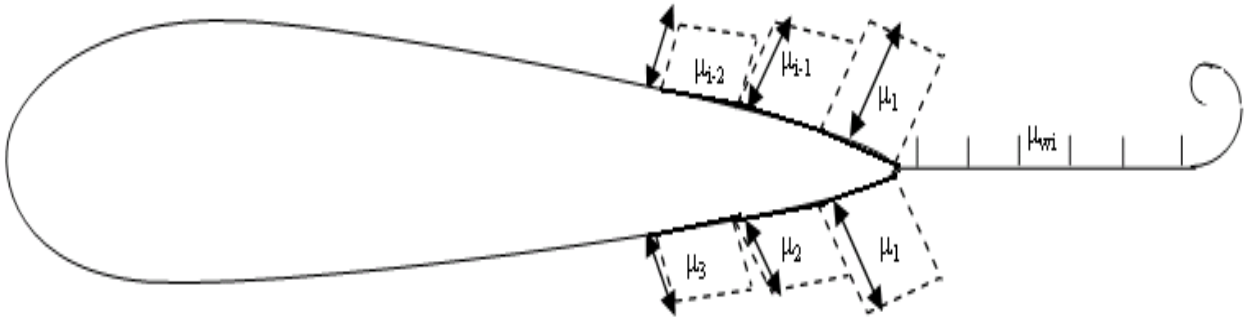


Figure 5.1: **Distribution of constant strength doublet panels over the airfoil surface and wake to model the unsteady aerodynamic loads over the airfoil in inviscid incompressible flow**

Panels of constant strength doublets are distributed over the surface the airfoil and the strength of the doublets for steady flow is determined by applying the boundary conditions:

$$\frac{\partial \phi}{\partial \mathbf{z}} (\mathbf{x}, \mathbf{z}\pm) = \mathbf{U}_\infty \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} - \alpha \right) \tag{5.1}$$

where ϕ is the velocity potential, x, z are the geometry variables of the airfoil, U_∞ is the free stream velocity and $z = g(x)$ is the function describing the airfoil surface. The influence of each panel on every other panel on the airfoil is calculated. For unsteady calculations, the influence of the wake on the airfoil is also very important. A time-marching scheme is used and at each time step, one doublet panel is shed into the wake from the airfoil trailing edge. The combined influence of all the wake panels on the airfoil panels is calculated for every time step.

The mathematical derivation for panel methods comes from the governing Laplacian equation for incompressible potential flow, namely,

$$\nabla^2 \phi = 0 \tag{5.2}$$

where ϕ is the velocity potential. Equation 2 is valid in steady and unsteady incompressible flow. Using Green's theorem and Eq. (5.2), it can be proved that[71], any surface in incompressible potential flow can be replaced by a summation of doublets and sources. Typically in thin airfoil theory, a doublet represents the lifting component and a source represents the airfoil thickness. To make the analysis simpler, we are considering only doublet representations over the airfoil. According to Ref. 70, the absence of source panels is not a disadvantage for lifting airfoils. Having determined the doublet potential for each panel, the unsteady Bernoulli equation (Eq. 5.3) can be used to determine the coefficient of pressure at every point over the morphing airfoil.

$$C_p = 1 - \frac{u^2}{{u_\infty}^2} - \frac{2}{{u_\infty}^2} \frac{\partial \phi}{\partial t} \tag{5.3}$$

where u is the local fluid velocity over the airfoil and u_∞ is the free stream velocity.

5.1 Steady Aerodynamics

Before computing unsteady aerodynamics, steady state computations are performed using distributed doublets over the surface of the airfoil with an infinite wake. After successfully performing calculations for steady flow, unsteady aerodynamic computations will be performed for validation with theoretical results. Figure 5.1 describes the panel distribution at the trailing edge of the airfoil. It should be understood that the same panel distribution is present throughout the airfoil surface.

Equation 5.1 is used with $g(x)$ being described by the parametric representation given by B-splines. The free stream velocity u_∞ is taken as unity without losing any generality. The influence of each doublet panel on the free stream velocity is then formulated as[70]

$$\begin{aligned} u_i &= \frac{\mu}{\mathbf{2}\pi}\left(\frac{\mathbf{z}}{\mathbf{(x-x_i)^2+z^2}} - \frac{\mathbf{z}}{\mathbf{(x-x_{i+1})^2+z^2}}\right) \\ w_i &= \frac{-\mu}{\mathbf{2}\pi}\left(\frac{\mathbf{x-x_i}}{\mathbf{(x-x_i)^2+z^2}} - \frac{\mathbf{x-x_{i+1}}}{\mathbf{(x-x_{i+1})^2+z^2}}\right) \\ a_{ij} &= (-\mathbf{sin}(\theta_\mathbf{j}-\theta_\mathbf{i}), \mathbf{cos}(\theta_\mathbf{j}-\theta_\mathbf{i})) \cdot (\mathbf{u_{ij}}, \mathbf{w_{ij}})^\mathbf{T} \end{aligned} \tag{5.4}$$

Here μ is the unknown doublet potential, $(x_i, 0)$ is the location of the doublet panel in its co-ordinate frame and θ_i is the orientation of the doublet panel i. The influence of this doublet i on every other doublet panel j is calculated using the velocity components u_{ij} and w_{ij}. The Kutta condition is enforced by making the vorticity at the trailing edge to be zero. This is given as

$$\mu_\mathbf{t1} - \mu_\mathbf{t2} + \mu_\mathbf{w1} = \mathbf{0} \tag{5.5}$$

where μ_{t1} and μ_{t2} are the two trailing edge doublet panels and μ_{w1} is the first wake panel. Equation (5.1) can be discretized for satisfying the no penetration condition as

$$\{\mathbf{U}_\infty + \mathbf{u}, \mathbf{W}_\infty + \mathbf{w}\}.\hat{\mathbf{n}} = \mathbf{0} \tag{5.6}$$

where u, w are the local velocity perturbations from the doublets and U_∞, W_∞ are the free stream velocity components. $\hat{n}$ is the direction of the unit normal to the airfoil surface. When Eqs. (5.4) and (5.5) are substituted in Eq. (5.6), a series of algebraic equations in the unknown doublet potentials, μ_i are obtained. The software code for performing this in MATLAB is given in Appendix B.

The doublet potential across the airfoil surface, μ_i, is solved and the pressure coefficient at every point on the airfoil surface is calculated using the Bernoulli equation, Eq. 5.3. Figure 5.2 describes the distribution of pressure over a symmetric airfoil at an angle of attack of 5^0. This matches the theoretical solution over the airfoil except at the trailing edge. This discrepancy near the trailing edge is due to numerical problems of satisfying the Kutta condition exactly at the trailing edge.

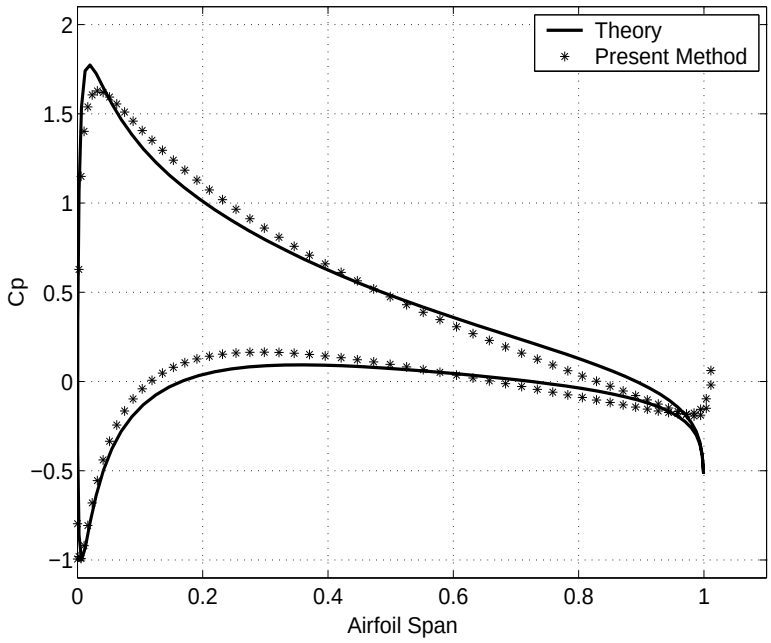


Figure 5.2: **Steady state pressure distribution at an angle of attack of 5^0 over a symmetric airfoil as predicted by the doublet panel code and compared with the theoretical solution**

The Kutta condition is satisfied numerically by forcing the vorticity at the trailing edge due to the doublet panels to be zero. In Ref. 70 also, the numerical scheme results are not shown at the airfoil trailing edge for steady flow. Instead, the theoretical solution is used. Theoretically, the velocity of flow should be zero at the trailing edge, as long as the trailing edge is not a cusp. Therefore the theoretical value for the coefficient of pressure should be unity. This is difficult to realize

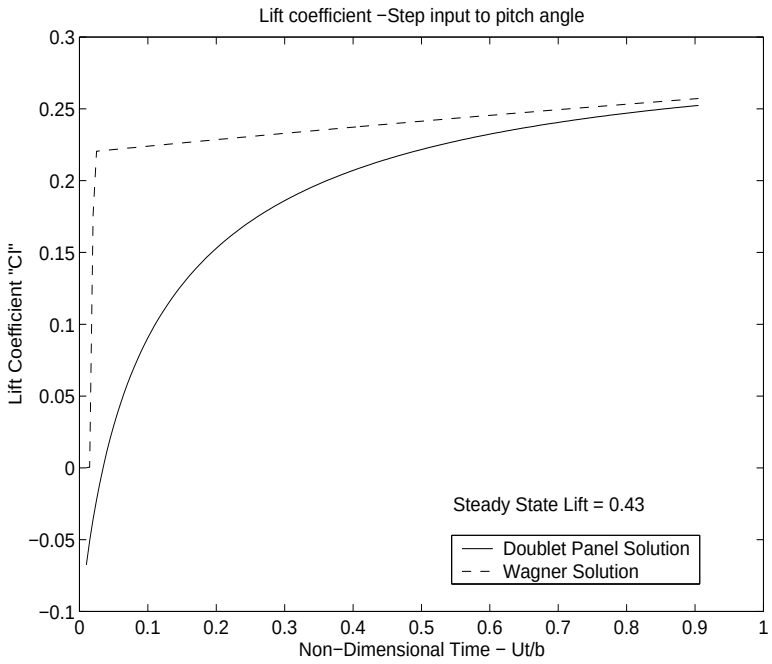


Figure 5.3: **Response of the Doublet Panel Code to an Impulsive Change in Angle of Attack and compared with the theoretical Wagner solution**

numerically without manipulating the trailing edge panel. However, the trailing edge coefficient of pressure is still finite and close to unity. Even though the trailing edge creates some numerical problems, the code is able to replicate the steady state pressure.

5.2 Unsteady Aerodynamics

The theoretical solution to the unsteady aerodynamic lift and pitching moment on an airfoil in inviscid flow is calculated using convolution integrals of the pitch rate with the Wagner function[13]. Physically, the Wagner function is a measure of the circulatory lift on the airfoil due to a unit step

change in angle of attack. It is given as

$$\mathbf{\Phi(s) = \frac{1}{2\pi i}\int_{-\infty}^{\infty} \frac{C(k)}{k} e^{iks} dk} \tag{5.7}$$

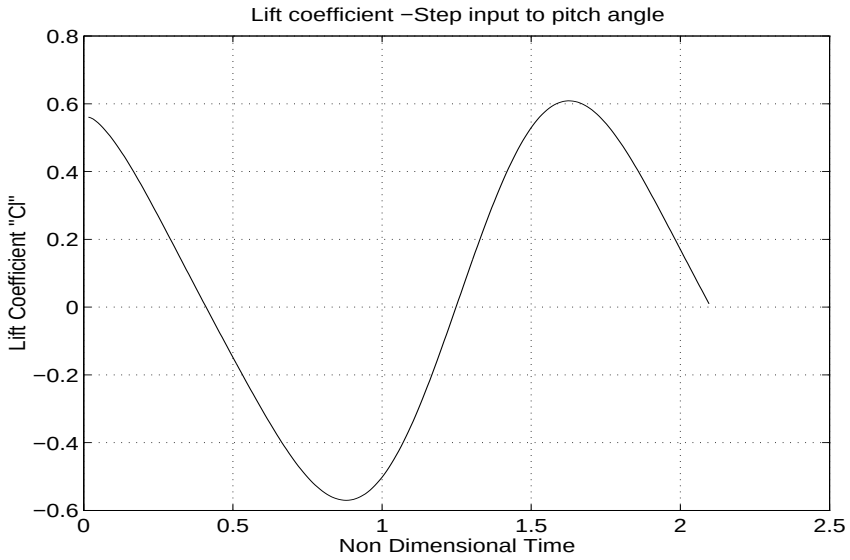
Here k is the reduced frequency, s is the non-dimensional time, $C(k)$ is the Theodorsen function which was explained in the Introduction. Reference 13 provides more details of the Theodorsen and Wagner functions. The Wagner function has a value of 0.5 just after the impulse has been imparted and then grows to a steady state value of 1 for infinite time.

The steady state panel code as given in Appendix B needs to be modified by introducing a time stepping procedure. At each time step, one doublet panel is added to the wake. Initially when the flow starts, that is at time, $t = 0$, there is no wake. For each time step, the wake potential is changed by the addition of one doublet panel. This growing wake contributes to the circulation over the airfoil. The effect of the wake is captured by using the known potential of the wake doublets in the no-penetration boundary condition. For simulating an impulsive angle of attack, the airfoil is at zero incidence to the free stream at initial time. At the first time step when the first wake panel is shed, the airfoil suddenly acquires an angle of incidence. The growth of circulation over the airfoil is captured by the wake shedding process. The effect of this is shown in Fig. 5.3.

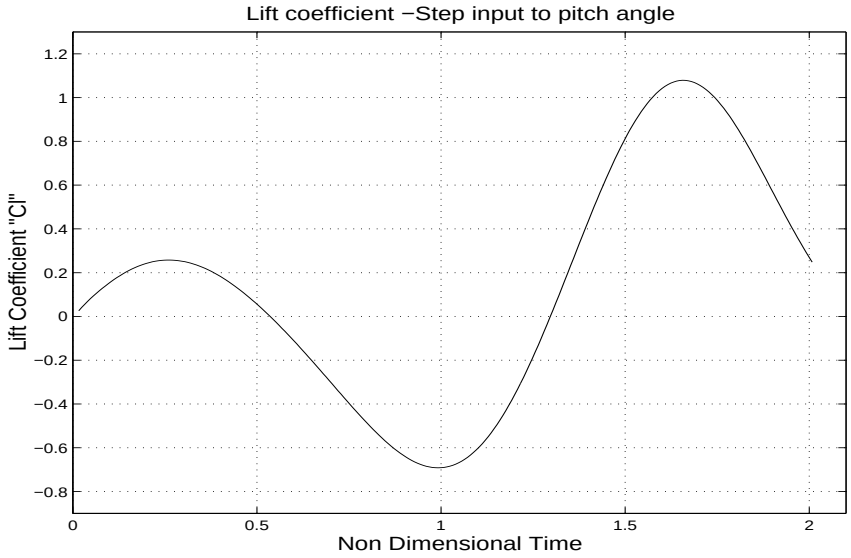
Figure 5.3 shows the growth of the lift with time as predicted by the present code. It can be seen that the solution approaches the Wagner solution. However, the initial impulsive start that is obtained using the theoretical result based on the Wagner function is not attainable by the doublet panel code. This is because the doublet panel code is a time domain solution. The Wagner function is derived in frequency domain and hence it is able to represent the impulsive jump in the lift

at zero time exactly. A time domain code on the other hand takes a finite time, no matter how small, to approximate this jump in lift. However, it should be noted that both solutions approach the same steady state solution. The response shown in Fig 5.3 is similar to that obtained by Basu and Hancock[72] for a symmetric Von-Mises airfoil. Basu and Hancock used a panel scheme consisting of distributed vortex and sources panels over the airfoil and wake. The authors[72] make no explanation for the discrepancy between the theory and the panel code, but simply state that" It can be seen that there are significant differences among various solutions up to $\frac{tU_\infty}{c} = 2.0$". Since in the time domain, an impulse is provided over a time step, the response of the system as shown in Fig. 5.3 will change depending on the time step size. Further, theoretically, the angular velocity of the airfoil is infinite at zero time. A time domain result will result in a finite velocity and finite inertial effects at the first time step. The time step used in the time stepping scheme determines the vortex shedding frequency into the wake. Hence it is required to select an optimum time step that captures the effect of the sudden change in angle of attack on the aerodynamics and the subsequent growth of circulation. The effect due to the finite angle trailing edge and the consequent wake modeling are very important in determining the rate of change of circulation.

This doublet panel method can also represent the unsteady aerodynamics over a pitching airfoil. The airfoil is given sinusoidal pitching amplitudes and the aerodynamic loads on the airfoil are calculated using the time stepping panel method. Figure 5.4(a) shows the unsteady lift over a symmetric airfoil with a constant amplitude sinusoidal pitching motion. Figure 5.4(b) shows the unsteady lift over a symmetric airfoil with a variable amplitude pitching motion. The theoretical solutions for both the problems in Fig. (5.4) are obtained using a convolution of the sinusoidal



a) Pitching airfoil with angle of attack, $\alpha(t) = 0.1\ sin(2t)$



b) Pitching airfoil with angle of attack, $\alpha(t) = 0.1t\ sin(2t)$

Figure 5.4: **Unsteady coefficient of lift over the airfoil as computed by the doublet panel method when the airfoil is placed into pitching motion with two different sinusoidal oscillations**

pitch rate with the Wagner function. It was seen previously (Fig. 5.3), that the time domain solution does not provide the same response to an impulsive change in angle of attack as computed from the theoretical Wagner solution. Therefore the convolution of the sinusoidal functions with the Wagner response will not exactly match the time domain solution obtained from the doublet panel code for a sinusoidally pitching airfoil but must still be similar.

5.3 Numerical Intricacies

The panel method of computing aerodynamic loads involves a large number of numerical computations. These computations are performed in Matlab. Appendix B provides the Panel code for steady and unsteady aerodynamic load calculations using distributed doublet panels. The following points are very useful when implementing panel methods and conform to the experience of the author.

1. The fluid flow over an airfoil is extremely sensitive to the shape of the airfoil. Using panel methods, the software code sees the shape of the airfoil through the effect of the distributed doublets. Hence sufficient number of doublet panels need to be distributed over the airfoil to capture the airfoil shape. According to Eqs. (5.3) and (5.4), the velocity components of the influence of the doublets on the free stream are a function of the doublet panel length. Very small panel lengths means large amount of doublet panels. This can lead to numerical inconsistencies. Hence an optimal number of panels that describe the airfoil shape but do not contribute to numerical errors needs to be used. In the present work, 80-120 doublet panels

were distributed over the airfoil.

2. Another source of numerical ill-conditioning is the trailing edge angle. The doublet panels on the top and bottom surface of the airfoil are very close to each other at the trailing edge. This distance again comes into picture in Eqs. (5.3) and (5.4). Trailing edge angles below 6^0 were found to produce a lot of numerical fluctuations in the aerodynamic pressure. The present work used a Van-De-Vooren airfoil with a trailing edge angle of 10^0.

3. The Kutta condition requires that the flow velocity vanishes at the trailing edge for a finite trailing edge angle. This is however, numerically difficult to achieve. In literature, it has been suggested that modifying the trailing edge panel co-ordinates by very small amounts[73] can enforce the Kutta condition better. The type of numerical Kutta condition used is very significant. In the present work, the trailing edge flow was in the direction that bisected the trailing edge. More than the position of the trailing edge, it was found that taking smaller panel sizes at the trailing edge helped enforce the Kutta condition. Also as mentioned before, the trailing edge angle needs to be carefully determined.

4. The time step size determines the wake shedding frequency. Hence care needs to be taken as to the exact time size. Also usage of the non-dimensional time as $T = \frac{t\,b}{u_\infty}$ is advantageous. Here b is the semi chord and t is the time in seconds. The theoretical equations for calculation of the unsteady aerodynamic forces require evaluation of convolution integrals such as

   $\int_0^t \Phi(t-\tau)\,\dot{\alpha(\tau)}dt$

   To evaluate such integrals numerically, the best procedure is to use Fourier transforms. Also

numerical differentiation can lead to a lot of numerical instabilities. It is better to avoid numerical differentiation wherever possible.

5. For unsteady flow, the modeling of the wake is critical to the development of the unsteady lift. Using panel methods, one panel is added to the wake at every time step. This gradually builds up the wake potential with the passage of time. An obvious question is what should be the length of each wake panel. Since the wake moves at the free stream velocity, a first assumption would be that the length of each wake panel is $U_\infty \, \Delta t$. However, this is false as can be readily assessed from the fact that a large free stream velocity implies a large wake panel. In implementation, the wake panel length should be of the same order as the length of the panels comprising the airfoil. Using non-dimensional time and also non-dimensionalizing the length of the panels with respect to the airfoil chord allows for the selection of the optimum wake panel length .

5.4 Dynamic Aeroelasticity

The panel method described in the previous sections is applied now to a one dimensional aeroelastic system, that is a system with only angular displacements in pitching motion. Accordingly the governing equation of this system will be

$$\ddot{\alpha} + \mathbf{k} \, \alpha = \mathbf{M_p(t)} \tag{5.8}$$

where α is the airfoil pitching angle, $M_p(t)$ is the unsteady pitching moment calculated using panel methods. This system will theoretically show no instabilities, but we use this system to test

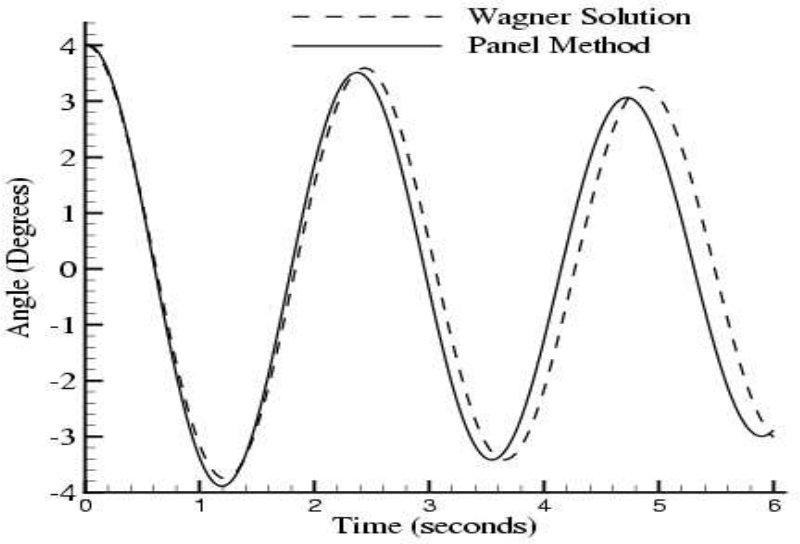


Figure 5.5: **Comparison between the Wagner solution and the panel code solution for the 1-D.O.F. pitching airfoil** $\ddot{\alpha} + 5\,\alpha = M(t)$

the panel method routine and also determine any differences between the panel method solution and the theoretical Wagner solution. As noted in the earlier section on unsteady aerodynamics, the theoretical solution treats the airfoil as a flat plate and hence its solution will not match the panel method calculation exactly. This fact is also noted in Ref. 73. However the two solutions should still be near similar for thin airfoils. The theoretical result is computed using Laplace Transformation of the governing differential equation, Eq. 5.8, with right hand side replaced by the Wagner solution for the unsteady pitching moment. The Wagner solution is given by

$$L(t) = \pi\rho b^2(\ddot{h} + U\,\dot{\alpha} - ba\ddot{\alpha}) - 2\pi\rho b\,U\Big(\Phi(s)w_a(0)+ \int_0^s \Phi(s-\tau)\dot{w_a}d\tau\Big)$$

$$M(t) = \pi\rho b^2(ba\ddot{h} - b(0.5-a)U\,\dot{\alpha} - b(\frac{1}{8}+a^2)\ddot{\alpha}) - 2\pi\rho b^2\,(0.5+a)U\Big(\Phi(s)w_a(0)+ \int_0^s \Phi(s-\tau)\dot{w_a}d\tau\Big)$$

$$s = \frac{Ut}{b}$$

$$\Phi(s) = 1 - 0.165e^{-0.0455s} - 0.335e^{-0.3s} \quad (5.9)$$

where $L(t)$ and $M(t)$ are the lift and pitching moment, α is the pitch angle, w_a is the downwash velocity and h is the plunge displacement. $\Phi(s)$ is the Wagner function which is evaluated using the approximate expression since an exact analytical expression is not possible. The numerical solution couples the derived doublet panel method with a 4^{th} order Runge-Kutta solver to obtain the response of the system described by Eq. 5.8. The stiffness is constant at 5 units. Figure 5.5 describes the result of this comparison. The response of the numerical panel code which is coupled with the Runge-Kutta solver to solve the system given in Eq. (5.8) is seen in Fig. 5.5 to be very similar to the theoretical solution to Eq. (5.8) as computed by using the Wagner solution (Eq. 5.9) in Eq. (5.8). Thus the numerical aeroelastic solver is verifiable with the theoretical solution.

For a two dimensional aeroelastic system, we shall add a plunge displacement also to the system described by Eq. (5.10). In this aeroelastic analysis, the concept of a time-varying torsional stiffness is analyzed as a possible means of flutter suppression. The equation of motion for such a 2-D

aeroelastic system will be

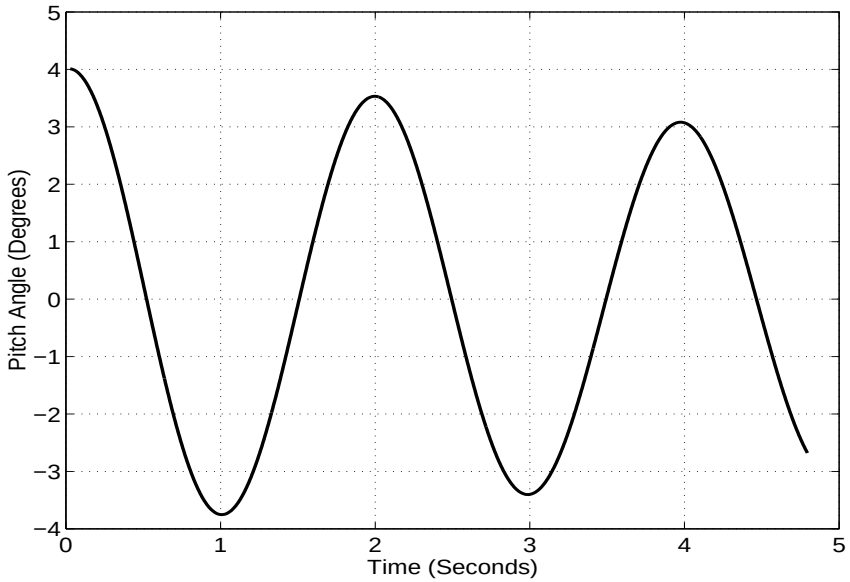
$$\mathbf{m}\ddot{\mathbf{h}} + \mathbf{s}_\alpha \ddot{\alpha} + \mathbf{k_h h} = -\mathbf{L_p(t)} \tag{5.10}$$

$$\mathbf{I}\ddot{\alpha} + \mathbf{s}_\alpha \ddot{\mathbf{h}} + \mathbf{k}_\alpha \mathbf{(t)}\alpha = \mathbf{M_p(t)}$$

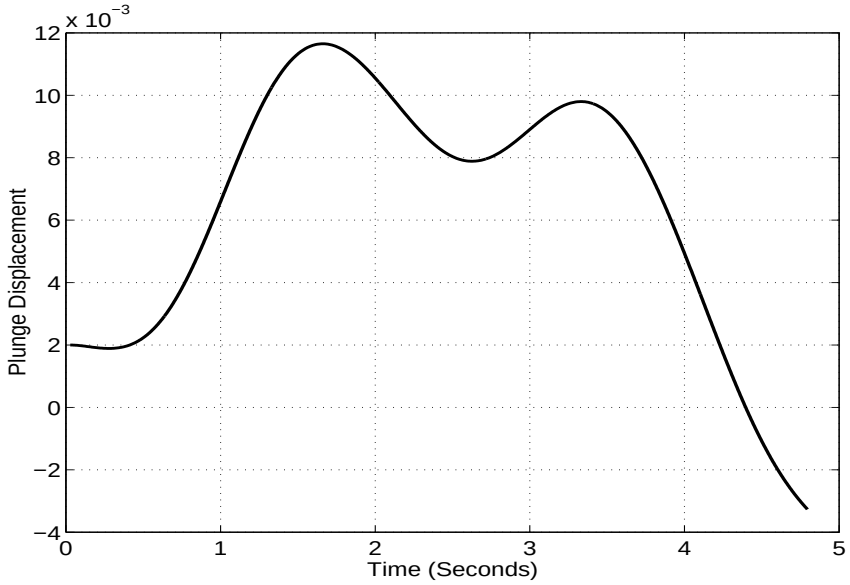
where the time varying torsional stiffness, $k_\alpha(t)$ is given as

$$\mathbf{k}_\alpha \mathbf{(t)} = \mathbf{k_0} + \mathbf{k_{01}}\ \mathbf{(1 - c_1\ e^{-c_2 t})} \tag{5.11}$$

Here in Eq. 5.10, $L_p(t)$ is the unsteady lift calculated using the panel method and $M_p(t)$ is the unsteady pitching moment calculated using the panel method. The method of analysis introduced in Chapter 2 regarding the matrix exponential time marching approach to solve linear time varying systems will be used herein. Accordingly this technique will be used to train neural networks to represent the dynamic aeroelastic response given by Eq. (5.10). As was noted in Chapter 3, recurrent neural networks can predict erroneous results especially for time varying systems. This was also observed in Fig. 3.2. Herein, using the method of matrix exponential time marching a feed forward network will be designed to predict the dynamic aeroelastic response of the 2-D time varying system given by Eq. (5.10).



a) Angular displacement of the airfoil



b) Vertical displacement of the airfoil

Figure 5.6: **Solution to the equation** $5\ddot{h} + 0.05\ddot{\alpha} + 0.25h = -L(t), 0.53I\ddot{\alpha} + 0.05\ddot{h} + 5\alpha = M(t)$ **using the time domain numerical code consisting of the Matrix Exponential time marching method coupled with the doublet panel method**

5.5 Neural Network Training to Represent Dynamic Aeroelasticity

The training of the neural network is performed using the Levenberg-Marquardt scheme that is available with the MATLAB software. A description of this technique was presented in Chapter 3 under the section titled "Neural Network Training". In order to use the concept of Matrix Exponential time marching that was explained in Chapter 2 to describe the time varying aeroelastic system, 3 neural networks are required to be trained in the manner described below. Therefore the number of neural networks is governed by the principal constituents of the matrix exponential time marching method.

1. ANN(1) Input - $\begin{pmatrix} t_i \\ t_{i+1} \\ k_\alpha(t_{i+1}) \end{pmatrix}$, Output - $[\mu]$

2. ANN(2) Input - $\begin{pmatrix} t_i \\ t_{i+1} \\ u_\infty \\ h \\ \dot{h} \\ \alpha \\ \dot{\alpha} \end{pmatrix}$, Output - $\begin{pmatrix} c_{l_i} \\ c_{l_{i+1}} \\ c_{m_i} \\ c_{m_{i+1}} \end{pmatrix}$

3. ANN(3) Input - $\begin{pmatrix} t_i \\ t_{i+1} \\ k_\alpha(t_{i+1}) \\ c_{l_i} \\ c_{l_{i+1}} \\ c_{m_i} \\ c_{m_{i+1}} \end{pmatrix}$ Output - $\{\lambda\}$

Here, the first network, ANN(1), describes the state transition matrix, μ of the time varying system numerically for the desired time interval, t_i, t_{i+1}. The next neural network, ANN(2), describes the non-conservative unsteady aerodynamic loading over the airfoil for the time interval, t_i, t_{i+1}. The last neural network, ANN(3) described the forced response, λ of the aeroelastic system at the time instant t_{i+1} given the initial conditions at time t_i and the outputs of the networks, ANN(1), ANN(2).

All three neural networks use 3 layers of neurons. The input and output layers have a linear transfer function. The hidden layer has a Hyperbolic tangent transfer function. Since the matrix exponential time marching approach to solve linear time variant systems is extremely accurate even with large time steps of the order of 0.1 seconds, the data sets required for training the neural networks over a given time duration can be reduced as compared to recurrent neural networks described in Chapter 3. This is very significant for training dynamic neural networks since large data sets can cause improper convergence of the training routine. However, using the present method, feed-forward neural networks can be used with a control over the size of the data sets required for training.

5.6 Aeroelastic Computations

The plunging motion of the airfoil also adds to the downwash velocity of the flow over the airfoil. According to Eq. (5.4) and (5.6), this effect of the added downwash, w due to the plunging motion changes the doublet potential. Figure 5.4 depicts the lift coefficient and pitching moment coefficient over an airfoil that is undergoing sinusoidal oscillations in pitch and plunge. This data is close to the theoretical solution. The unsteady aerodynamic loads from the doublet panel code can now be coupled with a structural dynamics solver based on the method of Matrix Exponential time marching to represent an aeroelastic solver. Figure 5.6 displays the aeroelastic solution to a constant stiffness 2-D system with the plunge stiffness, $k_h = 0.25$ units and the torsional stiffness, $k_\alpha = 5$ units.

Hence the variable stiffness system will be used at the flutter boundary to retrieve the unstable aeroelastic system back into the stable boundary. This analysis will be performed using the trained neural networks in order to provide a rapid aeroelastic solver that can be used as flutter suppression system. As mentioned earlier, the Levenberg-Marquardt routine in the MATLAB toolbox for neural networks is used to train the three neural networks titled ANN1, ANN2, ANN3 to represent the dynamic aeroelastic system. The networks are designed with respect to both a fixed system stiffness and a varying torsional stiffness. The variation in the torsional stiffness is according As can be seen, the system is stable, meaning the flutter boundary has not been crossed. Therefore till this flutter boundary, the constant stiffness system can be used. A neural network system will be developed which shall detect this onset of instability and then use a variable torsional stiffness to

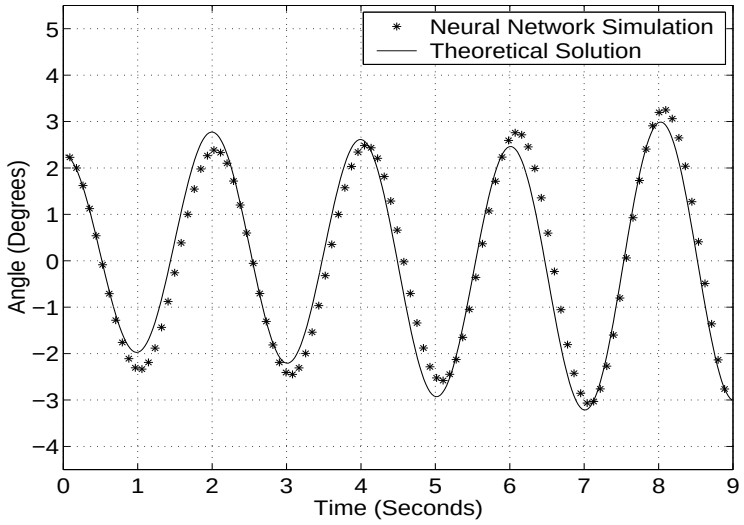


Figure 5.7: **Testing the array of neural networks trained using the method of Matrix Exponential Time Marching for obtaining the response of the constant stiffness aeroelastic system as described by Eq. 5.10 at the onset of flutter and comparing this response with the theoretical solution**

regain stability of motion. to Eq. (5.11).

Having trained, the network with a set of stiffness parameters at various free stream velocities, we test the ANN at different system stiffnesses. Since we have trained 3 different ANN units separately, the combination of the 3 units needs to be tested to see if the true aeroelastic response is achievable. Accordingly, a state of flutter is looked at. Considering the constant stiffness system of Fig. 5.6 but in a state of flutter, the trained neural networks are presented with its system parameters and the initial conditions with the flow data. The response is compared with the theoretical solution. This is shown in Fig. 5.7. As can be seen, the neural networks represent the numerical solution very well. However, for a varying stiffness system, since it is not possible to train the neural

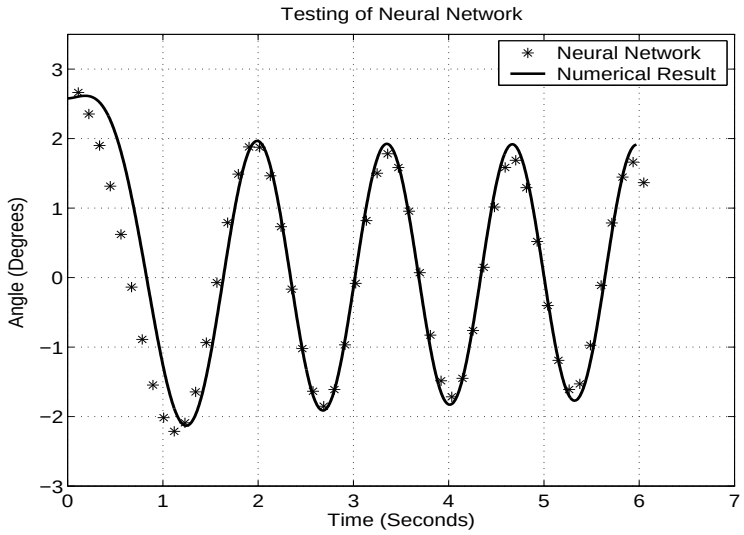


Figure 5.8: **Testing of the neural network array for the aeroelastic system described by Eq. 5.10 with a variable torsional stiffness given by** $k = 5 + 12\,(1 - 1.33e^{-t})$

network system for all possible stiffness variations, testing the neural networks for a different stiffness variation is necessary. Accordingly, during the training process, Eq. (5.11) uses certain constants, c_1 and c_2 and during testing, the constants will be changed. This tests the fidelity of the system to different system parameters. If the testing is successful, then it also establishes an adaptive aeroelastic controller wherein, stability in oscillations can be obtained using a variable stiffness system.

The ANN array will be tested for a variable stiffness aeroelastic system. Accordingly in Eq. (5.11), we consider $c_1 = 1.33$, $c_2 = 1$, $k_0 = 5$, $k_{01} = 12$ for the testing of the ANNs'. Note that, the parameters c_1 and c_2 different from those used for the training routine. Figure 5.8 displays the result of the testing. As we can see, the aeroelastic system as represented by neural networks does show some differences from the numerical aeroelastic system, but these errors reduce with time.

Moreover, the overall behavior of the neural network system follows the numerical solver quite accurately. Hence we conclude that the neural network system represented by three networks, ANN1, ANN2, ANN3 as described earlier can be used to describe the aeroelastic system given by Eq. 5.10.

The neural network is tested to see if the flutter reduced frequency that it predicts is close to the theoretical Theodorsen result. The flutter reduced frequency as predicted by the neural network solver for the constant stiffness system with parameters as $k_h = 0.25, k_\alpha = 5, m = 5, I = 0.53, s_\alpha = 0.05$ is 0.87 when the 2-D airfoil is given an initial pitch displacement of 2^0 angle of attack. Now a Theodorsen type Flutter analysis is done on this system. The Theodorsen equations for flutter are listed in Ref. 13. Using these equations, the Theodorsen flutter reduced frequency was calculated for the same system as represented by the neural network. The Theodorsen reduced frequency at flutter is 0.79. Hence the neural network is very accurate in calculating the Theodorsen flutter frequency and the percentage error between the neural network prediction and the theoretical result is 10.1%.

5.7 Flutter Suppression

This array of trained neural networks now provides an adaptive representation of the aeroelastic system with time varying stiffness. The representation is said to be adaptive because the neural network is able to respond correctly to a range of torsional variations. Each time the stiffness variation rate or magnitude changes, the numerical solver need not be used. On the other hand,

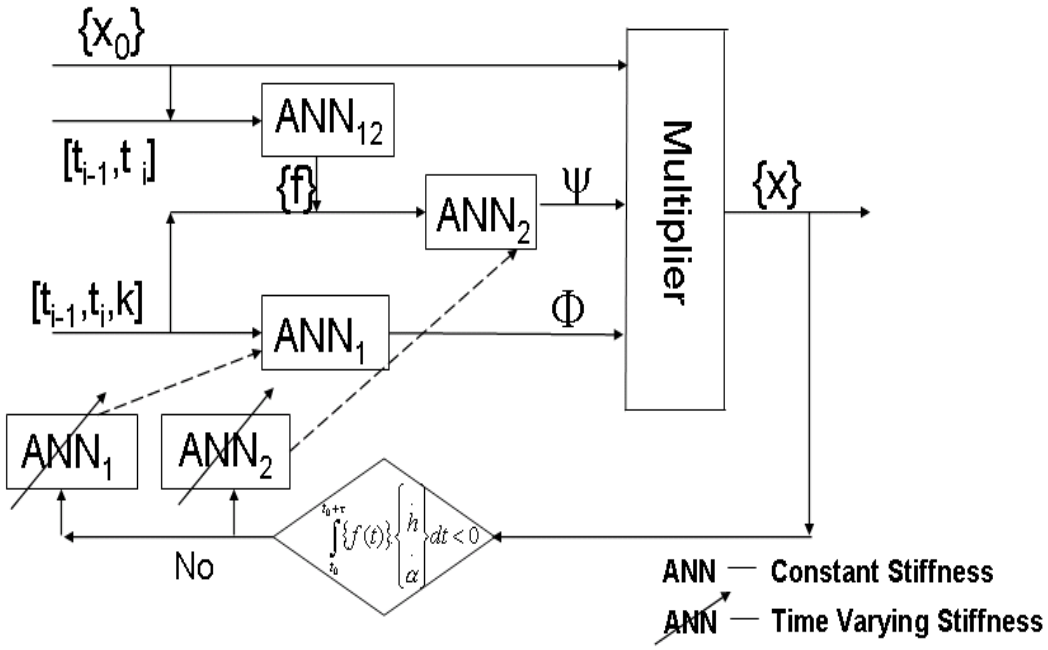


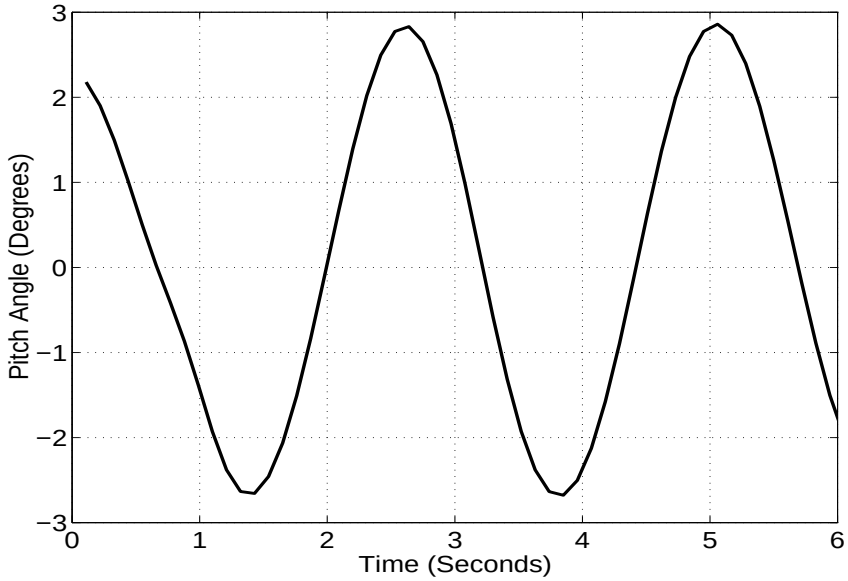
Figure 5.9: **Array of neural networks for calculating the dynamic aeroelastic response of a 2-D aeroelastic system. The array shows both the constant stiffness network and the variable torsional stiffness network for flutter suppression**

one can use the neural network system. This allows for the design of a flutter suppression toolbox. Consider the diagram in Fig. 5.9. This shows the three designed neural networks linked together to describe the output of the aeroelastic system. A feedback loop is also shown which evaluates the stability of the system. The stability is based on the response of the system during one period of oscillation. In the state space plot, if $|x^2 + \dot{x}^2|_{t+T} > |x^2 + \dot{x}^2|_t$, the system is unstable. Here, the period of oscillation is T and the stability is calculated at a time step $t + T$. Since this is a linear system, this approach is valid. In general, the procedure to detect flutter is to calculate the work done over a period of oscillation by the external non-conservative forces. Hence the system is at

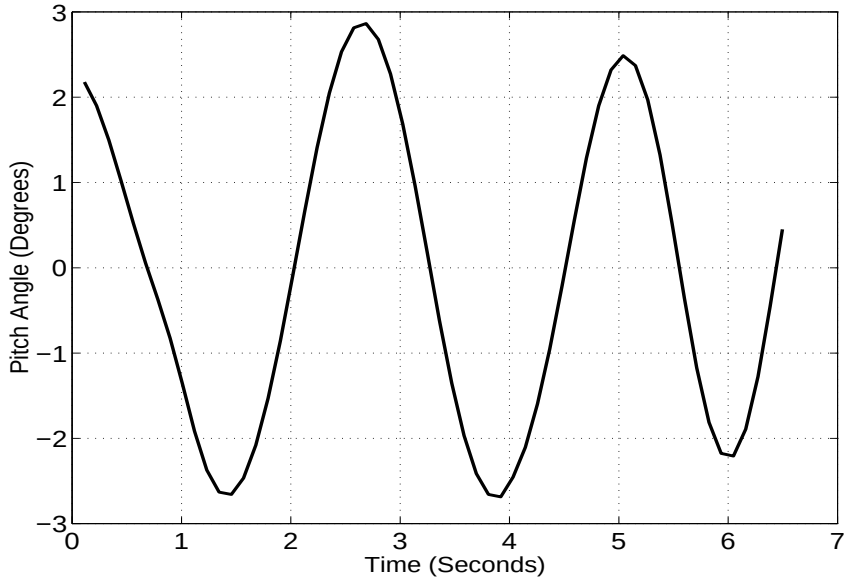
the flutter boundary if

$$\triangle \mathbf{W_{nc}} = \int_{\mathbf{t}}^{\mathbf{t+T}} \{\mathbf{f(t)}\}^{\mathbf{T}}.\{\dot{\mathbf{x}}\}\mathbf{dt} = \mathbf{0} \tag{5.12}$$

where $f(t)$ is the no-conservative aerodynamic loads and $\dot{x}$ is the vector containing the system velocities. If instability is detected by the solver over one period of oscillation, the neural networks for representing the structural response, that is , ANN(1) and ANN(3) are switched over to the adaptive mode wherein they can represent a time varying torsional stiffness according to Eq. 5.10. In Fig. (5.10), the initial dynamic aeroelastic response is calculated using the constant stiffness structural neural networks, ANN1 and ANN3. Once the system is tested to be unstable, the variable stiffness adaptive neural networks are switched on for the structural dynamics so as to bring the system again within the stability boundary. To achieve this, the neural network array that was tested and developed in Fig. (5.8) with different stiffness variations is used. The reduced frequency of the aeroelastic system is set to 1.1 which is just over the flutter reduced frequency for this system. It can be seen from Fig. 5.10(a) that the oscillations are unstable. Based on the procedure explained in Fig. 5.9, the neural network controller detects the unstable system after the first period and then switches over to the variable stiffness mode. The stiffness is now increased according to Eq. (5.11) till stability of oscillations is achieved. This is shown in Fig. 5.10(b) which depicts a stable oscillatory pattern for the time history of the airfoil's pitching angle. For Fig. 10(b), the stiffness variation that was used was $k_\alpha(t) = 5 + 12\ (1 - 1.33e^{-t})$. This was found to be sufficient to restore stability within one period of oscillation of the aeroelastic system. The computational time required for the neural network simulation is extremely small. For example the 5 seconds of neural network simulation shown in Fig. 5.10(a) was obtained within a few seconds on an



a) Unstable Pitching Oscillations



b) Flutter control by varying stiffness method

Figure 5.10: **Flutter suppression of a 2-D aeroelastic system governed by Eq. 5.10 using a variable torsional stiffness spring $k = 5 + 12\,(1 - 1.33e^{-t})$ with trained neural networks based on the approach of Matrix Exponential Time Marching**

SGI Origin machine using 1 processor in Matlab. Whereas the numerical aeroelastic routine coupling the structural dynamics solver with the doublet panel method takes almost 4 hours for the same response calculation. Since the neural network array allows such a rapid simulation, the controller can directly calculate the response to detect instability and prevent it using the variable stiffness approach.

5.8 Effectiveness of Neural Networks to Represent Dynamic Aeroelasticity

The Dynamic aeroelastic behavior of an airfoil was calculated using a time domain numerical code that coupled a panel method using distributed doublets with a time marching structural dynamics code. These results were then used in the design of artificial neural networks to capture the dynamic aeroelastic response. Three neural networks were trained separately, each representing one constituent of the aeroelastic behavior of the system. When these three neural networks were inter-connected, the dynamic aeroelastic response of the airfoil with a time-varying stiffness was obtained. The neural network architecture allows the representation of the fluid-structure coupling accurately and rapidly. The time step used in this neural network simulation was the order of 0.05 seconds which is an order of magnitude greater than the time step required by the numerical doublet panel code. This neural network representation is valid for any two dimensional time-varying linear aeroelastic system. The errors in the neural network approximation of the numerical result were shown to be within a 10% range. The Theodorsen flutter reduced frequency was also

within 11% of the neural network prediction of the flutter reduced frequency. The method of Matrix Exponential time marching upon which the neural network architecture is based provides for an accurate training scheme and hence the simulation of the neural network for different system stiffness parameters is accurate.

Chapter 6

Conclusions

Methods were developed based on trained neural networks to solve static and dynamic aeroelastic problems associated with morphing wing structures. The static aeroelastic problem that was studied was the shape control of a deforming bump on an airfoil surface. Such a static aeroelastic problem represents a fundamental aeroelastic problem in morphing wing theory, that is the deformation of a surface in a flow field. Here, this aeroelastic problem was combined with the concept of lateral dimensional moment generation for tailless aircraft maneuvering. This analysis required the use of CFD, since the control of flow separation played a major part in the design. The structural analysis was performed using the non-linear finite element solver package present in the software ANSYS. A neural network based approach for aeroelastic optimization of bumps on airfoils was demonstrated. The shape of the bumps was designed for maximum drag with minimum loss in lift and minimum strain energy of actuation. This design goal was based on the requirement for yaw moment generation caused by increased drag over one wing. The analysis was done at two free

stream velocities of Mach 0.3 and 0.7. For the low speed case, the bump was situated at the leading edge whereas for the high speed case, the bump was moved downstream. Two Neural networks were trained, one to represent the aerodynamic loading over the bump and the other to represent the structural deformation and actuator loading. An optimization scheme was devised based on the conjugate gradient scheme and using these trained neural networks. The results of the optimization showed the best bump designs to realize the objective and these results were verified with CFD and the finite element solver. For low speeds with free stream velocity of Mach number 0.3, a drag coefficient rise of 0.015 was realizable with a lift loss of the order of 12%. For transonic free stream speeds, initial bump displacements contribute to wave drag reduction and hence the overall drag over the airfoil is reduced. As the bump size is increased, the pressure drag rise is greater than the reduction in wave drag. The optimized bump shapes showed a coefficient of drag rise of 0.01 with a loss in lift of the order of 9%. The computed values of drag and lift from the neural network were within 10% of the corresponding CFD result. The strain energy of deformation as computed by the neural network for the optimized results was approximately 16% away from the ANSYS result.

The utilization of neural networks for dynamic aeroelastic studies was next explored. Calculation of the dynamic response requires the correct representation of the state variables of the concerned system at every time instant of interest. The system considered was a two dimensional aeroelastic system with a time-varying torsional stiffness. The concept of a variable stiffness wing has attracted a lot of attention recently and hence the investigation performed here utilized this concept of a time varying stiffness in the analysis of the dynamic aeroelastic equations of equilibrium. Ini-

tial attempts at using recurrent neural networks to represent such a time varying system did not prove to be computationally efficient since they required small time steps and were very sensitive to the training procedure. Hence an efficient and accurate time marching scheme was sought to obtain the response of linear time varying systems. Using this time-marching scheme, feed-forward neural networks could then be trained to represent the dynamic aeroelastic problem.

A scheme termed as Matrix Exponential Time Marching was devised for solving any linearly time variant system. This scheme used an analytical solution in the form of a matrix exponential as the state transition matrix within a certain time interval for single degree of freedom systems. Using this analytical solution, a time marching procedure was developed whereby the response of the linear time variant system could be determined for any time duration. This scheme was tested using many examples and was found to be extremely accurate even with large time steps. This scheme allowed the representation of the dynamic aeroelastic problem by three feed-forward neural networks. Using this scheme of Matrix Exponential time marching, these three neural networks were trained to represent the state of the system at any time interval. In order to train the neural networks to represent the unsteady aerodynamic loading over the airfoil, a panel method based on distributed doublets was developed. This panel method also used a time marching scheme to represent the unsteady circulation over the airfoil.

The training of the neural network was performed for both a constant stiffness aeroelastic system and a variable torsional stiffness system. Once trained, this array of neural networks was used as a flutter suppression system. Here the response of the constant stiffness system as simulated by the neural networks is tested for stability. If an instability is detected over a period of oscillation of

the airfoil, then the variable stiffness neural network array switches on, thereby using the concept of an increasing torsional stiffness to bring the aeroelastic system back into the stable domain of the response. This is achieved computationally. Practical experiments on the development of a variable stiffness wing need to be investigated.

In conclusion, this dissertation studied two novel aeroelastic problems in morphing wing aircraft using neural networks. Both static and dynamic aeroelastic problems that can be applied to morphing wing aircraft were analyzed. This analysis showed the efficiency of neural networks in representing linear and non-linear aeroelastic system. The major contributions of this work are in the representation of linear and non-linear aeroelastic response through the use of neural networks, its application to morphing wings and the efficient computation of solutions to arbitrarily time varying linear dynamical systems.

Bibliography

[1] Inman, D.J., Gern, F.H., Robertshaw, H.H., Kapania, R.K., Pettit, G., Natarajan, A. and Sulaeman, E., "Comments on Prospects of Fully Adaptive Wings", 8th International Conference on Smart Structures and Materials, Newport Beach, CA, March 5-8 2001.

[2] Pinkerton, J.L. and Moses, R.W., "A Feasibility Study to control Airfoil Shape Using THUNDER" NASA TM 4767, Nov. 1997.

[3] Kornbluh, R., R. Pelrine, Q. Pei, S. Oh, and J. Joseph, "Ultrahigh Strain Response of Field-Actuated Elastomeric Polymers", *Proceedings of SPIE Smart Materials* April 2000, Newport Beach, California

[4] Torra V., Isalgue A. and Lovey F.C., "Guaranteed behavior in shape memory alloys. Short- and long-time effects related to temperature and phase coexistence", *Journal of Thermal Analysis and Calorimetry*, Vol 66 (1), 2001, pp. 7-16

[5] Jacob, J.D., "On the Fluid Dynamics of Adaptive Airfoils", ASME International Mechanical Engineering Congress and Exposition, Nov.15-20, 1998, Anaheim, CA.

[6] Chen, P.C., Sarhaddi, D., Jha, R., Liu, D.D., Griffin, K. and Yurkowich, R., "Variable Stiffness Spar Approach for Aircraft Maneuver Enhancement Using ASTROS", *Journal of Aircraft*, Vol. 37(5), 2000, pp. 865-871.

[7] Garg D.P., Zikry M.A. and Anderson G.L., "Current and potential future research activities in adaptive structures: an ARO perspective", *Journal of Smart Materials and Structures*, Vol. 10 (4): 2001, pp. 610-623.

[8] Blair, M., "A Compilation of the Mathematics Leading to the Doublet Lattice Method", Report Number AD-A256304, 1992

[9] Sulaeman, E., "Effect of Compressive Force on the Aeroelastic Analysis of a Strut Braced Wing", Ph.D. Dissertation, Dept. of Aerospace and Ocean Engineering, Virginia Tech., Blacksburg, Virginia, 2002.

[10] Tannehil, J.C., Anderson, D.A. and Pletcher, R.H.,*Computational Fluid Mechanics and Heat Transfer,* Second Edition, Taylor and Francis, 1997.

[11] Kral, L.D., "Recent Experience with Different Turbulence Models Applied to the Calculation of Flow over Aircraft Components", *Progress in Aerospace Sciences*, Vol 34, 1998, pp. 481-541.

[12] Poling, D.R. and Telionis, D.P., "The response of airfoils to periodic disturbances - The unsteady Kutta condition", *AIAA Journal*, Vol 24., 1986, pp. 193-199.

[13] Bisplinghoff, R.L., Ashley, H. and Halfman R.L., *Aeroelasticity*, Dover Publications, 1996.

[14] Dowell, E.H. (Editor), Crawley, E.F., Curtiss, H.C., Peters, D.A., Scanlan, R.H. and Fernando, S., *A Modern Course in Aeroelasticity*, Kluwer Academic Publishers 1995.

[15] Nayfeh, A.H. and Balachandran. B., *Applied Nonlinear Dynamics: Analytical, Computational, and Experimental Methods*, Wiley-Interscience, NY, Jan 1995

[16] Khot N.S., Eastep F.E. and Kolonay R.M., "Method for enhancement of the rolling maneuver of a flexible wing", *Journal of Aircraft*, Vol 34 (5), 1997, pp. 673-678.

[17] Ramakrishnananda, B. and Wong, K.C., "Animating bird flight using aerodynamics" *The Visual Computer*, Vol 15 (10), 1999, pp. 494-508.

[18] Dhawan, S., "Bird Flight", *Sadhana-Academy Proceedings In Engineering Sciences*, Vol 16, Part 4., 1991, pp. 275-352.

[19] Ashkenas, L.I. and Klyde, D.H., "Tailless Aircraft Performance Improvements with Relaxed Static Stability," NASA CR 181806, March 1989.

[20] Gern, F.H.,Inman, D.J. and Kapania, R.K., "Structural and Aeroelastic Modeling of General Planform UCAV Wings with Morphing Airfoils", 42^{nd} *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Seattle, April 2001.

[21] Sommerer, A., Lutz, T. and Wagner, S., "Design of Adaptive Transonic Airfoils by means of Numerical Optimization", ECCOMAS, Barcelona, Sept 2000.

[22] Rathnasingham, R. "System identification and active control of a turbulent boundary layer", Ph.D. Thesis, Dept. of Aeronautics and Astronautics, MIT, May 1997.

[23] Cole, S.R., "Aeroelastic Effects of Spoiler Surfaces on a Low-Aspect Ratio Rectangular Wing", *Journal of Aircraft* Vol 29, No. 5, 1992 pp. 768-773

[24] Suleman, A., Moniz, P.A. and Costa, A.P., "Buffeting Suppression of a Wing Using Adaptive Shape Control", *Canadian Aeronautics and Space Journal*, Vol. 46(2), 2000

[25] Greenblat, D. and Wygnanski, I., "The Control of Flow Separation by Periodic Excitation", *Progress in Aerospace Sciences*, 36, 2000, pp 487-545

[26] Sinha, S.K. "Flow Separation Control with Microflexural Wall Vibrations", *Journal of Aircraft*, Vol. 38(3) 2001.

[27] Carpenter P.W. and Green P.N., ”The aeroacoustics and aerodynamics of high-speed Coanda devices, Part 1: Conventional arrangement of exit nozzle and surface”, *Journal of Sound and Vibration*, Vol. 208 (5), 1997, pp. 777-801.

[28] Silva, W.A., ”Application of Nonlinear-Systems Theory To Transonic Unsteady Aerodynamic Responses”, *Journal of Aircraft*, Vol 30 (5), 1993, pp. 660-668

[29] Raveh, D.E.,”Reduced Order Models for Nonlinear Unsteady Aerodynamics”, *AIAA Journal* Vol 39, No. 8, 2000, pp 1418-1429

[30] Richards, J.A., *Analysis of Periodically Time-Varying Systems*, Springer-Verlag 1983.

[31] Sinha, S.C. and Joseph, P., ”Control of generic dynamic systems with periodically varying parameters via Liapunov-Floquet transformation”, *ASME Journal of Dynamics, Measurements and Control*, 116, 1994, pp. 650-658.

[32] Youn, I. and Hao, A., ”Semi Active Suspensions with Adaptive Capability”,*Journal of Sound and Vibration* 180(3) 1995, pp 475-492,.

[33] Zadeh, L. and Desoer, C., *Linear System Theory, The State Space proach*, McGraw Hill, 1963.

[34] Douglas, J.M., *Process Dynamics and Control, Volume 1,Analysis of Dynamic Systems*, Prentice Hall, 1972.

[35] Hsu C.S., ”On Approximating a General Linear Periodic System”, *Journal of Mathematical Analysis and Applications* Vol 45, 1974, pp. 234-251.

[36] Jones, R.T., ”Wing Theory”, Princeton University Press, 1990.

[37] De Boor, C., ”A Practical Guide to Splines”, Springer-Verlag, New York 1978.

[38] Rogers, D.F, Adams, J.A., ”Mathematical Elements for Computer Graphics”, McGraw Hill 1990.

[39] Farin, G.E., ”Curves and Surfaces for Computer Aided Geometric Design”, Academic Press, Boston 1988.

[40] Madavan, N.K., Rai, M. and Huber, F.W., ”Redesigning gas-generator turbines for improved unsteady aerodynamic performance using neural networks”, *Journal of Propulsion and Power*, Vol. 17, No. 3, 2001, pp. 669-677

[41] Haykin, S., ”Neural Networks, A Comprehensive Foundation”, *Prentice Hall*, Second Edition, 1999

[42] Raisinghani, S.C. and Ghosh, A.K., ”Parameter Estimation of an Aeroelastic Aircraft using Neural Networks”, *Sadhana -Academy Proceedings In Engineering Sciences*, Vol. 25, Part 2, 2000., pp. 181-191.

[43] Scott, R.C. and Pado, L.E., ”Active control of wind-tunnel model aeroelastic response using neural networks”, *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 6, 2000, pp. 1100-1108.

[44] Bernelli-Sazzera, F., Mantegazza, P., Mazzoni, G. and Rendina, M., ”Active flutter suppression using recurrent neural networks”, *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 6, 2000, pp. 1030-1036.

[45] Bennet, D.J, Hollerbach, X.M., Xi, Y. and Hunter, I.W., ”Time-Varying Stiffness of Human Elbow Joint During Cyclic Voluntary Movements”, *Experimental Brain Research*, Vol. 88 (2), 1992 pp. 433-442

[46] Srinivasan, M.A, and R.H. LaMotte, ”Tactual Discrimination of Softness”, Journal of Neurophysics. Vol 73 (1), 1995

[47] Ackleh, A.S. and Fitzpatrick, B.G., ”Estimation of Time Dependent Parameters in General Parabolic Evolution Systems”, *Journal of Mathematical Analysis and Applications*, Vol. 203, 1996, pp. 464-480.

[48] Kotera, T. and Kawai, R., ”Vibrations of Strings with Time-Varying Length-(The Case having a Weight at One End)”, *Transaction of the Japan Society of Mechanical Engineering*,31(3), 1988, pp. 524-529, .

[49] Li, Q.S., ”A New Exact Approach for Analyzing Free Vibration of SDOF Systems with Non-periodically Time varying Parameters”, *Journal of Vibration and Acoustics*, 122(2), 2000, pp. 175-179.

[50] Moler, C. and Van Loan, C., ”Nineteen Dubious Ways to Compute the Exponential of a Matrix”, *SIAM Review* Vol. 20, 1978, pp. 801-836,.

[51] Cheng, H. and Wu, T.T., ”An Aging Spring”, Studies in Applied Mathematics, Vol 49(2), 1970, pp. 183-185 *Mechanical Systems and Signal Processing*, 11(3),pp. 375-390, 1997.

[52] Shahruz, S.M. and Tan, C.M., ”Response of Linear Slowly Varying Systems under External Excitations”, *Journal of Sound and Vibration* 131(2), pp. 239-247, 1989

[53] Amabili, M. and Rivola, A., ”Dynamic Analysis of Spur Gear Pairs : Steady State Response and Stability of the SDOF System with Time Varying Mesh Damping”,

[54] Hopfield, J.H., ”Neural networks and physical systems with emergent collective computational abilities”, Proceedings of the National Academy of Sciences, USA, Vol. 79: 1982, pp. 2554-2558.

[55] Elman, J. L., ”Finding Structure in Time,” *Cognitive Science*, vol. 14, 1990, pp. 179-211.

[56] Kohonen, T., ”Self organized formation of topologically correct feature maps”, *Biological Cybernetics*, Vol 43, pp. 59-69, 1982.

[57] Burden, R.L., and Faires, D.J., "Numerical Analysis", *International Thompson Publishing Company*, Seventh Edition, 1999.

[58] Marzocca, P. and Librescu, L., "Volterra Series Approach for Non-Linear Aeroelastic Response of 2-D Lifting Surfaces", paper # AIAA-2001-1459, 42^{nd} *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Seattle, April 2001.

[59] Mandic, D.P. and Chambers, J.A, "Recurrent Neural Networks for Prediction", *John Wiley and Sons LTD.*, 2001.

[60] J. W. Edwards and J. B. Malone," Current Status of Computational Methods for Transonic Unsteady Aerodynamics and Aeroelastic Applications", *Computing Systems in Engineering*, Vol. 3, No. 5, pp. 545-569, 1992.

[61] Guruswamy, G.P. " A Review of Numerical Fluids/Structures Interface Methods for Computations using High Fidelity Equations", NASA/TM-2000-209596, October 2000.

[62] Jameson A. and Vassberg, J.C., "Computational fluid dynamics for aerodynamic design - Its current and future impact", paper 2001-0538, 39^{th} AIAA Aerospace Sciences Meeting and Exhibit, Reno, Jan. 8-11, 2001.

[63] Bauer, S.X.S. "An Aerodynamic Assessment of Micro-Drag Generators(MDGs)", paper # 98-2621, 16 th AIAA Applied Aerodynamics Conference" June 15-18, 1998.

[64] Raney, D.L. and Park, M.A., "Flight Control Using Distributed Shape-Change Effector Arrays", paper # 2000-1560, 41^{st} AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference , April 3-6 2000.

[65] Choi S.W. and Chang K.S., "Navier-Stokes Computation of a Rapidly Deploying Spoiler", *Journal of Aircraft*, 37(4), 2000, pp. 655-661.

[66] Shih, T.H., Liou, W.W., Shabbir, A. and Zhu, J., "A New k-ϵ Eddy- viscosity Model for High Reynolds Number Turbulent Flows - Model Development and Validation", Computers and Fluids, 24(3), 1995 pp. 227-238.

[67] Liu, Y. and Kapania, R.K., "Equivalent Skin Analysis of Wing Structures using Neural Networks," 8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 6-8, September , 2000. AIAA Paper 2000-4802

[68] Webster, D.R., DeGraaff, D.B. and Eaton, J.K., "Turbulence of a Boundary Layer over a Two-Dimensional Bump", *Journal of Fluid Mechanics* Vol. 320, Aug. 1996, pp. 1061-1068.

[69] Wu, X. and Squires, K.D., "Prediction of the High-Reynolds Number-Flow over a Two-Dimensional Bump", *AIAA Journal* Vol. 36, No. 5, May 1998.

[70] Katz, J. and Plotkin, A., ”Low Speed Aerodynamics, From Wing Theory to Panel Methods”, McGraw Hill 1991.

[71] Moran, J., ”An Introduction to Theoretical and Computational Aerodynamics”, Wiley, 1984.

[72] Basu, B.C., Hancock, G.J., ”The Unsteady Motion of a Two-Dimensional Aerofoil in Incompressible Inviscid Flow”, *Journal of Fluid Mechanics*, Vol 7, p art 1, 1978, pp. 159-178.

[73] Bahbah, M.M. and Maruszewski, B., ”Effect of paneling, application of the Kutta condition and the way of positioning the control points on the numerical solution of panel methods. Part II” CAMES, Vol. 5: pp. 9 -19, 1998.

Appendix A

The matrix exponent,

$$exp(\int_0^t [C(t)]dt) = \begin{pmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{pmatrix}$$

where

$$C(t) = \begin{pmatrix} 0 & -1 \\ c_{21} & c_{22} \end{pmatrix}$$

is the matrix of time varying coefficients of the differential equation Eq. (2.2), that is $\{\dot{x}\} + [C(t)]\{x\} = 0$. The matrices $exp(\int_0^t [C(t)]dt)$ and $C(t)$ must commute , i.e. $e^{\int_0^t [C(t)]dt}\ [C(t)] = [C(t)]\ e^{\int_0^t [C(t)]dt}$ for a differential equation to be expressed in the form of Eq. (2.3). It is well known that is not the case for a matrix with variable elements. However we seek to know how valid is Eq. (2.3) when the matrices involved do not commute. For this purpose, we perform the following operations. Define

$$C(t)\ exp\left(\int_0^t [C(t)]dt\right) = \begin{pmatrix} 0 & -1 \\ c_{21} & c_{22} \end{pmatrix} \begin{pmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{pmatrix} = \begin{pmatrix} -d_{21} & -d_{22} \\ c_{21}d_{11} + c_{22}d_{21} & c_{21}d_{12} + c_{22}d_{22} \end{pmatrix} \quad \text{(A-1)}$$

$$exp\left(\int_0^t [C(t)]dt\right)\ C(t) = \begin{pmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{pmatrix} \begin{pmatrix} 0 & -1 \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} d_{12}c_{21} & -d_{11} + d_{12}c_{22} \\ d_{22}c_{21} & -d_{21} + d_{22}c_{22} \end{pmatrix} \quad \text{(A-2)}$$

Equating Eqs. (A-1) and (A-2), the following relations are obtained,

$$\begin{aligned} d_{21} &= -d_{12}\, c_{21} \\ d_{22} &= d_{11} - d_{12}\, c_{22} \end{aligned} \tag{A-3}$$

Now, we note that, for arbitrarily time-variant matrices, the above relations cannot in general hold and we therefore wish to obtain how approximate is the above equation for the case of time-variant parameters. From Eq. 4, the following relations are obtained,

$$\begin{aligned} d_{21} &= \beta \int_{t_0}^{t} \tilde{c_{21}} dt \\ d_{12} &= \beta\,(t_0 - t) \\ d_{22} &= \beta \left(\int_{t_0}^{t} \frac{\tilde{c_{22}}}{2} dt + \frac{\triangle}{tanh(\triangle)} \right) \\ d_{11} &= \beta \left(\int_{t_0}^{t} -\frac{\tilde{c_{22}}}{2} dt + \frac{\triangle}{tanh(\triangle)} \right) \end{aligned} \tag{A-4}$$

The terms β and $\triangle$ are as explained in Eq. 4. We superscribe c_{22} and c_{21} with a tilde, to indicate that they are arbitrarily time varying. Using the relations Eq. A-4 in Eq. A-3,

$$\begin{aligned} c_{21} &= -\frac{d_{21}}{d_{12}} = \frac{\int_{t_0}^{t} \tilde{c_{21}} dt}{t - t_0} \\ c_{22} &= -\frac{d_{22} - d_{11}}{d_{12}} = \frac{\int_{t_0}^{t} \tilde{c_{22}} dt}{t - t_0} \end{aligned} \tag{A-5}$$

Now, it is readily evident that the terms on the right hand side of Eq. A-5 is the average of the time varying quantity over a time interval $t - t_0$. If the terms c_{21} or c_{22} are time invariant, then the relationship in Eq. A-5 is exact. If the terms c_{21} and c_{22} are time variant, then a suitable time interval $t - t_0$ can be selected which will make the relations in Eq. A-5 near exact.

Appendix B

The MATLAB programs to calculate the steady pressure distribution over an airfoil and the unsteady aerodynamic loads is given below. The airfoil is constructed using B-Splines.

```
%This program calculates the steady pressure distribution over an airfoil.
clear all; close all;
np = input('enter the number of panels ');
alpha = input('enter the angle of attack ');
alpha=alpha*pi/180;
uinf = input('enter free stream velocity ');

% Having collected the data ,establish a Bspline representation and get the panels
v = [0 0 0 1 2 3 4 4 4];
for j=1:np,
[x, z] = Vooren(v,np,j);
if j == 1
panelx = x;
panelz= z;
else
panelx=cat(2,panelx,x);
panelz=cat(2,panelz,z);
end
end


lx = min(size(panelx));
lz = min(size(panelz));


%get the location of the collocation point in each panel and calculate the slope
for j = 1:np,
coloc(j,1) =(panelx(1,j)+panelx(lx,j))/2;
coloc(j,2) = (panelz(1,j)+panelz(lz,j))/2;
if j <= np/2
sn = -1;
```

```
else
sn = 1;
end


for k = 1:lx,
if (panelx(k,j) > coloc(j,1) & j > np/2)
break
elseif (panelx(k,j) < coloc(j,1) & j <= np/2)
break
end
end
slope(j) = atan2(sn*(panelz(k,j)-panelz(k-1,j)),sn*(panelx(k,j)-panelx(k-1,j)));
end


%calculate the influence coefficients
for j = 1:np,
for k = 1:np,
xt = coloc(j,1)-panelx(1,k);
zt = coloc(j,2)-panelz(1,k);
x2t = panelx(lx,k)-panelx(1,k);
z2t = panelz(lz,k)-panelz(1,k);
xl = xt*cos(slope(k))+zt*sin(slope(k));
zl = -xt*sin(slope(k))+zt*cos(slope(k));
x2 = x2t*cos(slope(k))+z2t*sin(slope(k));
z2 = 0;


r1 = (xl^2 + zl^2);
r2 = ((xl - x2)^2 + zl^2);


if j == k
ul = 0;
wl = -1/(pi*xl);
else
ul = 1/(2*pi)*(zl/r1-zl/r2);
wl = -1/(2*pi)*(xl/r1-(xl-x2)/r2);
end
u = ul*cos(-slope(k))+wl*sin(-slope(k));
w = -ul*sin(-slope(k))+wl*cos(-slope(k));
dob(j,k) = -u*sin(slope(j))+w*cos(slope(j));
B(j,k) = u*cos(slope(j))+w*sin(slope(j));
```

```
end
end

leng = sqrt((panelx(lx,np)-panelx(1,np))^2+(panelz(lz,np)-panelz(1,np))^2);
% Use the Kutta condition as MuW = Mu(np)-Mu(np/2) and calculate its influence
for j =1:np,
r1 = ((coloc(j,1)-panelx(1,1))^2+(coloc(j,2)-panelz(1,1))^2);
r2 = ((coloc(j,1)-(panelx(lx,np)+0.1))^2+(coloc(j,2)-panelz(lz,np))^2);
wl = -1/(2*pi)*((coloc(j,1)-panelx(1,1))/r1);
ul = 1/(2*pi)*((coloc(j,2)-panelz(1,1))/r1);
dob(j,np+1)= wl*cos(slope(j))-ul*sin(slope(j));
B(j,np+1) = ul*cos(slope(j))+wl*sin(slope(j));
if j == 1
dob(np+1,j) = -1;
elseif j == np
dob(np+1,j) = 1;
else
dob(np+1,j) = 0;
end
end
dob(np+1,np+1) = -1;


for j = 1:np+1,
for k = 1:np+1;
as(k) = dob(j,k);
end
act(j) = max(abs(as));
for k = 1:np+1,
dob(j,k) = dob(j,k)/act(j);
end
end

% Apply the normal boundary conditions
for j = 1:np,
Nm(j) =(uinf*sin(-alpha+slope(j)))/act(j);
end

Nm(np+1) = 0;


%Get the doublet potential
Mu = inv(dob)*(Nm');
```

```
c = 1;
for l2 = 1:np,
temp = 0;
if l2 > np/2
sn = 1;
else
sn = -1;
end
for l1 = 1:np+1,
temp = temp+B(l2,l1)*Mu(l1);
end
if l2  =1 & l2 = np
r = sqrt((coloc(l2+1,1)-coloc(l2-1,1))^2+(coloc(l2+1,2)-coloc(l2-1,2))^2);
vloc = sn*(Mu(l2+1)-Mu(l2-1))/r;
elseif l2 == np
r = sqrt((coloc(np-1,1)-coloc(np,1))^2+(coloc(np-1,2)-coloc(np,2))^2);
vloc = (Mu(np)-Mu(np-1))/r;
else
r = sqrt((coloc(2,1)-coloc(1,1))^2+(coloc(2,2)-coloc(1,2))^2);
vloc = -(Mu(1)-Mu(2))/r;
end

vel = (cos(alpha)*cos(slope(l2))+sin(alpha)*sin(slope(l2)))*uinf+temp+vloc/2;
uu(l2) = vel*cos(slope(l2));
vv(l2) = vel*sin(slope(l2));
if l2  = np/2+1 & l2 = np/2 & l2  = 1
xp(c) = coloc(l2,1);
yp(c) = coloc(l2,2);
cp(c) = -1+(vel^2)/uinf^2;
c = c+1;
end
end
figure(1);
plot(xp,cp,'*',xp,cp,'-r',xp,yp,'k'); title('Coefficient of Pressure')
ylabel('-Cp')
cl = 0;
ml = 0;


for l2 = 2:np/2-2,
leng = sqrt((panelx(lx,l2)-panelx(1,l2))^2+(panelz(lz,l2)-panelz(1,l2))^2);
ml = ml -(-cp(l2)*cos(slope(l2))+cp(np-l2+1)*cos(slope(np-l2+1)))*leng*cos(alpha)*(xp(l2)-panelx(1,1)/4);
cl = cl +(cp(np-l2+1)*cos(slope(np-l2+1))-cp(l2)*cos(slope(l2)))*leng*cos(alpha);
```

end

File Vooren.m
% This program gives an airfoil with a finite trailing edge
function $[B1x, B1y]$ = Vooren(x,np,num);
a = 0.3;ep = 0.1; kk = 2-(10*pi/180)/pi;
c = 1;
j1 = 2*pi-(num-1)*(2*pi)/(np);
j2 = 2*pi-num*(2*pi)/(np);
inc = -abs(j1-j2)/7;

for th = j1:inc:j2,
r1 = sqrt$((a * cos(th) - a)^2 + a^2 * sin(th)^2)$;
r2 = sqrt$((a * cos(th) - ep * a)^2 + a^2 * sin(th)^2)$;
th1 = atan2(a*sin(th),(a*cos(th)-a));
th2 = atan2(a*sin(th),(a*cos(th)-ep*a));
Bx(c) = $1 + 0.1124 + r1^k k/r2^{(kk-1)} * cos(kk * th1 - (kk - 1) * th2)$;
By(c) = $r1^k k/r2^{(kk-1)} * sin(kk * th1 - (kk - 1) * th2)$;
c= c+1;
end

k = 3;n = length(x)-3;
N = zeros(n+k-1,k+1);
count = 0;
t= zeros(length(Bx),1);
total = 0;
for d = 2:length(Bx),
temp = (Bx(d)-Bx(d-1))^2 + (By(d)-By(d-1))^2;
total = total+sqrt(temp);
end
for l = 2:length(Bx),
for d = 2:l,
t(l,1) = t(l,1) + sqrt((Bx(d)-Bx(d-1))^2+(By(d)-By(d-1))^2);
end
t(l,1)=t(l,1)/total *x(length(x));
end
q =1;

%Given the data points, calculate the control points using Cox-De-Boor algorithm
while q $<$ length(t)+1
while t(q,1) $<$ count +1
N(k+count,1) = 1;

```
for l = 2:k,
for j = k+count-(l-1):k+count,
if (x(j+l-1)-x(j))  = 0
N(j,l) = ((t(q,1)-x(j))*N(j,l-1))/(x(j+l-1)-x(j));
end
if (x(j+l)-x(j+1))  = 0
N(j,l) = N(j,l) + ((x(j+l)-t(q,1))*N(j+1,l-1))/(x(j+l)-x(j+1));
end
end
end
for h = 1:k+count,
M(q,h) = N(h,k);
end
q = q+1;
N = zeros(n+k+1,k+1);
if q == length(Bx) +1
count = count+1;
break
end
end
count = count+1;
if count == x(length(x)-1)
count = count -1;
t(q,1) = count+0.99;
end
end
% Calculate the new control polygon points and the Airfoil curve
Px = inv(M'*M)*M'*Bx';
Py = inv(M'*M)*M'*By';
B1x = M*Px;
B1y = M*Py;
```

% Panel code for Unsteady Aerodynamics

File UnsPanelFull.m

```
% Initiates the unsteady aerodynamics calculations
clear all; close all;
np = input('enter the number of panels ');
uinf = input('enter free stream velocity ');

v = [0 0 0 1 2 3 4 4 4];
for j=1:np,
[x, z] = Vooren(v,np,j);
if j == 1
panelx = x;
panelz= z;
else
panelx=cat(2,panelx,x);
panelz=cat(2,panelz,z);
end
end

lx = min(size(panelx));
b = panelx(lx,np)/2;

%Set the time step
tstep = 0.007;
[angul, h, tt2, cl2, ml2] = Morph(uinf,panelx,panelz,np,tstep);
plot(tt2,cl2,tt2,ml2,'c');
ylabel('Cm')
xlabel('time')
grid on
angul = angul*180/pi;
figure(2);
plot(tt2,angul,tt2,h);
ylabel('Angle Degrees')
xlabel('time')
grid on
```

Function that performs the time stepping for the unsteady aerodynamics

```
function [an1, an2, ttA, clA, mlA] = Morph(uinf,panelx,panelz,np,tstep)
alpha = 0;
```

```
% Get the Size Of the panel matrix
lx = min(size(panelx));
lz = min(size(panelz));
b = panelx(lx,np)/2;
mm = 7; sa = 0.05; ia = 0.38 * mm * b^2;
d2 = mm * ia - sa^2;
ff = 1;
cc = 1;


fid = fopen('NeuroData1.dat','w');
%get the location of the collocation point in each panel and calculate the slope
for j = 1:np,
coloc(j,1) =(panelx(1,j)+panelx(lx,j))/2;
coloc(j,2) = (panelz(1,j)+panelz(lz,j))/2;
if j ¡= np/2
sn = -1;
else
sn = 1;
end
for k = 1:lx,
if (panelx(k,j) > coloc(j,1) & j > np/2)
break
elseif (panelx(k,j) < coloc(j,1) & j <= np/2)
break
end
end
slope(j) = atan2(sn*(panelz(k,j)-panelz(k-1,j)),sn*(panelx(k,j)-panelx(k-1,j)));
end


%calculate the influence coefficients
for j = 1:np,
for k = 1:np,
xt = coloc(j,1)-panelx(1,k);
zt = coloc(j,2)-panelz(1,k);
x2t = panelx(lx,k)-panelx(1,k);
z2t = panelz(lz,k)-panelz(1,k);
xl = xt*cos(slope(k))+zt*sin(slope(k));
zl = -xt*sin(slope(k))+zt*cos(slope(k));
x2 = x2t*cos(slope(k))+z2t*sin(slope(k));
z2 = 0;
r1 = (xl^2 + zl^2);
```

```
r2 = ((xl − x2)^2 + zl^2);
if j == k
ul = 0;
wl = -1/(pi*xl);
if j ¿ np/2
p(j,k) = -0.5;
else
p(j,k) = 0.5;
end
else
ul = 1/(2*pi)*(zl/r1-zl/r2);
wl = -1/(2*pi)*(xl/r1-(xl-x2)/r2);
p(j,k) = -1/(2*pi)*(atan2(zl,(xl-x2))-atan2(zl,xl));
end
u = ul*cos(-slope(k))+wl*sin(-slope(k));
w = -ul*sin(-slope(k))+wl*cos(-slope(k));
dob(j,k) = -u*sin(slope(j))+w*cos(slope(j));
B(j,k) = u*cos(slope(j))+w*sin(slope(j));
end
% Use the Kutta condition as MuW = Mu(np)-Mu(1) and calculate its influence
xt = coloc(j,1)-panelx(lx,np);
zt = coloc(j,2)-panelz(lz,np);
x2t = 0.25*tstep*b*(cos(alpha));
z2t = 0.25*tstep*b*sin(alpha);
xl = xt*cos(-alpha)+zt*sin(alpha);
zl = -xt*sin(-alpha)+zt*cos(alpha);
x2 = x2t*cos(-alpha)+z2t*sin(alpha);
z2 = 0;
r1 = (xl^2+zl^2);
r2 = ((xl − x2)^2 + zl^2);
wl = -1/(2*pi)*(xl/r1-(xl-x2)/r2);
ul = 1/(2*pi)*(zl/r1-zl/r2);
u = ul*cos(alpha)+wl*sin(alpha);
w = -ul*sin(alpha)+wl*cos(alpha);
p(j,np+1) = -1/(2*pi)*(atan2(zl,(xl-x2))-atan2(zl,xl));
dob(j,np+1)= w*cos(slope(j))-u*sin(slope(j));
B(j,np+1) = u*cos(slope(j))+w*sin(slope(j));
if j == 1
dob(np+1,j) = -1;
elseif j == np
dob(np+1,j) = 1;
else
```

```
dob(np+1,j) = 0;
end
end
dob(np+1,np+1) = -1;
invIC = inv(dob);


%Computation Of velocities at each time step
%Start the Unsteady Computation
lmt = 0;
vel = 0.0;
vel2 = 0;vel3 = 0;
angul = [0.04 0.04];
h = [0 0.002];
c = 0;
count = 1;
for t = tstep:tstep:6.0,
if count > 1
velo2(count) = vel2;
hev(count) = vel3;
Tw(count) = velo2(count);
else
Tw(count) = vel;
hev(count) = 0;
end
thet = angul(count);
if count > 1
tau(count-1) = t;
cl(count-1) = 0;
ml(count-1) = 0;
end
for j = 1:np,
L(1) = 0;L(2) = 0;
q(j)= 0;
S(j) = 0;
% Calculate the normal velocity
Nm(j) = uinf*sin(-alpha-thet+slope(j))-hev(count)*cos(-alpha-thet+slope(j))...
+vel2*(coloc(j,1)-panelx(1,1)/2)*cos(slope(j));
%calculate the vertical velocity from the wake doublets
for k = 1:count-1,
add1= b*cos(alpha+angul(count-k+1))*tstep;
add2= b*sin(alpha+angul(count-k+1))*tstep;
L(1) = L(1)+add1;
```

```
L(2) = L(2)+add2;
xt = coloc(j,1)-(panelx(lx,np)+L(1));
zt = coloc(j,2)-(panelz(lz,np)+L(2));
x2t = add1;
z2t = add2;
xl = xt*cos(alpha+angul(count-k+1))+zt*sin(alpha+angul(count-k+1));
zl = -xt*sin(alpha+angul(count-k+1))+zt*cos(alpha+angul(count-k+1));
x2 = x2t*cos(alpha+angul(count-k+1))+z2t*sin(alpha+angul(count-k+1));
z2 = 0;
r1 = (xl^2 + zl^2);
r2 = ((xl-x2)^2+zl^2);
wl = -Muw(k)/(2*pi)*(xl/r1-(xl-x2)/r2);
ul = Muw(k)/(2*pi)*(zl/r1-zl/r2);
u = ul*cos(-alpha-angul(count-k+1))+wl*sin(-alpha-angul(count-k+1));
w = -ul*sin(-alpha-angul(count-k+1))+wl*cos(-alpha-angul(count-k+1));
Nm(j) = Nm(j)-(w*cos(slope(j))-u*sin(slope(j)));
S(j) = S(j)+u*cos(slope(j))+w*sin(slope(j));
q(j) = q(j)-Muw(k)/(2*pi)*(atan2(zl,(xl-x2))-atan2(zl,xl));
end
end
Nm(np+1) = 0;


%Get the doublet potential
Mu = invIC*(Nm');
Muw(count) = Mu(np+1);
%Save the potential of the previous time step
if count >1
phi1 = phi;
else
for j = 1:np,
phi1(j) =0;
phi(j) = 0;
end
end
for l2 = 1:np,
temp = 0;
phi(l2) =0;
if l2 > np/2
sn = 1;
else
sn = -1;
end
```

```
% Calculate velocity and potential at each panel
for l1 = 1:np+1,
temp = temp+B(l2,l1)*Mu(l1);
phi(l2) = phi(l2) + p(l2,l1)*Mu(l1);
end
if count > 1
phi(l2) = phi(l2)+q(l2);
temp = temp+S(l2);
end
if l2  =1 & l2 = np
r = sqrt((coloc(l2+1,1)-coloc(l2-1,1))^2+(coloc(l2+1,2)-coloc(l2-1,2))^2);
vloc = sn*(Mu(l2+1)-Mu(l2-1))/r;
elseif l2 == np
r = sqrt((coloc(np-1,1)-coloc(np,1))^2+(coloc(np-1,2)-coloc(np,2))^2);
vloc = (Mu(np)-Mu(np-1))/r;
else
r = sqrt((coloc(2,1)-coloc(1,1))^2+(coloc(2,2)-coloc(1,2))^2);
vloc = -(Mu(1)-Mu(2))/r;
end
%Find the net velocity at each panel
vel = (cos(alpha+thet)*cos(slope(l2))+sin(alpha+thet)*sin(slope(l2)))*uinf+temp+vloc/2;
vel = sqrt(vel^2+(Tw(count)*(coloc(l2,1)-panelx(1,1)/2)*cos(slope(l2))-hev(count)*cos(alpha-thet+slope(l2)))^2);
uu(l2) = vel*cos(slope(l2));
vv(l2) = vel*sin(slope(l2));
xp(l2) = coloc(l2,1);
yp(l2) = coloc(l2,2);
%Calculate the unsteady coefficient of pressure if count >= 2
cp(l2) = -1+(vel^2)/(uinf^2)+(2/uinf^2)*((phi(l2)-phi1(l2))/tstep);
else
cp(l2) = -1+(vel^2)/(uinf^2);
end
end


%Calculate the lift and pitching moment
if count > 1
for l2 = 1:np/2-1,
leng = sqrt((panelx(lx,l2)-panelx(1,l2))^2+(panelz(lz,l2)-panelz(1,l2))^2);
ml(count-1) = ml(count-1) +(cp(l2)*cos(slope(l2))-cp(np-l2+1)*cos(slope(np-l2+1)))*leng*cos(thet)*(xp(l2)-
panelx(1,1)/2);
cl(count-1) = cl(count-1) +(-cp(np-l2+1)*cos(slope(np-l2+1))+cp(l2)*cos(slope(l2)))*leng*cos(thet);
end
```

```
% Stiffness allocation
kk2(count-1) = 0.25;
kk(count-1) = 5;
kk1 = [kk(count-1);kk2(count-1)];


f11 = 2*(b^2)*(uinf^2)*1.225*(ml(length(ml)));
f21 = b*(uinf^2)*1.225*(cl(length(cl)));
f = [f11;f21];
xx=[angul(count);vel2;h(count);vel3];
[y2]= Runge2(xx,f,kk1,mm,ia,tstep);


angul = cat(2,angul,y2(1,1));
h = cat(2,h,y2(3,1));
vel2 = y2(2,1);
vel3 = y2(4,1);

if mod(tau(count-1),0.056) == 0
if tau(count-1) > 0.1
x0 = [angul(lmt+1);v11;h(lmt+1);v22];
inpcc = [tau(lmt);tau(count-1);uinf;x0(1,1);x0(2,1);x0(3,1);x0(4,1)];
oupcc = [ml(lmt);ml(count-1);cl(lmt);cl(count-1)];
else
x0 = [angul(count-6);0;h(count-6);0];
inpcc = [lmt;tau(count-1);uinf;x0(1,1);x0(2,1);x0(3,1);x0(4,1)];
oupcc = [ml(count-7);ml(count-1);cl(count-7);cl(count-1)];
end
fprintf(fid,'%f %f %f %f %f %f',inpcc);
fprintf(fid,'%f %f %f %f',oupcc);
cc=cc+1;
lmt = (count-1);
v11 = vel2;
v22 = vel3;
end
end


count = count+1;
end
for i = 3:length(tau),
an1(i-2) = angul(i);
mlA(i-2) = ml(i);
clA(i-2) = cl(i);
```

```
ttA(i-2) = tau(i);
an2(i-2) = h(i);
end
fclose(fid);


Function for Runge-Kutta time marching
function [y2] = Runge2(y0,g,k,m,ia,tstep)
s = [0 -1 0 0;k(1,1) 0 0 0;0 0 0 -1;0 0 k(2,1) 0];
A = [1 0 0 0;0 ia 0 0.05;0 0 1 0;0 0.05 0 m];
Ai = inv(A);
s = Ai*s;
g2 = [0;g(1,1);0;g(2,1)];
g2 = Ai*g2;
k1 = tstep*(g2-s*y0);
k2 = tstep*(g2-s*(y0+k1/2));
k3 = tstep*(g2-s*(y0+k2/2));
k4 = tstep*(g2-s*(y0+k3));
y2 = y0+1/6*(k1+2*k2+2*k3+k4);
```

Vita

Anand Natarajan was born on the afternoon of July 2^{nd} 1973 in the city of Trivandrum, India. He spent most of his life in the city of Bangalore in India. Anand finished his undergraduate education in mechanical engineering from the Regional Engineering College, Calicut in 1994. He then went on to finish his Masters in Mechanical Engineering from the Indian Institute of Science, Bangalore. He worked in the software industry in two noted firms, Parametric Technology Corporation(PTC) and Wipro Infotech. Anand then decided to join the Ph.D. program in the Dept. of Aerospace Engineering at Virginia Tech in August 1999.