NOTE TO USERS

FLUTTER SUPPRESSION OF AN AIRFOIL USING NEURO-FUZZY CONTROL

by

Chun Meng

B.E. (Mechanical Engineering)

East China University of Science & Technology (China), 1989

A thesis
presented to Ryerson University
in partial fulfillment of the
requirements for the degree of
Master of Applied Science
in the program of
Mechanical Engineering

Toronto, Ontario, Canada, 2003

Author's declaration

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Chun Meng

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Chun Meng

Borrower's page

Ryerson University requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

ABSTRACT

FLUTTER SUPPRESSION OF AN AIRFOIL USING NEURO-FUZZY CONTROL

© Chun Meng, 2003
Master of Applied Science
in the program of
Mechanical Engineering
Ryerson University

Flutter, a self-excited vibration of wings and control surfaces, can lead to catastrophic failure of aircraft structures. Classical methods have been applied successfully for flutter suppression and for increasing the flutter critical speed. With the demand of higher speed and more flexible aircraft, more advanced active flutter control techniques are required.

In this study, a neuro-fuzzy methodology for flutter suppression of a two dimensional airfoil is explored. A MATLAB simulation environment is used for the modeling and analysis. The airfoil model is simulated according to a set of aeroelastic equations of motion. A neuro-fuzzy controller, called NEFCON, is then embedded in the airfoil model for increasing the flutter speed.

NEFCON learns from the motion of the airfoil and automatically produces fuzzy rules. The simulation results show that these fuzzy rules can successfully increase the critical flutter speed. The performance of the fuzzy rules is tested with different airfoil parameters.

Acknowledgement

I would like to thank my first supervisor, Dr. Farrokh Sharifi, for his kindly guidance on my early Master's degree study. He takes great care of his students.

I would like to express my appreciation to Ms. Leah Stanwyck for her consistent and timely assistance on graduate study affairs. She is always so sweet and willing to help you.

I am grateful for the support of Mr. Reg Michie on software searching. Mrs. Diane Ewen provides a lot of help on paper writing, thank you.

I would like to thank my wife, X. Sun, for her love and tolerance. Without her help, it will be almost impossible for me to finish this thesis.

Finally, I would like to thank my supervisor, Dr. Hekmat Alighanbari, for his support, direction and encouragement. It is under his inspiration that I decided the research subject in an effort to explore the neuro-fuzzy method on flutter suppression. His valuable advices and suggestions have an important role in my success.

Table of Contents

Table of Figures

List of Tables

Nomenclature

a_h	dimensionless distance from airfoil mid-chord to elastic axis;
α	airfoil pitch displacement;
a_w	coefficient in the approximate formula for Wagner function;
α_T, ζ and kn	coefficients of the transfer function of the energy control law equation;
β	flap rotation;
β_c	flap rotation command;
b_w	coefficient in the approximate formula for Wagner function;
c_β(cbeta)	dimensionless distance measured from mid-chord to flap hinge;
c_w	coefficient in the approximate formula for Wagner function;
d_w	coefficient in the approximate formula for Wagner function;
l_w	coefficient in approximate formula for Wagner function;
ξ	nondimensional airfoil plunge displacement,h/b;
r_α(ralpha)	dimensionless radius of gyration about elastic axis;
r_β(rbeta)	dimensionless radius of gyration about flap hinge;
$T_{C\xi}$(T_{Cxi}), $T_{C\alpha}$(T_{Calpha}), $T_{G\xi}$(T_{Gxi}) and $T_{G\alpha}$(T_{Calpha}):	coefficients of energy control law equation;
τ	dimensionless time, tV/b;
μ_A(x)	membership value of an element (x) in set (A);

μ(mu)	airfoil-air mass ratio, $m/\pi\rho b^2$;
U	dimensionless free stream velocity, $V/b\omega_\alpha$;
V	free stream velocity;
ω_ξ(omegaxi)	uncoupled frequency ratio in plunge;
ω_β(omegabeta):	uncoupled frequency ratio in flap;
x_α(xalpha)	dimensionless distance from elastic axis to mass centre;
x_β(xbeta)	dimensionless distance from the flap hinge to the flap center of mass;
y_1	auxiliary variable;
y_2	auxiliary variable;
ζ_ξ(zetaxi)	viscous damping ratio in plunge;
ζ_α(zetaalpha)	viscous damping ratio in pitch;
ζ_β(zetabeta)	viscous damping ratio in flap hinge;
ρ	density of air;
ϕ	Wagner's function;

Chapter 1 Introduction

1.1 Purpose

The aircraft bring a new era in the human history. They make our dream of flying like a bird come true. In developing the aircraft, many diverse fundamental laws of nature are applied, some of which involve aeroelasticity [1]. Among aeroelastic phenomena, flutter, the self activated divergent oscillations of the aircraft wings and control surfaces, has caused intensive studies. Traditional control methods have been applied succes fully in flutter prevention.

Artificial intelligence has already numerous applications in various industries despite its short history, but its application in the aerospace area is quite new. When implemented in flutter suppression, artificial intelligence is often used to optimize classical control laws.

In the present study, the author will utilize the neuro-fuzzy method to suppress flutter of an airfoil. Flutter suppression of an airfoil is achieved by moving a flap attached to the airfoil. In the next chapters, the author will explain neural

networks theory, fuzzy logic theory and neuro-fuzzy method. The equations used to simulate the airfoil model and to develop the neuro-fuzzy controller model will be also presented. Finally, the author will present the flutter suppression results of the neuro-fuzzy controller and suggestions for future study.

1.2 Flutter

1.2.1 Flutter Definition

Flutter which is critical for an aircraft's operation is an aerodynamics induced oscillation of a wing, tail, or control surface. For a clear facet of this physical phenomenon, let us consider a cantilever wing, without sweepback and without flap, mounted in a wind tunnel at a small angle of attack and with a rigid support at the root. The wing oscillation will damp gradually to its original (stable) position after a transient shake when there is no air flow passing around the wing. When the airspeed increases gradually in the wind tunnel, the wing oscillation will first damp faster. As the speed increases further, a point can be reached at which the damping of the wing rapidly decreases, and there is a certain speed at which an oscillation can just maintain its amplitude. This speed is called the flutter speed. At speeds above this critical one, a small accidental disturbance of the wing can serve as a trigger to initiate an oscillation of great violence, which can result in a total structural failure in just seconds [2].

An aircraft wing is continuous and has an infinite number of degrees-of-freedom. For most practical theoretical studies, the simplified model of an airfoil wing with finite degree-of-freedom is used. A two degree-of-freedom airfoil model is shown in Figure 1. The model will be described later in detail in

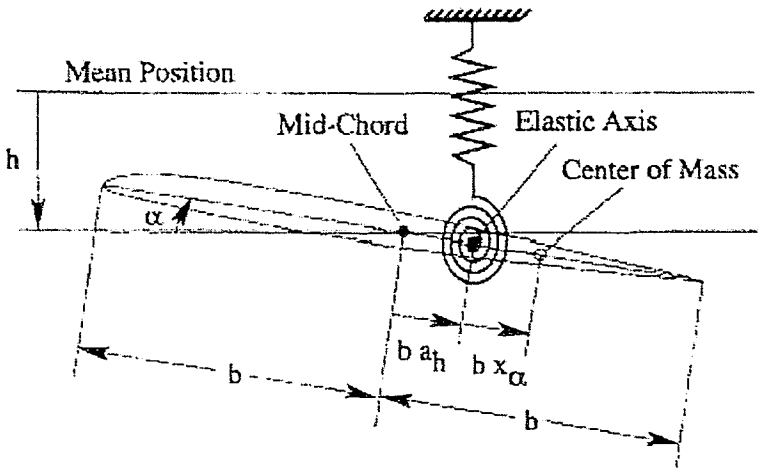


Figure 1. Three degree-of-freedom airfoil model (Alighanbari 2000)

Chapter 2. The flutter motion of the airfoil has both flexural and torsional components (Figure 2). Classical binary flutter requires at least two degree-of-freedom; for example, plunge and pitch degree-of-freedom. The phase shift between plunge and pitch motions plays an important role in the flutter phenomenon. It is mainly this phase difference that causes the flutter.

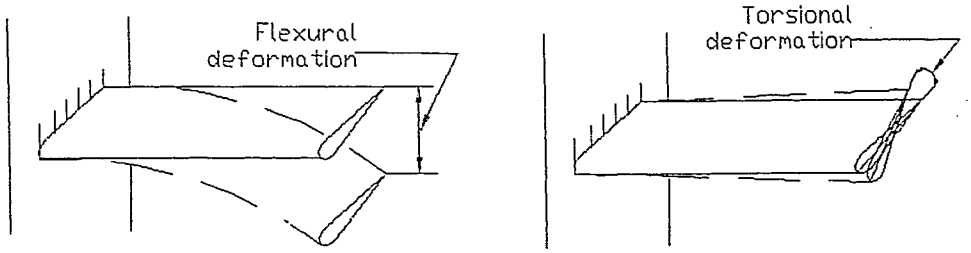


Figure 2. Airfoil deformations

1.2.2 Flutter Suppression

The critical effect of flutter has been attracting a lot of attention for several decades especially when people want to design faster fighter aircraft. Before the 1930s, little was known about flutter. Flying in those days did really need to take great risks with the new sports. It is almost impossible to know how many brave souls succumbed to flutter. In the 1930s, airplanes were expected to achieve high speed with better engines available. At that time, flutter analysis and prevention became more serious especially for faster fighter aircraft. There are basically two types of flutter suppression: passive flutter suppression and active flutter suppression.

a. Passive Flutter Suppression

This method includes mass balancing and stiffening of aircraft structure. From the data analysis on several aircraft in which wing flutter has been observed, and on others that showed no tendency to flutter, the following critical flutter speed equation for aircraft wings with mass-unbalanced ailerons was found [2]:

$U_{cr} = \frac{\omega c}{2k_{cr}}$ (U_{cr} = the critical mean flutter speed, feet per second; ω = the fundamental frequency of the wing in torsional oscillation in still air, radians per second; c = the length of the vibrating portion of the wing, feet; k_{cr} = the critical reduced frequency, 0.9 ± 0.12 radian). From this U_{cr} equation, it is obvious that the flutter speed can be raised by increasing the wing torsional stiffness. Systematic study has shown that mass balancing can increase critical flutter speed substantially [2]. Passive flutter suppression is sometimes achieved through an automatic control system, which can be referred to as semi-active flutter suppression. By using a microcomputer controlled micromotor to adjust the stiffness, Yang et al [3] increased the flutter speed of a two-dimensional wing effectively with this semi-active method.

Passive flutter suppression is an easy way to increase critical flutter speed. However, as it increases the aircraft weight in order to raise flutter speed, it may not be practical if light aircraft with high flutter speed are expected.

b. Active Flutter Suppression

With the demand of higher speed and light-weight aircraft and to overcome the shortcomings of passive flutter suppression, a new control method, called "active-suppression" was developed in the 1970s. Here, an onboard automatic control system actuates a control-surface on the wing in response to sensed structural motion of the aileron-flap in order to suppress flutter. The first active control practice was carried out by the U.S. Air Force in their Load Alleviation

and Mode Stabilization(LAMS) program, which resulted in a Boeing B-52 bomber flying 10 knots faster than its open-loop flutter speed.

1.3 Artificial Intelligence

1.3.1 Artificial Neural Networks

Artificial neural networks, often referred to as neural networks, is a branch of artificial intelligence (AI). It is developed based on present understanding of the human brain. There are various definitions of neural networks. Initially, a neural network is a parallel processor modeling the human brain that can learn [4]. It resembles the human brain in two ways: 1) acquiring knowledge through a learning process; 2) storing knowledge as the inter-neuron weights which are the connections among neurons that represent the inputs' contribution to the outputs.

Research about the human central nervous systems began as early as the Middle Age, but their detailed structure was only revealed a century ago. The brain's most basic components—neurons—are connected through trillions of synapses [4]. Each neuron (Figure 3) is composed of a central body, called soma, a number of root-like extensions, called dendrite, and a main transmission line, called axon.

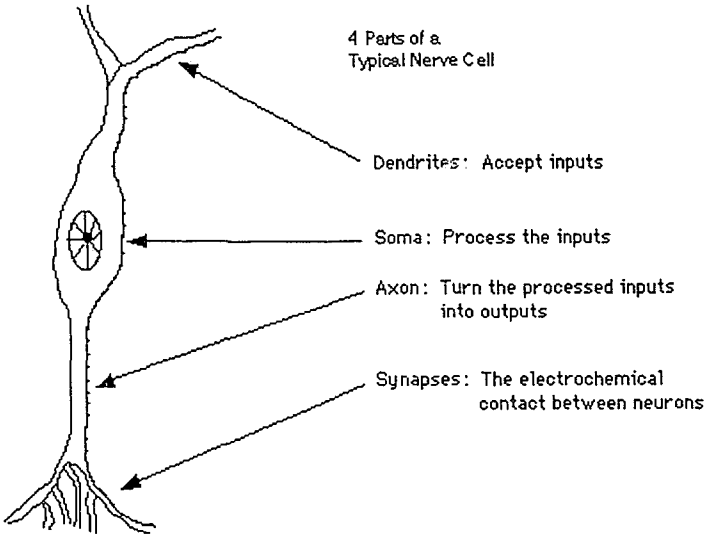


Figure 3. Biological neural network

Signals are transmitted electrically or chemically in the neural networks. The electrical transmission is within a neuron, while the chemical one happens among the neurons. The soma electrical potential changes as receiving chemical from synapses. When this potential reaches a certain level, an electrical pulse is transmitted by the axon and reaches other synapses, causing them to change the potential.

One of the most important characters of a neural network is its plasticity. It can form new connections or modify the strength among neurons in response to the circumstance changes. Even though single neuron is not as powerful as silicon logic gates, the brain can perform much faster by organizing neurons than the fastest computer in existence today [4,5,6] .

Learning ability is another critical character of a biological neuron network. In a person's early days, the connections in the nervous systems are formed at a very

fast manner while learning the outside world. Later on, these connections may be modified when new knowledge is available. This also leads to attempts to mimic the biological neural network for scientific computation.

An artificial neuron is analogous to a biological neuron (Figure 4). It is the basic computing cell in artificial neuron networks. The inputs, either from external circumstances or other neurons within the neuron network, are first summed up. Then, the difference between this summation and the threshold is passed to the

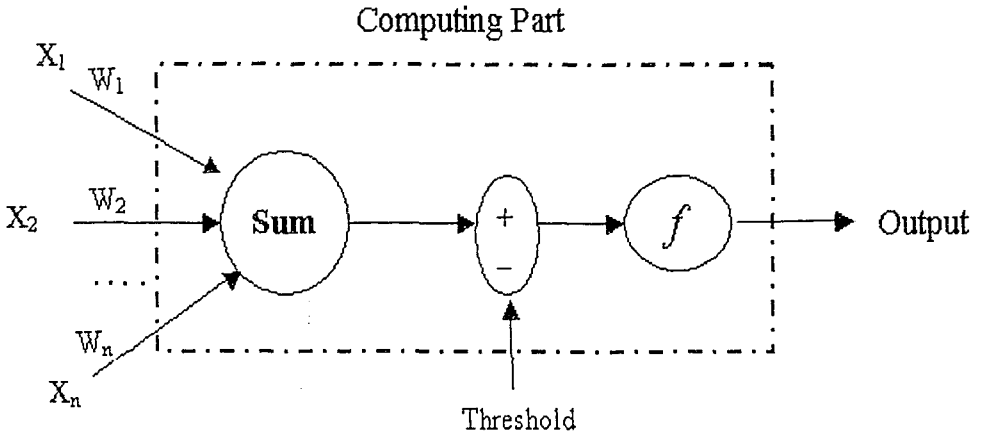


Figure 4. Artificial neuron

activation function. There are four common activation functions used for different purposes: step, sign, linear and sigmoid functions (Figure 5). In the backpropagation neural network, normally the sigmoid function is used [4]: $Y^{sigmoid}=\frac{1}{1+e^{-X}}$. The terms in biological neuron networks and artificial neuron networks are summarized in Table 1.

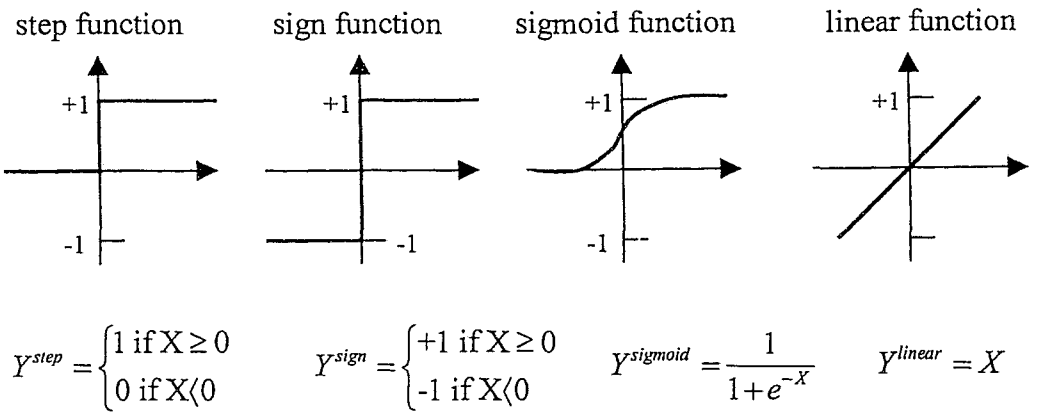


Figure 5. Neural networks activation functions

Biological Term	Artificial Term	Function
Axon	Connection	Signals transmission channel
Dendrite	Connection	Signal receiving, carrying for processing cell
Neuron	Neuron	Basic unit of networks
Soma	Sum, Activation Function	Computing cell
Synapse	Weights	Knowledge representations

Table 1. Biological neural network and artificial neural networks terms

The first artificial neuron model was introduced by W. McCulloch and W. Pitts in 1943. In the recent two decades, this artificial branch has expanded more rapidly than ever before. A lot of neural networks models are proposed [4, 5] and their applications have been mentioned quite often. In this study, the author will only discuss the feedforward backpropagation neural networks that will be

used in this study. The introduction on others types of neural networks is available in [5].

A multiple layer neural network is composed of an input layer, an output layer and at least one hidden layer. Every layer is made up of a number of neurons. The input layer neurons accept the external signals and pass them to the next layer neurons for process; the output layer neurons export the neural networks' outputs. The neurons in the hidden layer(s) act as signal processors. The weights pattern of a neural network is formed in the learning stage and can be changed if the neural networks is on-line. The number of hidden layers can be more than just one, but the increase in the hidden layers will increase the computation time dramatically. In practice, three-layer neural (Figure 6) networks are capable of approximating continuous functions to any degree of accuracy [7]. Here, the author will explain the supervised learning process with a three-layer feedforward backpropagation neural network (Figure 6). The typical learning process includes initialization, input feedforward and error backpropagation. Before starting the learning process, one should prepare the training data sets and know the expected results.

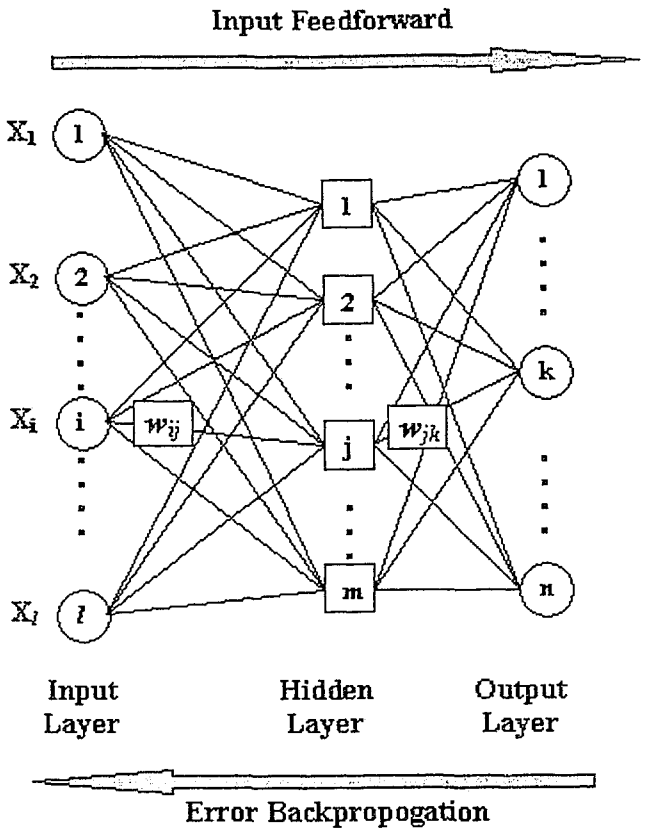


Figure 6. Three-layer neural networks

a. Initialization:

The weights and threshold levels for the neural networks are set to random numbers uniformly distributed within a narrow range. Normally, this range is $\pm 2.4/F_i$, where F_i is the total number of inputs of a neuron [5].

b. Input Feedforward:

During this stage, the training data and expected outputs are presented to the neural networks. The neural networks will do the calculation based on the

provided inputs and pass the training data or calculated intermediate results from one layer to the next layer. The output layer will present the final calculated results.

c. Error Backpropagation:

In this stage, the errors between the neural networks produced outputs and the expected output is calculated and passed from the output layer to the input layer. The weights in the neural networks are adjusted.

d. Iteration:

Iterate the steps *b* and *c* by presenting a new set of training data until the defined error is satisfied or other termination criteria are reached.

So far, the author has discussed the supervised training. The neural networks structure pattern for the same problem will not be unique. Its parameters depend on lots of factors, such as initial weights, learning rate, etc. There are also other learning algorithms in [4] and [5].

1.3.2 Fuzzy Logic

The fuzzy set concept was founded by Lofti Zadeh in 1964. It extended the classical two-value logic. Even though founded in 1960s, fuzzy logic was not accepted as a scientific theory until 1980s after a wide range successful applications [4]. Fuzzy sets, linguistic variables and fuzzy IF-THEN rules are the most basic concepts in fuzzy logic.

a. Fuzzy sets:

Unlike classical (crisp) logic, the fuzzy set has a smooth boundary. This fuzzy set concept can express knowledge more naturally. For example, when people talk about engine power, in classical logic, there is an accurate value (say 30hps) that splits the power domain into two subsets: weak and strong. The engine with power higher than 30hps will be said to be strong, while the engine whose power is less than 30hps will be said to be weak. *But what about a 29.9hps engine?* According to classical logic, it will be weak, but in fact, it is still quite strong. So in fuzzy logic, this can be expressed as a membership function value, say it is strong with a 0.8 membership function value (Figure 7).

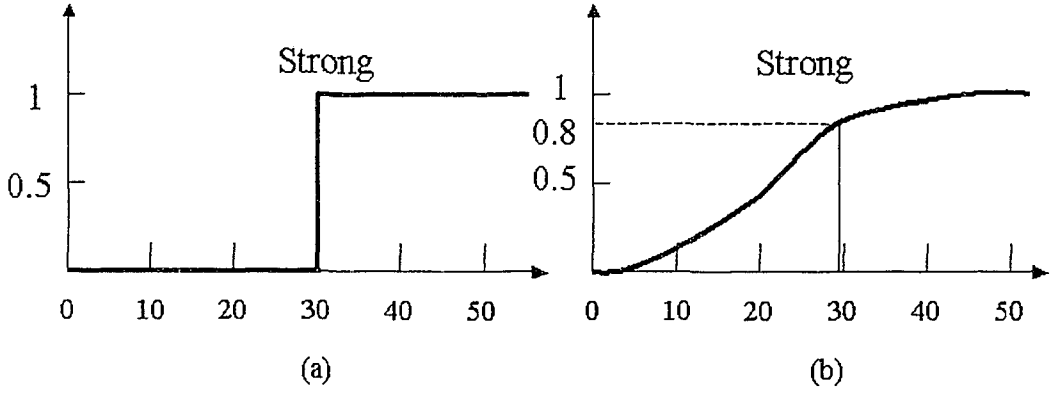


Figure 7. Classical (crisp)(a) and fuzzy(b) sets

Hence, instead of yes-or-no in classical logic, the element and set relationship is expressed in a matter of degree with value from 0 to 1 in fuzzy logic. The membership value is said to be zero when an element is completely out of a set; it is one when an element is totally in a set; and 0~1 when an element is partially in a set.

There are four commonly used fuzzy logic m ·ship functions: triangular, trapezoid, Gaussian and sigmoid function (Figure 8). No standards exist for fuzzy function selection. As a guideline, a membership function can be decided by consulting the experts in the relevant area, constructing formula, or learning it from the system performance. It is worth pointing out that with a simple formula and high computation efficiency, triangular and trapezoid membership functions are popular in fuzzy set partition, especially in control.

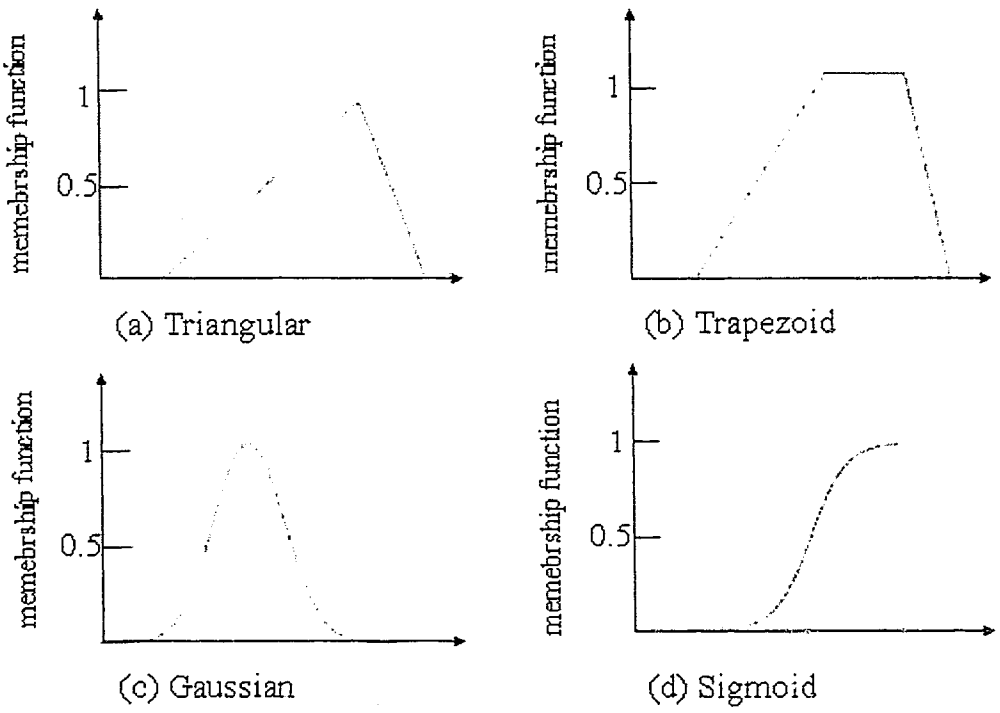


Figure 8. Fuzzy logic membership functions

b. Linguistic variables:

The linguistic variable concept is the foundation in fuzzy logic. It is the combination of both symbolic variables and numeric variables. The universe of discourse is expressed in a series of fuzzy sets, such as weak, medium and strong for the engine power example, so the linguistic variables can be described quantitatively with corresponding fuzzy terms, such as " engine power is strong". When a crisp value is given for the power, a degree of membership value can be found in a corresponding fuzzy set. This application will be seen later when fuzzy inference is presented.

c. Fuzzy rules:

The fuzzy rule concept was first introduced by Lofti Zadeh in 1973 [4]. A fuzzy rule is a statement that associates an antecedent in the "*if*" part with the consequent in the "*then*" part. It is also called the fuzzy "*IF-THEN*" rule. Unlike classical rules, a variable in a fuzzy rule's antecedent is expressed in fuzzy values, and the conclusion is also given in fuzzy values that should be defuzzified. The following two rules show the difference:

Conventional logic: *If the engine power is higher than 30hps, Then the propulsion force is 200Ns.*

Fuzzy logic: *If the engine power is strong, Then the propulsion force is high.*

The antecedent part of a fuzzy rule may include more than one condition, and the consequent part may also have more than one conclusion.

In the above discussion, it may have been noticed that both antecedent and consequent parts of a fuzzy rule are fuzzy values. But in real world engineering applications, the inputs are given in numerical values and the outputs are expected in crisp values. So a fuzzy rule must be interpreted. There are two common inference methods: Mamdani and Sugeno. In this study, the Mamdani inference method is used and the author will show this method with the following two inputs, one output system. The three fuzzy rules for the example are as follows:

1) *IF the engine power is medium AND the friction is low*
   THEN the speed is high.
2) *IF the engine power is strong OR the friction is normal*
   THEN the speed is normal.
3) *IF the engine power is low*
   THEN the speed is low.

The Sugeno inference method is introduced in [4]. Input fuzzification, rule evaluation, summation of fuzzy rules and defuzzification are the four steps in the Mamdani inference [4]:

Stage 1. Input Fuzzification:

In practical engineering applications, the inputs are given in numerical values. They are first converted to membership values in appropriate fuzzy sets, say for 18hps engine input, the membership values are 0.53 in a weak set and 0.3 in a medium set [Figure 9(a)].

Stage 2. Rule Evaluation:

The fuzzy operations are applied to antecedents of fuzzy rules for finding consequent membership values with the clipping method [Figure 9(b)-(d)].

For example, as shown in the figure, the "*AND*" is the operation for intersection-minimum, so the resulting consequent membership value for Rule 1 is $\min[\mu_{medium}(enginepower), \mu_{low}(friction)] = \min[0.3, 0.5] = 0.3$.

Stage 3. Fuzzy rules summation:

The clipped result of the individual fuzzy rule is unified to get one fuzzy result as shown in the lower part of Figure 9(e).

Stage 4. Defuzzification:

This is the final stage to get the crisp or numerical output from the fuzzy logic system for real world applications.

There are two common defuzzification methods: mean of maximum(MOM) and centre of gravity(COG). In this study, the COG is introduced. The formula for this method can be expressed as $COG = \frac{\int_a^b \mu_A(x)xdx}{\int_a^b \mu_A(x)dx}$ [4], where a and b are the upper and lower limits of the output universe. In practice, an approximation to

this integration is adopted, $COG = \frac{\sum_{x=a}^{b} \mu_A(x)x}{\sum_{x=a}^{b} \mu_A(x)}$. This means that the universe of discourse is divided into a limited number of points and the output is evaluated based on these points indicated by the arrow in the lower part of Figure 9(e)].

Figure 9. Fuzzy Rule Inference

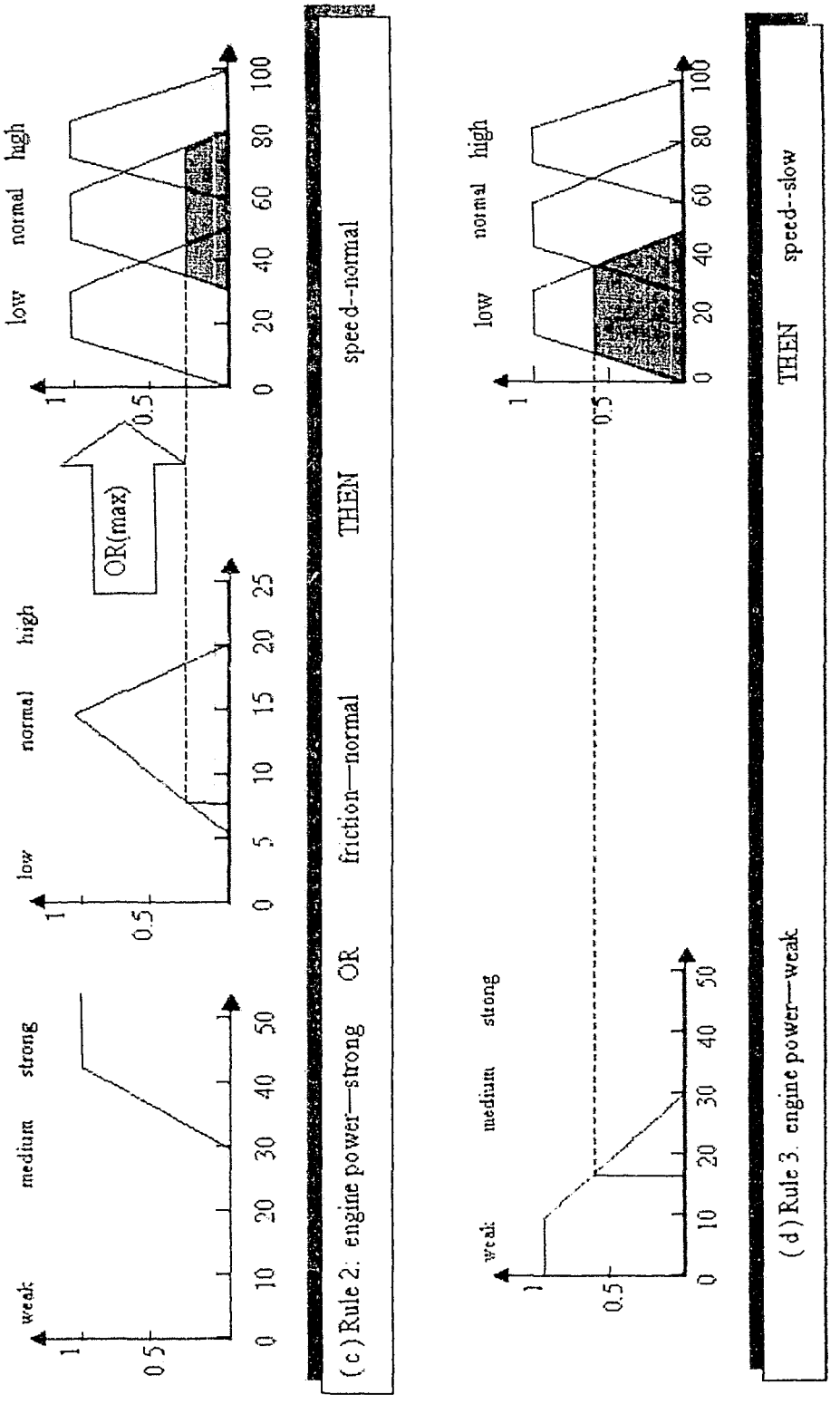


Figure 9. Fuzzy Rule Inference

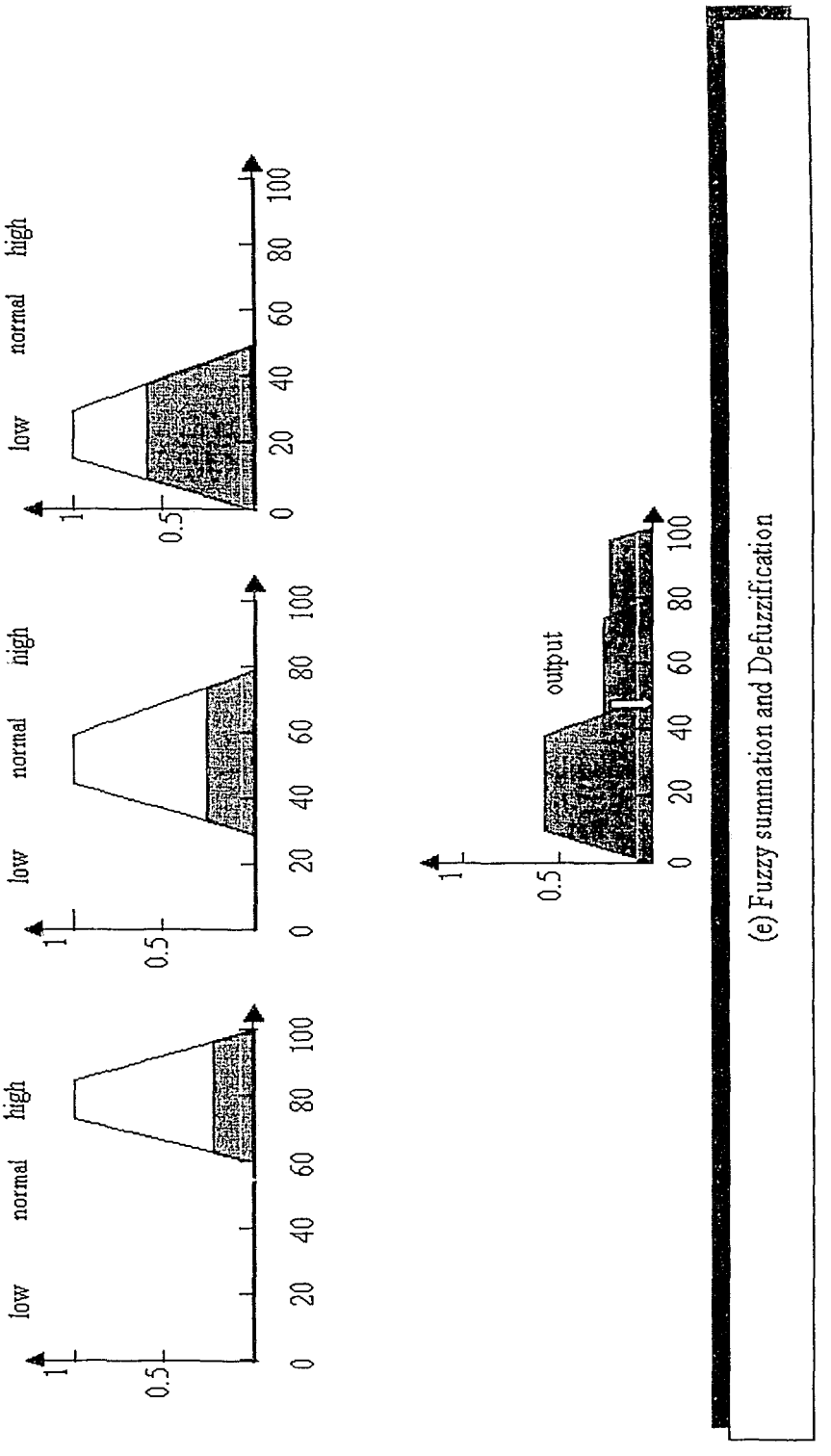


Figure 9. Fuzzy Rule Inference

1.3.3 Neuro-fuzzy Systems

Neural networks and fuzzy logic systems are both universal approximators and have seen a lot of successful real world applications. Both systems have their advantages and disadvantages. For neural networks, knowledge can be learned through backpropagation process, but it is difficult to explain neural networks, so it works as black box. For fuzzy logic systems, knowledge can be clearly expressed in fuzzy rules, but the fuzzy rules and the partition of the universe of discourse need expert knowledge, which limits its application.

It is natural to explore the possibility of combining these two artificial intelligent models to overcome their disadvantages. The hybrid system, normally called neuro-fuzzy systems, is first introduced by Lee and Lee [8] in the early 1970s. The neuro-fuzzy systems have a similar structure with multiple layer neural networks (Figure 10). The inputs are fed forward and the errors are backpropagated in the learning stage. As shown in Figure 10, a neuro-fuzzy system normally consists of one input layer, three hidden layers and one output layer.

Each layer has its particular function:

Layer one:

This is the input layer which communicates with external circumstances. The crisp inputs accepted externally are passed to the next layer. This layer does not process these crisp inputs.

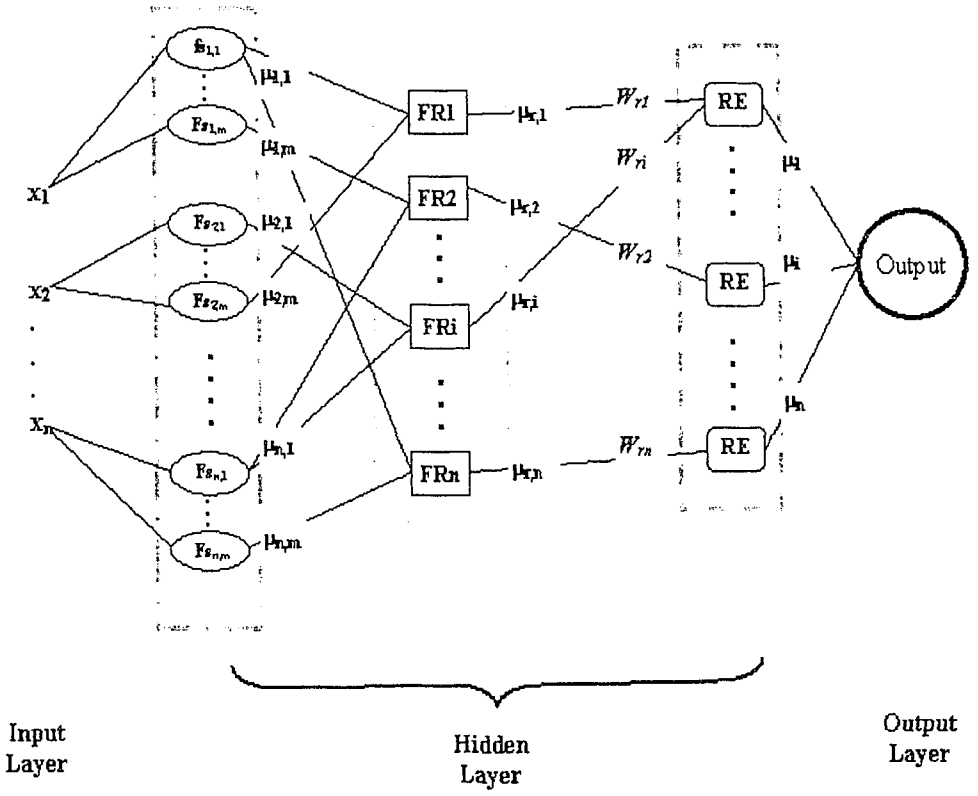


Figure 10. Neuro-fuzzy structure

Hidden layers:

The first hidden layer represents the antecedents of fuzzy rules. Its neuron number equals the total fuzzy sets of all the inputs in fuzzy rules. For the example used in section1.3.2, there are two inputs and each input universe is divided into three fuzzy sets, so six(2×3) neurons are included in this layer. When crisp inputs enter this layer, their membership values are decided. For example, 18hps engine power membership values are 0.3 in the weak set and 0.53 in the medium set. To decide the membership values, the triangular activation functions are used in this example. The parameters of the activation

functions can be adjusted in the learning stage in order to achieve expected neuro-fuzzy system outputs. The parameters effect on the membership value can be shown with Figure 11.

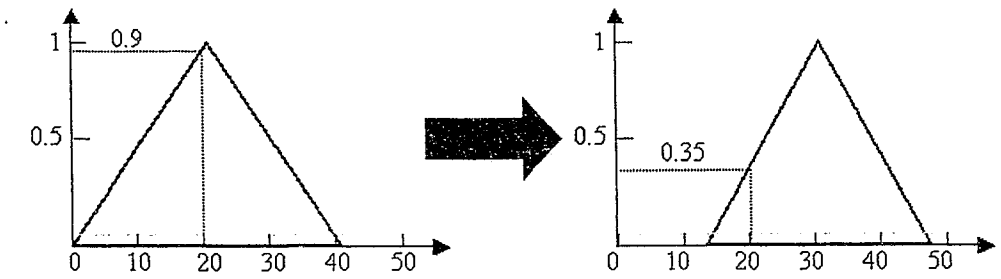


Figure 11. Activation function parameter effect

The second hidden layer represents the fuzzy rules with its neurons. Its neuron number equals that of the fuzzy rules. For this example, the hidden neurons number is three. The fuzzified results from the first hidden layer enter this layer to produce fuzzy rules output membership values. Fuzzy set operations will be applied if there is more than one antecedent in the fuzzy rules. In rule one of the example in section 1.3.2, the "AND"(min) operation is applied resulting in a membership value of 0.3.

Hidden layer three represents the membership function values of the fuzzy rules' consequent. If there is more than one input from the fuzzy rule layer, then the union fuzzy operation will be applied for the final membership function values.

Output layer: This layer produces the final crisp output using different defuzzification methods, such as COA or MOM.

The learning process of a neuro-fuzzy system can be the same as that of feedforward backpropagation neural networks. When a set of training inputs is provided to a neuro-fuzzy system, the output is produced and then compared with its expected value. The difference or the error will be backpropagated in order to modify the activation function and the weights.

Expert knowledge can be applied to a neuro-fuzzy system as in a standard fuzzy logic system; or appropriate sets of representative data can be used to train a neuro-fuzzy system to produce a set of fuzzy rules.

1.4 Previous works in flutter control

To the author's knowledge, there are no reports on the application of neuro-fuzzy controller in flutter suppression. The author will hence review some of the active flutter suppression works with the other methodology.

Haley and Soloway [9] validated the effectiveness of generalized predictive control in flutter suppression. This generalized predictive control was implemented to control the trailing-edge of an airfoil in a transonic wind tunnel. The experimental results showed that this control method with control parameters derived from the simulation model can increase critical flutter speed significantly.

Mukhopadhyay [10] designed suppression control laws using classical, linear quadratic Gaussian and minimax techniques. The control laws, based on the analytical state-space equations of the wing, adopted trailing-edge accelerator signals as flutter suppression inputs. The tests in a wind tunnel showed that the designed control laws could suppress flutter.

Crowther and Cooper [11] used neural networks in flutter prediction during flight flutter testing in aircraft manufacturing. The neural networks method prediction results were compared with the prediction results of a statistical method and the noise effect was also investigated. The authors concluded that the neural networks method showed improved accuracy under the noise-free condition. Under the noisy condition, neural networks prediction accuracy remained good if it was trained with noisy data, but the statistical method was unacceptably poor.

Scoot and Pado [12] were the first who presented experimental applications of neural networks on flutter suppression for an airfoil with a trailing-edge and upper- and lower-surface spoilers in a transonic wind tunnel. Neural networks were applied to flutter suppression in three ways. First, a neural network was trained to generate gains for every control law under different Mach number and pressure. The control laws with neural networks generated gains improved the flutter suppression over a fixed-gain control. Secondly, a neural network was used to produce flutter control signals based on the data derived from the plant model. The third flutter suppression application of neural networks was the

inverse model control. Both simulation and experimental results showed the successful performance of the last two methods in flutter suppression.

1.5 Objective of the thesis

Flutter can cause dramatic structure failure in a matter of seconds. Before the1930s, little was known about flutter. After that time, when airplanes were expected to achieve high speed and light weight, flutter analysis became more serious, especially for faster fighter aircraft.

Both passive and active methods can suppress flutter successfully. But passive suppression application is limited by its way on flutter suppression, and active flutter suppression can be applied in many different ways and will need complete understanding of airfoil behavior.

The present study intends to analyze a two-dimensional airfoil performance with flap as a controller and develop a neuro-fuzzy system to suppress the flutter and increase critical flutter speed. The neuro-fuzzy system will learn from the airfoil behavior. The fuzzy rules for flutter suppression will be produced by the neuro-fuzzy system automatically. The neuro-fuzzy results will be analyzed and compared with classical control methods.

1.6 Thesis overview

This study explores the neuro-fuzzy applications in airfoil flutter suppression. The significant point of this application is the attempt to develop a neuro-fuzzy system that can generate the effective fuzzy rule base automatically.

Chapter 1 includes two parts. The first part presents the theory on flutter and the previous approaches to flutter suppression. In the second part, the theory about neural networks, fuzzy logic and neuro-fuzzy system is introduced. The reasons for choosing neuro-fuzzy system are explained.

Chapter 2 deals with the airfoil motion equations. Eight ordinary differential equations(ODE) used for the Simulink® model are presented. The airfoil-flap combination model is built based on these ODEs under Simulink® environment.

The software(NEFCON) which is used in this study as the control part is introduced in Chapter 3. Detailed descriptions of NEFCON, NEFCON model and the complete model are available in this chapter.

Chapter 4 presents the fuzzy rule base generated by neuro-fuzzy system, the control result and the comparison of neuro-fuzzy result with other control schemes.

Chapter 5 provides the conclusion of this study and suggestions for future research.

Chapter 2 Governing Equations & Simulation Model

2.1. Mathematical model

Aeroelasticity is the theory dealing with the interaction between structures and aerodynamic forces. Two deformations, pitching and plunging, are considered in the flutter analysis of aircraft wings. Instead of analyzing a three dimensional wing, a three-degree-of-freedom (3DOF) airfoil with a flap is considered for preliminary study on neuro-fuzzy flutter suppression in this thesis.

The airfoil-flap combination (Figure 12) employed here consists of an airfoil and a trailing edge flap. The combination is mounted by a transitional and a torsional springs at the elastic axis. The plunging deflection denoted by h is measured at the elastic axis and is positive in the downward direction. The airfoil rotation about the elastic axis is represented by α and is positive nose-up. The flap is fixed to have only angular deflection (β) about the flap hinge, positive for the flap tail down. The airfoil-flap combination has a chord of $2b$. The elastic axis is located at a distance ba_h from the mid-chord, the mass center of the entire airfoil is located at a distance bx_α from the elastic axis and the flap

hinge is located at a distance bc_β from the mid-chord. The mass center of the flap is at a distance bx_β from the flap hing . All distances are positive toward the trail edge.

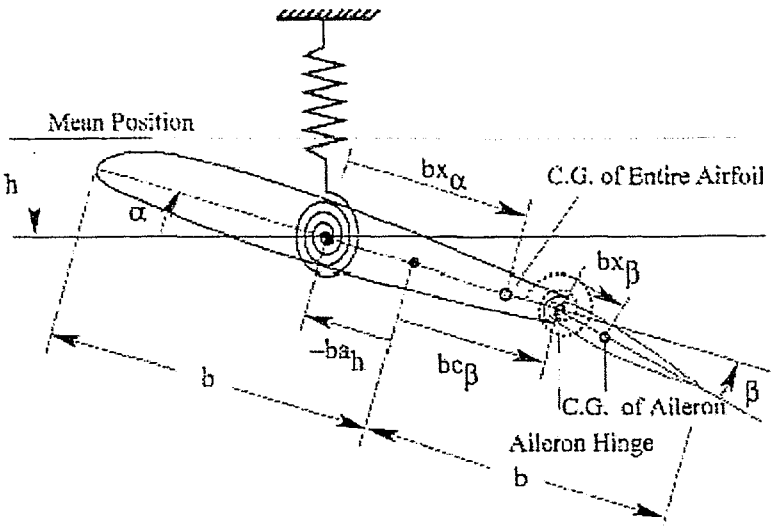


Figure 12. Three degree-of-freedom airfoil-flap (Alighanbari, 2000)

For active flutter suppression of the airfoil in this study, the control command rotation β_c should be applied to the flap. This β_c command can be derived from different control laws, such as classical feedback control techniques, linear quadratic Gaussian theory, eigenspace techniques and the aerodynamic energy method. In this study, the following rotation command [13] will be applied to the flap. The control result will be compared with that of the neuro-fuzzy control algorithm:

$$\beta_c = \left[\mathrm{T}_{c\xi}, \mathrm{T}_{c\alpha}\right]\begin{Bmatrix}\xi\\ \alpha\end{Bmatrix} + \frac{\alpha_T s^2}{s^2+2\zeta k_n s+k_n^2}\left[\mathrm{T}_{G\xi}, \mathrm{T}_{G\alpha}\right]\begin{Bmatrix}\xi\\ \alpha\end{Bmatrix} \qquad (2\text{-}1)$$

where α is the airfoil pitch angle, $\xi = h/b$ is nondimensional plunge displacement, and $T_{c\xi}$, $T_{c\alpha}$, $T_{G\xi}$, $T_{G\alpha}$ are control gains.

The denominator, $s^2+2\zeta k_n s+k_n^2$, is the transfer function for a second order system with damping ζ and natural frequency k_n. In practical control, the α_T, ζ and k_n parameters which decide the damping amount should be determined to optimize the control system. The transfer function equation (2-1) can be converted to the following time domain form [13]:

$$(T_{c\xi}+\alpha_T T_{G\xi})\xi''+(T_{c\alpha}+\alpha_T T_{G\alpha})\alpha''-\beta_c''+2\zeta k_n(T_{c\xi}\xi'+T_{c\alpha}\alpha'-\beta_c')+k_n^2(T_{c\xi}\xi+T_{c\alpha}\alpha-\beta_c)=0 \quad (2\text{-}2)$$

where β is the flap angle and $(\,)'$ denotes differentiation with respect to nondimensional time τ.

2.2. ODE and Control Model

Governing aeroelastic equations of the airfoil-flap combination are given as a set of ordinary differential equations (ODEs) by Alighanbari [14]:

$$m_1\xi''+m_2\alpha''+m_3\beta''+m_4\xi'+m_5\alpha'+m_6\beta'+m_7\xi+m_8\alpha+m_9\beta+m_{11}y_1+m_{12}y_2=0 \quad (2\text{-}3)$$

$$n_1\xi''+n_2\alpha''+n_3\beta''+n_4\xi'+n_5\alpha'+n_6\beta'+n_8\alpha+n_9\beta+n_{11}y_1+n_{12}y_2=0 \quad (2\text{-}4)$$

$$q_1\xi''+q_2\alpha''+q_3\beta''+q_4\xi'+q_5\alpha'+q_6\beta'+q_8\alpha+q_9\beta+q_{11}y_1+q_{12}y_2=-q_{13}\beta_c \quad (2\text{-}5)$$

$$-a_w\xi''-a_w(\frac{1}{2}-a_h)\alpha''-a_w\frac{T11}{2\pi}\beta''+y_1'=a_w\alpha'+\frac{a_wT10}{\pi}\beta'-b_wy_1 \quad (2\text{-}6)$$

$$-c_w\xi''-c_w(\frac{1}{2}-a_h)\alpha''-c_w\frac{T11}{2\pi}\beta''+y_2'=c_w\alpha'+\frac{c_wT10}{\pi}\beta'-d_wy_2 \quad (2\text{-}7)$$

T's are given in *Appendix I;*

m's, n's and *q's* are given in *Appendix II;*

a_w, b_w, c_w, d_w and a_h are given in *Nomenclature*;

y_1 *and* y_2 are auxiliary variables.

The β_c is the flap control command used to suppress flutter. We can now use Simulink® to solve these second order ODEs.

To illustrate the Simulink® model building method and procedure, we will use the following simple second order ODE :

$$9y'' + 23y' + 45y = 80 \tag{2-8}$$

This equation can be rewritten as:

$$y'' = \frac{1}{9}(-23y' - 45y + 80) \tag{2-9}$$

Now Simulink® must be started and a window for building a new model should be open. Then "Integration" blocks from the "Continuous"; "Scope" block from the "Sink" and the "Gain" blocks from the "Math" in the "Simulink" library are dragged into the new model window. These blocks should be connected according to the given equation (Figure 13). In this new model, one "Integration" block represents y" and the output from this block is equal to y' or input to y' block; the other "Integration" block is y'. The above equation (equation 2-9) means that the input of y'' is the right side, so the connection shown in the lower part of Figure 13 can be built and connected to y'' block input. The output from y'' block is connected to y' block for final y result. A

scope block is connected to the y' block output port for graphic result. For the function explanations on these blocks, one may refer to Matlab® Help.

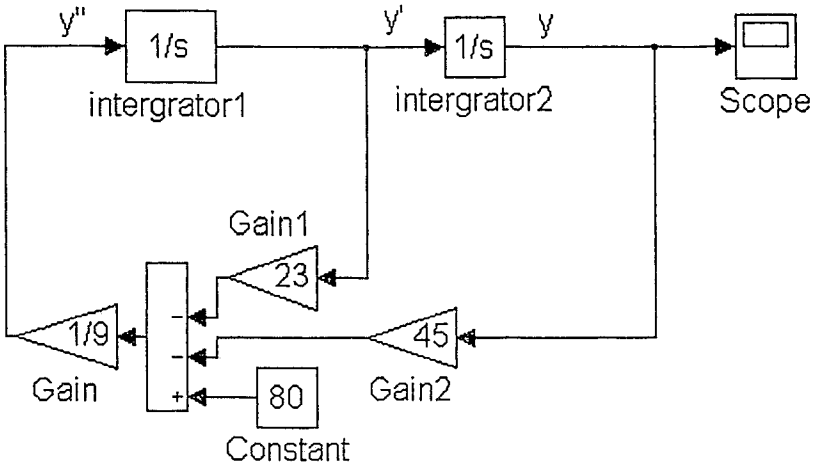


Figure 13. Sample ODE model in Simulink®

In a similar way, the author develops the Simulink® model for the airfoil-flap combination (equation 2-3~equation2-7). The detailed procedure is given in *Appendix III*. The model is shown in Figure 14.

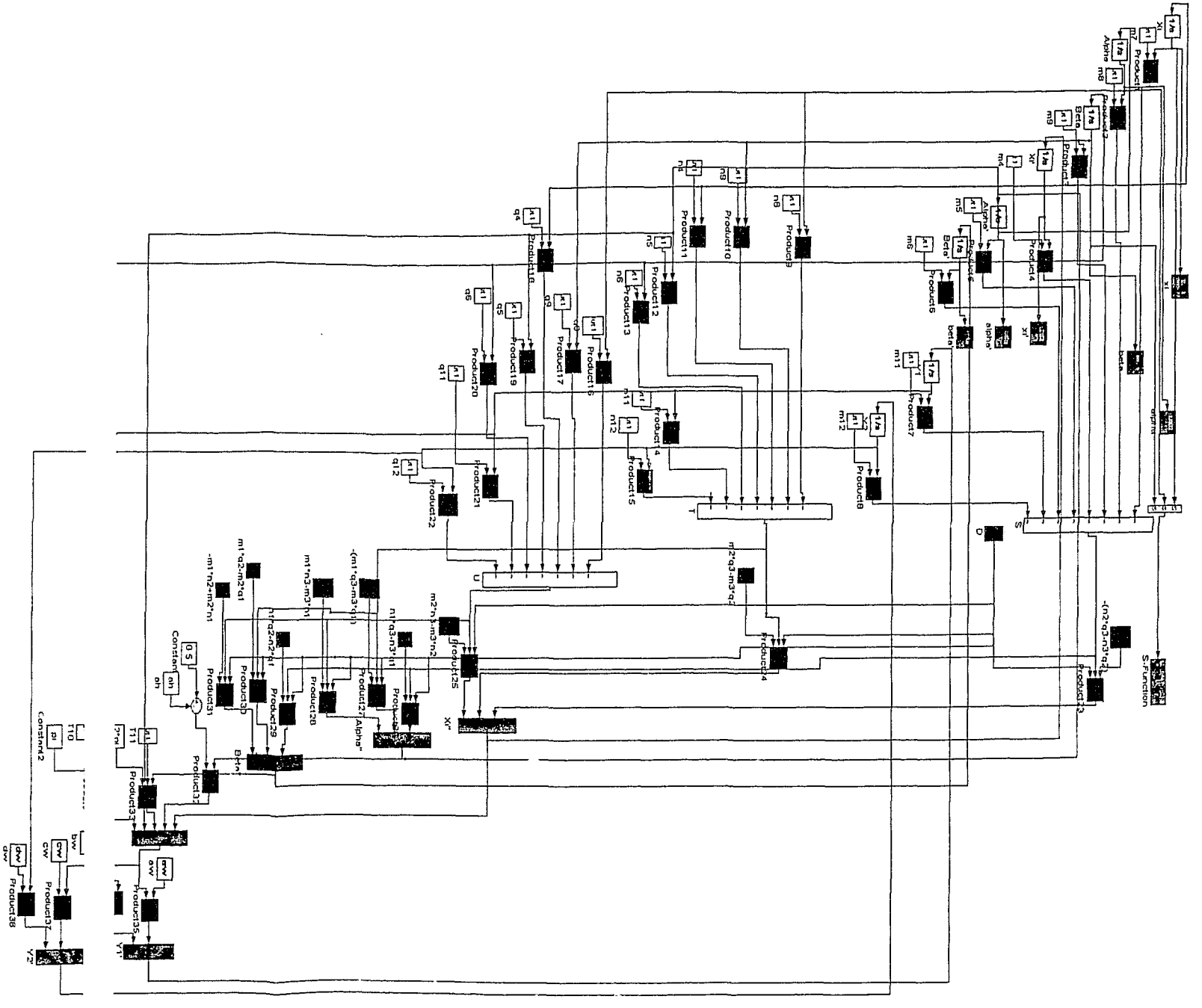
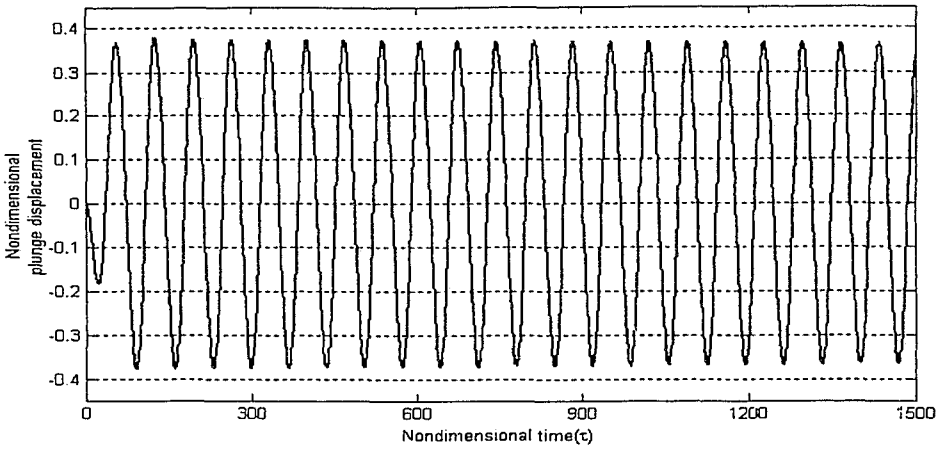
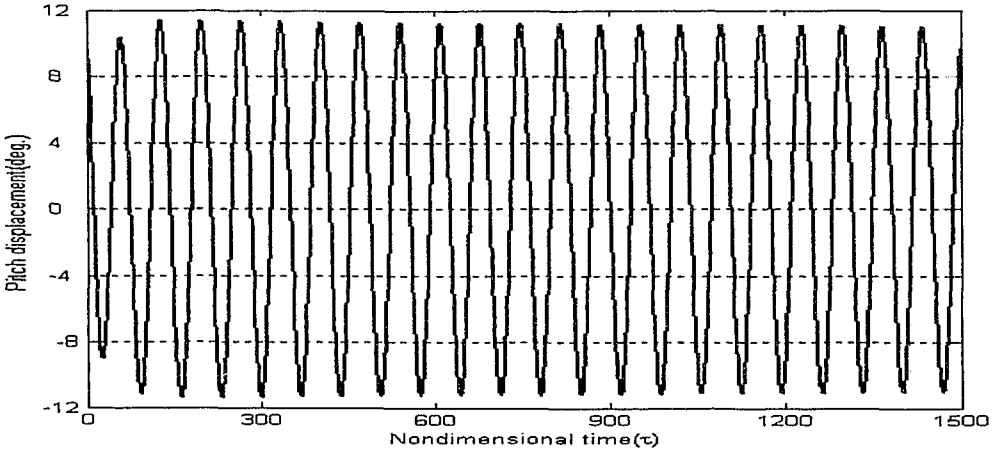


Figure 14. Airfoil-flap combination Simulink® model

With this Simulink® model, the critical flutter speed of the airfoil-flap combination is found to be $U = 6.139$(Figure 15), which is in agreement with the eigenvalue method. The proposed neuro-fuzzy controller will be included later in this Simulink® model for flutter suppression.

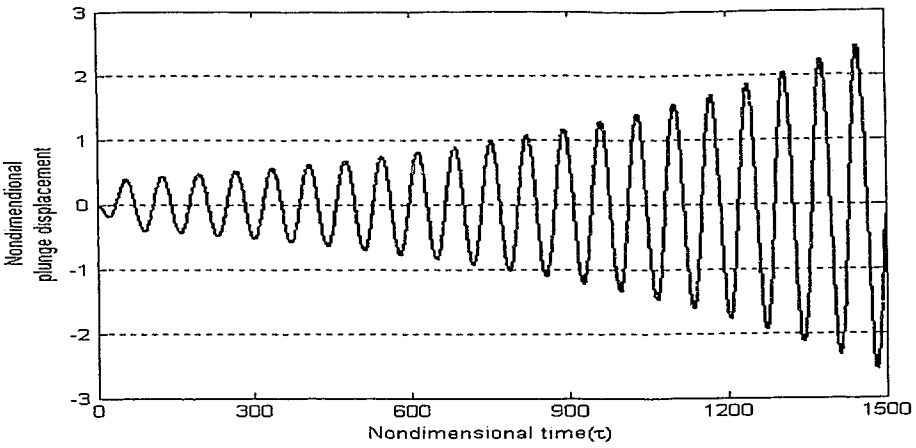


(a) nondimensional plunge displacement at flutter speed ($U = 6.139$)

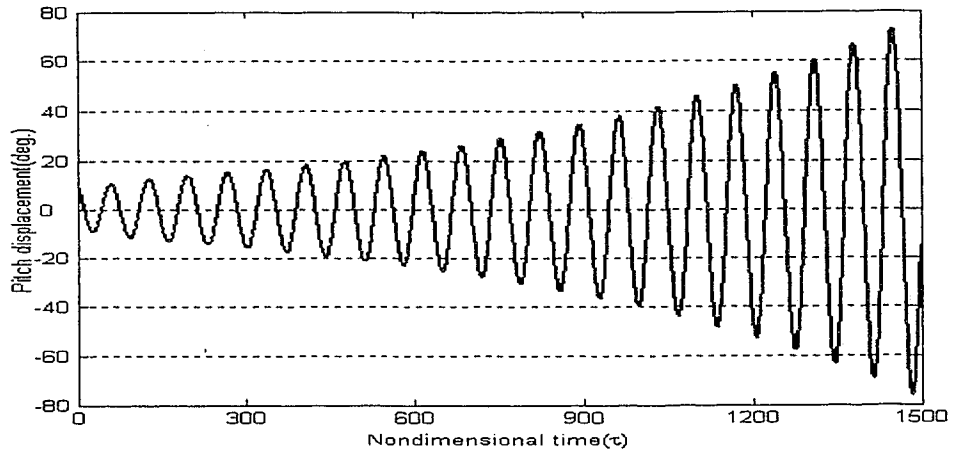


(b) pitch displacement at flutter speed ($U = 6.139$)

Figure 15. Airfoil displacement without controller



(c) nondimensional plunge displacement above flutter speed ($U = 6.16$)



(d) pitch displacement above flutter speed ($U = 6.16$)

Figure 15. Airfoil displacement without controller

Chapter 3 NEFCON and Flutter Control

3.1 NEFCON introduction:

Artificial neural network is inspired by the human brain and has been successfully applied in many fields, such as classification, recognition, and controls. Neural network's distinguishing characteristic is that it can learn and is adaptive to a changing environment. Neural networks can learn from given information to adjust itself to produce required results. Unfortunately, people don't know how it does this. Fuzzy logic can represent human knowledge in a natural way. The expert knowledge for fuzzy logic can be expressed in fuzzy "*IF-THEN*" rules. Expert knowledge on a system must be available for fuzzy logic application [4, 5].

The disadvantage of neural networks is that it works as a black box, which means it can not be interpreted. For fuzzy logic, its application is limited by the availability of expert knowledge. So, the combination of these two systems, neuro-fuzzy system, which adopts the neural networks' learning ability and

fuzzy logic's knowledge representation, can overcome the shortcomings. The combined neuro-fuzzy system can learn and express itself in a set of rules.

Neuro-Fuzzy Controller, which will be referred to as NEFCON, is a neuro-fuzzy system that adopts reinforcement learning. Under reinforcement learning algorithm, the controller does not rely on the accurate solution of the control problem, instead the controller can learn to control a system by the reinforcement signal or error signal. NEFCON controller uses an error signal in the learning process. It is supposed that the desired state of system is known. The difference between the current state of system and the desired state of system can be expressed in a suitable way. The knowledge of a certain problem can be used by NEFCON for practical control, or it can also learn from scratch if no knowledge of a problem is available. After learning the behavior of the system which will be controlled, NEFCON can express the control law for the system in a set of fuzzy rules.

The NEFCON system, like other neuro-fuzzy systems, has the multiple-layer-perceptron (MLP) structure (Figure 16). The NEFCON, that is used here for explanation, has two inputs (ξ_1 and ξ_2), five rules (R_1~R_5) and one output (η). The inputs to the neuro-fuzzy system are the state variables of the system to be controlled, and the output of the NEFCON system will be applied to the system as a control command. The neurons in the only hidden layer correspond to the fuzzy rules. For instance, R_2 can be:

$$If\ \xi_1\ is\ A_1^{(1)}\ and\ \xi_2\ is\ A_2^{(1)}\ then\ \eta\ is\ B^{(1)}.$$

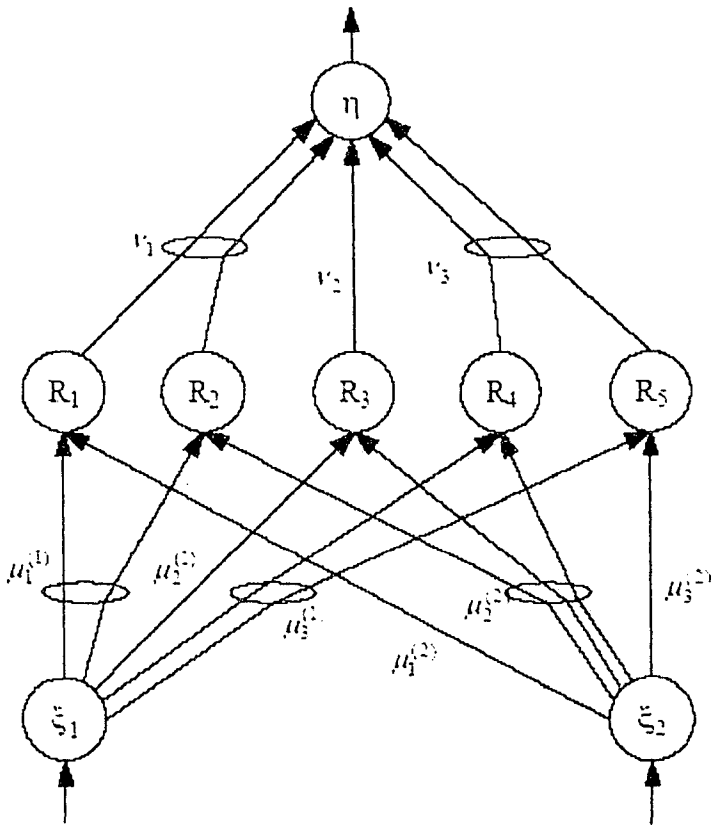


Figure 16. NEFCON structure diagram (Nurnberger, Nauck and Kruse)

where $A_1^{(1)}$, $A_2^{(1)}$ *and* $B^{(1)}$ are the fuzzy sets of inputs ξ_1, ξ_2 and output η respectively.

The connections between the input layer and the fuzzy rule layer, and between the fuzzy rule layer and the output layer are fuzzy weights. This is shown in Figure 17. For classical neurons, the connections between two layers are crisp numbers used as weights. For NEFCON, these weights are actually fuzzy membership function values decided by the inputs and membership function in corresponding fuzzy sets.

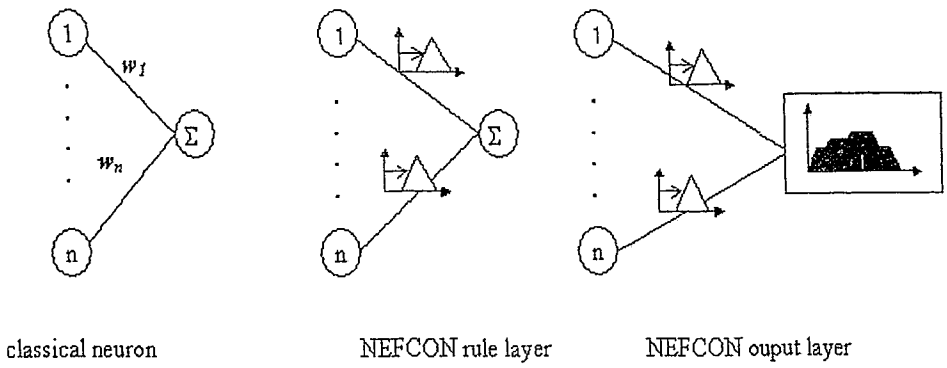


Figure 17. Classical neuron and NEFCON comparison

As shown in Figure 16 by the ellipse, some connections share the same weights. For example, the input to rule R_4 and R_5 connections share the same $\mu_3^{(1)}$, which means the linguistic variable for input ξ_1 in the antecedent part of R_4 and R_5 comes from the same fuzzy set. The shared weights will be modified identically for all fuzzy rules in the learning process if necessary.

3.2 NEFCON learning algorithm

When control laws are not available for a system control problem, the NEFCON controller can be used as it can learn from scratch and express the control law in a set of fuzzy rules. This procedure begins by learning the behavior of the system to be controlled. The NEFCON learning algorithm includes two stages: learning fuzzy rules and learning fuzzy sets.

a. Learning fuzzy rules:

This stage is dedicated to building an initial fuzzy rule base. There are two classes of rule learning: decremental rule learning and incremental rule learning.

The author will present the incremental rule learning here. Details on the decremental rule learning are available in [16].

The incremental rule learning consists of two phases. In phase one, the highest membership function values are calculated in corresponding fuzzy sets for each newly entered set of state variables. If there are no rules in the existing fuzzy rule base with these variables in the antecedent, an output is "guessed" based on present error. The fuzzy rule produced is added to the fuzzy rule base. This phase is repeated the number of stipulated times. In phase two, this input set is propagated through the NEFCON system to update the existing fuzzy rules contribution. This can also be iterated the number of predefined times. In this phase, fuzzy rules which are used less than a certain percentage are deleted from the rule base.

b. Learning the fuzzy sets:

It is assumed that the fuzzy rule base which is built the way described in last section is adequate, and the possible lack of performance is due to an inappropriate rule organization. The purpose of the learning fuzzy sets is to modify the membership functions of the NEFCON system for optimal performance.

The difference between the desired output and the current output will be used as a reinforcement signal. This process is very similar to the backpropagation process in the neural networks. The reference signal is backpropagated from the output layer to the input layer. Only the parameters of the membership function activated in the fuzzy rules at present stage will be modified to reduce the error. This indicates that those fuzzy rules which are not applied to the current state will not be changed. At the same time, the fuzzy rules with higher activation are assumed to have a bigger influence. This will also affect the modification degree together with the learning rate and the error value. This process will be repeated for a predefined number of times.

3.3 Control model with NEFCON

The NEFCON controller is added to the airfoil-flap combination model, which was built in section 2.2 based on the set of airfoil aeroelastic equations of motion (equation 2-3~2-7) under the Simulink® environment.

3.3.1 New NEFCON creation

The author simplifies the Simulink® model built in section 2.2 by creating an airfoil subsystem(named "Airfoil Model", orange part in Figure 18) with outputs: ξ, α, β, $\dot{\xi}$, $\dot{\alpha}$ and $\dot{\beta}$, and input: β_c. The "Nefcon System" coming with the NEFCON software is added into the airfoil window (Figure 18). The two small windows of Figure 18 are used for new NEFCON controller parameters setting. "Nefcon Signal Generator" is selected as NEFCON signal generator is

used, and "Nefcon Learning System" will be used to learn the airfoil-flap control. According to equation (2-2), the author will use four inputs--ξ, α, $\dot{\xi}$ and $\dot{\alpha}$—for flutter suppression. The final result is shown in the upper part of Figure 18.

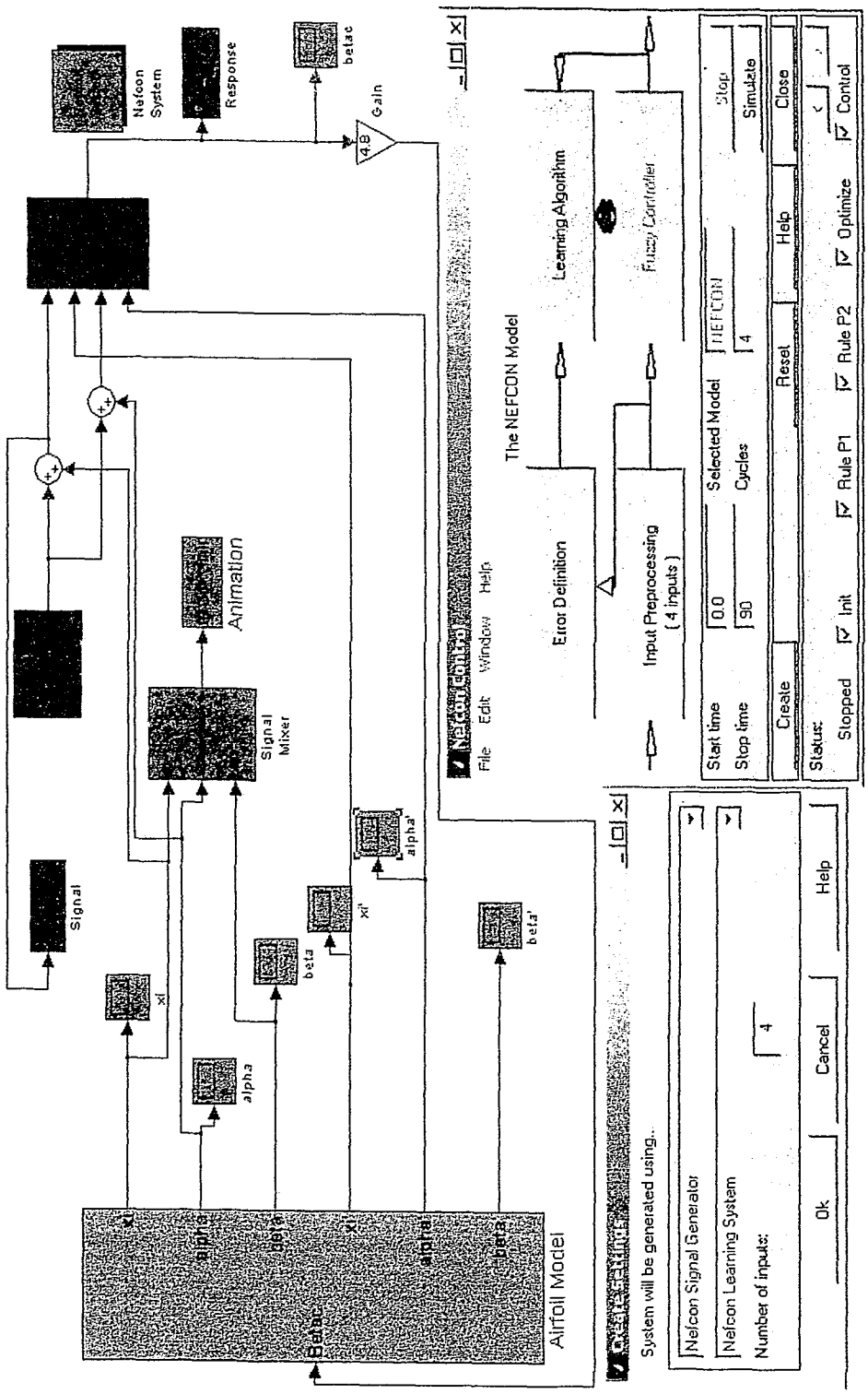


Figure 18. NEFCON model creation

3.3.2 NEFCON learning set-up

The NEFCON controller is now ready to learn to control the airfoil system. Before starting the NEFCON controller to learn the airfoil system behavior, one should set up the learning environment first. Figure 19 shows the "Nefcon Control" window for set-up.

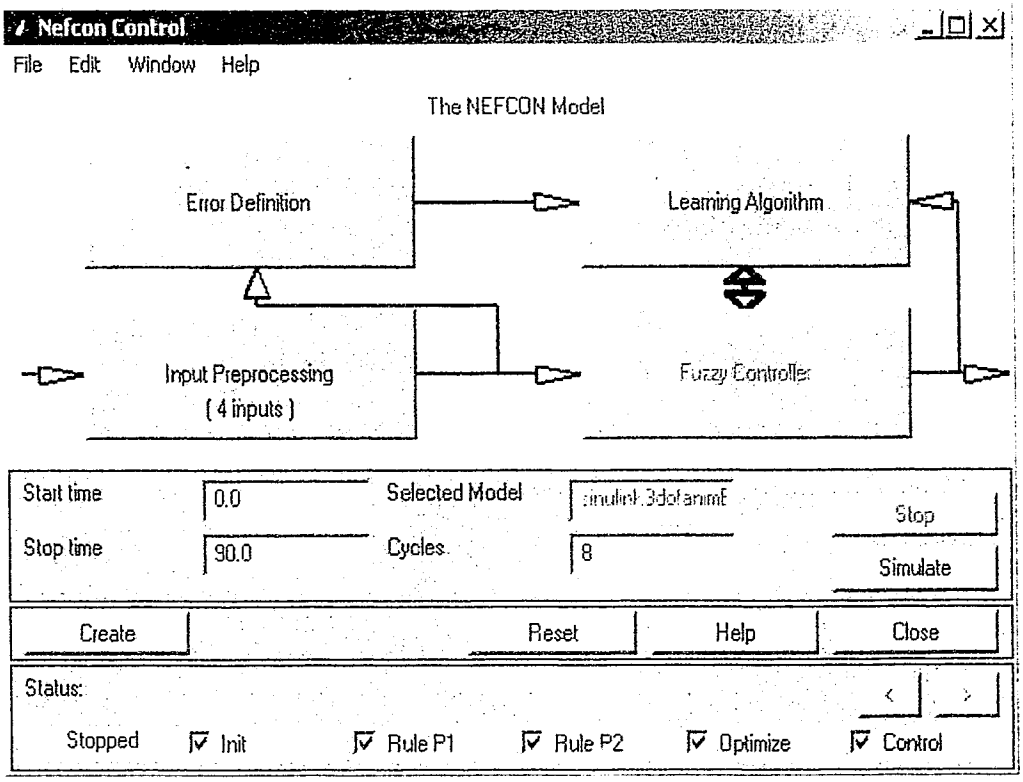


Figure 19. Nefcon Control window

3.3.2.1 Error definition (Figure 20)

There are four choices for error definition:

a. Using a Fismatix:

This method uses a fuzzy inference system for error definition. As there are four inputs into the controller, the inference system will be too complicated, so it is not used here.

b. Using Input One as error:

The input one means the difference between the desired state value and the current response. In this airfoil-flap system, there are many inputs and this input can not be modified, so it is not selected.

c. Bounded error calculation:

The NEFCON bounded signal generator is used to calculate the error. This method is not used here as the author does not select bounded signal generator which is used for manual error set up.

d. Using a Matlab Function:

In this method, the error range can be easily controlled and hence the learning process. Thus, it is adopted in this study (Figure 20).

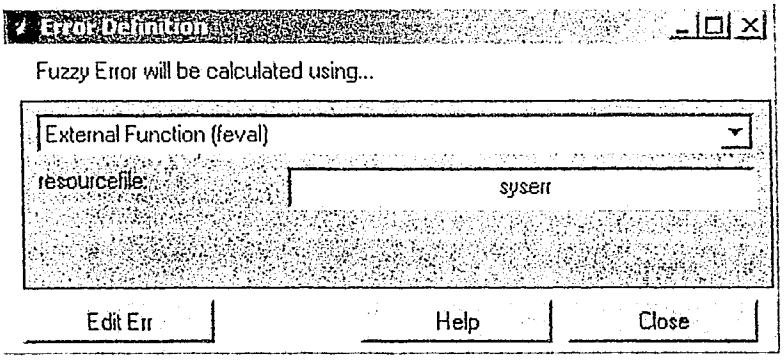


Figure 20. Nefcon error definition

3.3.2.2 Input processing (Figure 21)

This window is designed to define the input number, input range and gains. For the flutter suppression problem, the sets are:

	Number	Range				Gain
Input	4	No.1	No.2	No.3	No.4	1
		-2~2	-1.5~1.5	-2~2	-1.5~1.5	
Output	1	N/A	N/A	N/A	N/A	1

Table 2. NEFCON input setting

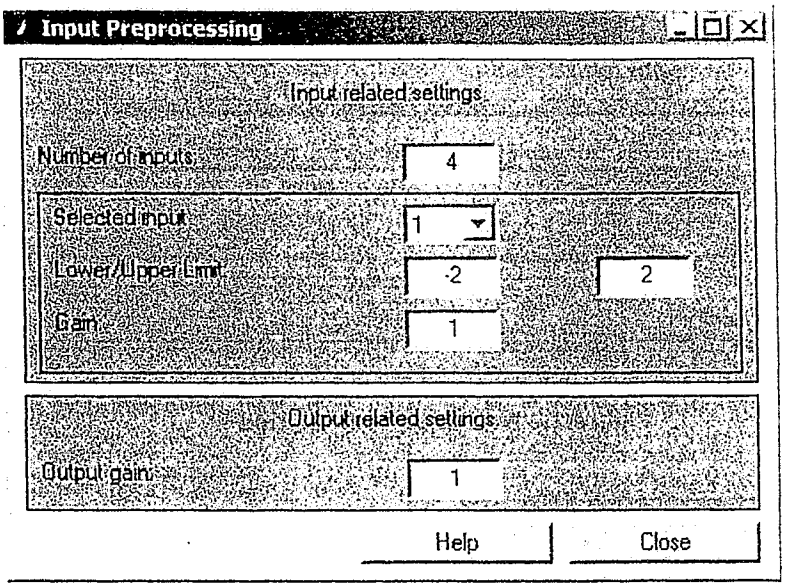


Figure 21. Nefcon input processing

3.3.2.3 Learning algorithm (Figure 22)

This window is divided into two parts: rules learning and optimization.

Rule learning:

There are four types of learning algorithms which can be selected:

a. Bottom-Up Learning:

The universe of discourse of inputs and outputs is evenly partitioned. The equally distributed membership functions are assigned to each partitioned set. The number of the partition and the membership function types can be defined in the "Partitioning" column. The rule base is initially empty. If any prior rule base is loaded through "fismatrix" column, then its membership functions and rules will be the initial data and more rules will be added during the learning

stage. Under this learning algorithm, the fuzzy rule base is initially empty and new rules are added by learning the system to be controlled, so this guarantees that every rule is active in system control and is selected in this thesis.

The learning rate is set to 1; rules used less than 5% will be deleted in phase two. The cycle time of this airfoil-flap combination is found to be very small, so the learning time is set to 90 seconds. The cycle numbers in two phases are all set to one.

b. NEFCON:

The universe of discourse of inputs and outputs is evenly partitioned. The equally distributed membership functions are assigned to each partitioned set. The number of partitions and the membership function type can be defined in the "Partitioning" column. If prior rule base is loaded through "fismatrix", then its membership function and rules will be the initial data. Otherwise, a full set of rules will be produced based on the number of inputs, output and fuzzy sets.
The "bad" rules will be deleted during the learning process. As this is decremental algorithm which is not selected for this study, so this "NEFCON" is not used.

c. Disable_Initial Fismatrix:

The fuzzy rule learning process is disabled. A valid initial rule base must be loaded in the "fismatix". As the effective control rules are not available, this algorithm can not be adopted in this study.

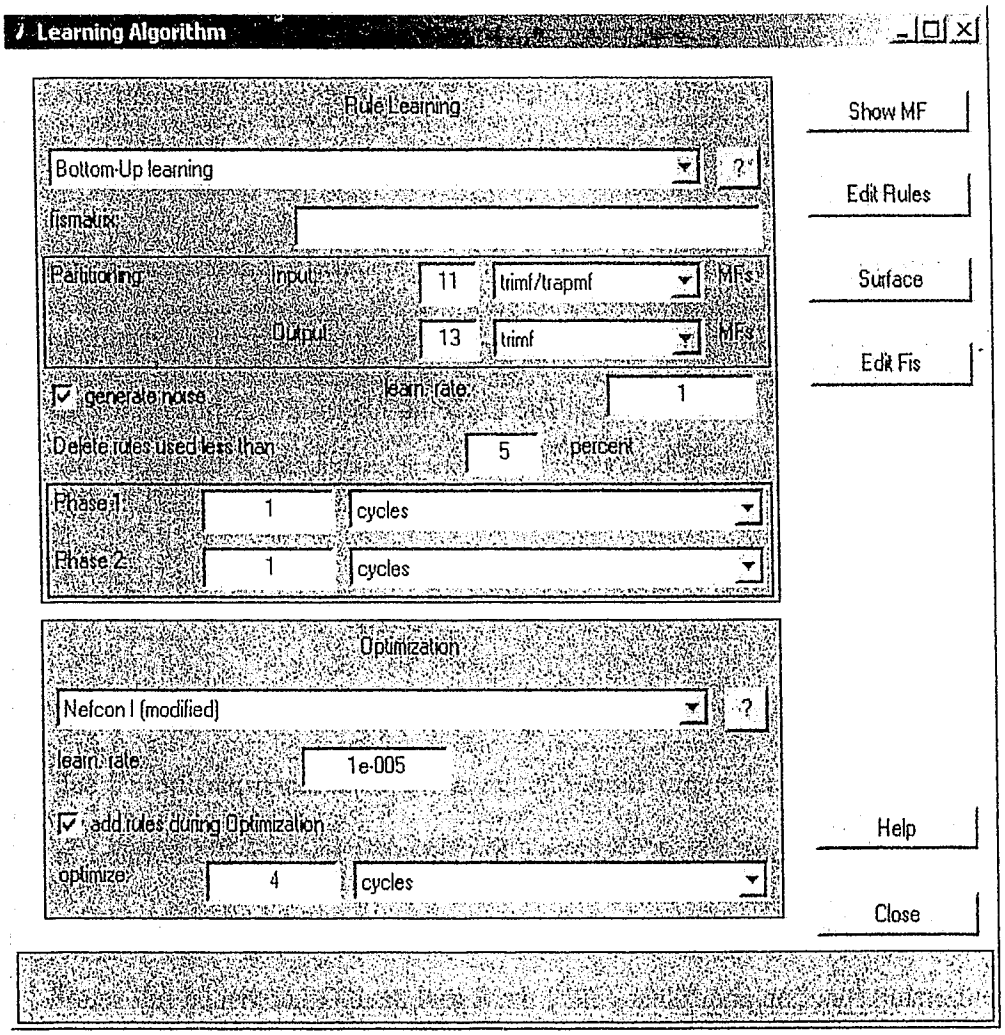


Figure 22. Nefcon learning algorithm

d. Disabled_Current Fismatrix:

The fuzzy rule learning process is disabled. And the current rule base will be used as initials. As the initial fuzzy rules are not available, this algorithm can not be used here.

Optimization:

The fuzzy rule base is optimized by shifting and/or modifying the support of the fuzzy sets. There are three possible optimization methods. The NEFCON I(modified) optimization method is introduced as before in section 3.2.b. It uses the current error to modify the membership function. The NEFCON II optimizes the membership functions based on the error change trend. If the optimization is disabled, then no optimization will be made. The NEFCON I method is adopted with four cycles at 10^{-5} learning rate, where the learning rate will decide fuzzy sets shifting and/or modifying degree.

3.3.3 NEFCON training

Up to now, all the learning parameters have been decided, except the airfoil-flap structural data, initial motion data and speed. All these data are included in an m-file called *initm3dof* and should be connected to the model by the Matlab® command:

set_param('airfoilaileron3dof', 'InitFcn', 'initm3dof')

where:

"airfoilaileron3dof": the name of the model to assign the initial value;

"initm3dof": the m-file contains the initial values to be assigned to the model.

After the learning environment for NEFCON is set up, the learning process can be ignited by "Simulate" (Figure 18). The cycle of the airfoil-flap combination oscillation is very small, so the learning time is set to the 90sec. During this

period, the NEFCON system will get enough motion information about the airfoil-flap combination. Another point to be mentioned is that after a number of trials on NEFCON learning, it is found that the produced flap command(β_c) should be magnified by a gain before applied to flutter suppression.

The NEFCON is trained at a speed higher than the critical flutter speed. The fuzzy rules produced at higher speed can successfully suppress flutter that happened at a low speed after the amplifier is adjusted.

The final fuzzy rule base is given in the standard Matlab® Fuzzy Controller window (Figure 23) with ten rules.

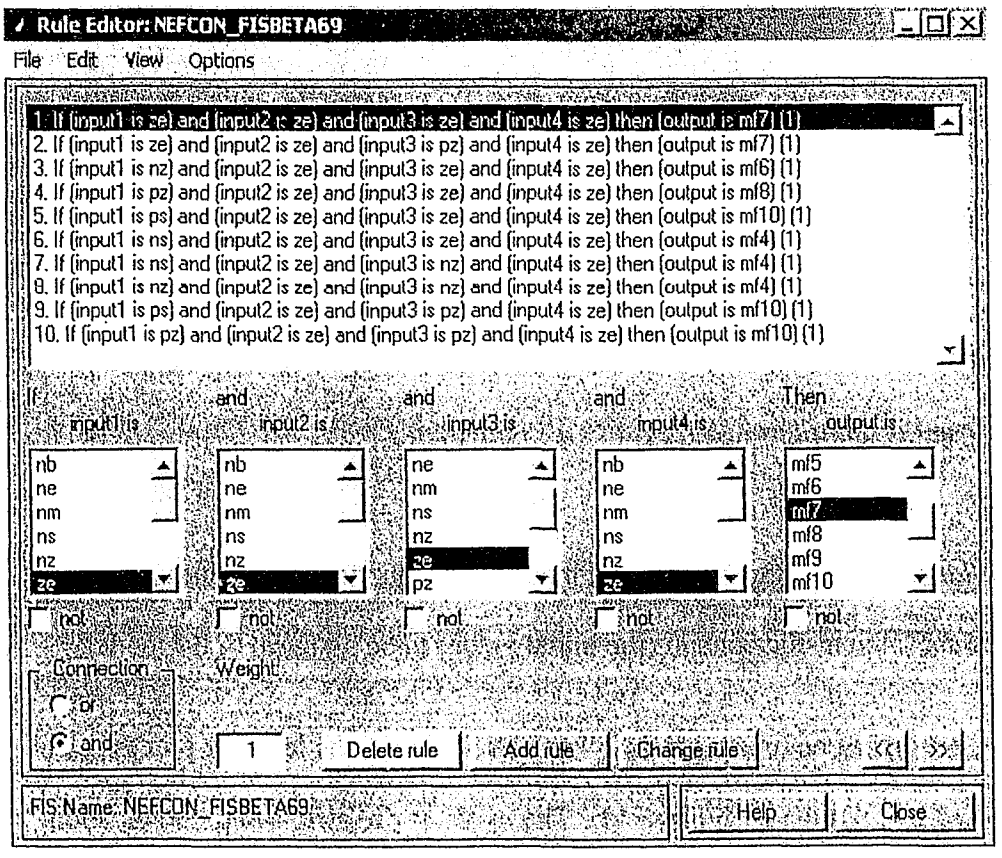


Figure 23. NEFCON produced fuzzy rule base

Chapter 4 Flutter Suppression

In this chapter, both the neuro-fuzzy control and the modified energy method (equation 2-2) are implemented in flutter suppression. The flutter suppression results of these two methods are compared. The neuro-fuzzy controller flutter suppression performance is tested under different conditions.

4.1 Modified Energy Method Flutter Control

The modified energy based control method (equation 2-2) is implemented in a Simulink® model (Figure 24). There are basically three parts in this model: controller (red), airfoil model (orange) which is built in Chapter 2 and results display (gray). The inputs to the controller are the acceleration ($\ddot{\xi}$), velocity ($\dot{\xi}$) and displacement (ξ) in plunge and acceleration (α''), velocity (α') and displacement (α) in pitch. These control inputs come from the airfoil-flap model (orange part). The output of the controller (β_c) passes to the airfoil model for flutter suppression. The airfoil model is designed to simulate the airfoil motion in plunge, pitch and flap rotation. The display part uses Simulink® scope blocks to show graphically the airfoil motion.

For an airfoil-flap combination, the modified energy method based controller (equation 2-2) is implemented with the following parameters which have been used by other researchers [13]: $\mu = 100$, $x_\alpha = 0.25$, $\alpha_h = -0.5$, $r_\alpha^2 = 0.25$, $r_\beta^2 = 0.0015$ and $x_\beta = 0.002$. For a control surface that covers 20% of the airfoil chord, $c_\beta = 0.6$; and $T_{c\xi} = 0$, $T_{c\alpha} = -1.86$, $T_{G\xi} = 4$ and $T_{G\alpha} = 2.8-4(\alpha_h+0.4)$. The coefficients in the transfer function (equation 2-1) are decided by optimization: $\alpha_T = 4$, $\zeta = 0.6$ and $k_n = 0.04$. No structural damping is considered in this study. The critical airfoil-flap flutter speed without control is found to be $U = 6.139$ (Figure 25). With control applied (equation 2-2), the critical flutter speed can be increased up to $U = 8.25$ (Figure26). Unstable condition occurs at critical flutter speed ($U = 6.139$) under modified energy control. However it is not a part of this study, the author does not explore this problem.

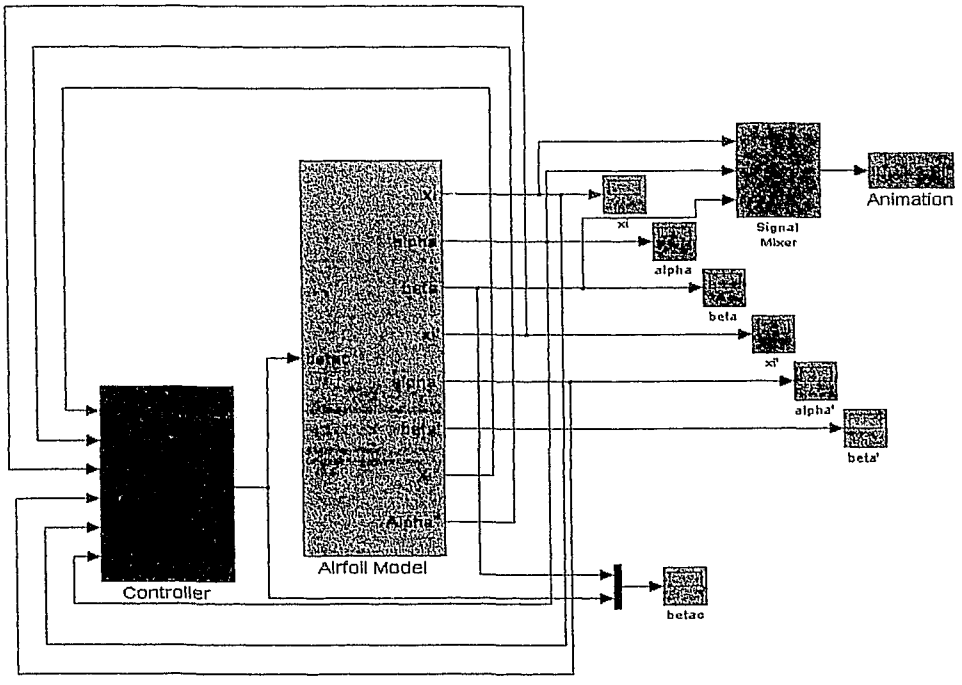
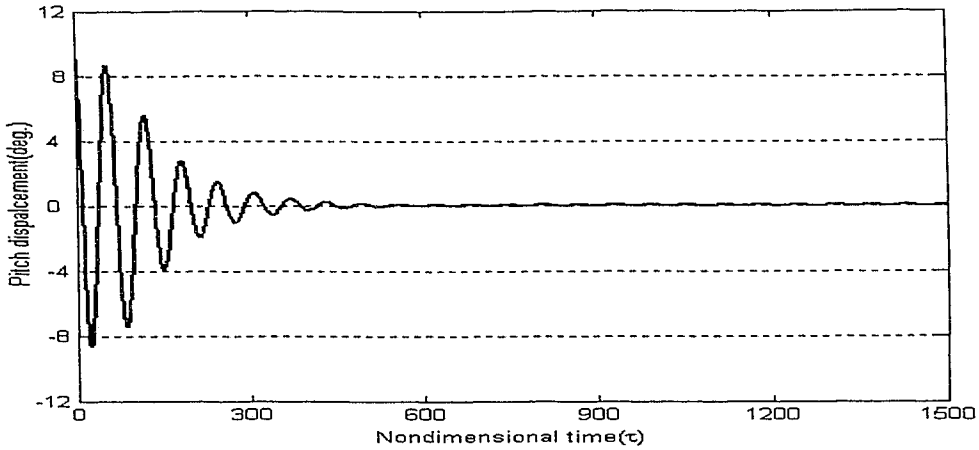
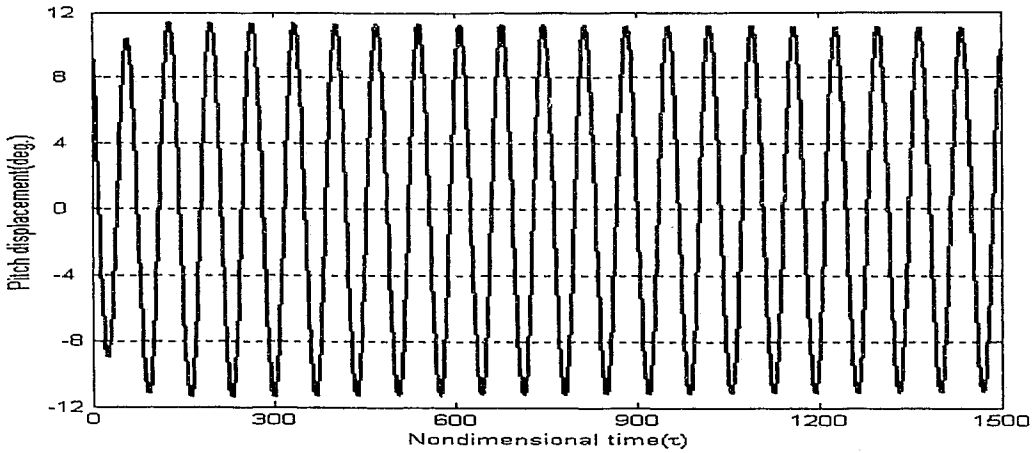


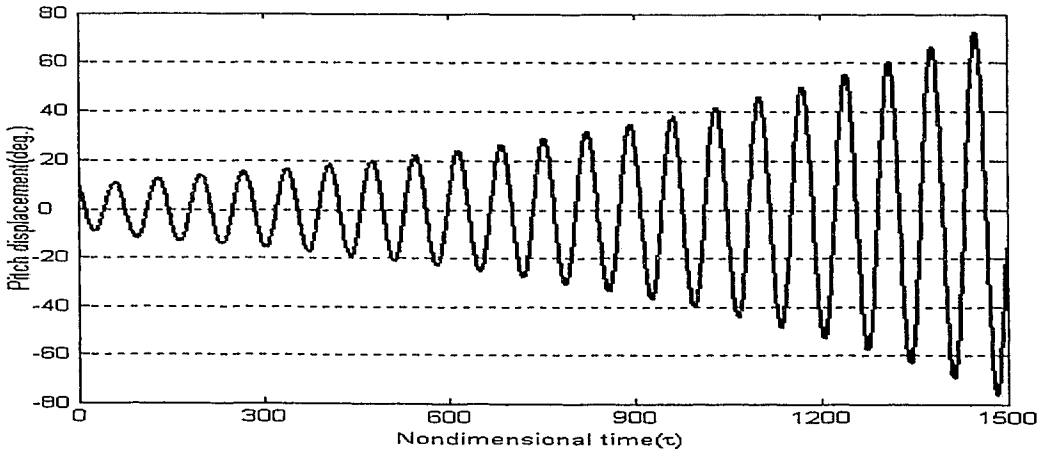
Figure 24. Energy method based flutter suppression scheme



(a) pitch displacement below flutter speed ($U = 6$)

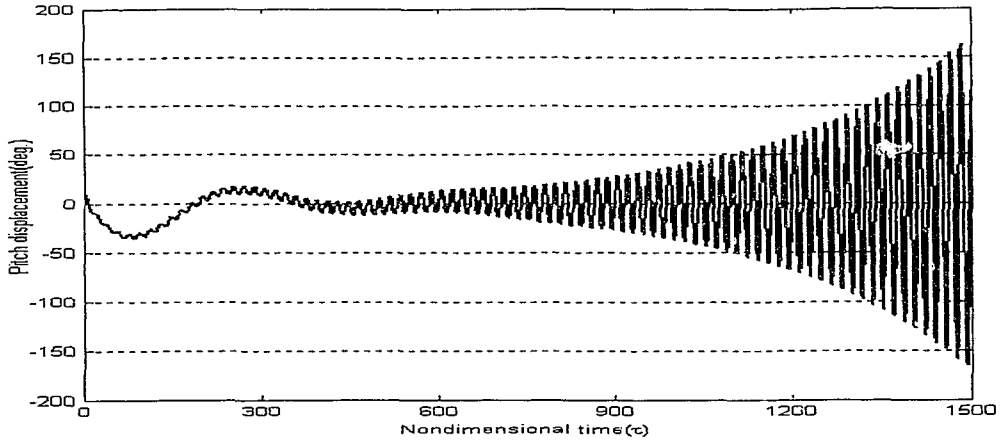


(b) pitch displacement at flutter speed ($U = 6.139$)

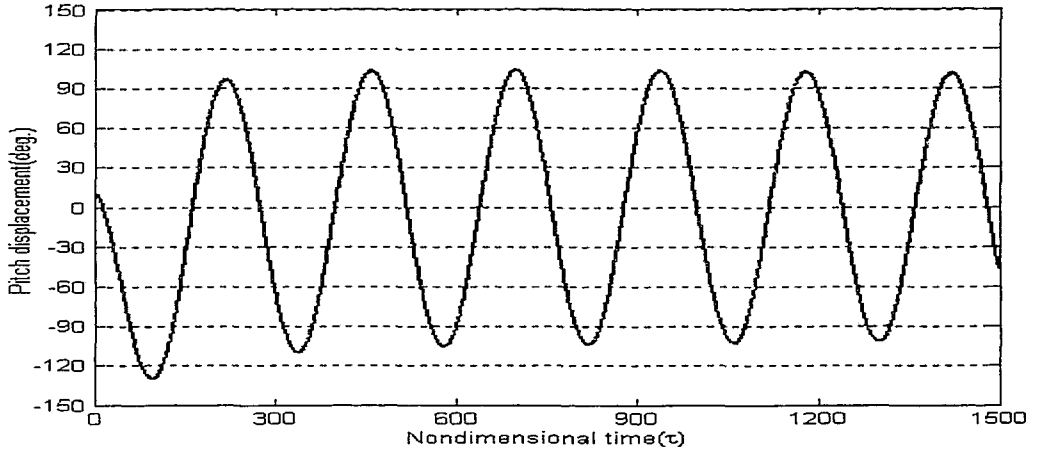


(c) pitch displacement above flutter speed ($U = 6.16$)

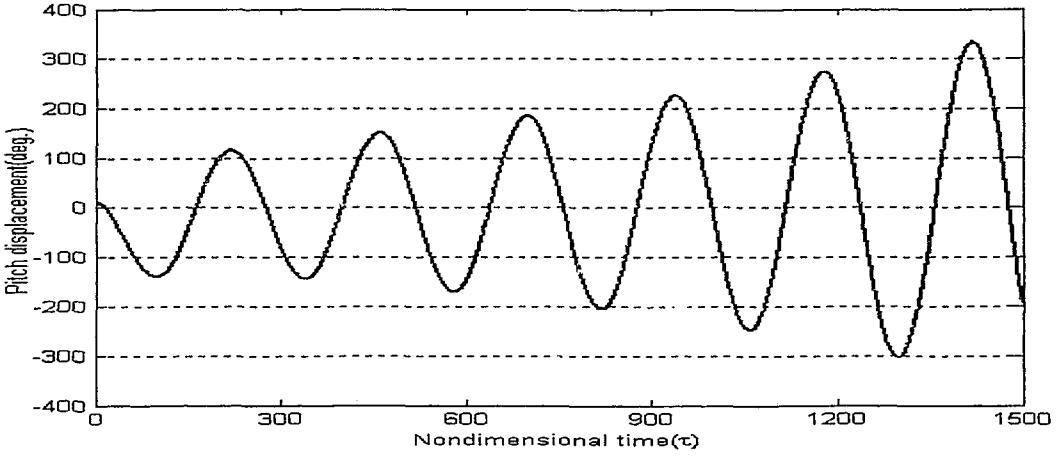
Figure 25. Airfoil-flap flutter speed at $U = 6.139$ without control



(a) pitch displacement below flutter speed ($U = 6.139$)



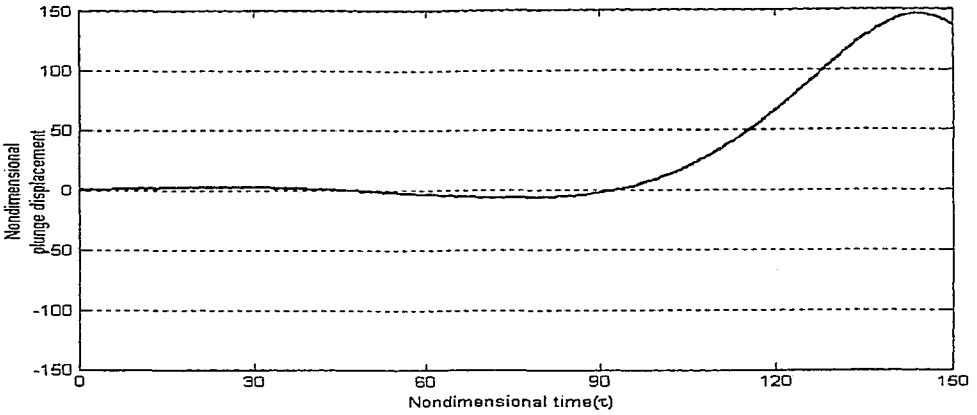
(b) pitch displacement at flutter speed ($U = 8.25$)



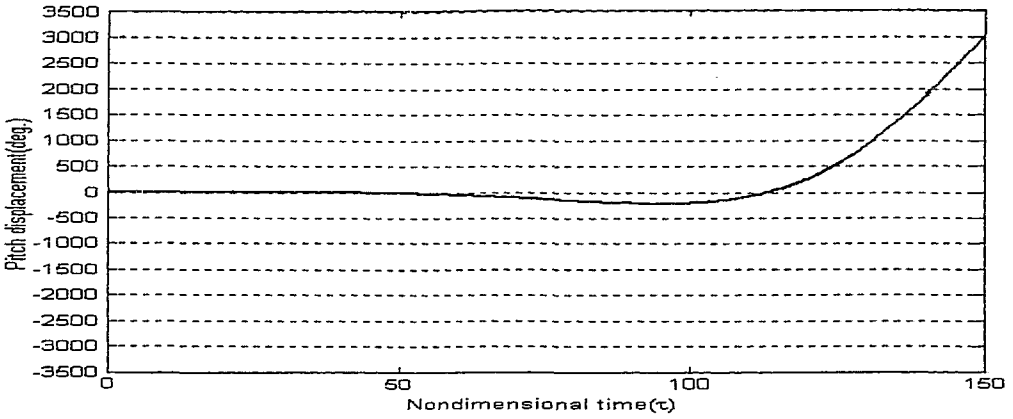
(c) pitch displacement above flutter speed ($U = 8.3$)

Figure 26. Airfoil-flap flutter speed at $U = 8.25$ with control

The following figures (Figure 27, Figure 28) show the airfoil motion both without and with modified energy control at $U = 8$.

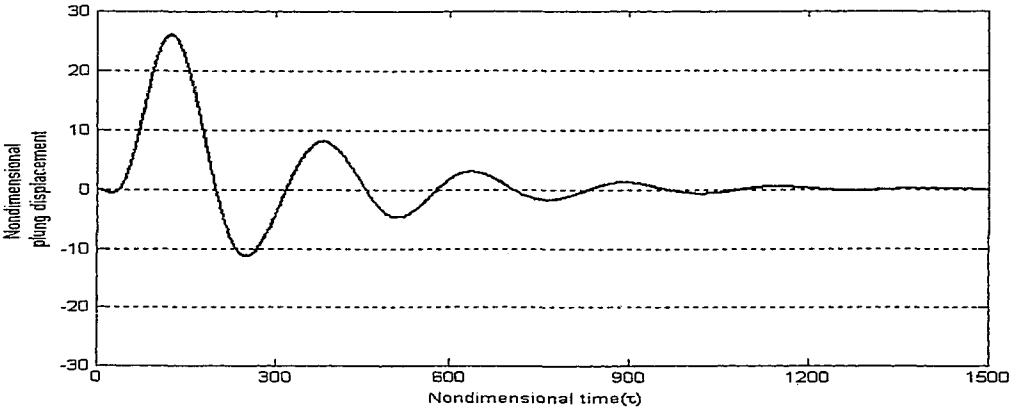


(a) nondimensional plunge displacement



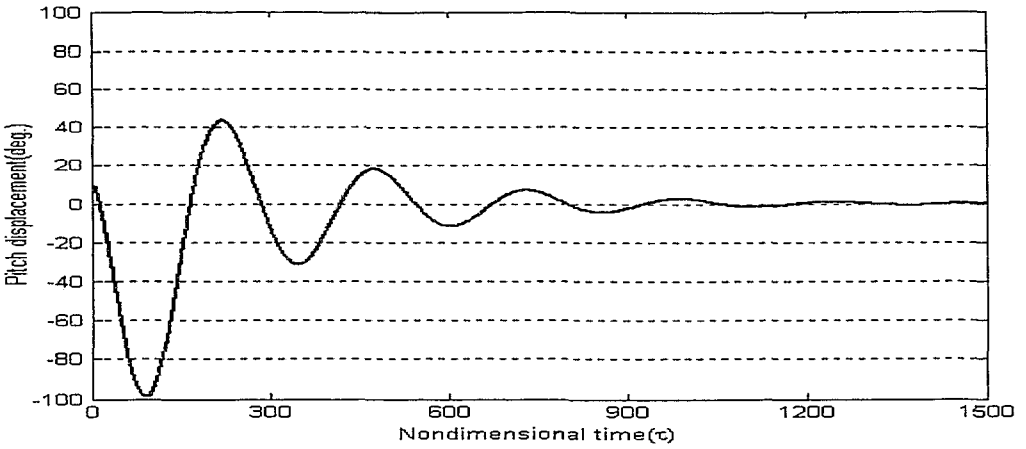
(b) pitch displacement

Figure 27. Airfoil motion without control at $U = 8$



(a) nondimensional plunge displacement

Figure 28. Airfoil motion with control at $U = 8$



(b) pitch displacement

Figure 28. Airfoil motion with control at $U = 8$

4.2 NEFCON flutter suppression

The NEFCON fuzzy rule base produced in the previous chapter has been optimized and is applied for flutter suppression.

The neuro-fuzzy rule base is loaded into the NEFCON controller for the airfoil-flap system. The learning and optimization processes are disabled.

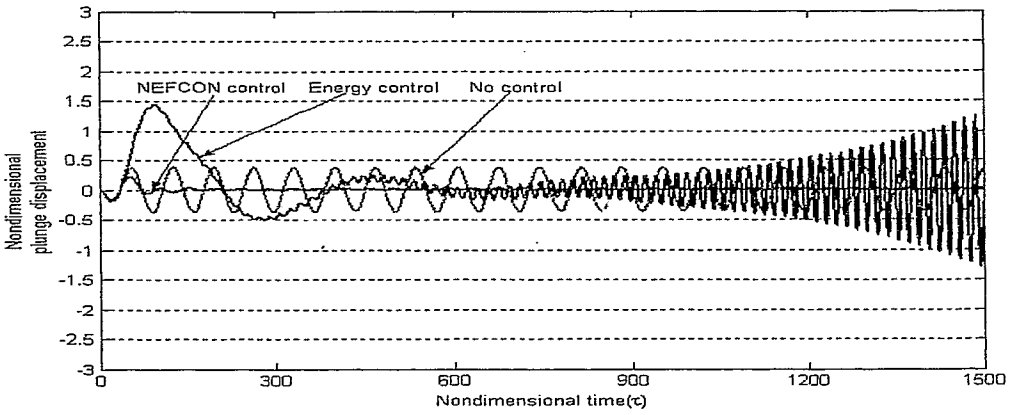
As the fuzzy rule base used here is produced at speed U =6.9, the magnifying factor must be optimized for other lower speeds. A set of amplifying factors applied to different speeds can be found in Table 3.

speed	6.14	6.3	6.7	6.9
gain	0.9	1.1	1.11	1.2903

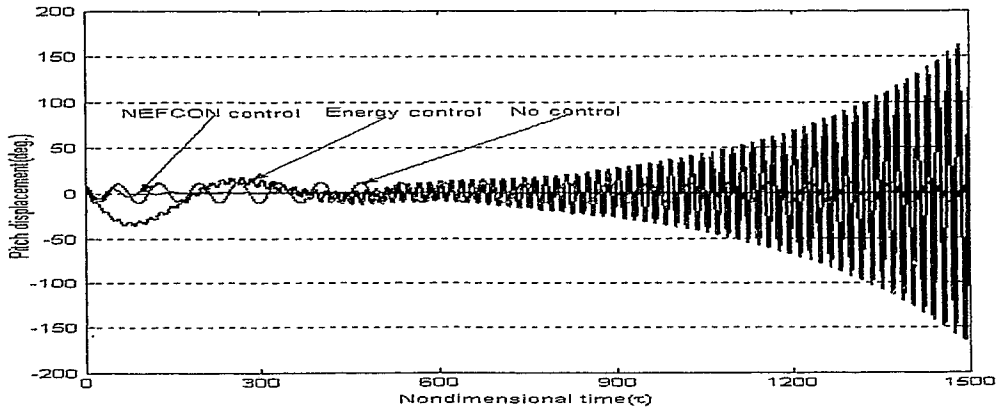
Table 3. NEFCON amplifying gains for different speeds

Based on the above set of data, Matlab polynomial fitting method is used to find an equation on speed-gain relation which can be expressed as: Gain = $4.8*U^3$-$94.1*U^2$+$614.3*U$-1335.5. This equation must be put in the Simulink® model of the NEFCON model.

The fuzzy rules can suppress flutter successfully and quickly for speeds up to 6.7. Figure 29 and Figure 30 show the performance of the Nefcon controller. When speed increases to U = 6.9, the suppression happens slowly and drift happens at τ = 1500 (Figure 31).

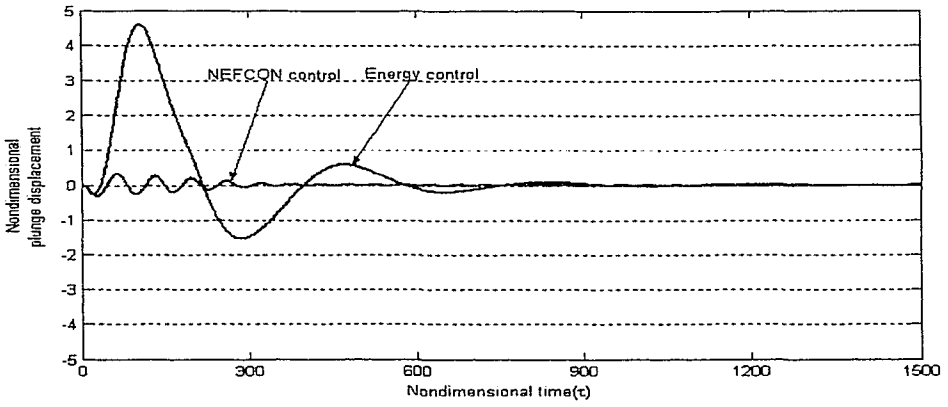


(e) nondimensional plunge displacement

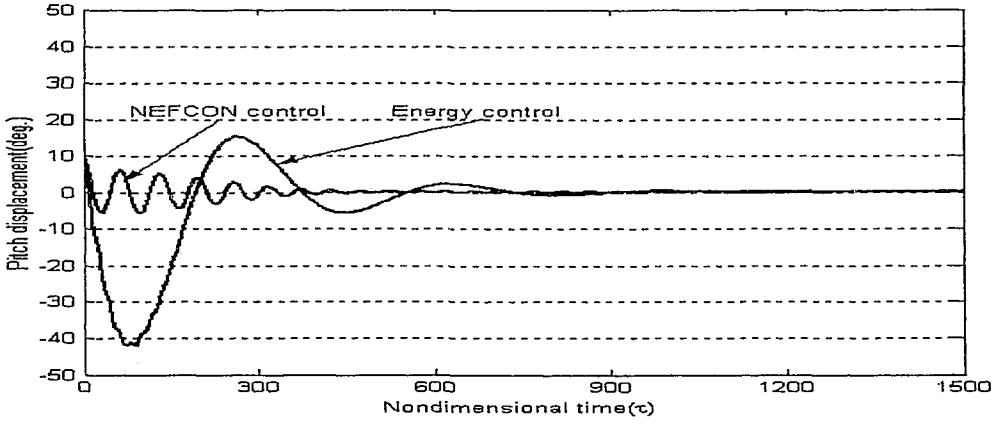


(b) pitch displacement

Figure 29. Flutter suppression (U = 6.139)

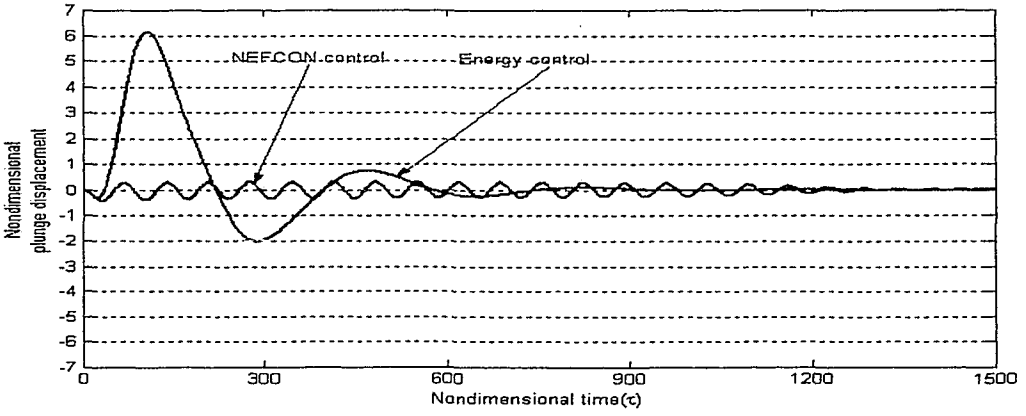


(a) nondimensional plunge displacement



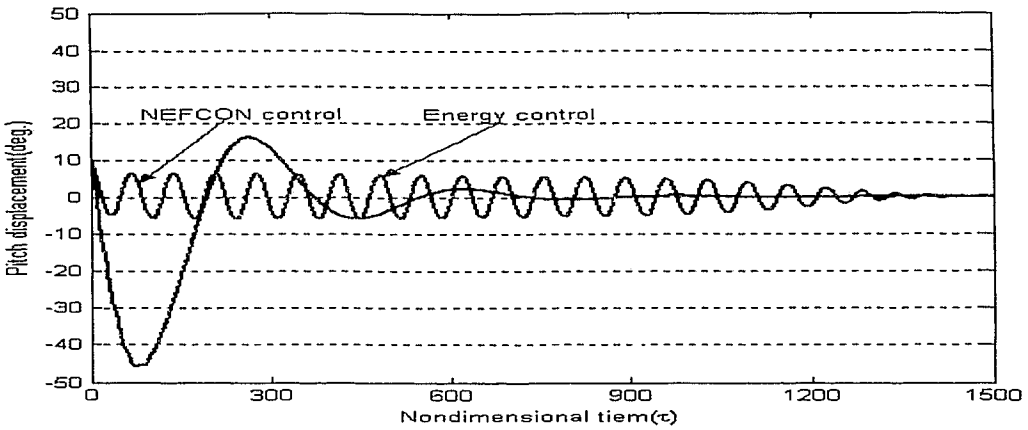
(b) pitch displacement

Figure 30. Flutter suppression ($U = 6.7$)



(a) nondimensional plunge displacement

Figure 31. Flutter suppression ($U = 6.9$)



(b) pitch displacement

Figure 31. Flutter suppression ($U = 6.9$)

4.3 Neuro-fuzzy control and energy method control comparison

Both of the control methods can successfully suppress flutter. With the modified energy control method, the plunging displacement amplitude will first jump to a higher level; whereas under the NEFCON control, the plunging displacement amplitude is maintained within a small range and then will be suppressed (Figure 29⊔ Figure 31). So under the modified energy method, the airfoil-flap combination needs to be stronger to resist the higher plunge deformation.

The energy method can suppress flutter at much higher speeds such as $U = 8$ (Figure 28), but not the NEFCON method.

The significant result from this study is that the neuro-fuzzy control methodology can be applied to flutter suppression successfully without any

prior control command knowledge required. The NEFCON controller adopted in this study can learn itself from the motion of the airfoil-flap to produce a set of fuzzy rules for flutter suppression. The NEFCON controller can then use these fuzzy rules to suppress the flutter successfully. The NEFCON control method should be considered first in flutter suppression.

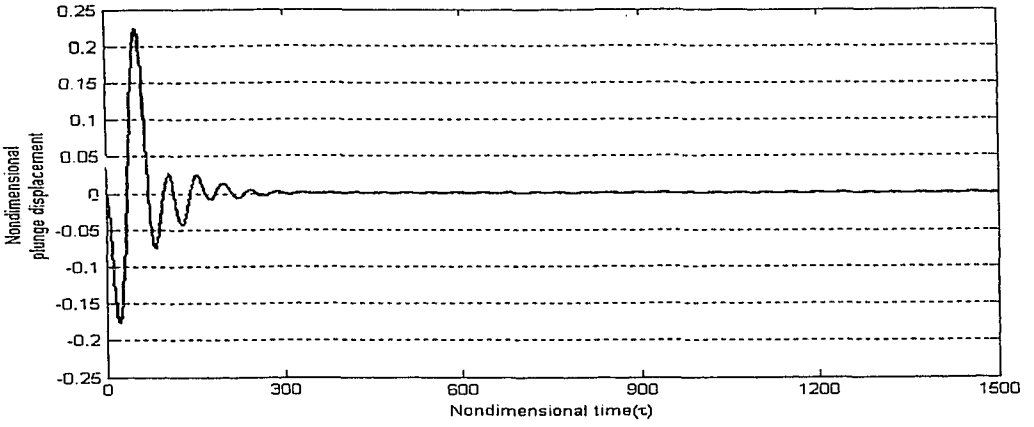
4.4 Neuro-fuzzy controller performance test

In this section, two performance tests on the fuzzy rules produced in Chapter 3 are implemented. First, the value of the ω_ξ parameter is changed slightly (±2%) for fuzzy rule sensitivity test. Second, four parameters-ω_ξ, ω_β,r_α *and* r_β- will be increased with a higher percent (say10%) of the initial values to find out the flutter suppression performance of the fuzzy rule base under the new airfoil-flap system. The author will also try to find out which parameters are critical to the flutter suppression in the second test.

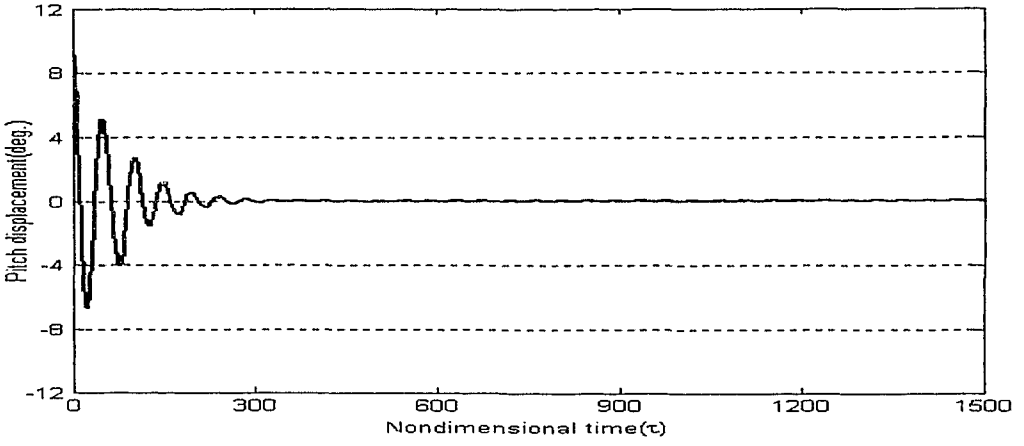
4.4.1 Sensitivity Test—2% change in ω_ξ

2% increase in parameter (ω_ξ = 0.204)

The neuro-fuzzy controller can successfully suppress flutter until the airspeed is 6.7 (Figure 32, Figure 33). When airspeed reaches 6.9, the amplifying factor should be changed to 1.315 for successful flutter suppression (Figure 34). The successful flutter suppression at airspeed 6.7 indicates that the slight change in the parameter ω_ξ does not affect the neuro-fuzzy controller performance.

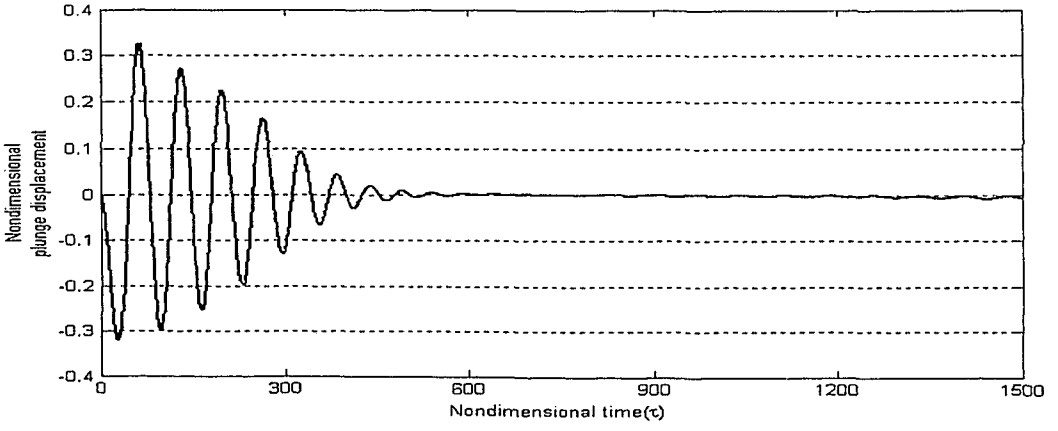


(a) nondimensional plunge displacement



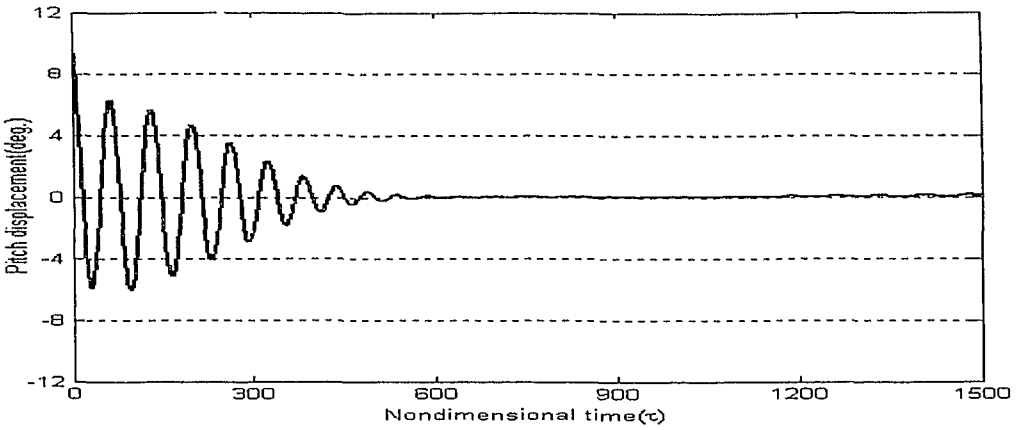
(b) pitch displacement

Figure 32. Neuro-fuzzy control under $\omega_\xi = 0.204$ and $U = 6.139$



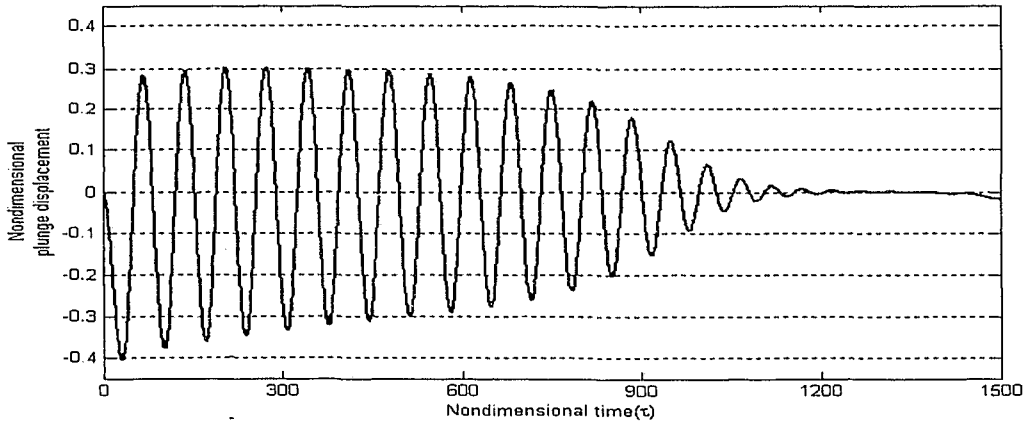
(a) nondimensional plunge displacement

Figure 33. Neuro-fuzzy control under $\omega_\xi = 0.204$ and $U = 6.7$

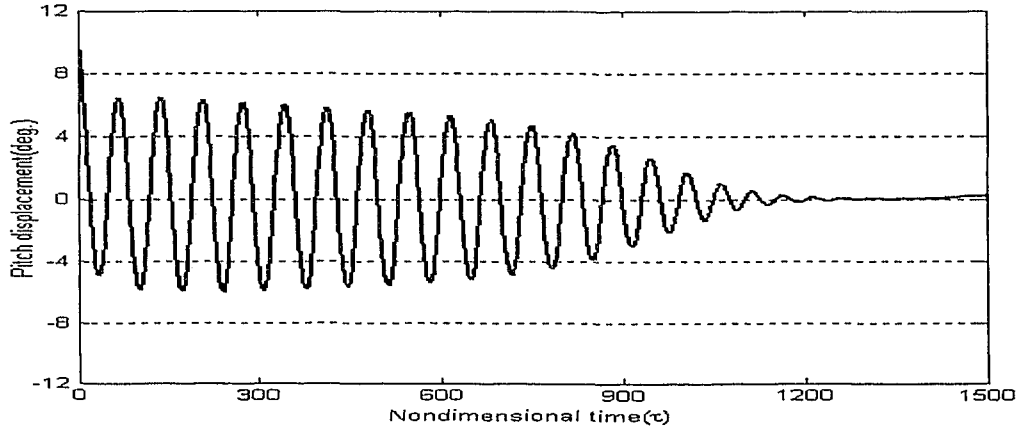


(b) pitch displacement

Figure 33. Neuro-fuzzy control under $\omega_\xi = 0.204$ and $U = 6.7$



(a) nondimensional plunge displacement

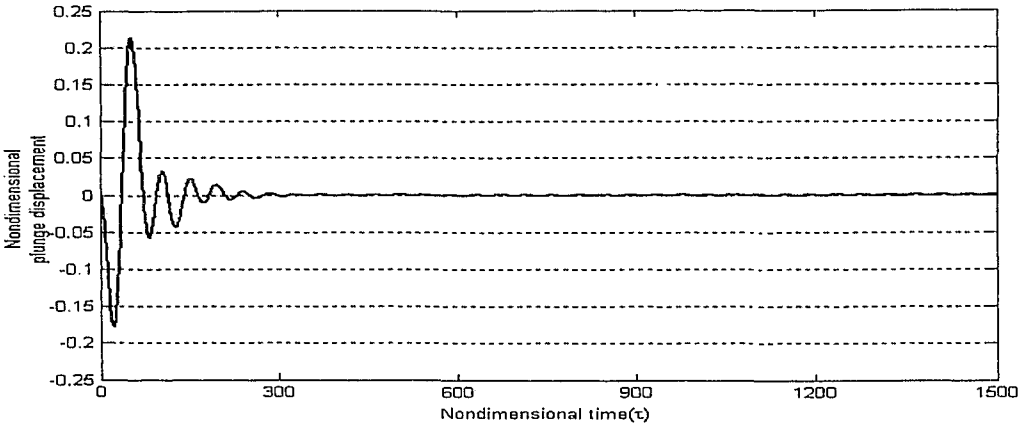


(b) pitch displacement

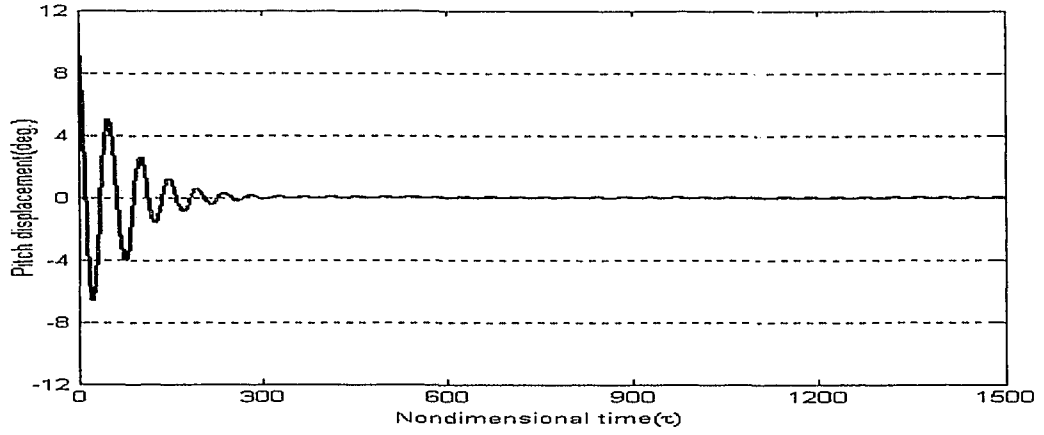
Figure 34. Neuro-fuzzy control under $\omega_\xi = 0.204$ and $U = 6.9$ with (amplifying factor = 1.315)

2% decrease in parameter ($\omega_\xi = 0.196$)

From Figure 35~Figure 37, the same conclusion as that in previous section on the neuro-fuzzy controller's performance can be drawn for this decreased ω_ξ parameter.

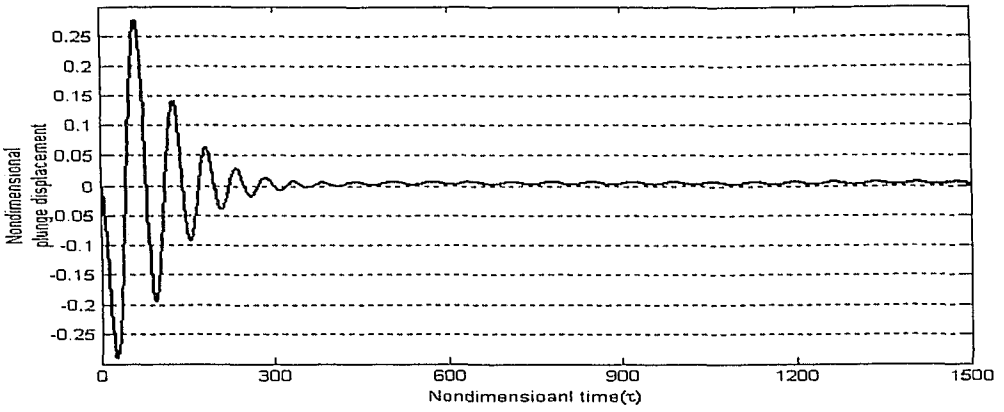


(a) nondimensional plunge displacement

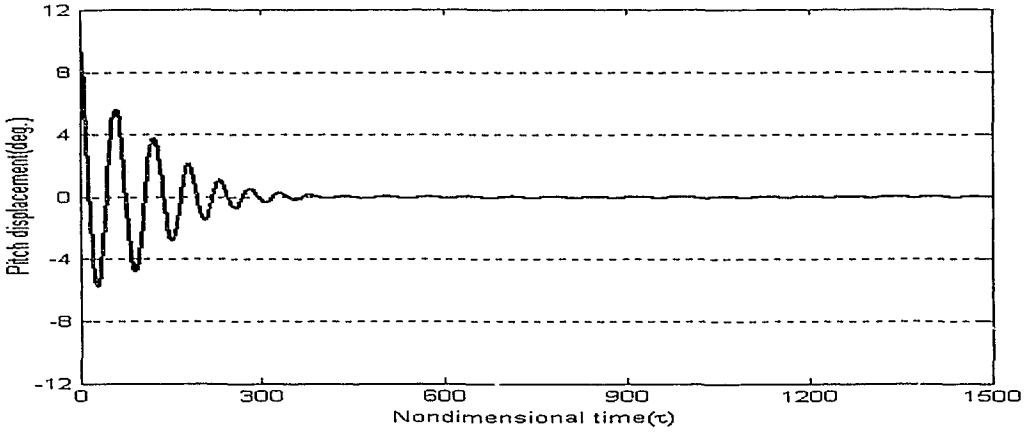


(b) pitch displacement

Figure 35. Neuro-fuzzy control under $\omega_\xi = 0.196$ and $U = 6.139$

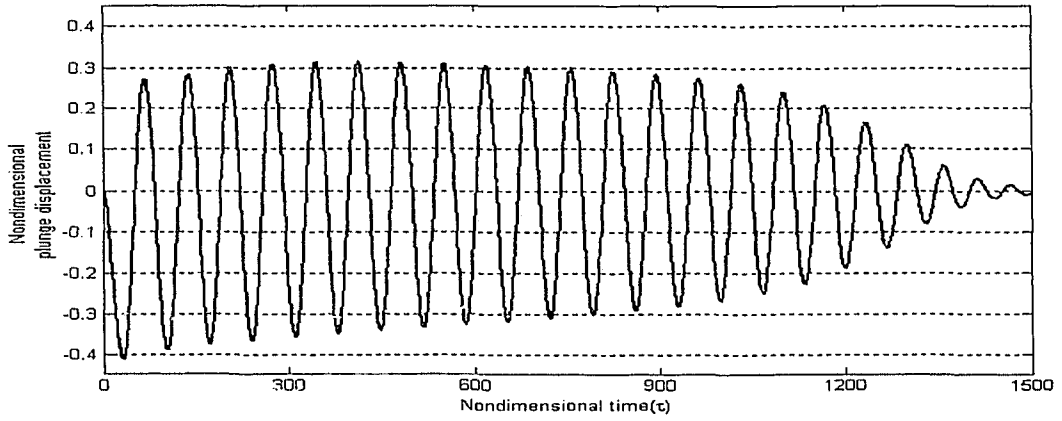


(a) nondimensional plunge displacement



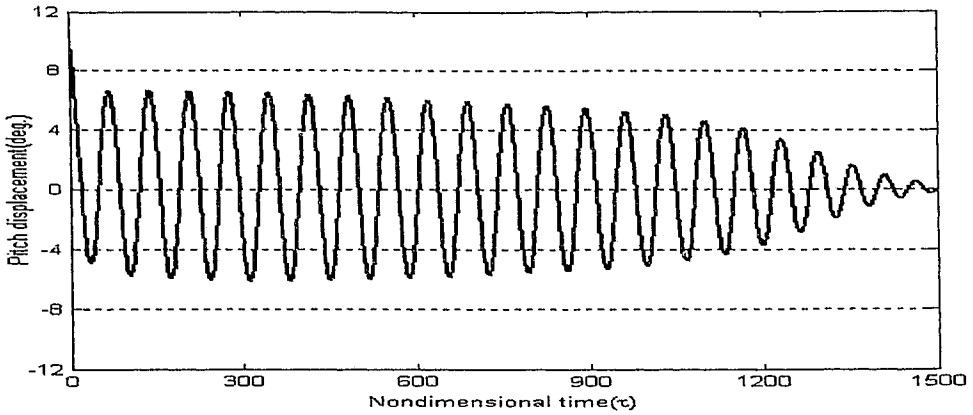
(b) pitch displacement

Figure 36. Neuro-fuzzy control under $\omega_\xi = 0.196$ and $U = 6.6$



(a) nondimensional plunge displacement

Figure 37. Neuro-fuzzy control under $\omega_\xi = 0.196$ and $U = 6.9$ (amplifying factor = 1.268)



(b) pitch displacement

Figure 37. Neuro-fuzzy control under $\omega_\xi = 0.196$ and $U = 6.9$ (amplifying factor = 1.268)

4.4.2 Performance under new system—10% parameters increase

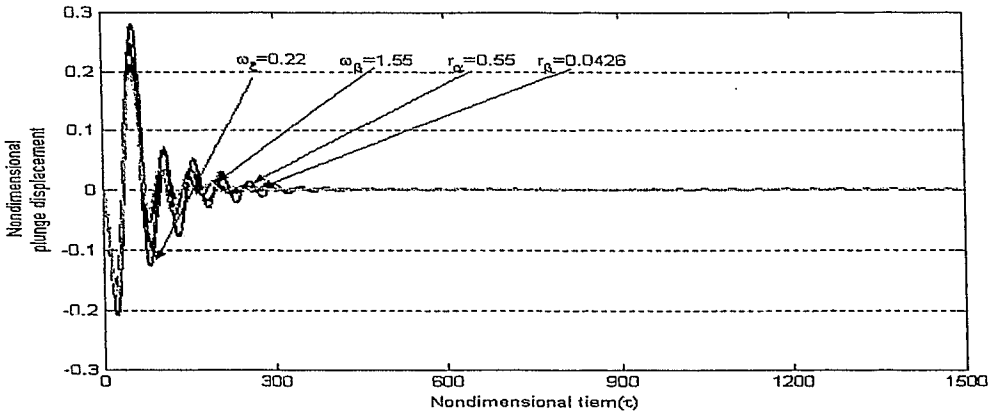
Unlike the slight (±2%) parameter change, a 10% increase will change the critical airfoil-flap combination flutter speed (Table 4). The performance test starts at the new flutter speeds and continues at several critical speeds.

10% increased parameter	ω_ξ	ω_β	r_α	r_β
New flutter speed	6.07	6.149	6.59	6.153

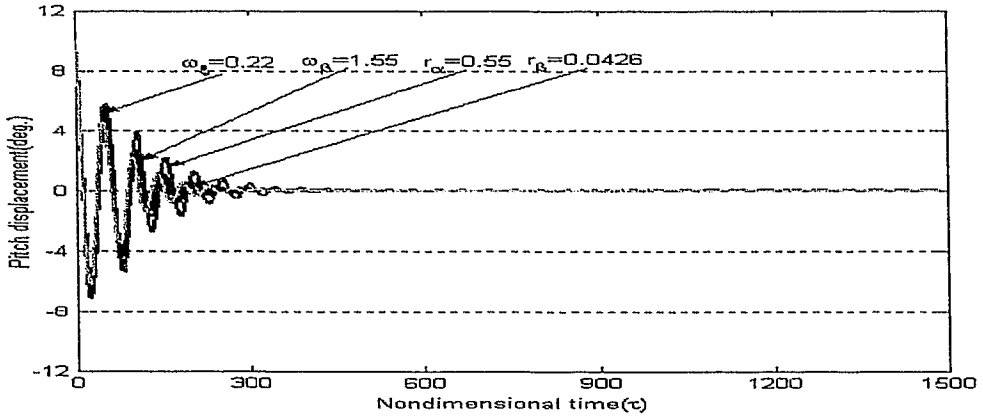
Table 4. New critical flutter speed under increased parameters

At new flutter speed

The following figure (Figure 38) indicates that the neuro-fuzzy controller can successfully suppress flutter at new flutter speeds.



(a) nondimensional plunge displacement



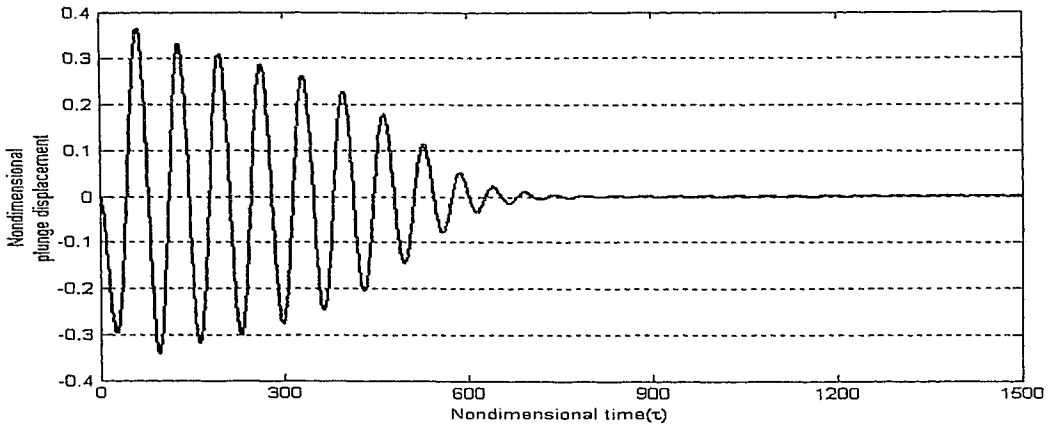
(b) pitch displacement

Figure 38. Neuro-fuzzy control at new flutter speeds

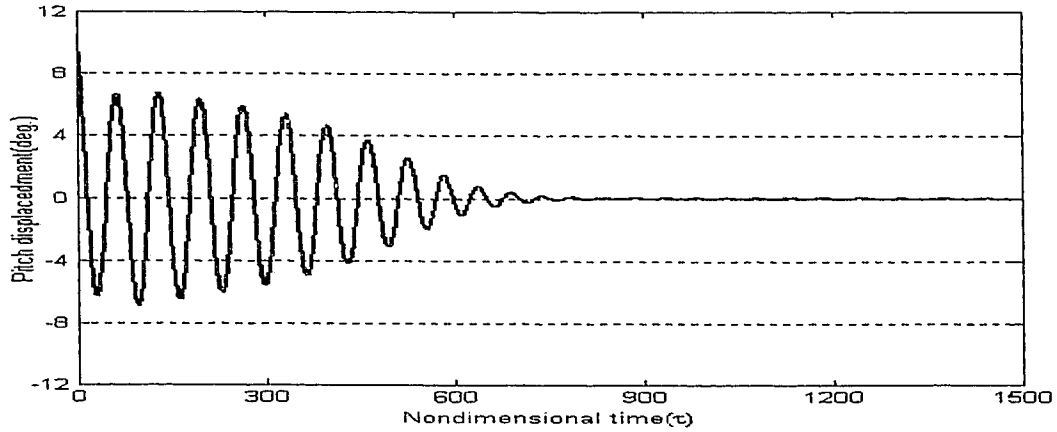
Air speed U = 6.65

It can be seen in Figure 39 and Figure 41 that the increase of two parameters (ω_ξ, r_α) will not degrade the neuro-fuzzy control performance in flutter suppression. After changing the amplifying factor produced by the equation-$4.8*U^3-94.1*U^2+614.3*U-1335.5$ to 1.0(for ω_β) and 0.9(for r_β), the neuro-fuzzy controller can suppress flutter better under the increased ω_β and r_β (Figure 43, Figure 44) compared with the results shown in Figure 40 and Figure 42 . As the air speed increases, the control result will become more and more sensitive to

the amplifying factor of the neuro-fuzzy controller output. The equation produced by the regression method for amplifying factor calculation is not accurate enough, so the adjustment of this amplifying factor does not indicate that the neuro-fuzzy controller is inefficient.

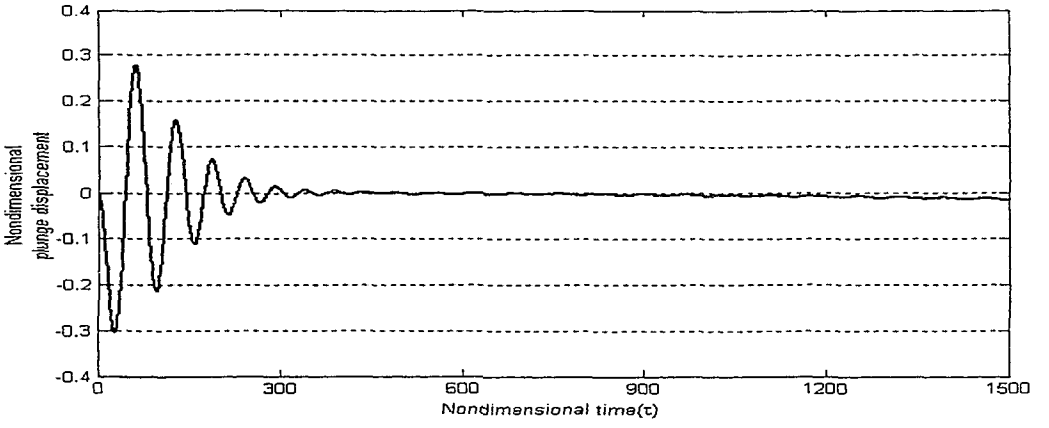


(a) nondimensional plunge displacement

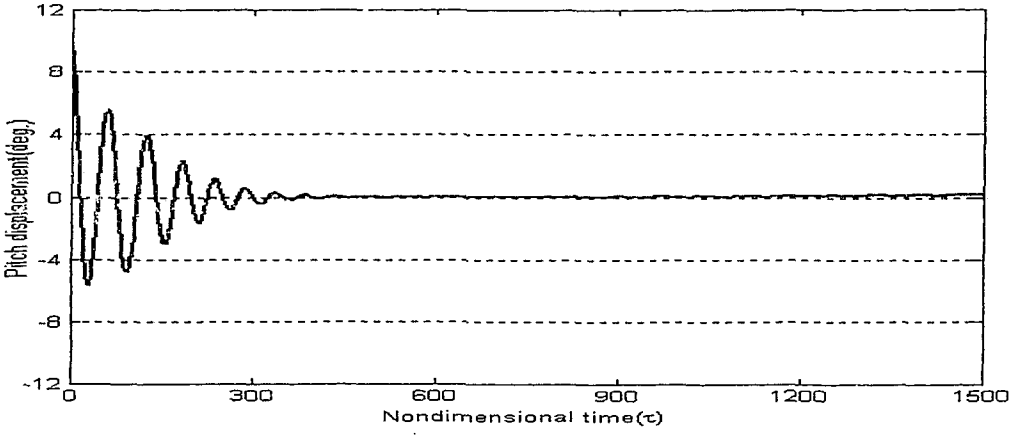


(b) pitch displacement

Figure 39. Neuro-fuzzy control at ω_ξ=0.22 with other parameters unchanged

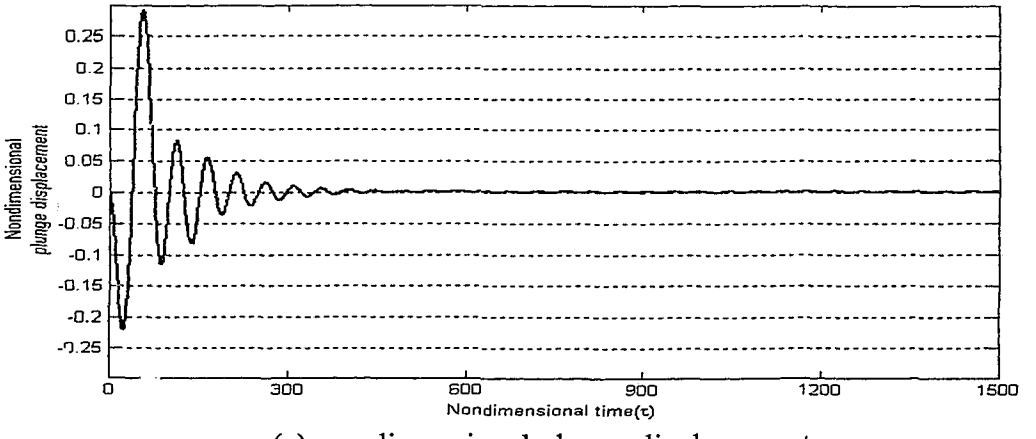


(a) nondimensional plunge displacement



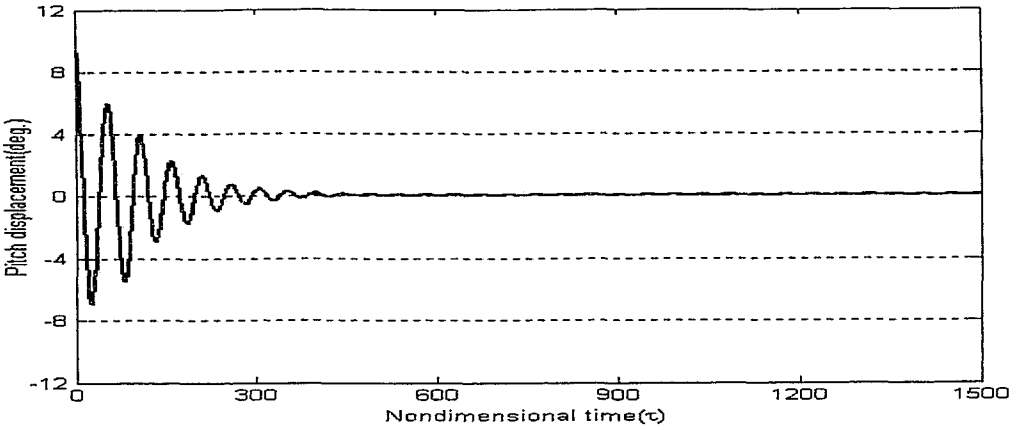
(b) pitch displacement

Figure 40. Neuro-fuzzy control at $\omega_\beta = 1.55$ with other parameters unchanged



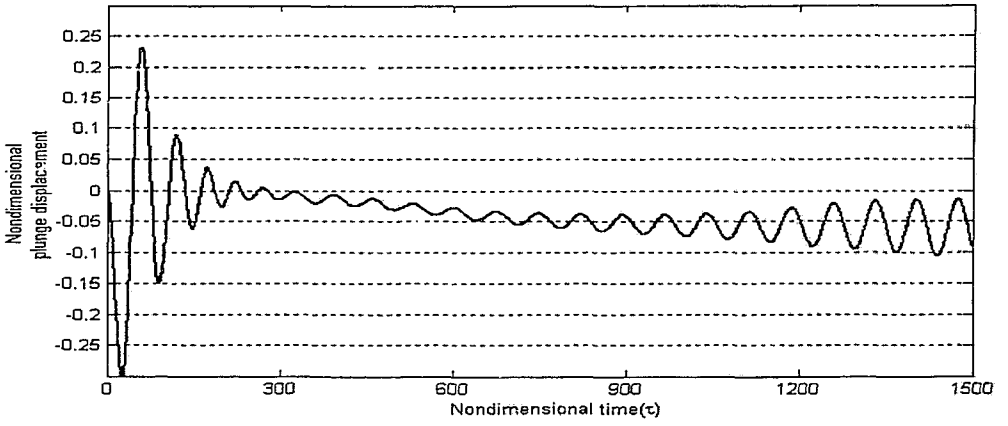
(a) nondimensional plunge displacement

Figure 41. Neuro-fuzzy control at $r_\alpha = 0.55$ with other parameters unchenged

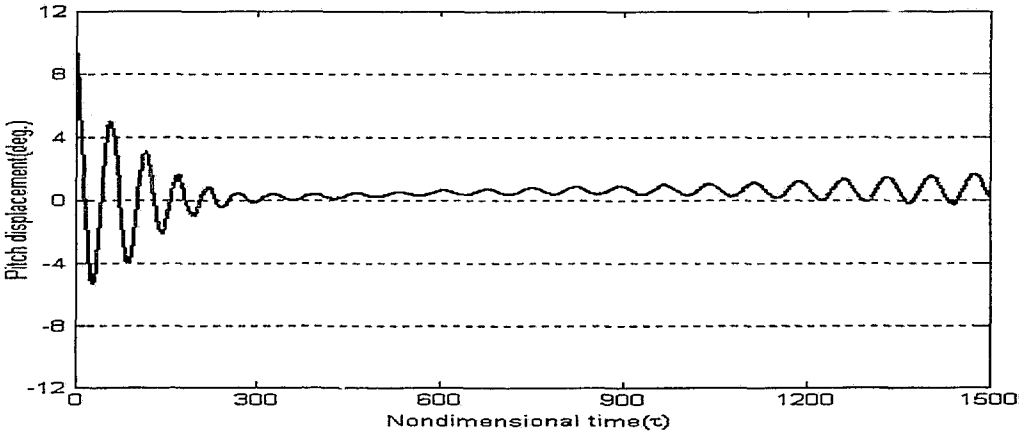


(b) pitch displacement

Figure 41. Neuro-fuzzy control at $r_\alpha = 0.55$ with other parameters unchanged

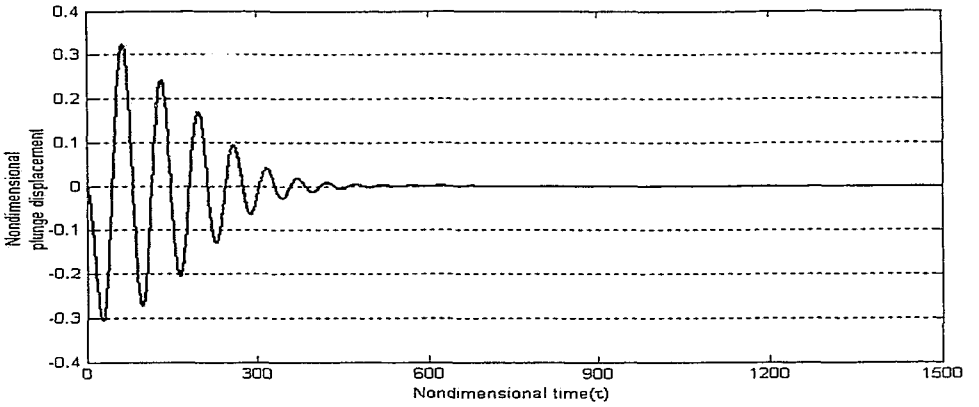


(a) nondimensional plunge displacement

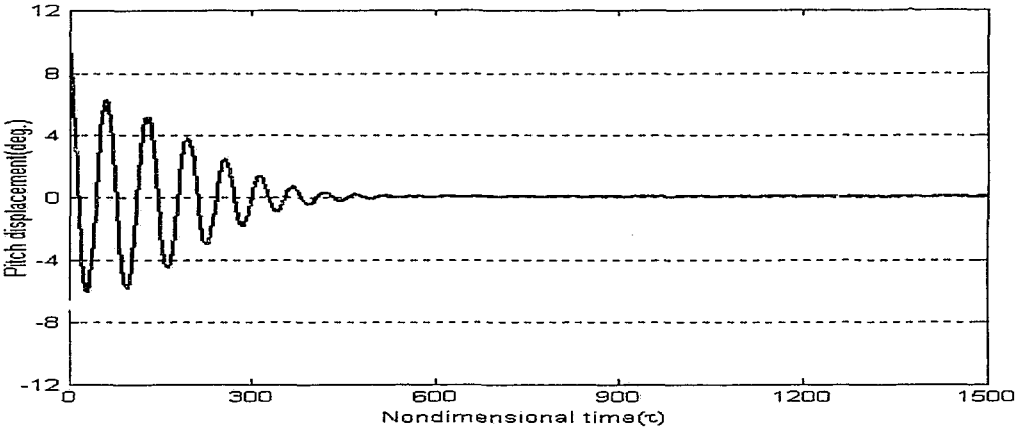


(b) pitch displacement

Figure 42. Neuro-fuzzy control when $r_\beta = 0.0426$ with other parameters unchanged

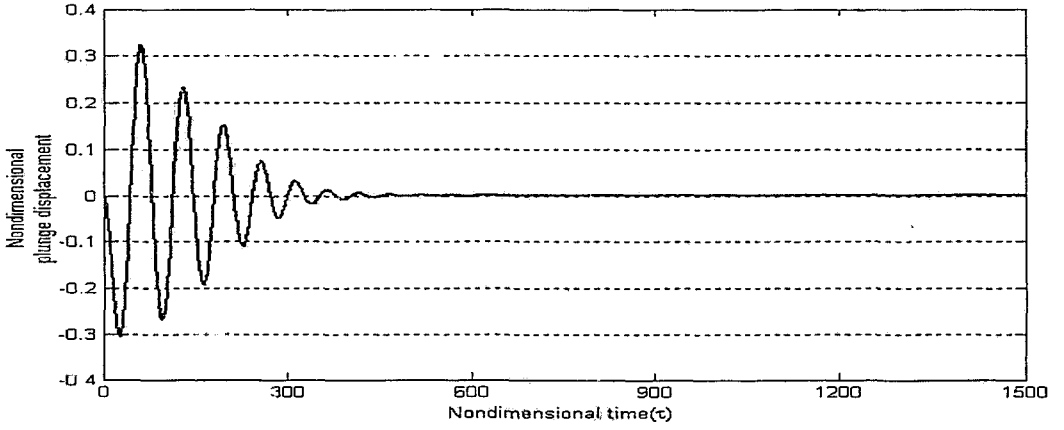


(a) nondimensional plunge displacement



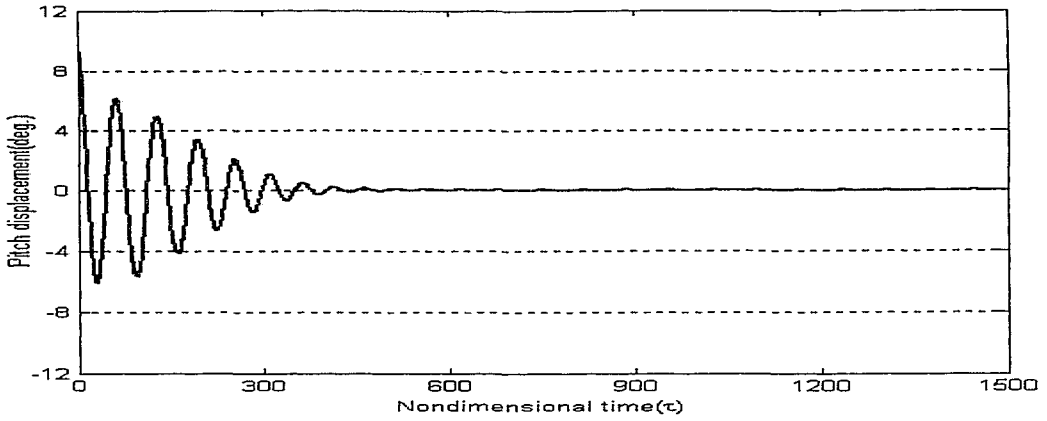
(b) pitch displacement

Figure 43. Neuro-fuzzy control at ω_β= *1.55* with other parameters unchanged (amplifying factor 1.0)



(a) nondimensional plunge displacement

Figure 44. Neuro-fuzzy control at r_β= *0.0426* with other parameters unchanged (amplifying factor 0.9)

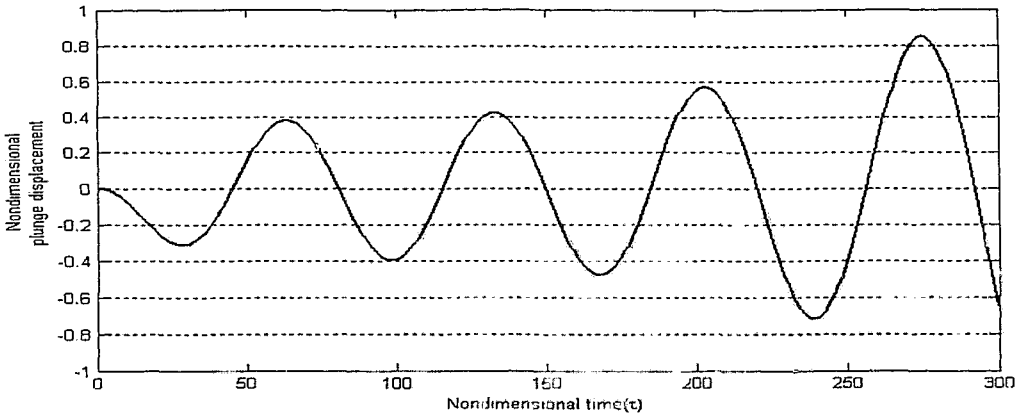


(b) pitch displacement

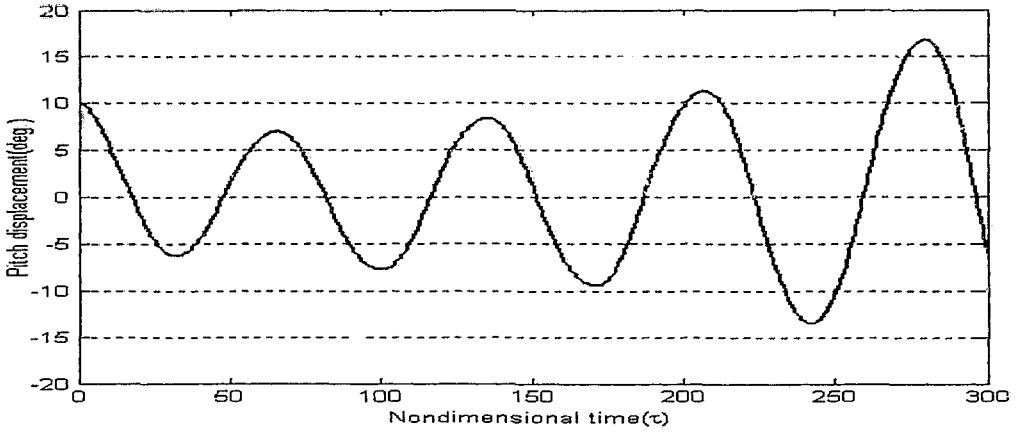
Figure 44. Neuro-fuzzy control at r_β= *0.0426* with other parameters unchanged (amplifying factor 0.9)

Air speed U = 6.7

At this air speed, the neuro-fuzzy controller will not be able to suppress flutter with increased ω_ξ (Figure 45). The increase in r_α does not affect the neuro-fuzzy control flutter suppression (Figure 47). After changing the amplifying factor to 1.05(for ω_β) and 0.9(for r_β), the neuro-fuzzy controller can successfully suppress flutter with increased parameters- ω_β and r_β. The reason for this change can be the same under $U = 6.65$ (Figure 46, Figure 48).

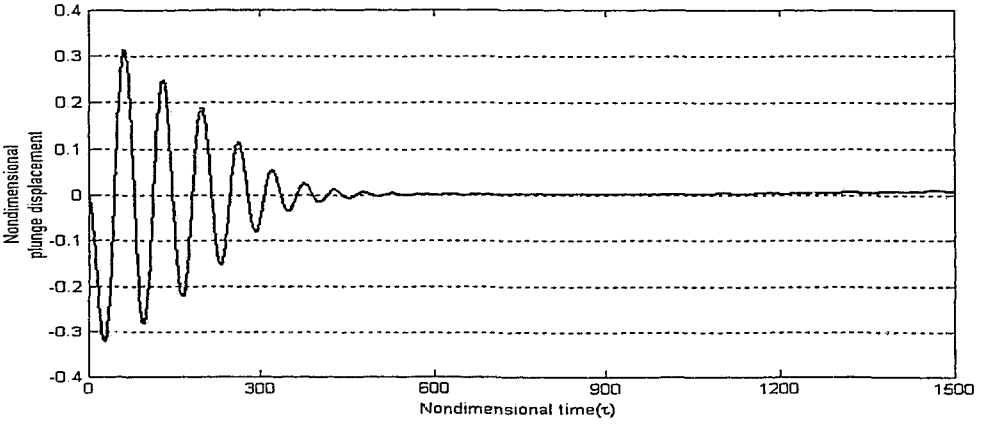


(a) nondimensional plunge displacement



(b) pitch displacement

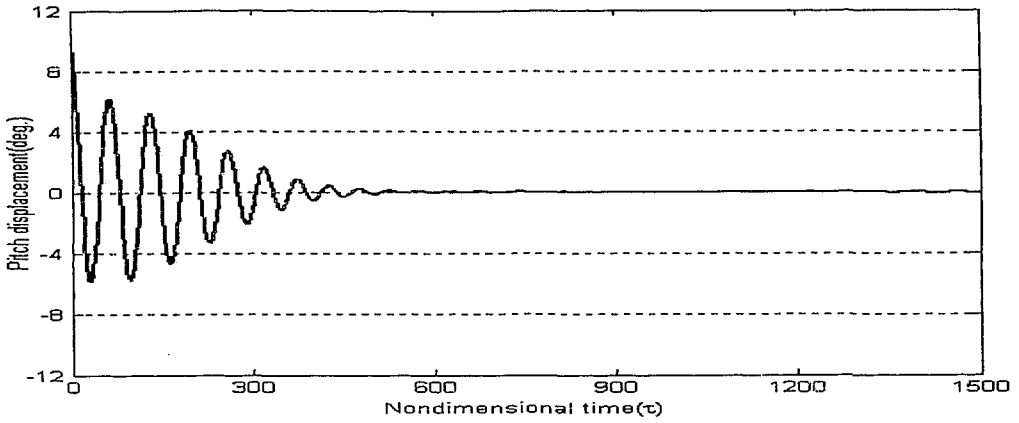
Figure 45. Neuro-fuzzy control at ω_ξ=2.2 with other parameters unchanged



(a) nondimensinal plunge displacement

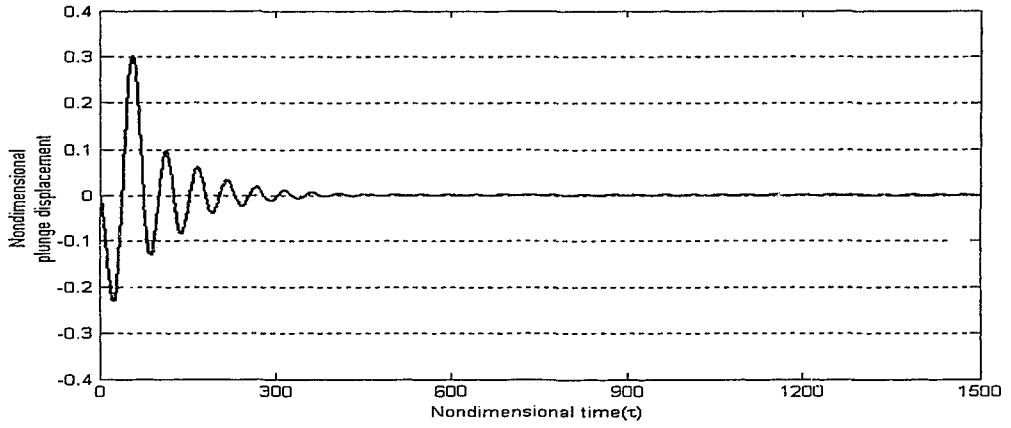
(b)

Figure 46. Neuro-fuzzy control at ω_β= 1.55 with other parameters unchanged (amplifying factor 1.05)

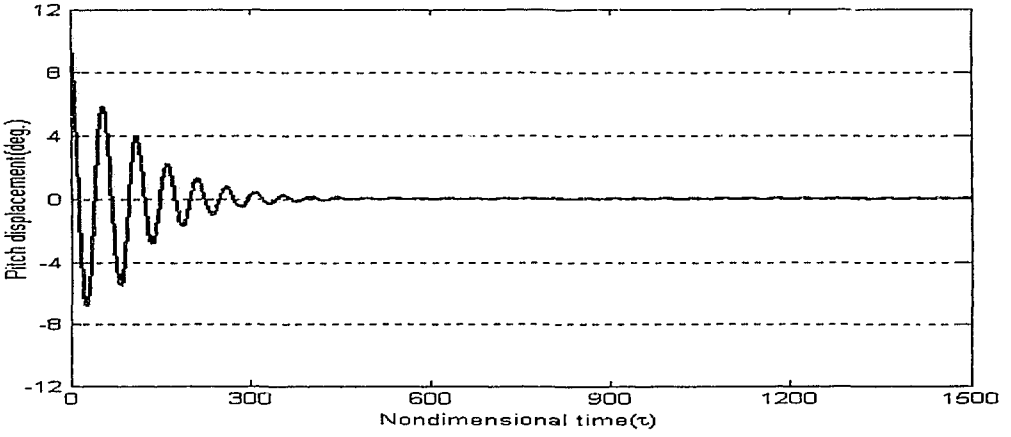


(c) pitch displacement

Figure 46. Neuro-fuzzy control at $\omega_\beta = 1.55$ with other parameters unchanged (amplifying factor 1.05)

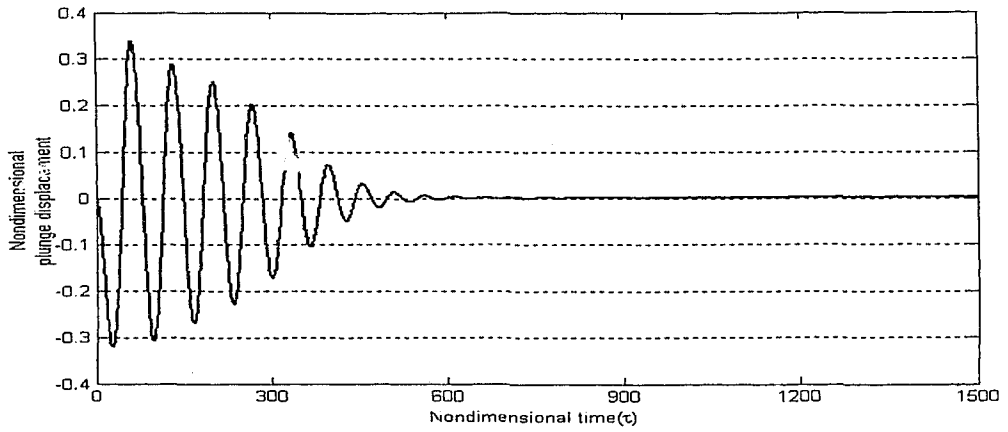


(a) nondimensional plunge displacement

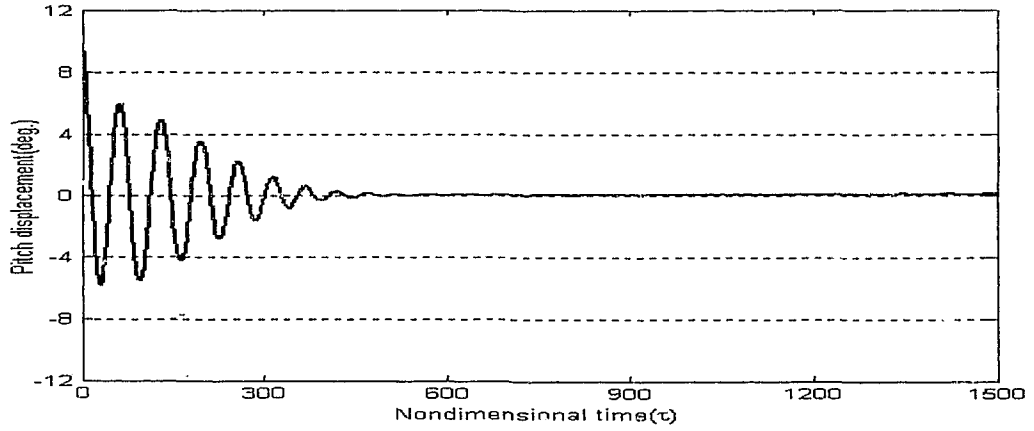


(b) pitch displacement

Figure 47. Neuro-fuzzy control at $r_\alpha = 0.55$ with other parameters unchanged



(a) nondimensional plunge displacement



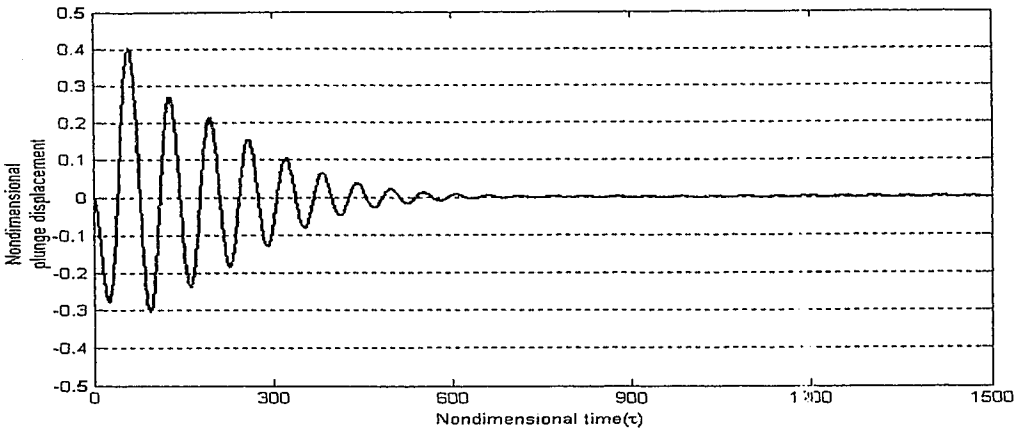
(b) pitch displacement

Figure 48. Neuro-fuzzy control at $r_\beta = 0.0426$ with other parameters unchanged (amplifying factor 0.9)

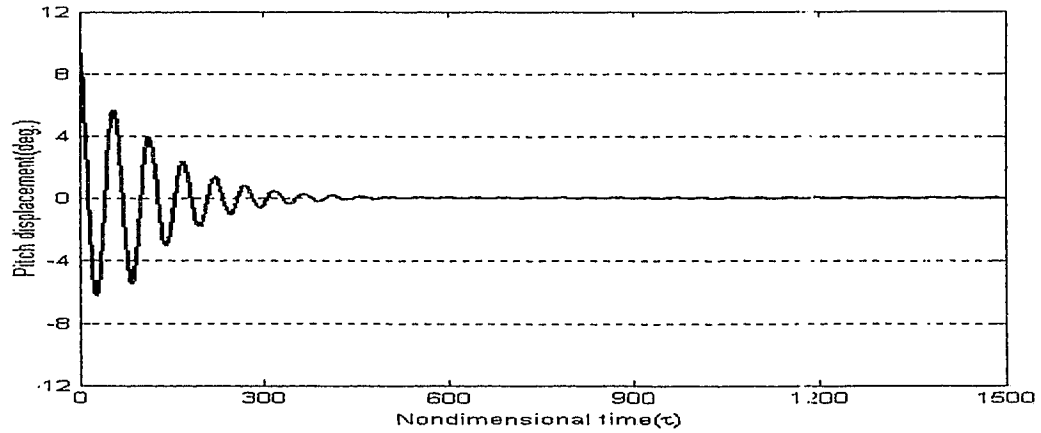
Air speed $U = 6.9$

At this air speed, the neuro-fuzzy controller will work only under the increased r_α (Figure 49).This is not surprising as the fuzzy rules used here are produced at this speed (U = 6.9) and the parameter changes will affect the neuro-fuzzy controller's performance easily.

From the performance of the neuro-fuzzy controller's performance at different air speeds under increased airfoil-flap combination parameters, it can be concluded that the fuzzy rules are capable in flutter suppression.



(a) nondimensional plunge displacement



(b) pitch displacement

Figure 49. Neuro-fuzzy control when $r_\alpha = 0.55$ with other parameters unchanged

Chapter 5 Conclusion

The neuro-fuzzy controller application to the flutter suppression of a three degree-of-freedom airfoil-flap combination in a two-dimensional incompressible inviscid flow is studied. The airfoil Simulink® model is simulated according to a set of airfoil aeroelastic equations of motion. A NEFCON controller is then included in the Simulink® model of the airfoil to learn the airfoil behaviour and increase the critical flutter speed. A fuzzy rule base with ten rules is produced by the NEFCON controller from learning the dynamic behavior of the airfoil.

The NEFCON output, flap rotation command, is decided by the fuzzy rules and magnified by a gain. The NEFCON controller can increase critical flutter speed by 12.4%.

For the purpose of comparison, the modified energy control law is also applied on the same airfoil-flap combination. When the modified energy method is applied, the plunging displacement first jumps to a high level and then reaches a stable state. The modified energy method increases the critical flutter speed by 35%. The present study shows that the neuro-fuzzy control methodology can be

successfully applied to flutter suppression without any prior knowledge of the flutter suppression command required.

Further research is suggested on the application of neuro-fuzzy control in flutter suppression of airfoils. Further study is required to improve the NEFCON controller's performance by adjusting its structural parameters, finding out why the gain factor must be applied and why different gain factors are needed for training and control.

References

[1]. Bisplinghoff, R. L., Ashley, H. and Halfman, R. L., "Aeroelsticity," Dover Publications, Inc., Mineola, New York, 1996.

[2]. Fung, Y.C., "Introduction to the theory of aeroelasticity," John Wiley and Sons,Inc. London. Champman & Hal, Limited, 1955.

[3]. Yang, Z.C, Zhao, L.C. and Jiang, J.S. "A semi-active flutter control scheme for a two-dimensional wing", Journal of Sound and Vibration (1995) 184(1),1-7

[4]. Negnevitsky, M., "Artificial intelligence: a guide to intelligent systems," Addison-Wesley, 2002.

[5]. Haykin, S., "Neural networks: a comprehensive foundation," Macmillian College Publishing Company, 1994.

[6]. Rojas,R., "Neural networks: a systematic introduction," Springer, 1996.

[7]. Hornik, K., Stinchcombe, M., and White, H., "Multilayer feedforward networks are universal approximators," Neural Netwoks 2 (1989) 359~366

[8]. Rutkowska, D., "Neuro-Fuzzy architectures and hybrid learning," Physica-Verlag A Spronger-Verlag Company, 2002.

[9]. Haley,P., Soloway, D., "Generalized predictive control for active flutter suppression," Journal of Guidance, Control, and Dynamics, Vol.24,No.1, January-February 2001.

[10]. Mukhopadhyay, V., "Transonic flutter suppression control law design using classical and optimal techniques with wind-tunnel results," 40th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials(SDM) Conference, April 12-15, 1999/St. Louis, MO.

[11]. Crowther, W.J., Cooper, J.E., "Flight test flutter prediction using neural networks ," School of Engineering, University of Manchester, M13 9PL, 23 October , 2000.

[12]. Scott, R. C., Pado, L.E., "Active control of wind-tunnel model aeroelastic response using neural networks," Journal of Guidance, Control and Dynamics Vol.23, No.6, November-December 2000.

[13]. Alighanbari,H., " Flutter analysis and chaotic response of an aircraft accounting for structural nonlinearity," PH.D Thesis McGill University,1995

[14]. Alighanbari, H., "The aeroelastic response of a 3-dof airfoil with structural nonlinearities," Third International Conference on Nonlinear Problems in Aviation and Aerospace May 10-12, 2000.

[15]. Alighanbari,H., "Stability and bifurcation of an airfoil-aileron combination in two-dimensional incompressible inviscid flow," Proceedings of 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Denver, CO, USA,April,2002.

[16]. Alighanbari,H., "Aeroelastic response of an airfoil-aileron combination with freeplay in aileron hinge," Journal of Aircraft, Vol.39, No.4: Engineering Notes, 2002.

[17]. Nurnberger,A.,Nauck,D., and Kruse,R., "A neuro-fuzzy development tool for fuzzy controllers under Matlab/Simulink," Proc. 5th European Congress on Intelligent Techniques and Soft Computing (EUFIT'97), pp. 1029-1033, Aachen, 1997.

[18]. Nauck,D., Klawonn,F., and Kruse,R., "Foundations of neuro-fuzzy systems,: John Wiley & Sons,1997.

[19]. Nurnberger, A., Nauck,D., Kruse,R., "Neuro-fuzzy control based on the NEFCON-model: recent developments," Soft Computing 2 (1999) 168-182 Springer-Verlag 1999.

[20]. Nauck, D., and Kruse,R., "A fuzzy neural network learning fuzzy control rules and memebership functions by fuzzy error beckpropogation," Proc. IEEE Int. Conf. Neural Networks ICNN'93, San Francisco, Mar.28-Apr.1(1993), 1022-1027.

[21]. Nürnberger,A., Nauck,D., and Kruse,R., "Neuro-Fuzzy control based on the NEFCON model under MATLAB/SIMULINK" 2ND On-line World Conference on Soft Computing in Engineering Design and Manufacturing, 1997.

[22]. Nürnberger,A., and Kruse,R., "Neuro-Fuzzy techniques under MATLAB/SIMULINK applied to a real plant," Proc.IEEE International Conference on Fuzzy Systems 1998 (FUZZ-IEEE '98), pp. 572-576, Anchorage, Alaska, 1998.

[23]. Lanas, A. I., Pacheco1, M.A., Vellasco1, M.M., Tanscheit, R., "Neuro-Fuzzy control of a multivariable nonlinear process," The 7th European Congress on Intelligent Techniques and Soft Computing. EUFIT `99 - Magdeburg, Germany, 1999 .

[24]. Mukhopadhyay,V., "Transonic flutter suppression control law design and wind-tunnel test results," Journal of guidance, control, and dynamics, Vol.23, No.5, September-October 2000.

[25]. Alighanbari, H., and Price, S.J., "The post-hopf-bifurcation response of an airfoil in incompressible two dimensional flow," Nonlinear dynamics 10: 381-400 © 1996 Kluwer Academic Publishers. Printed in the Netherlands.

[26]. Alighanbari, H., and Hashemi, S.M., "Bifurcation analysis of an airfoil containing a cubic structural nonlinearity and subjected to two-dimensional

incompressible flow," 43rd AIAA /ASME/ASCE/AHS Structures, Structural Dynamics, and Materials Conference, April 22-25, 2002/ Denver, CO.

[27]. Bernelli-Zazzera,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, November-December 2000.

[28]. Lee, B.H., and Wong, Y.S., "Neural network parameter extraction with application to flutter signals," Journal of Aircraft, Vol.35, No. 1: Engineering Notes, 1998.

Appendix I ***Equations for T's***

$$T1=\frac{1}{3}\times scb\times(2+cbeta^2)+cbeta\times acs$$

$$T3=(\frac{1}{8}+cbeta^2)\times acs^2+0.25\times cbeta\times scb\times acs\times(7+2\times cbeta^2)$$

$$-\frac{1}{8}\times(1-cbeta^2)(5\times cbeta^2+4)$$

$$T4=-acs+cbeta\times scb$$

$$T5=(1-cbeta^2)-acs^2+2\times cbeta\times scb\times acs$$

$$T7=(\frac{1}{8}+cbeta^2)\times acs+\frac{1}{8}\times cbeta\times scb\times(7+2\times cbeta^2)$$

$$T8=\frac{1}{3}\times scb\times(2\times cbeta^2+1)+cbeta\times acs$$

$$T10=scb+acs$$

$$T11=acs\times(1-2\times cbeta)+scb\times(2-cbeta)$$

$$T12=scb\times(2+cbeta)-acs\times(2\times cbeta+1)$$

where:

$$zbeta=rbeta^2+(cbeta-ah)\times xbeta$$

$$ahbar=\frac{1}{2}-ah$$

$$scb=\sqrt{1-cbeta^2}$$

$$acs=\arccos(cbeta)$$

Appendix II ***Simulink® blocks for m's, n's and q's***

These are the m's, n's and q's equations:

$$m_1 = \frac{\mu + 1}{\mu};$$

$$m_2 = \frac{xalpha \times \mu - a_h}{\mu};$$

$$m_3 = \frac{-T1 + xbeta \times \pi \times \mu}{\pi \times \mu};$$

$$m_4 = 2\frac{U \times lw + zetaxi \times omegaxi \times \mu}{U \times \mu};$$

$$m_5 = \frac{2 \times lw(0.5 - a_h) + 1}{\mu};$$

$$m_6 = \frac{-T4 + lw \times T11}{\pi \times \mu};$$

$$m_7 = \left(\frac{omegexi}{U}\right)^2;$$

$$m_8 = \frac{2 \times lw}{\mu};$$

$$m_9 = \frac{2 \times lw \times T10}{\pi \times \mu};$$

$$m_{10} = 0;$$

$$m_{11} = \frac{-2}{\mu};$$

$$m_{12} = \frac{-2}{\mu};$$

$$n_1 = \frac{xalpha \times \mu - a_h}{ralpha^2 \times \mu};$$

$$n_2 = \frac{a_h^2 + 0.125 + ralpha^2 \times \mu}{ralpha^2 \times \mu};$$

$$n_3 = \frac{zbeeta \times \pi \times \mu + T1 \times a_h - T1 \times cbeta - T7}{ralpha^2 \times \pi \times \mu};$$

$$n_4 = \frac{lw(1 + 2 \times a_h)}{ralpha^2 \times \mu};$$

$$n_5 = \frac{\left[-2 \times U \times a_h \times lw - U \times lw + U\right] \times (0.5 - a_h) + 2 \times zetaalpha \times ralpha^2 \times \mu}{U \times ralpha^2 \times \mu};$$

$$n_6 = \frac{T11 + 2 \times T1 - lw \times T11 + 2 \times T4 \times a_h - 2 \times T4 \times cbeta - 2 \times T8 - \times 2 \times a_h \times lw \times T11}{ralpha^2 \times \pi \times \mu};$$

$$n_7 = 0;$$

$$n_8 = \frac{-lw \times U^2 + ralpha^2 \times \mu - 2 \times lw \times a_h \times U^2}{ralpha^2 \times U^2 \times \mu};$$

$$n_9 = \frac{T10 - 2 \times a_h \times lw \times T10 + T4 - lw \times T10}{ralpha^2 \times \pi \times \mu};$$

$$n_{10} = 0;$$

$$n_{11} = \frac{1 + 2 \times a_h}{ralpha^2 \times \mu};$$

$$n_{12} = \frac{1 + 2 \times a_h}{ralpha^2 \times \mu};$$

$$q_1 = \frac{-T1 + xbeta \times \pi \times \mu}{\pi \times rbeta^2 \times \mu};$$

$$q_2 = \frac{zbeta \times \pi \times \mu + T1 \times a_h - T1 \times cbeta - T7}{\pi \times rbeta^2 \times \mu};$$

$$q_3 = \frac{-T3 + rbeta^2 \times \pi^2 \times \mu}{rbeta^2 \times \pi^2 \times \mu};$$

$$q_4 = \frac{T12 \times lw}{\pi \times rbeta^2 \times \mu};$$

$$q_5 = \frac{2\left(cbeta^2 - 1\right)\sqrt{1 - cbeta^2} - 6 \times T1 - 3 \times T4 + 6 \times T12 \times lw(0.5 - ah)}{6 \times \pi \times rbeta^2 \times \mu};$$

$$q_6 = \frac{4 \times zetabeta \times omegabeta \times rbeta^2 \times \pi^2 \times \mu + U \times T12 \times lw \times T11 - U \times T4 \times T11}{2 \times U \times rbeta^2 \times \pi^2 \times \mu};$$

$$q_7 = 0;$$

$$q_8 = \frac{T12 \times lw}{\pi \times rbeta^2 \times \mu};$$

$$q_9 = \frac{-U^2 \times T4 \times T10 + U^2 \times T5 + omegabeta^2 \times rbeta^2 \times \pi^2 \times \mu + U^2 \times T12 \times lw \times T10}{rbeta^2 \times U^2 \times \pi^2 \times \mu};$$

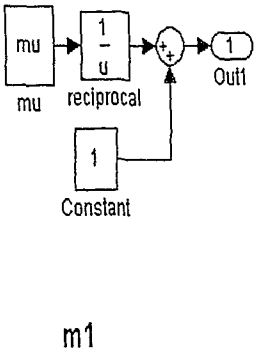
$$q_{10} = 0;$$

$$q_{11} = \frac{-T12}{\pi \times rbeta^2 \times \mu};$$

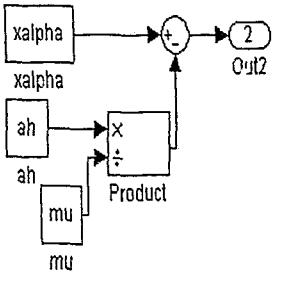
$$q_{12} = \frac{-T12}{\pi \times rbeta^2 \times \mu};$$

$$q_{13} = \left(\frac{omegabeta}{U}\right)^2$$

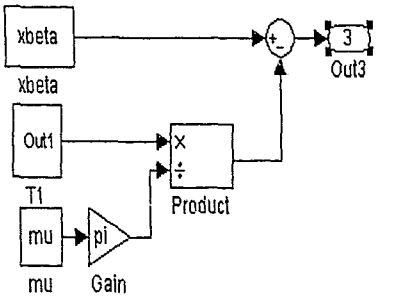
The followings show the Simulink® models of ms, ns and qs:



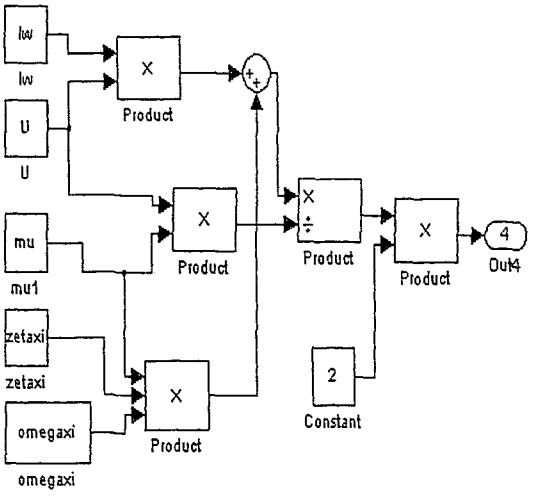
m1



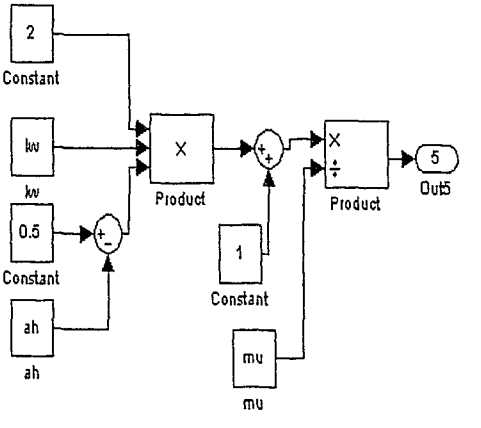
m2



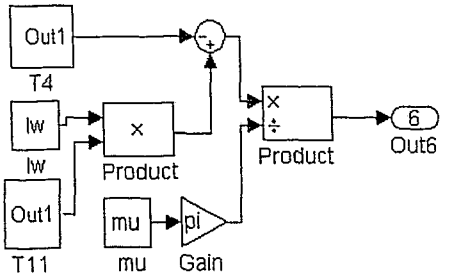
m3



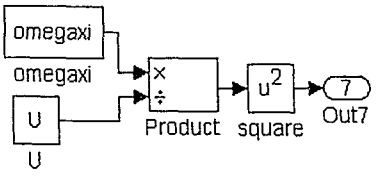
m4



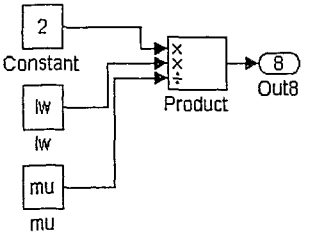
m5



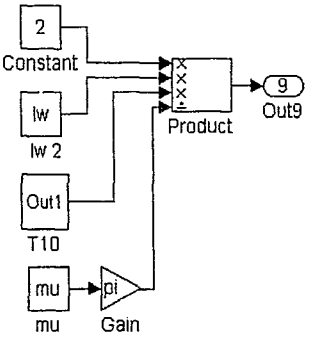
m6



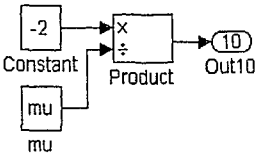
m7



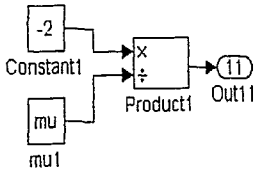
m8



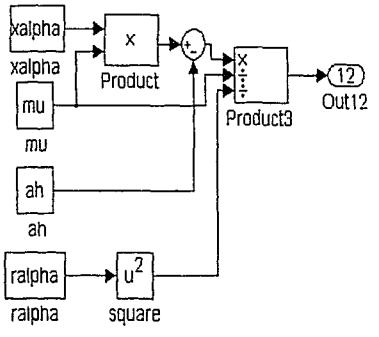
m9



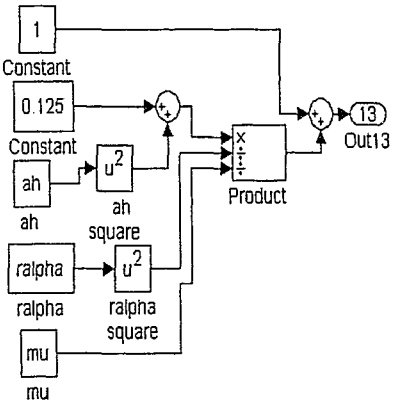
m11



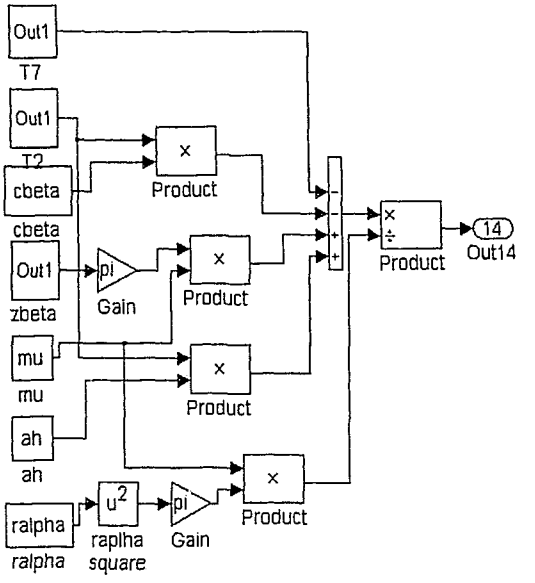
m12



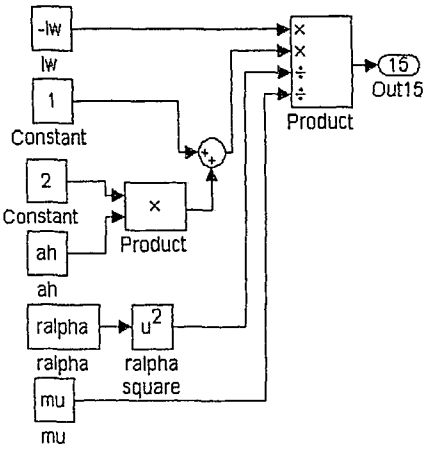
n1



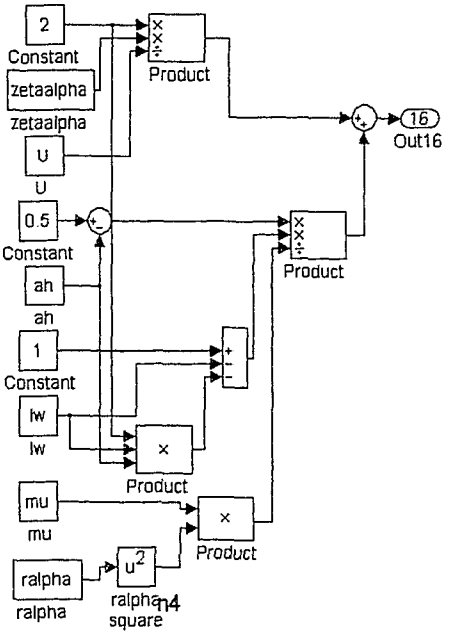
n2


Out1
T7
Out1
T2
cbeta
cbeta
Out1
zbeta
mu
mu
ah
ah
ralpha
ralpha
Product
Product
Product
Product
Product
Gain
Gain
raplha
square
Out14
14

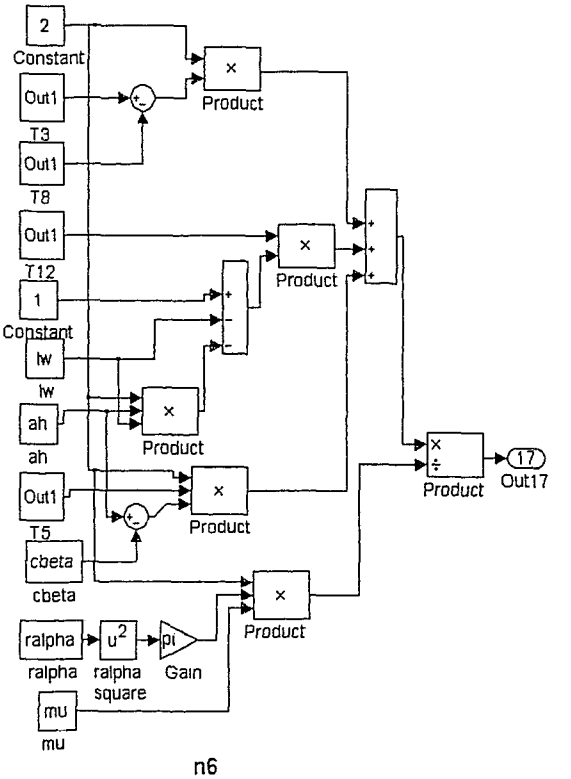
n3


-lw
lw
1
Constant
2
Constant
ah
ah
ralpha
ralpha
mu
mu
Product
Product
Product
ralpha
square
Out15
15

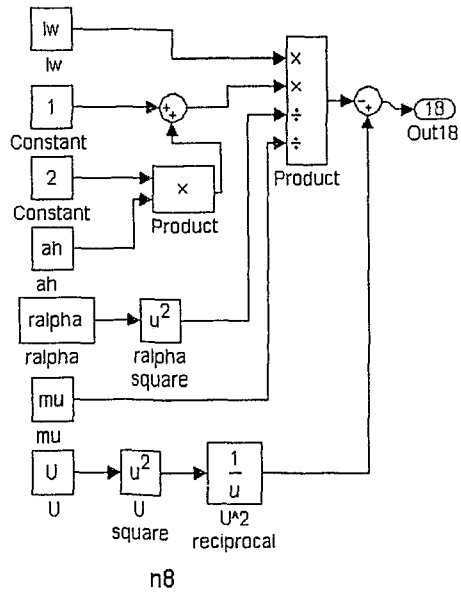
n4


2
Constant
zetaalpha
zetaalpha
U
U
0.5
Constant
ah
ah
1
Constant
lw
lw
mu
mu
ralpha
ralpha
Product
Product
Product
Product
Product
ralpha
square
Out16
16

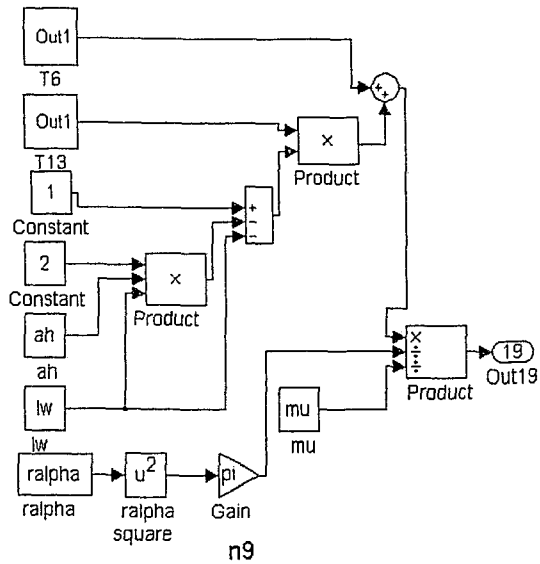
n5


2
Constant
Out1
T3
Out1
T8
Out1
T12
1
Constant
lw
lw
ah
ah
Out1
T5
cbeta
cbeta
ralpha
ralpha
mu
mu
Product
Product
Product
Product
Product
Product
ralpha
square
Gain
Out17
17

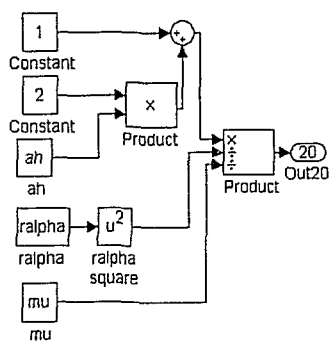
n6



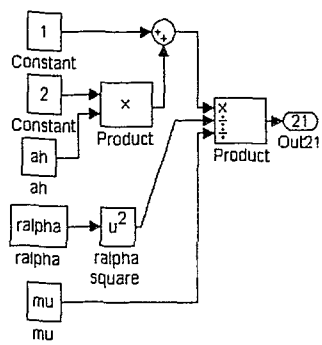
n8



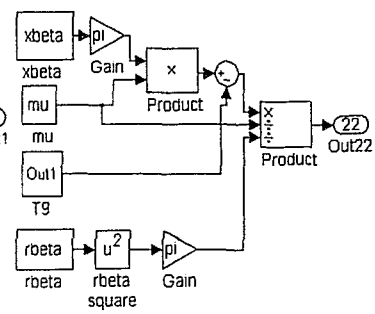
n9



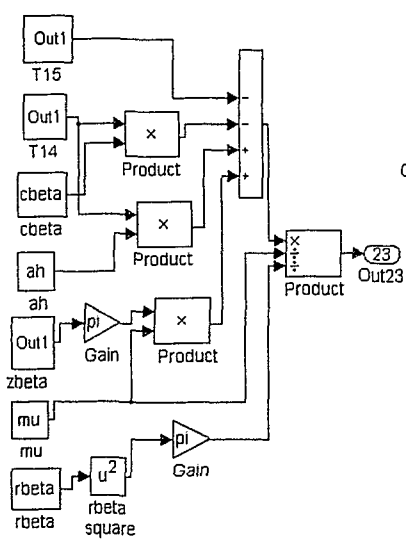
n11



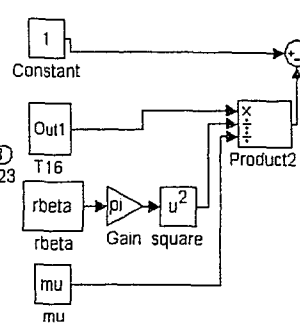
n12



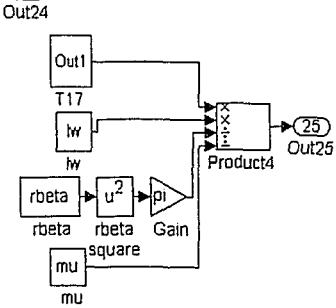
q1



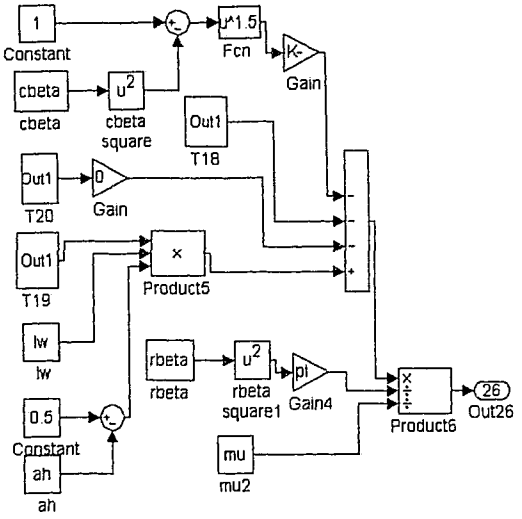
q2



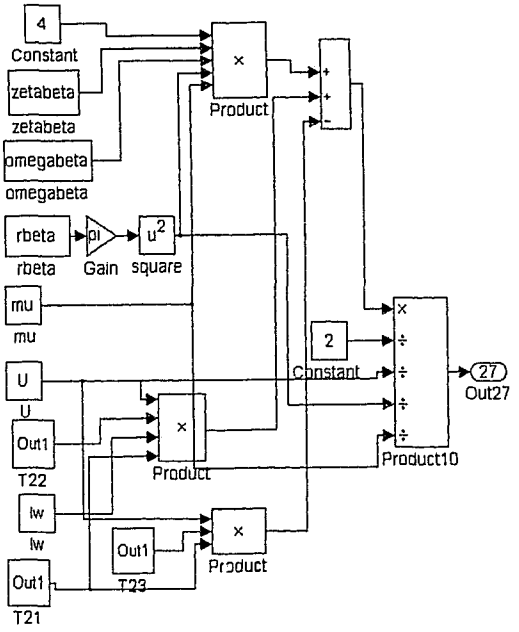
q3



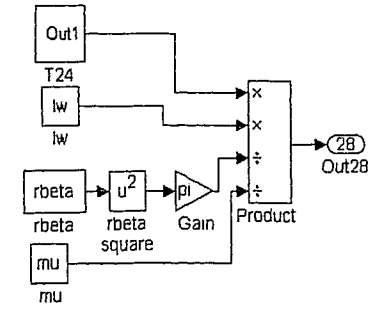
q4



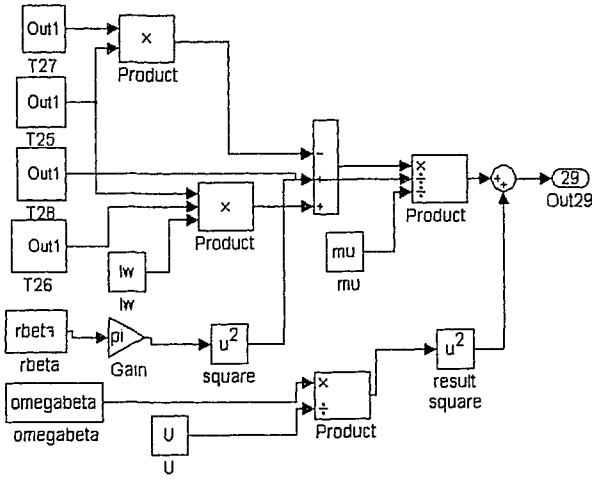
q5



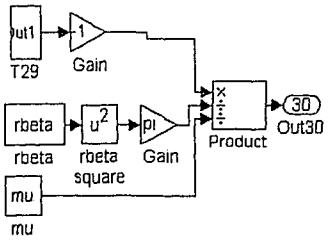
q6



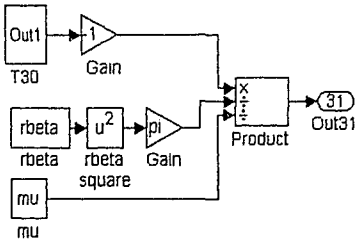
q8



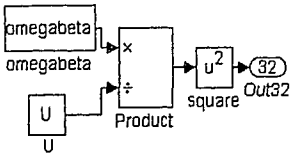
q9



q11



q12



q13

Appendix III ***Ordinary differential equations calculation***

$$m_1\xi'' + m_2\alpha'' + m_3\beta'' + m_4\xi' + m_5\alpha' + m_6\beta' + m_7\xi + m_8\alpha + m_9\beta + m_{11}y_1 + m_{12}y_2 = 0 \quad (2\text{-}3)$$

$$n_1\xi'' + n_2\alpha'' + n_3\beta'' + n_4\xi' + n_5\alpha' + n_6\beta' + n_8\alpha + n_9\beta + n_{11}y_1 + n_{12}y_2 = 0 \quad (2\text{-}4)$$

$$q_1\xi'' + q_2\alpha'' + q_3\beta'' + q_4\xi' + q_5\alpha' + q_6\beta' + q_8\alpha + q_9\beta + q_{11}y_1 + q_{12}y_2 = -q_{13}\beta_c \quad (2\text{-}5)$$

In order to solve the ξ, α and β, the above equations are rewritten as:

$$m_1\xi'' + m_2\alpha'' + m_3\beta'' = -(m_4\xi' + m_5\alpha' + m_6\beta' + m_7\xi + m_8\alpha + m_9\beta + m_{11}y_1 + m_{12}y_2)$$

$$n_1\xi'' + n_2\alpha'' + n_3\beta'' = -(n_4\xi' + n_5\alpha' + n_6\beta' + n_8\alpha + n_9\beta + n_{11}y_1 + n_{12}y_2)$$

$$q_1\xi'' + q_2\alpha'' + q_3\beta'' = -(q_4\xi' + q_5\alpha' + q_6\beta' + q_8\alpha + q_9\beta + q_{11}y_1 + q_{12}y_2 + q_{13}\beta_c)$$

The right parts of the above three equations are expressed in the following way:

$$S = -(m_4\xi' + m_5\alpha' + m_6\beta' + m_7\xi + m_8\alpha + m_9\beta + m_{11}y_1 + m_{12}y_2)$$

$$T = -(n_4\xi' + n_5\alpha' + n_6\beta' + n_8\alpha + n_9\beta + n_{11}y_1 + n_{12}y_2)$$

$$U = -(q_4\xi' + q_5\alpha' + q_6\beta' + q_8\alpha + q_9\beta + q_{11}y_1 + q_{12}y_2 + q_{13}\beta_c)$$

so:

$$m_1\xi'' + m_2\alpha'' + m_3\beta'' = S$$

$$n_1\xi'' + n_2\alpha'' + n_3\beta'' = T$$

$$q_1\xi'' + q_2\alpha'' + q_3\beta'' = U$$

Solving these new formed equation, we have:

$$\xi'' = -(n_2 \times q_3 - n_3 \times q_2) \times \mathrm{S/D} + (m_2 \times q_3 - m_3 \times q_2) \times \mathrm{T/D} - (m_2 \times n_3 - m_3 \times n_2) \times \mathrm{U/D}$$

$$\alpha'' = (n_1 \times q_3 - n_3 \times q_1) \times \mathrm{S/D} - (m_1 \times q_3 - m_3 \times q_1) \times \mathrm{T/D} - (-m_1 \times n_3 + m_3 \times n_1) \times \mathrm{U/D}$$

$$\beta'' = -(n_1 \times q_2 - n_2 \times q_1) \times \mathrm{S/D} + (m_1 \times q_2 - m_2 \times q_1) \times \mathrm{T/D} + (-m_1 \times n_2 + m_2 \times n_1) \times \mathrm{U/D}$$

where:

$$\mathrm{D} = -\mathrm{m}_1 \times n_2 \times q_3 + \mathrm{m}_1 \times n_3 \times q_2 + n_1 \times m_2 \times q_3 - n_1 \times m_3 \times q_2 - q_1 \times m_2 \times n_3 + q_1 \times m_3 \times n_2$$

We can now build the airfoil-flap combination model under Simulink® (see Figure 14).

Appendix IV ***M-files for simulation***

Initial condition:

% This file contains the airfoil-flap structure parameters, initial motion information
% and energy control law parameters. This file will be called every time when
% simulation begins.

$\mu = 100;$

$\omega_\xi = 0.2;$

$\omega_\beta = 1.5;$

$r_\alpha = 0.5;$

$r_\beta = \text{sqrt}(0.0015);$

$a_h = -0.5;$

$x_\alpha = 0.25;$

$x_\beta = 0.002\ ;$

$c_\beta = 0.6;$

$\zeta_\xi = 0;$

$\zeta_\alpha = 0;$

$\zeta\beta = 0;$

$lw = 1;$

$aw = 0.165;$

$bw = 0.0455;$

$cw = 0.335;$

$dw = 0.3;$

$U = 6.9;$

$Tc_\xi = 0;$

$Tc\alpha = -1.86;$

$Tg_\xi = 0.0698;$

$Tg_\alpha = 2.8;$

$\alpha_T = 4;$

$\zeta = 0.6;$

$kn = 0.04;$

Animation:

```
function [sys,x0]=AIRFOIL_Anim(t,x,u,flag,ts);
% AIRFOIL_Anim S-function for animating the motion of a 3DOF airfoil-flap.
% The animation shows the relative pitch and plunge motions and the
% associated control surface deflections.

%  This file is written based on bact_anim.m by
%  Marty Waszak, 7-31-95
%       NASA Langley Research Center
%       Hampton, VA 23681-0001
%       (804) 864-4015
%       m.r.waszak@larc.nasa.gov

global yAF xAF xFP yFP AIRFOIL
% global icount B_Movie

if flag==2,
   if any(get(0,'Children')==AIRFOIL),
    if strcmp(get(AIRFOIL,'Name'),'AIRFOIL Animation'),
        set(0,'currentfigure',AIRFOIL);

         xyAF=[ ...
                -12  0
                 8  0 ];
                xyFP=[ ...
                 8  0
                 12  1 ];


        xAF=xyAF(:,1);              yAF=xyAF(:,2);
```

```
xFP=xyFP(:,1);                yFP=xyFP(:,2);

        dLE = abs(xAF(1));

        dH2 = abs(xAF(2));
        dFP = abs(xFP(2) - xFP(1));


        hscale = 50;
        ascale = 50*pi/180;
        dscale = 10*pi/180;
        bias   = -2*dscale;

        xi  = -u(1)*hscale/5;
        alpha  = u(2)*ascale;
        beta = u(3)*dscale;


        xLE = -dLE*cos(alpha);          yLE = xi + dLE*sin(alpha);

        xH2 =  dH2*cos(alpha);          yH2 = xi - dH2*sin(alpha);


        xFP = xH2 + dFP*[cos(beta)];
        yFP = yH2 - 30*dFP*[sin(beta)];

        x1=[xLE; xH2];          y1=[yLE; yH2];
        x2=[xH2; xFP];          y2=[yH2; yFP];

        hndl_vec=get(gca,'UserData');
        hndl1 = hndl_vec(1);
```

```
            hndl2 = hndl_vec(2);

            set(hndl1,'XData',x1,'YData',y1);
            set(hndl2,'Xdata',x2,'YData',y2);

        drawnow;

%       icount = icount + 1;
%       B_Movie(:,icount) = getframe;

    end
  end
  sys=[];

elseif flag == 4 % Return next sample hit

  % ns stores the number of samples
  ns = t/ts;

  % This is the time of the next sample hit.
  sys = (1 + floor(ns + 1e-13*(1+ns)))*ts;

elseif flag==0,

   % Initialize the figure for use with this simulation
   AIRFOIL_AnimInit('AIRFOIL Animation');
    [flag, AIRFOIL] =figflag('AIRFOIL Animation');
   axis([-15 15 -15 15]);
   hold on;

   xyAF=[ ...
```

```
      -12 0
       8 0 ];
   xyFP=[ ...
       8 0
       12 -1 ];


  xAF=xyAF(:,1);     yAF=xyAF(:,2);
  xFP=xyFP(:,1); yFP=xyFP(:,2);

  x1=[xAF];  y1=[yAF];
    x2=[xFP]; y2=[yFP];

  % Draw the reference line for the model
    plot([-15 15],[0 0],'w','LineWidth',1);
    plot([0 0],[-10 10],'w','LineWidth',1);
    text(-12,-12,'(This is only for motion demos. purpose!)')

%
hndl=plot(x1,y1,'y',x2,y2,'r',x3,y3,'g','EraseMode','background','LineWidth',3);
   hndl=plot(x1,y1,'y',x2,y2,'r','EraseMode','normal','LineWidth',3);
   set(gca,'UserData',[hndl]);
   %set(gcf,'Color','w');

%  icount = 1;
%  B_Movie(:,icount) = getframe;

   sys=[0 0 0 3 0 0];
%  sys=[0 0 #outputs #inputs 0 0];
   x0=[];
end;
```

Animation initialization:

```
function figNumber=AIRFOIL_AnimInit(namestr)
% BACT_ANIMINIT Initializes a figure for the BACT SIMULINK animation.

%  Ned Gulley, 6-21-93
%  Copyright (c) 1984-94 by The MathWorks, Inc.

if (nargin == 0)
    namestr = 'SIMULINK Animation';
end

[existFlag,figNumber]=figflag(namestr);

if ~existFlag,
   % Now initialize the whole figure...
   position=get(0,'DefaultFigurePosition');
     position(2) = [50];
   position(3:4)=[400 300];
   figNumber=figure( ...
      'Name',namestr, ...
      'NumberTitle','off', ...
    'BackingStore','off', ...
    'Position',position);
   axes( ...
      'Units','normalized', ...
      'Position',[0.05 0.05 0.70 0.95], ...
      'Visible','off', ...
    'DrawMode','fast');

   %=================================
```

```
% Information for all buttons
labelColor=[0.8 0.8 0.8];
yInitPos=0.90;
top=0.95;
bottom=0.05;
left=0.80;
btnWid=0.15;
btnHt=0.10;
% Spacing between the button and the next command's label
spacing=0.04;

%=================================
% The CONSOLE frame
frmBorder=0.02;
yPos=0.05-frmBorder;
frmPos=[left-frmBorder yPos btnWid+2*frmBorder 0.9+2*frmBorder];
h=uicontrol( ...
   'Style','frame', ...
   'Units','normalized', ...
   'Position',frmPos, ...
  'BackgroundColor',[0.5 0.5 0.5]);

%=================================
% The CLOSE button
labelStr='Close';
callbackStr='close(gcf)';
closeHndl=uicontrol( ...
   'Style','pushbutton', ...
   'Units','normalized', ...
   'Position',[left bottom btnWid btnHt], ...
   'String',labelStr, ...
```

```
    'Callback',callbackStr);
end;

cla reset;
set(gca,'DrawMode','fast');
axis off;
```