Control Power Optimization using Artificial Intelligence for Hybrid Wing Body Aircraft

Rupanshi Chhabra

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
In
Aerospace and Ocean Engineering

Rakesh K. Kapania
Joseph A. Schetz
Sameer B. Mulani

June 30, 2015
Blacksburg, Virginia

Keywords: Artificial Intelligence, Artificial Neural Network, Optimization, Hybrid Wing Body, Genetic Algorithm, Hinge Moments

Control Power Optimization using Artificial Intelligence for Hybrid Wing Body Aircraft

Rupanshi Chhabra

ABSTRACT

Traditional methods of control allocation optimization have shown difficulties in exploiting the full potential of controlling a large array of control surfaces. This research investigates the potential of employing artificial intelligence methods like neurocomputing to the control allocation optimization problem of Hybrid Wing Body (HWB) aircraft concepts for minimizing control power, hinge moments, and actuator forces, while keeping the system weights within acceptable limits. The main objective is to develop a proof-of-concept process suitable to demonstrate the potential of using neurocomputing for optimizing actuation power for aircraft featuring multiple independently actuated control surfaces and wing flexibility. An aeroelastic Open Rotor Engine Integration and Optimization (OREIO) model was used to generate a database of hinge moment and actuation power characteristics for an array of control surface deflections. Artificial neural network incorporating a genetic algorithm then performs control allocation optimization for an example aircraft. The results showed that for the half-span model, the optimization results (for the sum of the required hinge moment) are improved by more than 11%, whereas for the full-span model, the same approach improved the result by nearly 14% over the best MSC Nastran solution by using the neural network optimization process. The results were improved by 23% and 27% over the case where only the elevator is used for both half-span and full-span models, respectively. The methods developed and applied here can be used for a wide variety of aircraft configurations.

Acknowledgements

I would like to thank my advisor, Dr. Rakesh K. Kapania, for giving me the opportunity to work as part of his group. Thank you for your support, guidance, patience and most importantly, for having faith in me. My special thanks to Dr. Sameer B. Mulani, for giving me a great start in this project and always being able to answer my work related questions; and thank you Dr. Joseph A. Schetz, not only for being on my committee, but also for being so curious to learn more about artificial intelligence.

Thanks to Nathan Love for mirroring the Hybrid Wing Body aircraft model and also for all the time he spent brainstorming with us; Moustaïne Adegbindin for working on Forward Swept Wing and keeping all the people looking at the pretty pictures. I must thank Wei Zhao for always being able to lend a knowledgeable ear to my research. A big thanks to all my friends for their support and comments throughout the development of my thesis.

The research was funded by NASA Langley Research Center through the National Institute of Aerospace, Hampton, VA, (under the contract number NNL13AA08B, Task Order: NNL15XX##T) through the ARMD Seedling Fund project "Artificial Intelligence Based Control Power Optimization on Tailless Aircraft". From the Virginia Tech Team, thanks to the Project Monitor and Principal Investigator, Frank H. Gern (NASA Langley) for providing the X-48B hybrid wing body aircraft finite element model. We would also like to thank Jesse Quinlan (NASA Langley) for his suggestions and help. We sincerely appreciate the insights provided by Boeing researchers, D. Brown and N. H. Princen, relating optimization.

I would never have even been in a position to complete this work without the loving support of my parents, and my little sister Jayati, who have always encouraged me through all of my academic ventures. Special thanks to my boyfriend, Yogin, for believing in me that I could do this, for motivating me by taking so much interest in my work and also for proofreading all of my reports and thesis drafts.

Rupanshi Chhabra

Contents

List of Tables

Page Nos.

List of Figures

Page Nos.

Nomenclature

Abbreviations

AELINK	=	Aeroelastic Linkage Coefficients
AIC	=	Akaike's Information Criterion
AICc	=	Corrected Akaike's Information Criterion
ANN	=	Artificial Neural Network
AOA	=	Angle of Attack
ARMD	=	Aeronautics Research and Mission Directorate
AUV	=	Autonomous Underwater Vehicle
BDF	=	Bulk Data File
BWB	=	Blended Wing Body
CFD	=	Computational Fluid Dynamics
DLM	=	Doublet-Lattice Method
DMI	=	Direct Matrix Input
FEM	=	Finite Element Method
GA	=	Genetic Algorithm
GPS	=	Global Positioning System
HMS	=	Sum of Absolute Value of Hinge Moments
HWB	=	Hybrid Wing Body
LHS	=	Latin Hypercube Sampling
LM	=	Levenberg-Marquardt
MATLAB	=	Matrix Laboratory developed by MathWorks
MLP	=	Multi-Layer Perceptron
MSC	=	MacNeal - Schwendler Corporation
MSE	=	Mean Squared Error

NASA	=	National Aeronautics and Space Administration
OREIO	=	Boeing Open Rotor Engine Integration on a BWB
PMC	=	Primitive Monte Carlo
RMSE	=	Root Mean Squared Error
SOL	=	Solution (MSC.Nastran)
SBC	=	Schwarz's Bayesian Criterion
SSE	=	Sum of Squared of Errors
TE	=	Trailing Edge
UAS	=	Unmanned Aerial System
UAV	=	Unmanned Aerial Vehicle
VLM	=	Vortex-Lattice Method

Symbols

A	=	Output bias
A_{ij}	=	Aerodynamic influence coefficient matrix
Ac_i	=	Activation Function
B	=	Vector connecting the hidden layer to the output neuron
C	=	Vector of H bias terms for the hidden neurons
c_e	=	Mean aerodynamic chord of the elevator of the same area
C_L	=	Lift coefficient
C_{m_0}	=	Pitching moment
C_{H_e}	=	Hinge moment coefficient
D_{jk}^1 , D_{jk}^2	=	Real and imaginary parts of substantial differential matrix
$[D_{jk}]$	=	substantial derivative matrix for the aerodynamic displacements
$[D_{jx}]$	=	substantial derivative matrix for the extra aerodynamic points
e	=	Vector of Network Errors
f_j	=	Pressure on lifting element

$\{F_k\}$	=	Aerodynamic forces
$\{F_g\}$	=	Structurally equivalent forces
g	=	Gradient
$[G_{kg}]$	=	Interpolation matrix
h	=	Number of neurons in the hidden layer
H_e	=	Hinge moment
$[H]$	=	Hessian matrix
J	=	Jacobian matrix
k	=	Reduced frequency
K_{aa}	=	Structural stiffness matrix
$\boldsymbol{l_i}$	=	Transverse segment of the horseshoe vortex
M_{aa}	=	Structural mass matrix
n	=	Number of samples used for training of a network
p	=	Total number of parameters, i.e. weights plus biases
P_a	=	Vector of applied loads
q	=	Flight dynamic pressure
$q(x,y)$	=	Disturbance velocity due to the modeled surface
Q_{aa}	=	A-set aerodynamic influence coefficient matrix
S_e	=	Elevator area behind the hinge line
S_{kj}	=	Integration matrix
u_k	=	Displacement at aerodynamic grid points
$\{u_k\}$	=	Structural grid point deflections
$\{u_g\}$	=	Aerodynamic grid point deflections
$\{u_x\}$	=	Extra aerodynamic points used to describe aerodynamic control surface deflections and overall rigid body motions
V	=	Velocity vector flow field
V_∞	=	Freestream velocity vector field
w_j^g	=	Static aerodynamic downwash
w_j	=	Downwash
$\{w_j\}$	=	Aerodynamic degrees of freedom

ω	=	Angular frequency
$\varphi(x, y)$	=	Perturbation velocity potential
w_{ji}	=	Weights in a Neural Network
x	=	Input for a Neural Network
X	=	Linear input neurons
z	=	Target for a Neural Network

This page has been intentionally left blank

Chapter 1

Motivation

"The question of whether a computer can think is no more interesting than the question of whether a submarine can swim. "

— Edsger W. Dijkstra

Autonomy [2, 3] has replaced the human operator in many applications and this replacement makes it necessary to advance the development of the autonomous systems techniques. These autonomous systems operate in all kinds of environments, like air (Unmanned Aerial Vehicles), in space (Mars Rover [4]), in or under water (Sentry (Fig. 1.1), Seaglider: Autonomous Underwater Vehicles (AUVs) [5]), and even on land (Google Self-Driving Car (Fig. 1.2) [6], Autonomous Robots). Autonomy is extensively used for unmanned deep space missions and also in support of human missions. Research has been conducted in each of these application areas to determine the potential roles and benefits of autonomy.

Figure 1.1 Sentry, an autonomous underwater vehicle capable of exploring the ocean down to 6,000 meters
(Copyright © Woods Hole Oceanographic Institution, used with permission of Christopher Reddy at WHOI)

Unmanned Aerial Vehicles (UAVs) also known as Unmanned Aerial Systems (UAS) are inherently very different from any manned vehicle. UAVs have attained a great amount of attention in these recent years. Today, UAVs cover a very wide range of applications like archaeology, architecture, cultural heritage, large scale local mapping, 3D city modeling,

Figure 1.2 Google Self-Driving Car
(© 2015 Google Inc, used with permission. Google and the Google logo are registered trademarks of Google Inc.)

change detection in urban and suburban areas, characterization of river and other landscapes, landfill surveys, agriculture and forestry, natural and man-made hazards, landslide investigations, geology, environmental and construction monitoring, search and rescue, traffic supervision and traffic accident recording, crime fighting and coordination of police services, fire brigade crisis management, multimedia applications and generation of video games, etc. The diversity of applications is already enormous and ever increasing as time goes by [3]. This is one of the reasons, why so many companies are trying to get into this technology. The other reasons are the seeming ease of operation of such aircraft and the economic benefits which this technology offers. UAVs have been around much longer than most people realize and all these developments in UAVs trace their beginnings to World War I. During World War II, these aircraft could be controlled by radio signals. By the 1950s, these vehicles could return from a mission and be recovered. Today, UAVs are involved in a wide range of missions [7]. Advancements in avionics, navigation via Global Positioning System (GPS) and flight electronics have further fuelled the usage of UAVs.

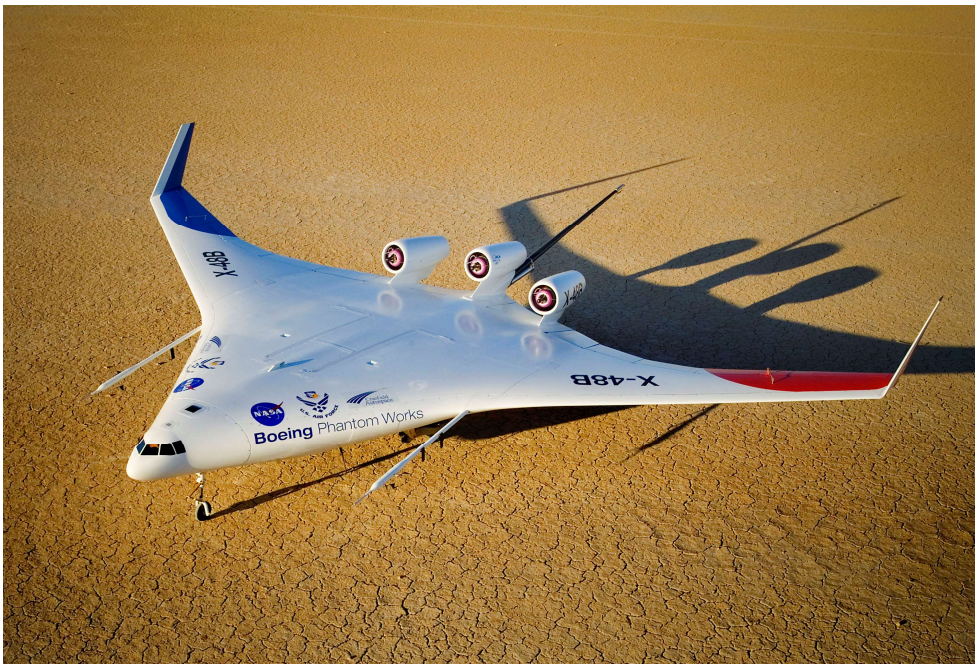


Figure 1.3 Boeing X-48B (NASA Photo)

The Boeing X-48B (see Fig. 1.3) is an experimental UAV aimed at studying the characteristics of the hybrid/blended wing body (HWB/BWB) for an aircraft. The HWB concept represents a potential breakthrough in subsonic transport efficiency. This is why, HWB configurations have attracted considerable interest lately. In the conventional tube-wing configurations, the fuselage contributes slightly to the total lift while adding considerable skin friction drag. HWBs are designed in a way that their payload volume is located inside the wing such that their fuselage contributes as much into the generation of lift, as the wing. They are tailless [8] flying wings which also reduce the drag. But the HWB is a cross between the conventional type and flying wing design.

The Boeing X-48B has 13 elevons on its trailing edges (TE), 2 rudders on the vertical fins, 1 elevator and 8 high-lift devices as shown in Fig. 1.4. Multiple, large, rapidly moving control surfaces are required, as this tailless configuration has short moment arms for pitch and directional control. Even to perform basic trim, control, pitch stability augmentation and wing load alleviation, trailing-edge devices and winglet rudders are called upon [10]. In addition, to improve engine noise shielding, the OREIO vertical tails are highly canted, resulting in a significant amount of pitch authority for the rudder. Therefore, some of the control surfaces like the elevator and the rudder can perform multiple functions. This makes the control surface allocation a critical issue, thereby, making it difficult to find the most efficient way to fly X-48B [10,11]. This presents an opportunity for the application of some form of autonomy or artificial intelligence.

Reducing actuation power is an enabler for ultra-efficient commercial transport aircraft that, therefore, directly impacts common challenges of simultaneously reducing fuel consumption, emissions, and noise of future air traffic. HWB aircraft and other innovative concepts suitable for this research offer the potential to achieve significant fuel burn savings of over 25%, with resulting emission reductions, as well as community noise benefits, thereby countering the impact of dramatic increases in future air traffic volume [9].

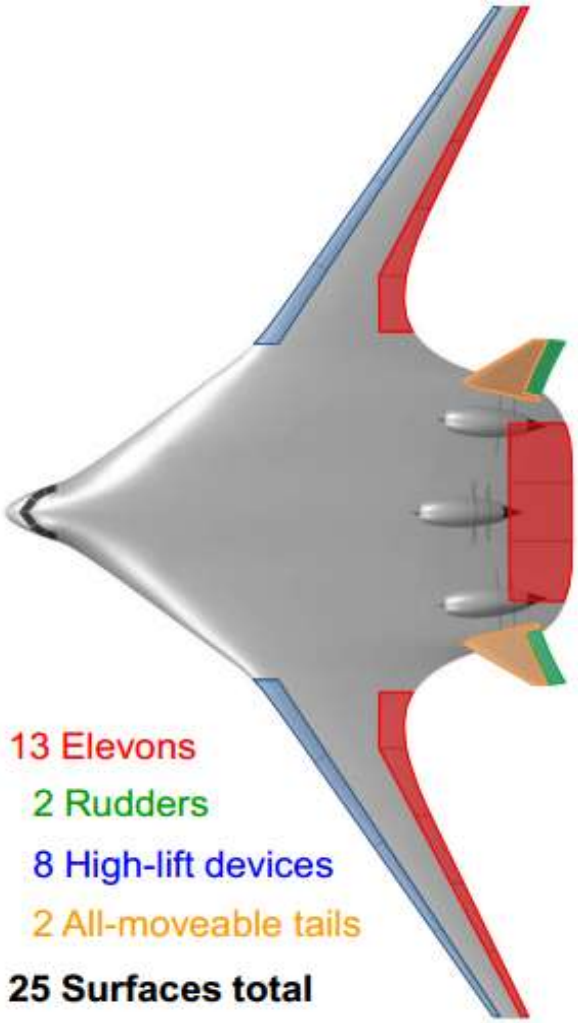


Figure 1.4 Boeing X-48 OREIO HWB aircraft [*Gern, F.H., Vicroy, D.D., Mulani, S.B., Chhabra, R., Kapania, R.K., Schetz, J.A., Brown, D., and Princen, N.H., "Artificial Intelligence Based Control Power Optimization on Tailless Aircraft", ARMD Seedling Fund Phase I Final Report, NASA Technical Memorandum, NASA/TM-2014-218671, November, 2014.*]

Higher levels of autonomy and automation using artificial intelligence capabilities enable a wider variety of more advanced technology and enable humans to focus on those tasks for which they are better suited. The author's motivation for this research manifests in the venture of allocating appropriate, physically possible control surface deflections using artificial intelligence while requiring minimum actuation power.

1.1 Hybrid/Blended Wing Body Aircraft

A Hybrid/Blended wing body aircraft is a configuration where the wing and fuselage are integrated which essentially results in a large flying wing. BWB aircraft were previously called ‘tailless airplanes’ or ‘Flying-Wing aircraft’ [12, 13]. The shape of the aircraft is the first thing that attracts one’s attention. The aircraft’s appearance is like a single extended wing because of the BWB design.

Some 85 years after the Wright Flyer, the then McDonnell Douglas aerodynamicist, Robert Liebeck proposed the concept of Blended Wing Body (BWB) aircraft [14]. However, the BWB aircraft concept has been on the drawing board even longer. Even though, the BWB configuration has shown promise in terms of aerodynamic efficiency, and has also shown

Figure 1.5 X-48B developed by Boeing Phantom team and NASA (NASA Photo)

quite remarkable advantages against the conventional airframe, which included improved fuel efficiency, longer flight range, reliability and even lower manufacturing costs, such a concept has only been applied to military aircraft to obtain a low radar cross-section [12]. HWB’s shape also reduces total wetted area. In this imaginative layout there is no need for a conventional tail [15].

In recent years, the Boeing Phantom team, NASA and the Air Force Research Laboratory (AFRL) have developed X-48 as a BWB concept aircraft (Fig. 1.5), and it has been observed very closely in its flight testing phase since 2006 [8].

In 2012, the remotely operated X-48C HWB aircraft, designed by Boeing and built by Cranfield Aerospace Limited in partnership with NASA, took its first flight from Rogers Dry Lake at Edwards Air Force Base, California. The X-48C model, was formerly the X-48B BWB aircraft, which was modified to evaluate the low-speed stability and control of a low-noise version of an HWB aircraft design [16].

April 9, 2013 marked the end of the flight testing of X-48C [17]. Studies on HWB lead to the understanding that a HWB aircraft offers a tremendous promise of significantly greater fuel efficiency and reduced noise. Also, it could be controlled as effectively as a conventional aircraft.

Hybrid Wing Body (HWB) platforms feature multiple control surfaces, with large control surface geometries leading to large hinge moments and high control power demands. Due to the large number of control surfaces on a HWB aircraft, there is no unique relationship between control inputs and resulting aircraft response, i.e. different combinations of control surface deflections may result in the same maneuver, but with large differences in control power. Stability augmentation and control power optimization for the HWB aircraft concept is an enabler for its success [9].

1.2 Problem Definition

This project investigates the potential of applying artificial intelligence methods like neurocomputing to the control allocation optimization problem of HWB aircraft concepts. While traditional methods of control allocation optimization may have limitations in exploiting the full potential of controlling large arrays of control devices, artificial neutral networks (ANN), inspired by biological nervous systems, have successfully been applied to a variety of complex optimization problems [9, 16].

The flight conditions specified for this problem are:

Load Case: 2.5g

Maneuver: Pitch, symmetric (full payload, residual fuel)

Mach: 0.5

Dynamic Pressure: 1.769 psi

Model Specifications (Half Span, HWB X-48B) (Fig. 1.6):

Span: 2552.1 in

Area: 576720 in^2

Chord: 818.35 in

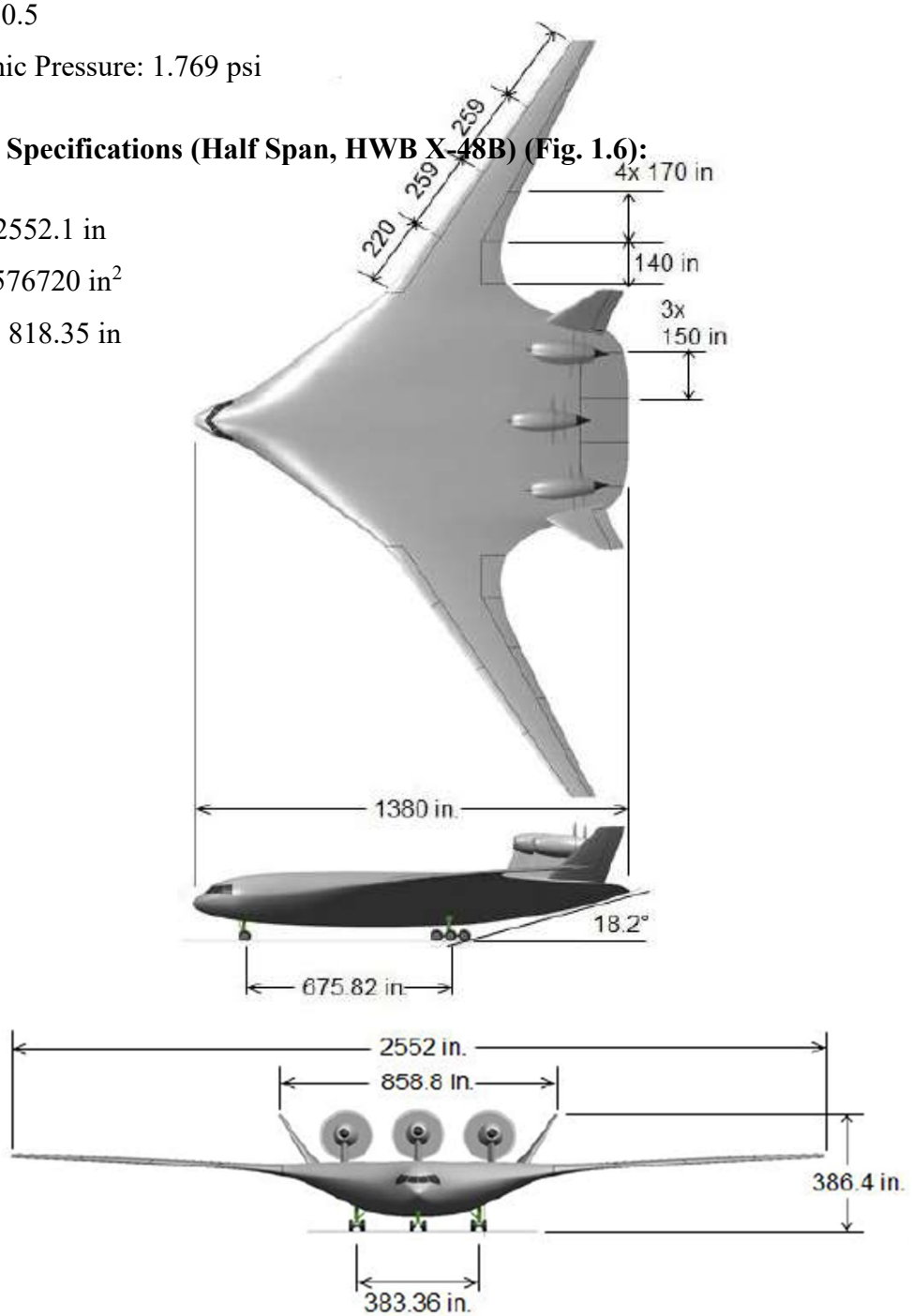


Figure 1.6 Model Specifications of Boeing X-48B HWB OREIO aircraft (NASA Photo)

1.3 Proposed Solution

This project employs a finite element based aeroelastic HWB model (Fig. 1.7), inspired by the Boeing X-48B OREIO aircraft, and wind tunnel and flight test data to develop a database using MSC Nastran that can be used to develop an artificial neural network (ANN) to perform control allocation optimization for tailless aircraft at the conceptual design level.

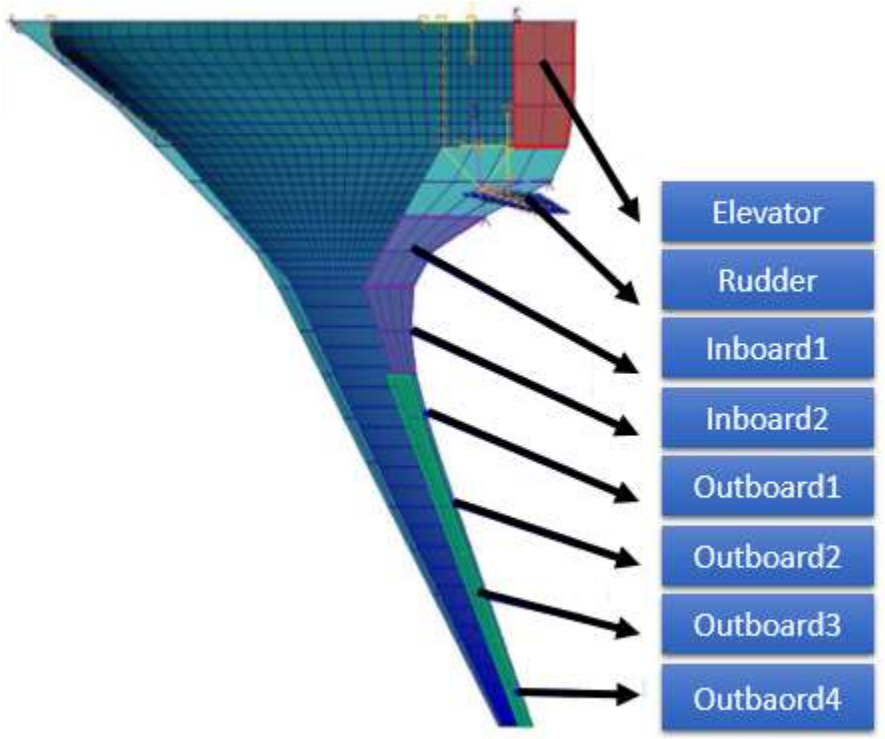


Figure 1.7 Control Surfaces in Boeing X-48B OREIO aircraft

ANNs are inspired by biological nervous systems, and neuro-computing has been successfully applied to a variety of complex optimization problems. The prior use of artificial intelligence to overcome the shortcomings of conventional methods for control allocation has not been explored to a significant extent in the open literature. A literature search yielded very few results, most of them not applicable to this critical need for HWB configurations [9, 17].

Artificial intelligence (AI) offers new ways to optimize functions. Genetic Algorithm (GA) is one of the model of machine learning that is used as a search and optimization tool. It derives its behavior from a metaphor of the mechanisms of evolution in nature. GA is used in this project to optimize control power while minimizing hinge moments and structural loads. The optimum result obtained from the GA and ANN, is then checked using MSC Nastran.

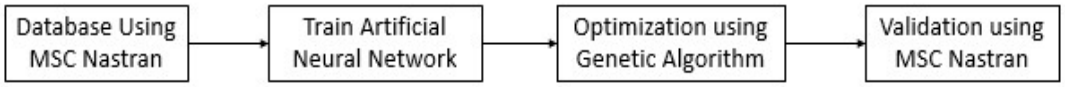


Figure 1.8 Proposed Solution

Each control surface on an aircraft has a hinge of some sort. Actuators are implemented and even sized, based on the magnitude of the force required to rotate the control surface about the hinge [18]. On deflecting the control surface, there is an aerodynamic moment experienced about that hinge, called the hinge moment. The pilot must provide a moment to counter that hinge moment, if he needs to deflect that control surface. The force exerted by the pilot is augmented with mechanical ratios provided by the control surface actuators.

The actuation power is a function of the control surface deflection, since the actuation power is computed as the multiplication of deflection of the control surface and hinge moment generated by the control surface due to that deflection [19]. As an initial figure of merit, the sum of absolute value of hinge moment (HMS) was chosen here, for control power to be minimized.

Also, the size of a control surface implies a constraint on its design to minimize hinge moments. Hinge moments are related to the scale of the control surface in such a way that if the area increases as the square of the scale, the moment increases with the cube of the scale [10].

The proposed concept applies to the design and development stages of the future air vehicles. Artificial intelligence can be used to develop optimized control laws in a much more efficient manner than by using traditional methods; and such optimized control laws are an enabler for an entire fleet of revolutionary aircraft. Without the necessity to implement artificial intelligence on the aircraft itself, no certification issues are associated with such a solution.

A collaborative team of researchers from NASA Langley, Virginia Tech, and Boeing Research and Technology are exploring the use of artificial neural networks to develop innovative control algorithms minimizing control power, hinge moments, and actuator forces to keep system weights within acceptable limits.

1.4 Collaborative Team

Virginia Polytechnic Institute and State University

Profs. Rakesh K. Kapania, **Joseph A. Schetz**, **Sameer B. Mulani (Presently, at The University of Alabama)**,
Rupanshi Chhabra: Neurocomputing and actuation power optimization
Nathan Love: Aeroservoelastic finite element modeling (Full-Span Model)
Moustaïne Adegbindin: Forward Swept Wing Aircraft Example

NASA Langley Research Center

Frank H. Gern (PI): Project management
Dan D. Vicroy, Jesse R. Quinlan: Aeroservoelastic finite element modeling (Half-Span Model)

Boeing Research and Technology

Norman H. Princen, Derrell Brown: Actuator dynamics; Control surface geometry, effectiveness and deflection limits; provide wind tunnel and flight-test data.

Chapter 2

Artificial Intelligence

History is filled with stories of the creation of intelligent machines. Artificial Intelligence (AI), too, has a long history, but it is still actively growing and changing. Prior to the 1970s, AI had generated considerable interest and also considerable hype from the research community. Many interesting systems had been developed, but these fell short of the predictions made by some in the community which led to a loss of interest in the technology as well as the hardware built for AI. In 1943 [21], neurophysiologist Warren McCulloch and mathematician Walter Pitts wrote a paper on how neurons might work. To describe how neurons in the brain might work, they modeled a simple neural network using electrical circuits. Neural networks breathed new life into this evolving field, providing additional ways for classification and learning.

AI is increasingly prevalent in our everyday lives. Today, it is used in a variety of industries from gaming, to finance, to robotics and to media. AI is enjoying a boom in its capabilities, due to the rise in processing power and the growing abundance of digitally available data. Today's deep learning systems can teach themselves to perform some tasks, from pattern recognition to translation, almost as well as humans can, by mimicking the layers of neurons in a human brain. As a result, things that once called for a living mind, from

interpreting pictures to playing video games, are now within the scope of computer programs [22].

Most of us use Facebook today. When uploading pictures on Facebook, DeepFace (an algorithm unveiled by Facebook in 2014), recognizes individual human faces in images 97% of the time [22, 23]. Different kinds of neural networks are used in the context of psychologically inspired models of human cognitive function. Apple's Siri is a rudimentary form of AI that millions of people use every day [24]. One of the good examples of AI is Cleverbot (Fig. 2.1), a light-hearted online AI experiment to chat with. It provides a simple back-and-forth correspondence, but should you decide to break the flow of conversation, it may be unable to provide suitable feedback [25]. Another developing example of AI is Cyc. Cyc not only recalls the data in its databases but can also come to knowledge based conclusions [26].

Figure 2.1 Cleverbot, an AI to chat with.
(©2015 Rollo Carpenter, used with permission of Rollo Carpenter)

AI is already powerful enough to make a dramatic difference to human life. It can now enhance human endeavor by complementing what people can do. Despite a century of poking and prodding at the brain, psychologists, neurologists, sociologists and

philosophers are still a long way from an understanding of exactly how a mind might be made or what one is.

The closer we get to building truly semantic and autonomous systems, the more complex may be the consequences of their abuse and malfunction. Today, all we see is that if a computer program produces an error, it terminates abruptly, but it will be a completely different scenario to have a semantic error hidden deep within the vocabulary that abuses the inherent logic of the system, making it take undesirable action, such as becoming aggressive or behaving in other unacceptable ways or making undesirable decisions on its own.

Some scientists, businessmen, researchers, marketers, engineers are in support of expanding our understanding of brain to improve AI, whereas some are against it. The world has been warned by one of the best physicists in the world, Stephen Hawking about the capability of AI. In his words, "The development of full artificial intelligence could spell the end of the human race" [27].

2.1 Artificial Neural Networks (ANNs)

2.1.1 Introduction

Even after working in this field for a very little amount of time, one realizes that the term "Neural Networks" is a very evocative term. It suggests machines that are something like brain. They indeed have something to do with brains and their study also makes contact with other branches of science, engineering and mathematics. The most non-technical definition of neural networks according to Kevin Gurney is,

"*An interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the inter-unit connections strengths, i.e. weights, obtained by a process of adaptation to, or learning from, a set of training patterns"* [28].

In order to understand this definition better, a quick look at the neurophysiological knowledge of biological neurons and of networks of such neurons may be helpful. It is estimated that there are about 100 billion neurons or nerve cells, in a human brain. Natural neurons communicate via electrical signals that are short-lived impulses. Neurons receive signals through *synapses* located on the dendrite or membrane of the neuron. Each neuron typically receives many thousands of connections from other neurons, and it is therefore constantly receiving a multitude of incoming signals, which eventually reach the cell body. When the signals received are strong enough, i.e. if the resulting signal exceeds some threshold then, the neuron is *activated* and generates a voltage impulse in response. This is then transmitted to other neurons via a branching fiber known as the *axon*. This signal might be sent to another synapse, and might activate other neurons, as illustrated in Fig. 2.2. [28, 29].

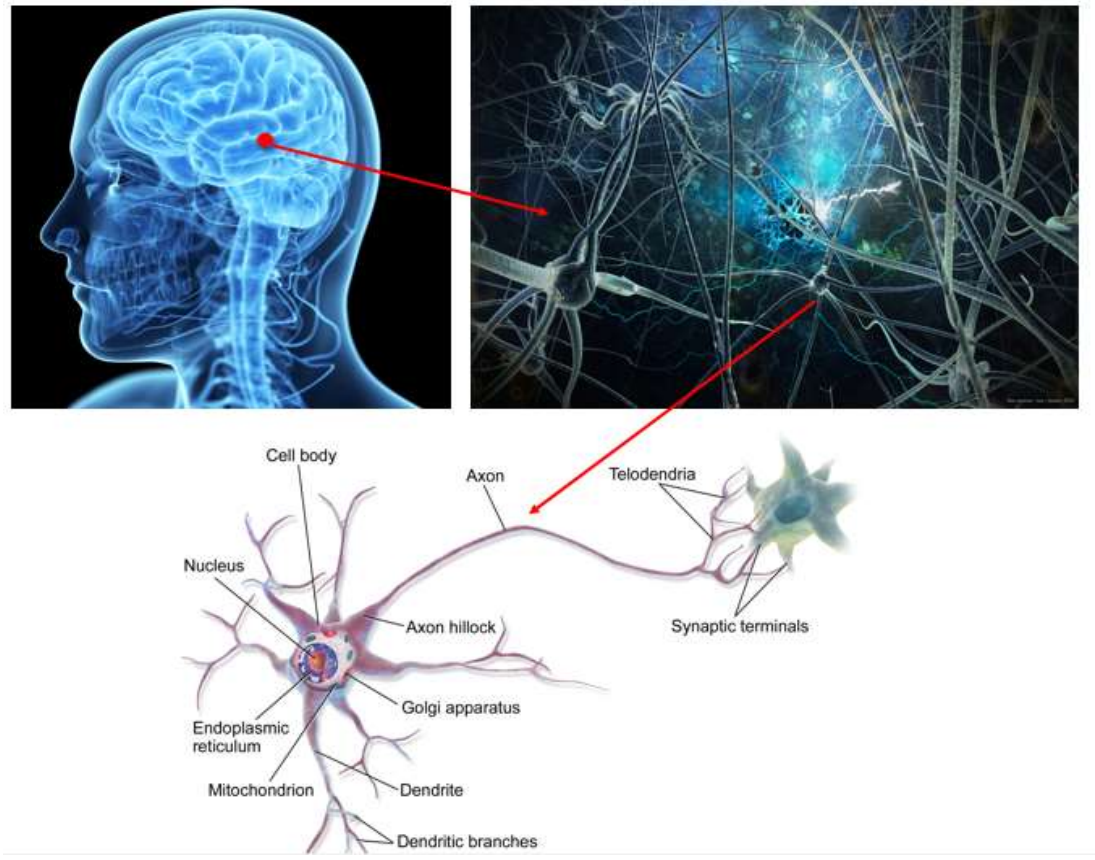


Figure 2.2 Biological neuron in human brain [*Mehrotra, K., Chilukuri M., and Sanjay R., Elements of Artificial Neural Networks. Boston: MIT Press, 1997*], used under fair use, 2015

It is this architecture and style of processing that we hope to incorporate in neural networks. A system of artificial neurons is referred to as a "network"; this network may be a single node or a large collection of nodes, where each one is connected to the other. Here, neurons refer to the mathematical model of a neuron (see Fig. 2.3), rather than the biological counterpart from which the mathematical model was derived. In neural networks, the *nodes* are the artificial equivalents of biological neurons. *Weights* are used to model *synapses*, so that each input is multiplied by a weight before being sent to the equivalent of the cell body. The weighted signals are then added together to supply a node *activation*. The activation is then compared to a *threshold*. If the activation is more than the threshold, the unit produces an output with some value (conventionally "1"), otherwise it outputs zero [30-32].

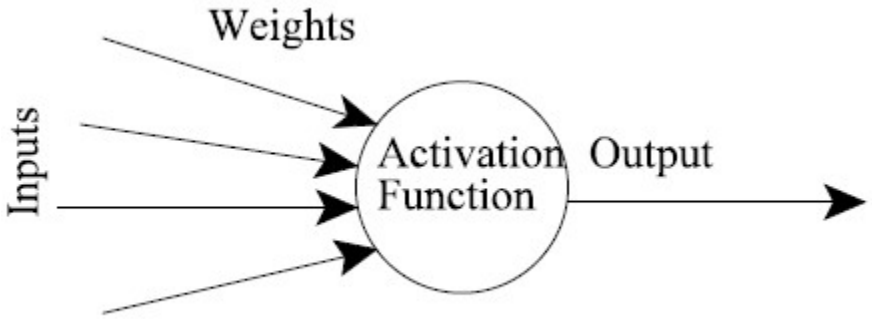


Figure 2.3 A mathematical model of an Artificial Neuron

2.1.2 Network Architecture

ANNs combine artificial neurons in order to process information. But, the complexity of real neurons is highly abstracted when modeling artificial neurons. ANNs basically consist of *inputs*, which are multiplied by weights, and then computed by a mathematical function which determines the activation of the neuron. This activation function can be used to give the desired behavior to the neuron. Another function computes the *output* of the artificial neuron:

$$z = \sum_i w_i x_i \quad (1)$$

Where w_i is the weight at the inputs x_i, and z is the output [33].

If the weight of the artificial neuron is high, the input would be strong. By adjusting the weights of an artificial neuron, one can obtain the output wanted for specific inputs. When there are hundreds of neurons in an ANN, it is quite difficult to specify all weights. There are algorithms which can adjust the weights in order to obtain the desired output from the network.

2.1.2.1 Training/Learning

The information is stored in the network weights which evolve by a process of stimulus from a set of pattern examples. Using different types of training paradigms, the weights are updated. When the required weight updates are made, the output is compared with the target, and accordingly, the new changes are made to the weights. This sequence of events is repeated iteratively many times until the network's behavior converges so that its response to each new pattern is close to the corresponding target. The process is known as the *training algorithm* which includes, any ordering of pattern presentation, criteria for terminating the process, etc. This training algorithm dictates the global algorithm that affects all the weights and biases of a given network. [28]

Supervised Learning

Supervised learning is a type of machine learning algorithm where is used a known dataset to train the network and then make predictions using that network. The training dataset needs to include the input data as well as the response values. Using this dataset, the supervised learning algorithm seeks to create a model that can make predictions of the response values for a new dataset. Using larger training datasets over a small sample space often yield models that can generalize well for new datasets, and hence, have higher predictive power.

Supervised learning includes two categories of algorithms [34]:

Classification: for response values, where the data can be separated into different categories, such as for responses that can have just a few known values, such as 'true' or 'false'.

Regression: for response values which are continuous. Regression for responses are real numbers. Regression models are often more computationally efficient.

2.1.2.2 Feedforward Network

Feedforward networks have one-way connections from input to output layers. The neurons are arranged in a layered structure in which each signal arises from an input and passes via *n* neurons before reaching an output. This feedforward structure is only one of many available networks and is generally used to place an input pattern into one of several classes according to the resulting pattern of outputs. They are most commonly used for prediction, pattern recognition, and nonlinear function fitting. There are many different kinds of feedforward networks, namely feedforward backpropagation, cascade-forward backpropagation, feedforward input-delay backpropagation, linear, and perceptron networks [31, 34].

2.1.2.3 Multi-Layer Perceptron

In general, it is difficult to incorporate prior knowledge into a neural network. Therefore, the network can only be as accurate as the data that is used to train the network. This is why, it is important that the data covers the range of inputs and outputs for which the network is going to be used.

A *single layer perceptron* is a perceptron having only one layer of variable weights and one layer of output neurons. A *multi-layer perceptron* (MLP) has a minimum of three layers, an input layer, an output layer and a hidden layer (see Fig. 2.4). The input layer consists of *input neurons* which are identity neurons. They forward the information exactly

as received. The hidden layer consists of *information processing neurons* and *binary neurons.* An information processing neuron processes the input information somehow. A binary neuron sums up all inputs by using the weighted sum as propagation function.

MLPs can be trained well to generalize within the range of inputs for which they have been trained. However, they do not have the ability to accurately extrapolate beyond this range, so it is important that the training data span a good range of the input space [34, 35].

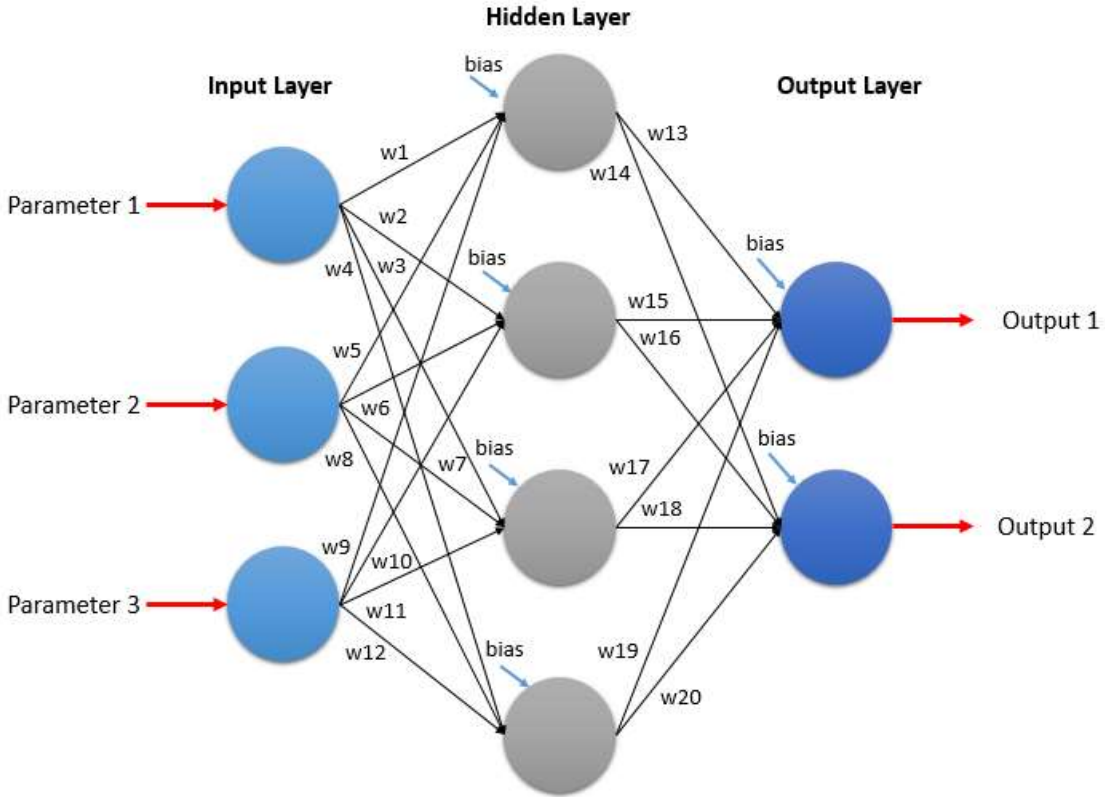


Figure 2.4 Architecture of Multi-Layer Perceptron. *Parameter* represents the input variable, *w(n)* are the weights, *Output* is the output obtained

2.1.2.4 Backpropagation

The backpropagation algorithm was proposed in 1986 by Rumelhart, Hinton and Williams for setting and modifying weights [36, 37]. The availability of a rigorous method to set intermediate weights to train hidden layers of ANNs proved to be very useful in multi-layer ANNs. It also opened the way to overcome the single-layer shortcomings, which almost lead to the end of ANNs.

Backpropagation is generally a gradient descent procedure with the error function receiving all weights as arguments and assigning them to the output error. It has been one of the most studied and used algorithms for neural networks learning. The backpropagation algorithm looks for the minimum of the error function in weight space using the method of gradient descent. The combination of weights which minimizes the error function is considered to be a solution of the learning problem. This method requires computation of the gradient of the error function at each iteration step. An appropriate transfer function should be used in perceptrons [37, 38].

The activation function of the artificial neurons in ANNs implementing the backpropagation algorithm is a weighted sum, the sum of the inputs x_i multiplied by their respective weights w_{ji}:

$$Ac_j(x, w) = \sum_{i=0}^{n} x_i w_{ji} \quad (2)$$

One can see that the activation depends only on the inputs and the weights.

2.1.3 Neural Networks in MATLAB

MATLAB allows one to concentrate on solving a problem without having to spend many hours pursuing neural network literature and developing the algorithms. Since the first neural model by McCulloch and Pitts (1943), there have been developed hundreds of different models considered as ANNs. The differences in them might be the functions, the

accepted values, the topology, the learning algorithms, etc. MATLAB provides a user-friendly and helpful GUI to develop almost all of these ANNs.

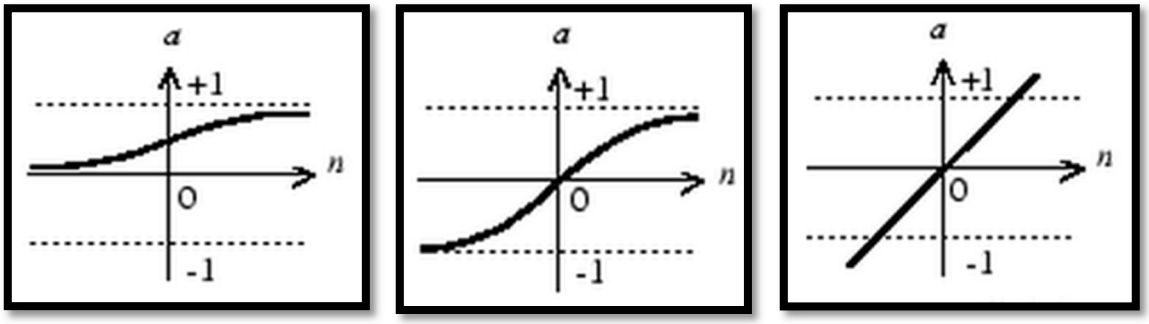
2.1.3.1 Neural Network Toolbox in MATLAB

The multilayer feedforward neural network is the one used by the Neural Network Toolbox™. Neural Network Toolbox supports a variety of supervised and unsupervised network architectures. With the toolbox's modular approach to building networks, network architectures are customized for this specific problem.

The toolbox provides functions and apps for modeling complex nonlinear systems that are not easily modeled with a closed-form equation. Neural Network Toolbox supports supervised learning with feedforward, radial basis, dynamic networks and learning vector quantization. With the toolbox one can design, train, visualize, and simulate neural networks. Neural Network Toolbox can be used for applications such as data fitting, pattern recognition, clustering, time-series prediction, and dynamic system modeling and control [34].

2.1.3.2 Network Topology

Transfer Functions



$a = logsig(n)$ $a = tansig(n)$ $a = purelin(n)$

Figure 2.5 Graphical representation of Transfer Functions [Beale, M.H., Hagan, M.T. and Demuth, H.B., *Neural Network Toolbox User's Guide*, 2015, The MathWorks, Inc., Natick, MA]

The three transfer functions (Fig. 2.5) most commonly used for multilayer networks are *logsig*, *tansig* and *purelin*. In multilayer networks, as the neuron's net input goes from negative infinity to positive infinity, the log-sigmoid transfer function (*logsig)* generates outputs between *0* and *1*. Multilayer networks can also use the tan-sigmoid transfer function (*tansig)*. Sigmoid output neurons are often used for pattern recognition problems, while linear output neurons are used for function fitting problems. The linear transfer function (*purelin)* is usually used in the input layer.

Architecture of the Network

Feedforward networks often have an input layer and one or more hidden layers of sigmoid neurons followed by an output layer of linear neurons (see Fig. 2.6). The use of linear neurons in the output layer is used for function fitting problems. To constraint the outputs of a network between a range, sigmoid transfer function should be used.

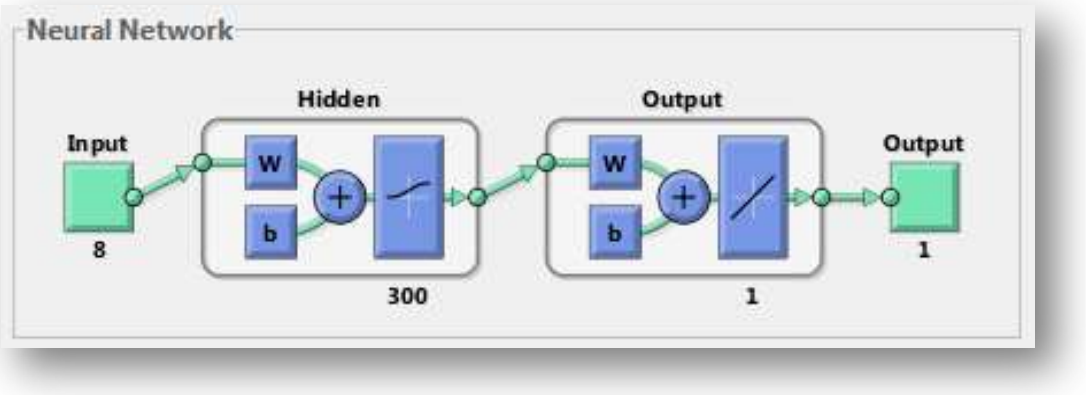


Figure 2.6 Example of architecture of Neural Network in MATLAB

In multilayer networks, the layer number determines the superscript on the weight matrix. With an appropriate number of layers and a sufficient number of neurons in the hidden layer, this network can be used for function approximation.

In MLPs, the complexity of the model is determined by the number of neurons in the hidden layer. The equation for the MLP model can be written as [39]:

$$Y = A + \sum_{n=1}^{h} B \tanh(C + DX) + noise \quad (3)$$

where,

- *A*, the output bias
- *h*, number of neurons in the hidden layer
- *X*, linear input neurons
- *B*, a vector of *h* weights connecting the hidden layer to the output neuron
- *C*, a vector of *h* bias terms for the hidden neurons
- *D*, a vector of *h* weights connecting the linear input neurons to the hidden layer

Now that the architecture of the multilayer network has been defined, the design process is described in the following sections.

Selection of the Number of Hidden Layers

There may not be any hidden layer needed if the inputs and outputs are linearly related. Linear and generalized linear models are useful in a wide variety of applications. If the function is mildly nonlinear and there is a low number of variables with a small dataset, generalization may be better with a simple linear model than with a complicated nonlinear model. If there are just one or two input variables, there seems to be no advantage to using more than one hidden layer.

In MLPs with a wide variety of nonlinear functions, one hidden layer with a large number of neurons would suffice for a good approximation. But there is no theory yet to decide how many hidden neurons are needed to approximate any given function.

Selection of the Number of Neurons in the Hidden Layer

In order to decide on the number of neurons in the hidden layer, the following should be carefully considered [41]:

- the numbers of input and output neurons

- the number of training samples
- the amount of noise in the targets
- the complexity of the function
- the architecture of the ANN
- the type of hidden unit activation function
- the training algorithm

In most cases, there is no specific rule to determine the number of neurons for the hidden layer other than the "trial-and-error" rule. Several ANNs need to be trained and analyzed before one can get the required kind of ANN. Checking the generalization error is one way to go about analyzing the trained network.

If the network has less than the required number of hidden neurons, the mean-squared error (MSE) also known as the training error will be high therefore, generalization error will also be high due to underfitting and high statistical bias. If the network uses more than the required number of neurons in the hidden layer, the generalization error would still be high due to overfitting, even though training MSE is low.

In order to decide on the kind of architecture for the ANN, there are many books that specify different kinds of rules. The most common one is that, the size of the hidden layer is to be somewhere between the input layer size and the output layer size. Some of the other rules are as follows:

- Number of hidden neurons = (Number of inputs + outputs) * (2/3) [41, 42].
- In an MLP with one hidden layer, there can never be more than twice the number of neurons in the hidden layer as there are in the input layer [42].
- The exact number of neurons in the hidden layer needs to capture 70-90% of the variance of the input data set [44].
- Number of hidden neurons with the lowest root-mean-squared error (RMSE) [44].

There are many counterexamples available today to disprove all of these rules. This is because, these rules ignore the number of training cases, the amount of noise in the targets,

and the complexity of the function. Some of the error criteria that take these factors into account are:

Schwarz's Bayesian criterion (SBC) [41, 46]:

$$SBC = (n)log\left(\frac{SSE}{n}\right) + (p)log(n) \tag{4}$$

Akaike's information criterion (AIC) [41, 47]:

$$AIC = (n)\log\left(\frac{SSE}{n}\right) + 2(p) \tag{5}$$

Corrected AIC (AICc) [41, 48]:

$$AICc = (n)\log\left(\frac{SSE}{n}\right) + \frac{(n+p)}{(1-(p+2)/n)} \tag{6}$$

where

- SSE is the sum of squared errors for the training set,
- n be the number of training samples,
- p be the total number of parameters (weights and biases)

These criteria can actually be used to choose the number of neurons for the hidden layer in an ANN. Also, the value (SSE/n) is the MSE which can be obtained from the ANN itself.

For each criterion, the model with the smallest value is chosen. Lower criterion value implies either fewer explanatory variables, better fit, or both.

When the model has a finite number of parameters and the usual regularity conditions hold, SBC is consistent for model selection, whereas AIC and AICc are not very consistent for model selection. However, when the true model has an infinite number of parameters, AIC and AICc are comparatively more efficient than SBC. But, various empirical studies have

shown that in some cases, AIC overfits, while SBC mostly produces reasonable results [32, 41].

Generalization Error

Methods for estimating generalization error are given as following:

- Single-sample statistics (AIC, SBC etc.) [49]: In linear models, statistical theory provides several simple estimators of the generalization error under various sampling assumptions. These estimators adjust the training error for the number of weights being estimated. These statistics can be used as generalization error in nonlinear models in case of a large training set.

- Split-sample or hold-out validation [50]: The most commonly used method for estimating generalization error in neural networks is to reserve part of the data as a test set. This test set needs to be a representative sample of the cases where the network needs to be tested. After training, the network is run on the test set, and the error on the test set provides an unbiased estimate of the generalization error, provided that the test set was chosen randomly.

- Cross-validation [50, 51]: In order to use all the data for training, split-sample validation is improved to form a new method known as the cross-validation. Here, the initial dataset is divided into k subsets of approximately equal size. The network is trained k times, each time leaving out one of the subsets from training, but using only the omitted subset to compute whatever error criterion is of interest. The disadvantage of cross-validation is that one has to retrain the net many times.

- Bootstrapping [50, 51]: Bootstrapping is an improvement on cross-validation that often provides better estimates of generalization error at the cost of even more computing time. In this method, instead of repeatedly analyzing subsets of the data, subsamples of the data are repeatedly analyzed. Each subsample should be a random sample with replacement from the full sample.

Training Algorithm

Neural Network Toolbox supports a variety of training algorithms, including several gradient descent methods, conjugate gradient methods, the Levenberg-Marquardt algorithm, and the resilient backpropagation algorithm. One can access training algorithms from the command line or via the graphics user interface.

The Levenberg-Marquardt backpropagation is a network training algorithm that updates weight and bias values according to Levenberg-Marquardt optimization. This algorithm appears to be the fastest method for training moderate-sized feedforward neural networks. Even though it requires more memory than other algorithms, it is usually the first choice in any supervised algorithm.

The Levenberg-Marquardt (LM) algorithm was designed to approach second-order training speed without having to compute the Hessian matrix [34]. When the performance function is the sum of squares, then the Hessian matrix is given as:

$$H = J^T J \tag{7}$$

and the gradient is computed as

$$g = J^T e \tag{8}$$

where J is the Jacobian matrix and e is a vector of network errors.

The Jacobian matrix contains the first derivatives of the network errors with respect to the weights and biases. The LM algorithm uses this approximation in the following Newton update:

$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e \tag{9}$$

When the μ is zero, it becomes Newton's method. Due to the impressive accuracy of Newton's method, μ is decreased after each successful step. Hence, the performance function is always reduced at each iteration of the algorithm.

In MATLAB, the LM algorithm has a built-in function which uses the mean-square error (MSE) as the performance function.

2.1.3.3 Training

During training, progress is constantly updated in the training window. There are some criteria that can be used to monitor as well as stop the network training, like gradient magnitude, validation checks, training time, performance value and number of training epochs (iterations.

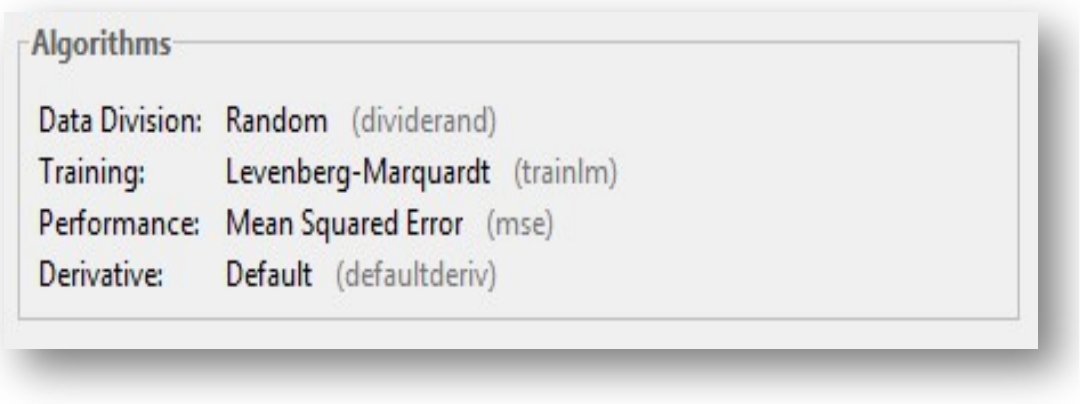


Figure 2.7 Example of Algorithms used in Neural Network in MATLAB

Out of all these choices, of most interest are the performance, the magnitude of the gradient of performance and the number of validation checks, as they are used to terminate the training. In a good neural network, these values decrease as the training comes to an end. Training is terminated when the average error gradient is less than or close to 1e-8, or when for 10 consecutive iterations, the relative improvement in the error function is less than or equal to 1e-12 [34]. See Fig. 2.7 for an example set of choices.

2.1.3.4 Analysis and Testing

Four basic plots obtained after the training window can determine the training effectiveness of the neural network: performance plot, training state plot, error histogram, and regression plot (see Fig. 2.8).

- Performance Plot: The value of the performance function versus the iteration number is shown here. It plots training, validation, and test performances.
- Training State Plot: It shows the progress of other training variables, such as the number of validation checks, the gradient magnitude, etc.
- Error Histogram Plot: The distribution of the network errors is shown in this plot.
- Regression Plot: It shows a regression between network outputs and network targets (Fig. 2.8).

When the training is complete, the network performance needs to be checked to determine if any changes need to be made to the training process, the network architecture, or the datasets.

The validation and test curves are found to be very similar. If the test curve had increased significantly before the validation curve increased, then it is possible that some overfitting might have occurred.

The network can be validated by creating a regression plot. If the training were perfect, the network outputs and the targets would be exactly equal, but the relationship is rarely perfect in practice. The R value is an indication of the relationship between the outputs and targets. If R = 1, this indicates that there is an exact linear relationship between outputs and targets and indicates a good fit.

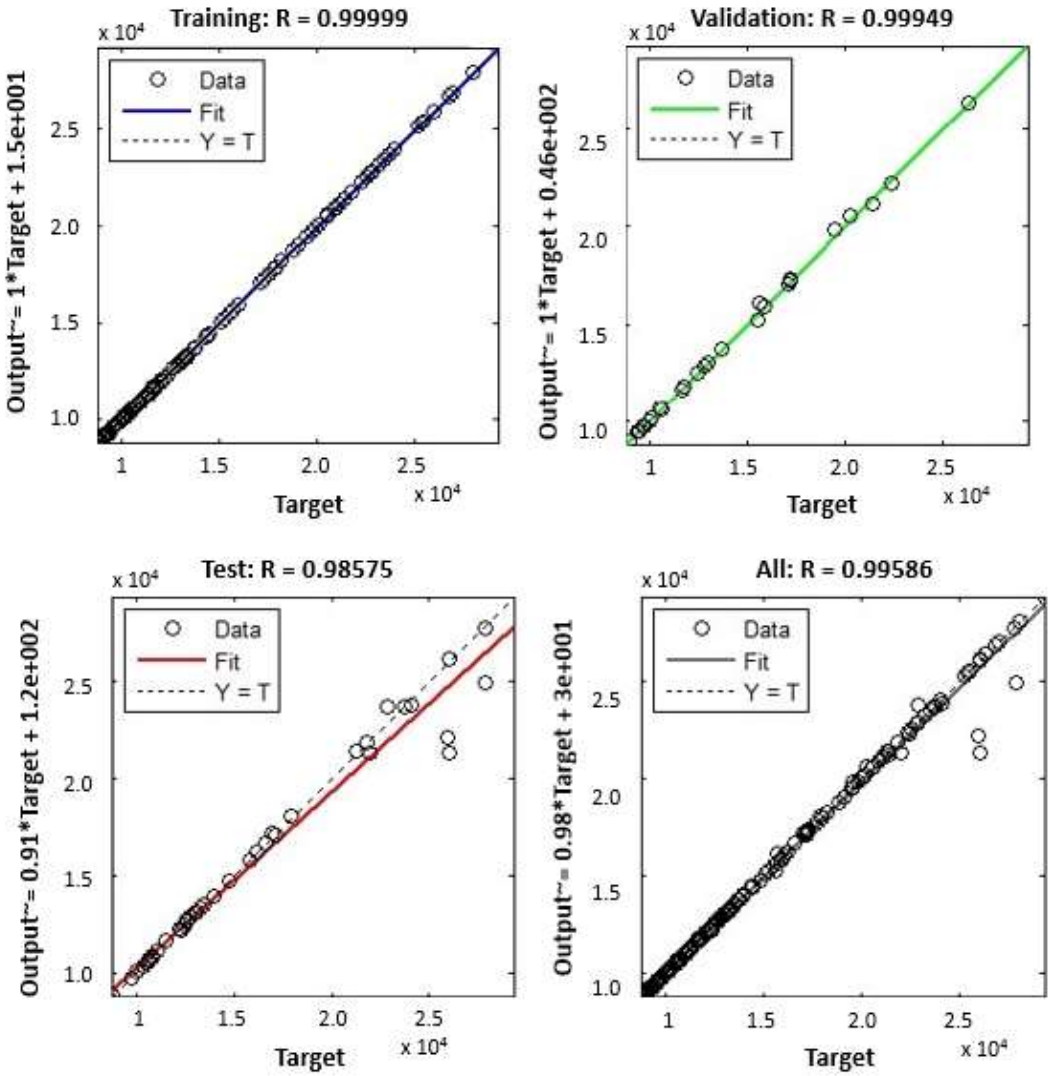


Figure 2.8 Regression plots for an example network

When a network is put to test, one may present the network with a sample it has not seen before. If the net has learned the underlying structure of the problem domain, then it should be able to give a value as the output which is close to the real answer and then the net is said to *generalize* well. Good generalization is, therefore, one of the key properties of neural networks.

Ways to Improve a Network

If the network does not seem to be sufficiently accurate, the following steps help to better understand the network as well as improve the accuracy of the network.

- Training the network several times may give a better idea about the performance of the network overall. Each time a feedforward network is initialized, the network parameters are different and might produce different solutions.
- Changing the number of hidden neurons. Making the hidden layer too large, may cause the problem of under-characterization. This means that the network needs to optimize more parameters than there are, leading to generation of noise in the network.
- Trying a different training function.
- Increasing the size of the training data set. Providing additional data for the network is more likely to produce a network that generalizes well to new data.

After the network is trained and validated, the trained net can be used to calculate the network response to any input.

One of the important characteristics to be kept in mind is that each time an ANN is trained, the same input can result in a different solution. This is due to different initial weight and bias values and different divisions of data into training, validation, and test sets, each time. As a result, different ANNs trained on the same problem can give different outputs for the same input. To ensure that an ANN is robust and is of good accuracy, it requires retraining the same net several times [34].

2.1.4 Limitations

Levenberg-Marquardt requires a lot of memory for training the network. There are other fast algorithms available too. Even though they are not as accurate, they will not use up that much memory.

Multilayer networks can approximate any reasonable function arbitrarily well. However, backpropagation and its variations might not always find a solution.

Neural networks are highly sensitive to the number of neurons in their hidden layers. A lesser number of neurons may lead to underfitting, whereas more neurons may contribute to overfitting. Improving the network's ability to generalize helps prevent overfitting. When a network memorizes the training set, but has not learned how to generalize to new inputs that is known as "overfitting". Such a network shows a relatively small error on the training set, but a much larger error when presented with the new data.

2.2 Genetic Algorithm Background

In the early 1960s, the very first genetic algorithms were developed. Researchers such as W.W. Bledsoe, G.J. Friedman, G.E.P. Box and H.J. Bremermann developed evolution-inspired algorithms for machine learning and function optimization [52]. Ingo Rechenberg introduced the evolutionary strategy in 1965. These techniques were programmed to model the aspects of natural evolution, where there was no population or crossover or mutation. The use of the idea of a population was introduced in later versions.

2.2.1 Introduction to Genetic Algorithm

Genetic algorithms (GAs) are a family of computational models inspired by evolution. These algorithms are made to encode a potential solution to a specific problem on a simple data structure and apply recombination to these structures to preserve critical information.

GA is a model of machine learning that is used as a search and optimization tool. It derives its behavior from a metaphor of the mechanisms of evolution in nature. GAs create a population of individuals and enable the fittest candidate to survive and reproduce based on random information search and exchange imitating the natural biological selection. The production of new individuals in the new generations depends on the parents, the offspring are created using parts and portions of the parents; the fittest candidate, preserving the best biological features and thus improving the search process [53].

2.2.2 Terminologies involved in Genetic Algorithm [54]

Fitness Functions: The *fitness function* is the function that is to be optimized, also known as objective function. The code for Global Optimization Toolbox in MATLAB tries to find the minimum of this fitness function.

Individuals: An *individual* is any point to which one can apply the fitness function. The value of the fitness function for an individual is its score. An individual is referred to as a *genome,* and the vector entries of that genome are known as *genes*.

Populations: A *population* is an array of individuals.

Diversity: *Diversity* is defined as the average distance between individuals in a population. Diversity in the population enables the GA to search a larger region of the space (see Fig. 2.9).

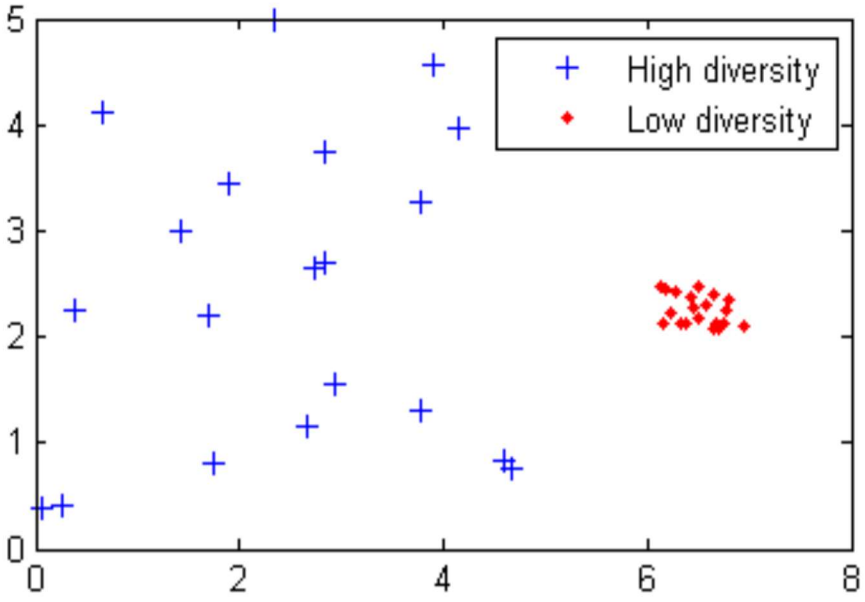


Figure 2.9 Diversity in population makes easier for GA to search a larger region

Generations: At each iteration, the GA performs a set of computations on the current population and produces a new population, which forms a new *generation*.

Fitness Values: The *fitness value* of an individual is the value of the fitness function for that individual. The *best* fitness value for a population is the smallest fitness value for any individual in the population for the minimization problem.

Operators: GA starts with a random set of points in the population and uses operators to produce the next generation of the population. The different operators are *scaling, selection, crossover,* and *mutation*.

Parents and Children: To create the next generation, the GA selects individuals, that have better fitness values in the current population, called *parents*, and uses them to create the next generation of individuals, called *children*.

2.2.3 GA in MATLAB

Global Optimization Toolbox is capable of searching for global solutions to problems that contain multiple maxima or minima. It includes genetic algorithm, global search, multi-start, simulated annealing solvers and pattern search. These solvers can be used to test optimization problems where the objective or constraint function is continuous, discontinuous, does not possess discontinuous derivatives, stochastic, or includes simulations or black-box functions with undefined values for some parameter settings [54].

Genetic Algorithm (GA) in MATLAB allows one to solve both constrained and unconstrained optimization problems based on a natural selection process. The algorithm repeatedly modifies a population of individual solutions. Over successive generations, the population "evolves" toward an optimal solution.

The GA toolbox can be used to solve problems that are difficult to solve for standard optimization algorithms, including problems in which the objective function is discontinuous, non-differentiable, stochastic, or highly nonlinear.

The standard genetic algorithm proceeds as follows [56]:

- An initial population of individuals is generated at random or heuristically.
- At every evolutionary step, the individuals in the current population are decoded and evaluated according to some predefined quality criterion, referred to as the fitness, or fitness function.
- To create a new population, a few individuals are selected according to their fitness and they are the parent solutions.
- The expected number of times an individual is chosen is approximately proportional to its relative performance in the population. Therefore, best fit individuals have a better chance of reproducing, while low-fitness ones are more likely to disappear.

An efficient optimization algorithm uses two techniques to find a global minimum: exploration and exploitation. Exploration is used to investigate new areas in the search space, and exploitation makes use of the knowledge from the points found in the past up to that instant, to find better points. These two requirements are contradictory, and a good search algorithm needs to find a tradeoff between the two [52].

2.2.4 Mechanism of Genetic Algorithm

Selection on its own, cannot introduce any new individuals into the population, i.e., it cannot find new points in the search space. Genetically inspired operators, crossover, and mutation create the best-fit solutions (see Fig. 2.10) [30].

Elite children are the individuals in the current generation with the best fitness values, which automatically pass to the next generation.

Crossover is performed between two selected individuals, called parents, by exchanging parts of their genomes to form new individuals, called offspring. Substrings are exchanged after a randomly selected crossover point. This operator tends to enable the evolutionary process to move toward more promising regions of the search space.

The mutation operator is introduced to prevent premature convergence to local optima by randomly sampling new points in the search space. It is carried out by flipping bits at random, with some probability.

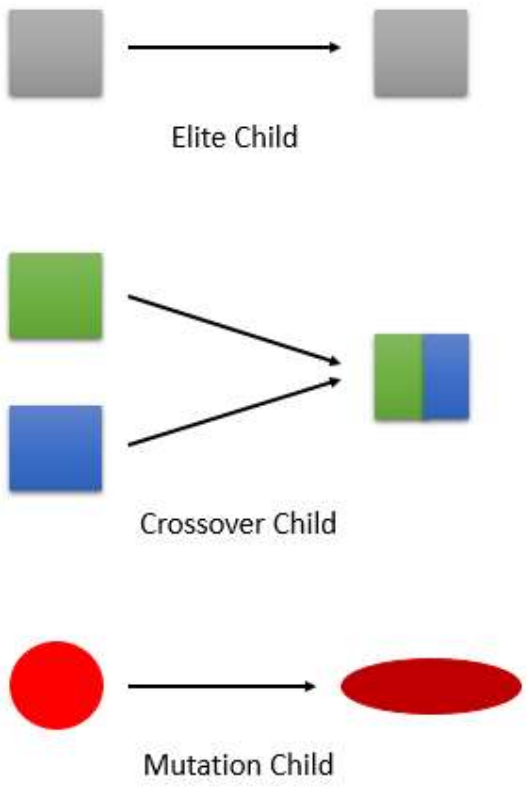


Figure 2.10 Selection Operators in Genetic Algorithm

2.2.5 Benefits of Genetic Algorithm

Some of the advantages of GA are [54]:

- GAs are capable of parallel processing.
- A vast solution set can be scanned by a GA at a very fast rate. GAs are well suited for complex, discontinuous, noisy fitness functions.

- Since GAs efficiently search the global space, they are capable of converging to the global minimum effectively.
- GAs maintain robustness through evolution. This demonstrates the performance of the algorithm under various circumstances.
- A GA also does not impose restrictions on the domain of the search or optimization problem.

2.2.6 Genetic Algorithm with Neural Network

Genetic Algorithms and Neural Networks in a broad sense dwell in the class of evolutionary computing algorithms, which are used in everyday problems like forecasting the stock market, turnovers, or the identification of credit bonus classes for banks [53]. Both GAs and NNs have gained more importance in modern days. In spite of their apparent

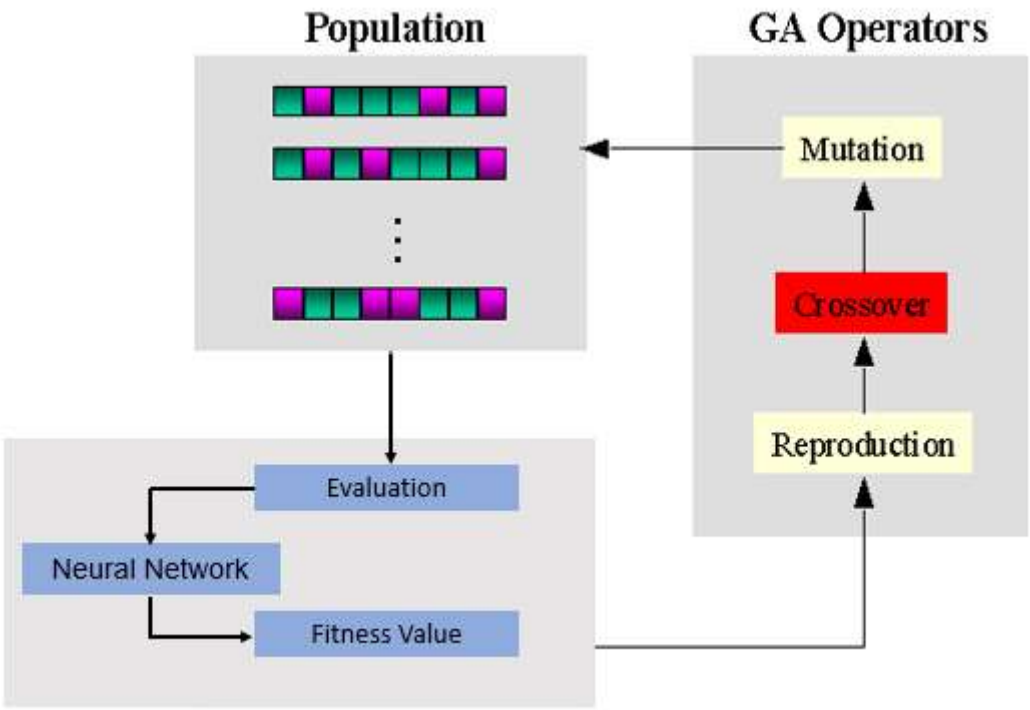


Figure 2.11 GA Mechanism

design differences, they also share several features in common which are sufficiently interesting for the innovation-oriented applications.

Both GAs and NNs are adaptive. They can learn and can deal with highly nonlinear models and noisy data. They do not require smooth functions. For both, the modeling and coding ought to be executed carefully, since they are complex. These algorithms work best for real world applications. GAs can optimize parameters of a neural net and GAs can also be used to fine tune the NN parameters [53].

In this project, GA is being used to optimize control surface deflection to minimize the sum of absolute of hinge moments of all the control surfaces (HMS). GA provides the optimum combination of control surface deflections and extracts its expected HMS from the trained ANN.

Chapter 3

Static Aeroelastic Analysis

Structural integrity is necessary for any aircraft. Every aircraft company pays special attention towards achieving such requirements and is called “Structural Technology group”. This group consists of structural engineers, metallurgists and aerodynamicists. A structural technology group is responsible for overall load prediction, aircraft component strength analysis and structural stability prediction. Stability derivatives are invaluable for characterizing the performance and handling of an aircraft and hinge moment coefficients are a special type of stability derivative that require special handling and display.

The structural technology group plays an important role in the early design process as well as in the post-certification process. Initial design process is very important for an aircraft design project to succeed, since any major flaws could result in aircraft production delay and greatly increase the cost of the overall design process. During the early 20^{th} century, trial-and-error methods were used in the designing airplane parts. In the recent times, however, computer simulations and testing is introduced right from the beginning of a design process. Almost all previous knowledge in designing an aircraft is applied. Static aeroelastic analysis is one such method included in the initial design process. It not only provides necessary parameters for a structural integrity of an airframe, but also provides

the control power requirements to handle such a design. This can save a lot of time and money by ensuring that a design is meeting the power requirements set forth by an entire airplane development project.

In the current work, MSC Nastran's structural finite elements were used to build the structural model. Geometric, structural, inertial and damping data were provided to MSC Nastran, and the structural stiffness, mass and damping matrices were generated for the aeroelastic analyses. Aircraft stability derivatives are then obtained from the loads and deflections.

3.1 Static Aeroelasticity

The mutual interaction between inertial forces, aerodynamic forces and elastic forces, and the influence of this interaction on an airplane design is studied under "Aeroelasticity" [57]. Airplane structures in modern times are very flexible, which introduces a variety of aeroelastic phenomenon. An aeroelastic phenomenon occurs when structural deformation induces aerodynamic forces which in-turn produces structural loads. This interaction either gets smaller and smaller converging into stable equilibrium or diverges into an unstable situation destroying the airplane structure. As a result, aeroelastic analysis plays a crucial role in an airplane design process.

When mutual interaction between inertial forces, aerodynamic forces and elastic forces is included in the analysis, it is referred to as dynamic aeroelastic analysis. However, when mutual interaction of only aerodynamic and elastic forces is taken into consideration, it is referred to as static aeroelastic analysis. The term "static" refers to time independent analysis [58].

Independently, structural analysis problems dealing with external forces are statistically determinate as loads (external and internal) are independent of the deformation. Similarly, in aerodynamic analysis, deformation is independent of the loads. However, when mutual interaction between aerodynamic and elastic forces are considered, loads and structural deformation are inter-dependent making the aeroelastic problem indeterminate. In other

words, both loads and deflections are solved simultaneously, making these problems computationally expensive.

3.1.1 Trimmed Flight

A BWB or any other aircraft configuration is considered trimmed (at the nominal cruise condition) when the aerodynamic center of pressure is coincident with the center of gravity, and all of the trailing-edge control surfaces are faired. Positive static stability requires that the nose-down pitching moment be minimized [11]. Stability is a property of an equilibrium state. To discuss stability, the meaning of equilibrium has to be defined first. If an aircraft is to remain in steady, uniform flight, the resultant forces and the resultant moments about the center of gravity must be equalized to zero. An aircraft satisfying this requirement is in a state of equilibrium. The state of equilibrium is called trimmed flight in flight dynamics. The property of an aircraft to preserve the trimmed flight conditions e.g. its altitude or to resist displacements and if displaced, to develop forces and moments tending to restore the original conditions is called stability [59]. The stability of an aircraft is categorized as static and dynamic.

The longitudinal static stability refers to the aircraft's stability in the pitching plane. If an aircraft is longitudinal by stable, a small increase in the angle of attack will cause a change of the pitching moment, which decreases the angle of attack. On the other hand, a small decrease in the angle of attack will cause in a change of the pitching moment, which increases the angle of attack. This content can be explained with the help of following equations: [58].

$$C_L = -C_z \tag{10}$$

For level flight,

$$L = W \tag{11}$$

$$\frac{W}{q_\infty S_{ref}} = -C_z \tag{12}$$

$$C_z = C_{z_0} + C_{z_\alpha}\alpha + C_{z_{\delta_e}}\delta_e + C_{z_q}\frac{q\bar{c}}{2V} + C_{z_{\dot{\alpha}}}\frac{\dot{\alpha}\bar{c}}{2V} + C_{z_{\dot{z}}}\frac{\ddot{z}}{g} + C_{z_{\ddot{\theta}}}\frac{\ddot{\theta}\bar{c}}{2g} \tag{13}$$

$$C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_{\delta_e}}\delta_e \tag{14}$$

Where, lift is denoted by *L, W* is for weight, $C_{z_0}, C_{z_\alpha}, C_{z_{\delta_e}}, C_{z_q}, C_{z_{\dot{z}}}, C_{z_{\ddot{\theta}}}, C_{m_0}, C_{m_\alpha}, C_{m_{\delta_e}}$ are stability derivatives, α is angle of attack, q is dynamic pressure, δ_e is the elevator deflection, $\ddot{\theta}/g$ are the inertial derivatives and *c* is for chord.

These equations are derived for tailless aircraft, for trimmed equilibrium, quasi-steady flight, and constant mass, for small angles of attack. If the aircraft is put in a state of equilibrium, e.g. by adjusting control inputs, then the aircraft is flying in a trimmed condition. The trim conditions are given as Eq. 17 and Eq. 18:

$$W_{total} = W_{aircraft} + W_{fuel} + W_{payload} \tag{16}$$

$$\sum \mathrm{L} = L_{wing} + L_{fuselage} = nW_{total} \tag{17}$$

and

$$\sum \mathrm{M} = 0 \tag{18}$$

Where *n* = load case

3.1.2 Hinge Moments

In conventional aircraft, the hinged part of the vertical stabilizer is called the rudder, which is used to deflect the tail to the left and right as viewed from the front of the fuselage; the hinged part of the horizontal stabilizer is called the elevator which is used to deflect the tail up and down; and the outboard hinged part of the wing is called the aileron which rolls the

wings from side to side. The hinge moment is the moment acting about the hinge line of a control surface that must be overcome to move the control surface when a pilot exerts a force command on the control stick.

At any given dynamic pressure and Mach number, the hinge moment varies with angle of attack, control surface deflections, and trim tab deflection. The effects that contribute to the hinge moment are difficult to analytically predict, but they are necessary to properly design the aircraft control system [60]. Hinge moment governs the magnitude of augmented pilot force required to move the corresponding actuator to deflect the control surface. To minimize the size and thus the cost and power requirements of the actuation system, the control surface deflections need be optimized, so that the control forces are as low as possible [2, 10].

The study of hinge moments is important to be able to predict the forces (moments) required by the human pilot or the actuator system. The feedback of this required force or moment is an additional cue to help the pilot fly the aircraft efficiently [19].

Hinge moments are not measured on conceptual aircraft models, as they cannot be discerned from the total forces measured by the sting balance during a wind tunnel experiment. Panel code analysis, CFD, and existing wind tunnel and flight test data are used during the conceptual design process to conservatively approximate hinge moments. These estimation techniques often lead to overestimation of required control surface actuator size, increasing the weight and cost of the aircraft [19]. The desire to not only improve hinge moment estimates but also optimize it early in the design process motivated the current research to create this optimization schedule.

For the pitching maneuver, the studies primarily focus on the elevator hinge moment, but the procedure is the same for any other control surface. If the hinge moment is represented by H_e , then the hinge moment coefficient is defined as:

$$C_{H_e} = \frac{H_e}{\frac{1}{2}\rho V^2 S_e c_e} \tag{19}$$

Where, S_e = the elevator area behind the hinge line

c_e = the mean aerodynamic chord of the elevator of the same area

Generally, it is assumed that the hinge moment depends linearly on the elevator deflection. Within small to moderate angles, the linear relationship of the hinge moment as a function of control surface deflection and angle of attack may be described as follows:

$$C_{H_e} = C_{H_o} + \frac{\partial C_{H_e}}{\partial \alpha}\alpha + \frac{\partial C_{H_e}}{\partial \delta_e}\delta_e \tag{20}$$

3.2 Static Aeroelasticity in MSC Nastran

The development of NASTRAN (NASA Structural Analysis) began in 1965 and the MacNeal-Schwendler Corporation (MSC) was one of the principal developers of the software. Since 1965, MSC Nastran has continued to evolve and extend its capabilities. Today, MSC Nastran can perform stress, load, aerodynamic and control system analysis and design for an aircraft using a common finite element representation [58]:

- Determination of applied loads, internal stresses and stability derivative values, while considering aeroelastic deformations of the structure
- Hinge moments calculated for all control surfaces
- Restrained stability derivatives and unrestrained derivatives are both provided
- Results for rigid aircraft as wells as elastic case are available.
- Control surfaces can be easily modeled and analyzed for their performance characteristics and for their aeroelastic effectiveness as a function of Mach number and dynamic pressure.

3.2.1 MSC Nastran Solution 144 (SOL 144)

As described above, static aeroelastic problems deal with the mutual interaction of aerodynamic and structural forces on a flexible vehicle that results in a redistribution of the aerodynamic loading as a function of airspeed. Static aeroelastic analysis for this work is

intended to obtain both structural and aerodynamic data. The structural data of interest include loads, deflections, and stresses. The aerodynamic data include stability and control derivatives, trim conditions and pressures and forces. In MSC Nastran, a solution sequence to address these needs is called Solution 144 (or SOL 144). Solution 144 computes aircraft trim conditions with subsequent recovery of structural responses, aeroelastic stability derivatives and static aeroelastic divergence dynamic pressures [58].

MSC Nastran's structural finite elements were used to build the structural model. Geometric, structural, inertial and damping data are provided to MSC Nastran, and the structural stiffness, mass and damping matrices are generated for the aeroelastic analyses. Matrices of aerodynamic influence coefficients are computed from the data describing the geometry of the aerodynamic panels. The location of the structural grid points does not decide the choice of aerodynamic grid points for the aerodynamic model. One subsonic and three supersonic lifting surface aerodynamic theories are available in the software. The subsonic theory is the Doublet Lattice Method (DLM) whereas Mach Box method, Piston theory and ZONA51 method for multiple interfering lifting surfaces are available for supersonic conditions. Since the current BWB configuration was analyzed for the subsonic regime, the DLM method was used for aerodynamic analysis. The analysis at subsonic conditions uses Vortex-Lattice aerodynamic theory which is a steady case of the Doublet-Lattice method.

Static equilibrium equations are solved using the finite element method to obtain the structural load distribution on an aircraft in trimmed conditions. Splining techniques are used to generate a transformation matrix from structural grid point deflections to aerodynamic grid point deflections. This provides the aerodynamic stability derivatives like, moment and lift coefficients and curve slopes due to control surface rotation, and trim variables like angle of attack and control surface setting, as well as aerodynamic and structural loads, structural deflections and element stresses.

The user supplies finite element models for the definition of the structure and aerodynamic loading, including information on the flight condition. Stability and control derivatives are printed for each unique flight condition (Mach number and dynamic pressure). Then, a trim

analysis is performed that determines unknown trim values and performs standard data recovery for each trim subcase. Correction factors are then applied to compute values for adjustments for the effects of camber and twist, experimental pressures etc. These factors are specified using the Direct Matrix Input (DMI) section.

Aerodynamic elements are represented by strips, boxes, or segments of bodies that are combined to idealize the vehicle for computation of aerodynamic forces. The elements are similar to structural elements and are defined by their geometry and associated degrees of freedom. For the Vortex-Lattice Method, MSC Nastran uses trapezoidal boxes with their edges parallel to the free-stream velocity. MSC Nastran automatically generates these aerodynamic elements and grid points to ensure that many of the theoretical requirements are met.

3.2.2 Aerodynamic Analysis in MSC Nastran

As mentioned previously, six different aerodynamic theories are implemented in MSC Nastran. However, this work focuses on Vortex-Lattice subsonic lifting surface theory.

The basic relationship between the lifting pressure and the dimensionless vertical or normal velocity induced by the inclination of the surface to the airstream (in other words, downwash or normal-wash) is given by,

$$\{w_j\} = [A_{ij}]\{f_j/q\} \tag{21}$$

Where,

w_j = downwash

$A_{ij}(m,k)$ = aerodynamic influence coefficient matrix as a function of Mach number (m) and reduced frequency (k)

f_j = pressure on lifting element j

q = flight dynamic pressure

The substantial differential matrix of the deflections to obtain downwash is given by,

$$\{w_j\} = \left[D_{jk}^1 + ikD_{jk}^2\right]\{u_k\} + \{w_j^g\} \tag{22}$$

Here,

w_j^g = static aerodynamic downwash which includes the static incidence distribution that may arise from an initial angle of attack, camber, or twist.

k = reduced frequency, $k = \omega b/V$ where ω is the angular frequency, b is a reference length and V is the free-stream velocity.

u_k = displacement at aerodynamic grid points

D_{jk}^1 , D_{jk}^2 = real and imaginary parts of substantial differential matrix

Forces and Moments are obtained by integrating pressure using,

$$\{P_k\} = \left[S_{kj}\right]\{f_j\} \tag{23}$$

Where S_{kj} = integration matrix.

The above three equations can be combined to give a resulting aerodynamic influence co-efficient matrix:

$$[Q_{kk}] = \left[S_{kj}\right]\left[A_{ij}\right]^{-1}\left[D_{jk}^1 + ikD_{jk}^2\right] \tag{24}$$

The aerodynamic method computes the S, D^1 and D^2 matrices using user-supplied Mach Numbers and reduced frequencies. The DLM computes the A matrix and then uses matrix decomposition and forward and backward substitution in the computation of Q. Now for static aeroelastic analysis, reduced frequencies are 0 as analysis is time independent. Equation (24) then becomes,

$$[Q_{kk}] = \left[S_{kj}\right]\left[A_{ij}\right]^{-1}\left[D_{jk}\right] \tag{25}$$

Vortex Lattice Method (VLM) is used to obtain S and D in Eq. (24). VLM is steady state case for Doublet Lattice method [63]. Vortex lattice methods were first formulated in the late '30s, and the method was first called "Vortex Lattice" in 1943 by Faulkner. A lifting surface in VLM is modeled as an infinitely thin sheet of discrete vortices. The flow field is then simulated around the body and pressure and force distribution is obtained around the modeled surface. Lift and induced drag are then obtained from the modeled behavior of the control surface. Aerodynamic coefficients and their derivatives also can be computed from the obtained flow field. VLM method extends Prandtl lifting line theory [64] where the control surface in an aircraft is modelled by a number of horseshoe vortices.

VLM is built upon Potential flow theory and based on solutions to Laplace's equation. In most of the engineering applications, approximation to a real fluid as an ideal fluid works quite well. VLM like potential flow theory neglects viscous effects. As a result, boundary layers are not resolved, and viscous drag cannot be obtained. The inviscid assumption also results in loss of turbulence and dissipation. However, lift induced drag can be obtained, and some stall phenomena could be captured. Apart from the inviscid assumption, flow is also assumed to be incompressible and irrotational. Angle of attack is assumed to be small and control surfaces are assumed to be infinitely thin. In 2D, the flow field can be then written as,

$$\boldsymbol{V} = \boldsymbol{V}_{\infty} + q(x, y) \tag{26}$$

Where, V is the velocity vector flow field, V_{∞} is the freestream velocity vector field and $q(x, y)$ is the disturbance velocity (with u and v components) due to the modeled surface.

The disturbance velocity vector is defined as,

$$\boldsymbol{q}(x, y) = \boldsymbol{\nabla \varphi}(x, y) \tag{27}$$

Where, $\varphi(x, y)$ is the perturbation velocity potential, which satisfies Laplace's equation.

The lifting surface on an airplane is divided into a lattice of quadrilateral panels and a horseshoe vortex with strength Γ on each panel is placed. The bound vortex of the horseshoe vortex is placed on the ¼-chord element line of each panel. A control point is placed on the ¾-chord point of each panel. Trailing vortices extend to infinity. Thus, perturbation velocity is obtained by summing the contributions from all the horseshoe vortices,

$$\boldsymbol{q_i} = \boldsymbol{\nabla \varphi_i} = \sum_j \boldsymbol{A_{ij}} \Gamma_j \tag{28}$$

The strengths of the vortices on each of the panels are obtained by applying a Neumann Boundary conditions at the control point of the panel. According to the Boundary condition,

$$\boldsymbol{V_i} \cdot \boldsymbol{n_i} = \left(\boldsymbol{V_\infty} + \sum_j \boldsymbol{A_{ij}} \Gamma_j\right) \cdot \boldsymbol{n_i} = \boldsymbol{0} \tag{29}$$

The above equation is solved for unknown vortex strength Γ_i for each panel. The total force vector and moment vector on the control surface can then be obtained using,

$$\boldsymbol{F_i} = \boldsymbol{\rho} \Gamma_i (\boldsymbol{V_i}) \times \boldsymbol{l_i} \tag{30}$$

$$\boldsymbol{F} = \sum_i \boldsymbol{F_i} \tag{31}$$

$$\boldsymbol{M} = \sum_i \boldsymbol{F_i} \times \boldsymbol{r_i} \tag{32}$$

Where, $\boldsymbol{l_i}$ is the transverse segment of the horseshoe vortex and $\boldsymbol{r_i}$ is the midpoint of i^{th} panel. Lift and drag can be then obtained from the force field $\boldsymbol{F}$.

Interpolation is used to connect aerodynamic and structural grids. This results in independence between the selection of aerodynamic and structural grid points. In MSC Nastran, a structural model could involve one-, two- or three-dimensional array of grid points. An aerodynamic model could use surface elements for lifting surface theory or line elements for strip theory. MSC Nastran provides a general interpolation method to

interconnect various combinations. Any aerodynamic panel or body can be further subdivided into sub regions that use a separate function for each. For this work, 3-D structural model was used with surface elements of lifting surfaces for aerodynamic theory.

MSC Nastran provides three types of interpolation utility: Linear splines, Surface splines and user-defined explicit interpolation. A model may use several splines including a combination of the three types. Linear splines are recommended for high-aspect ratio wings or bodies, whereas surface splines are recommended for low aspect ratio wings and bodies where structural grid points are distributed over an area. For a BWB configuration, the vertical tail uses linear splines, whereas everywhere else surface splines are used.

In MSC Nastran, structural degrees of freedom are chosen as independent degrees of freedom, whereas aerodynamic degrees of freedom are dependent. An interpolation matrix relates the dependent degrees of freedom to the independent ones. Two such transformations are required in aeroelastic analysis: the interpolation from structural deflections to the aerodynamic deflections and the relationship between the aerodynamic forces and the structurally equivalent forces acting on the structural grid points. The following relation transforms structural grid point deflections $\{u_k\}$ to the aerodynamic grid point deflections $\{u_g\}$ using interpolation matrix $[G_{kg}]$.

$$\{u_k\} = [G_{kg}]\{u_g\} \tag{33}$$

The transformation between aerodynamic forces $\{F_k\}$ and structurally equivalent forces $\{F_g\}$ acting on the structural grid points is obtained from the principal of virtual work.

$$\{F_g\} = [G_{kg}]^T\{F_k\} \tag{34}$$

Interpolation matrix $[G_{kg}]$ depends on the type splines used. Detailed discussion about each type of splines are given in Ref. [58].

The downwash relation of Eq. (34), for static aero-elasticity becomes,

$$\{w_j\} = [D_{jk}]\{u_k\} + [D_{jx}]\{u_x\} + \{w_j^g\} \quad (35)$$

Where:

$\{w_j\}$ = vector of aerodynamic degrees of freedom (e.g. angles of attack)

$\{u_k\}$ = vector of aerodynamic deformations (displacements)

$\{u_x\}$ = vector of "extra aerodynamic points" used to describe aerodynamic control surface deflections and overall rigid body motions

$\{w_j^g\}$ = represents an initial static aerodynamic downwash. It primarily includes the static incidence distribution that may arise form initial angle of attack, camber or washout.

$[D_{jk}]$ = substantial derivative matrix for the aerodynamic displacements

$[D_{jx}]$ = substantial derivative matrix for the extra aerodynamic points.

The aerodynamic forces (from Eq. 35) can be transferred to the structural forces using the spline matrix and can be reduced to a-set to form the aerodynamic influence coefficient matrix Q_{aa} as,

$$[Q_{aa}] = [G_{ka}]^T [W_{kk}][S_{kj}][A_{jj}]^{-1}[D_{jk}][G_{ka}] \quad (36)$$

Second aerodynamic influence coefficient matrix Q_{ax} for forces at the structural grid points due to unit deflections of the aerodynamic extra points can be written as:

$$[Q_{ax}] = [G_{ka}]^T [W_{kk}][S_{kj}][A_{jj}]^{-1}[D_{jx}] \quad (37)$$

The complete equations of motion in the a-set degrees of freedom can be written as:

$$[K_{aa} - qQ_{aa}]\{u_a\} + [M_{aa}]\{\ddot{U}_a\} = q[Q_{ax}]\{u_x\} + \{P_a\} \quad (38)$$

Where,

K_{aa} = Structural stiffness matrix

M_{aa} = Structural mass matrix

P_a = Vector of applied loads (Including mechanical, thermal, and gravity loads plus aerodynamic terms due to user input pressures and/or downwash velocities)

Equation (38) is the basic set of equations for static aeroelastic analysis. Rigid body motions are also included to represent the free-flying characteristics of an air vehicle. To include rigid body motions, Eq. (38) is partitioned into r-set (supported) and l-set (left over) degrees of freedom. Ref. [58] provides detailed information on how these equations are solved for the aerodynamic stability derivatives (lift and moment coefficients due to control surface deflections), trim variables (angle of attack and control surface setting), aerodynamic and structural loads, structural deflections, and element stresses.

Lift is calculated in the MSC Nastran according to Eq. (13), where,

$\frac{q\bar{c}}{2V}$ = Pitch

$\frac{\dot{\alpha}\bar{c}}{2V}$ = 0 (Quasi-Steady Analysis)

$\frac{\ddot{z}}{g}$ = Load Factor

$\frac{\ddot{\theta}\bar{c}}{2g}$ = Pitching Acceleration

And, all the C_z terms are the stability derivatives, calculated by MSC Nastran and printed in the output file.

Chapter 4

Proposed Methodology for Optimum Control Power

As mentioned before, the HWB aircraft has 13 elevons and 2 rudders. Due to these large number of variables, the optimization methodology for appropriate allocation of deflections for all the control surfaces to achieve minimum hinge moments forms the following hypothesis.

4.1 Hypotheses

Hypothesis 1

A dataset is generated using MSC Nastran, which includes the control surface deflections and the HMS values (Fig. 4.1). Then using this dataset, an ANN is trained, which accurately represents the static aeroelastic model behavior.

Hypothesis 2

This trained ANN can then be used as a surrogate model in the optimization process using GA. The optimum control allocation will be provided by GA, whereas the HMS value for that particular control allocation will be provided by the trained ANN (Fig. 4.1).

Hypothesis 3

The HMS and the control allocation obtained from the optimization process involving the ANN and GA is indeed the optimum. Also, MSC Nastran is the only way available to validate the optimum HMS (Fig. 4.1).

4.1.1 Dataset Generation using MSC Nastran

The first step towards optimization is the generation of the initial dataset. For this step, a BDF file with the characteristics of a half span model of X-48B aircraft was provided by NASA Langley. In MSC Nastran, a half span model is enough, to perform symmetric maneuvers. This implies that, there are only 2 inboards, 4 outboards, 1 rudder and a half of elevator in the model. From the review of MSC Nastran, in order to trim the aircraft, we need to leave two of the control surfaces free. In a pitching maneuver, elevator is most effectively used to trim the aircraft. Therefore, for this case, the elevator along with the angle of attack (AOA) were left free for MSC Nastran's trim analysis.

The steps for generating the dataset using MSC Nastran are as follows:

- Provide the BDF file with the deflections for all the control surfaces, except for the elevator and angle of attack. It needs to be kept in mind that the control deflections should not have values which are difficult or impossible to attain in real-life scenarios. Therefore, the control surfaces were restrained to deflections between -30 and +30 degrees.
- Save those control surface deflections in a separate file, as these are the design variables.
- MSC Nastran performs trim analysis using SOL 144 for those specific deflections.
- SOL 144 provides its output in a F06 file along with all the other data.

- Hinge moments are not direct outputs of MSC Nastran. Therefore, the values for hinge moments are extracted manually from the F06 file.
- The sum of absolute of the hinge moments for all the control surfaces gives us the objective function for the optimization.

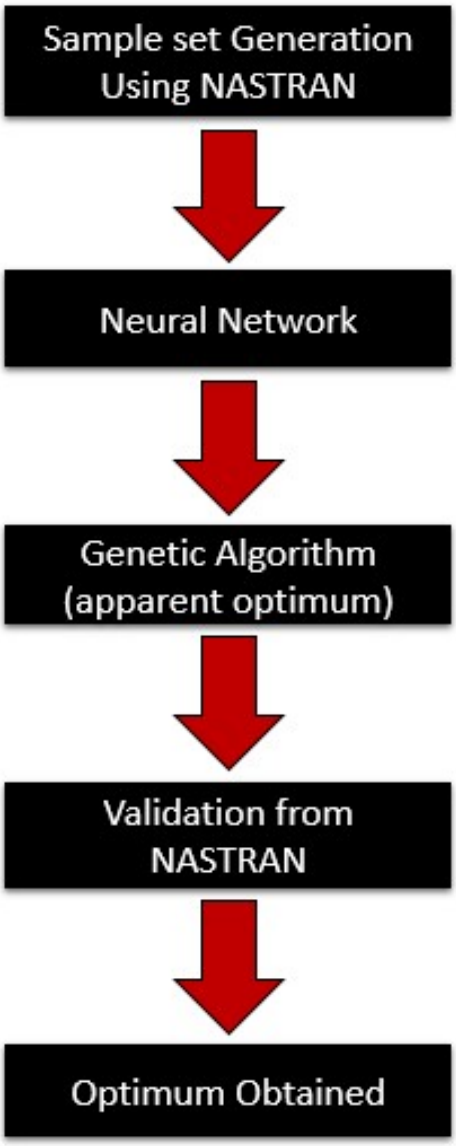


Figure 4.1 Initial Optimization Schedule

The problem with this procedure is that there can be over a million combinations of deflections for these 7 control surfaces using just 3 points to get a relation between the

deflections and HMS values. Therefore, a uniformly spaced dataset is created using *Latin Hypercube Sampling* (LHS) [68] over AELINK cards. AELINK [70] defines relationships among aeroelastic static trim variables (which specify rigid body motions to be used as trim variables in static aeroelasticity), and aerodynamic control surface deflections such that:

$$u^D + \sum_{i=1}^{n} Ci\, u_i^I = 0.0 \tag{38}$$

Where:

u^D = dependent variable
u_i^I = independent variable

LHS method is very simple and does not involve any tricky simulations. Moreover, it has a smaller variance with respect to Primitive Monte Carlo (PMC) sampling. These AELINK cards are generated using *lhsdesign* [34] in MATLAB, which creates randomly spaced combinations with values between *0* and *1*. These values are and then mapped to values between *-1.2* and *1.2*. The control surface deflections for 6 dependent elevons and a rudder would be directly related to the elevator deflection and indirectly dependent on the AOA.

Therefore, instead of providing the BDF file with the deflections, the AELINK cards were provided to MSC.Nastran along with the BDF file of the model. A MATLAB script then read the trim variables, stability and control derivatives, and hinge moments for each individual control surface, storing the AELINK Cards, control surface deflections and HMSs for post-processing and optimization. A dataset of *3000* samples was generated for the *2.5g* maneuver case.

4.1.1.1 Data Pruning

Could a training sample be detrimental to learning? Contrary to the common belief that more training data yield better generalization, the quality of the sample also matters and that the neural network might be better off when some training samples are discarded. A

general approach called data pruning is used to identify the samples that are troublesome for learning. Data Pruning is a selective selection approach that is used to identify and eliminate samples that do not fit in our aimed sample space [70].

The relationship between the control surface deflections and HMS is linear but with a lot of noise. A slight change in the deflection of elevator can yield a remarkable change in HMS. This is not true for any other control surface. This implies that, the sensitivity of the elevator deflection brings noise to the dataset. This noise makes it less probable for the ANN to learn the relationship between the input values and the target values with minimum number of samples. During dataset generation step using MSC.Nastran, no constraints over the HMS can be applied. Therefore, in order to avoid generating even more but unnecessary samples, one can simply target a small sample space with adequate number of samples.

Minimization of HMS is the primary objective of this study. So, the ANN needs to be of good accuracy, particularly for lesser values of HMS. A neural network is as accurate and responsive as the dataset used for its training. Hence, instead of using the whole dataset for training of ANN, only those samples were used where the HMS values were lower than the mean of HMSs for the complete dataset. Out of *3000* samples, the pruning gave *1782* samples which had HMS below the mean HMS. These samples were used for the next step.

4.1.2 Response Surface using an Artificial Neural Network

Artificial intelligence is applied to the HWB control allocation problem for optimizing the control surface schedules and minimize control power. In order to optimize for control power, Genetic algorithm is used to search entire global search space. The objective function can be obtained from MSC Nastran, using the population for each generation in GA. However, MSC Nastran is computationally expensive and thus would become bottleneck in the optimization process. Since NN can provide objective function in fast turn-around time, trained neural network is used to model objective function for GA.

In order to train the neural network, both the AELINK coefficients and direct values of the individual control surface deflection angles are used. The ANN for AELINK coefficients will be called or ANN-AELINK and the ANN using the control surface deflections will be called ANN-Deflections. In order to choose the number of neurons for both the cases, trial and error method as well as SBC error calculation was used.

4.1.3 Optimization using Genetic Algorithm

The response surface generated by the neural network is to find the optimum arrangement using GA. In this problem, GA makes small random changes in the individuals in the population to create children using mutation. Mutation promotes GA to search a broader space by providing genetic diversity. Usage of GA to enhance the network results and the quality of the neural network training progress is indicated by the fitness value. GA starts with the minimum HMS samples obtained in the initial dataset and with each generation of the GA, the fitness value is reduced until it is flattened out after about *60* generations, indicating that the neural network is fully trained at this point. The resulting optimum indicated is then validated against an MSC Nastran computation.

4.1.4 Validation

There should be a way to tell that the HMS obtained from the ANN is correct. This means, that one needs to check the accuracy of the ANN and if it was able to generalize well. This is why; the HMS obtained from the ANN, for the optimum combination from GA, needs to be validated.

For the ANN-AELINK, the optimum combination of AELINK cards obtained from the GA is fed back into MSC Nastran. Moreover, for the ANN-Deflections, the combination obtained from the GA is used to calculate the AELINK cards for that combination and then this AELINK is fed back into MSC Nastran along with the BDF file of the model to perform trim analysis once again.

In both the cases, the hinge moments are extracted from the F06 file (created by MSC Nastran) and the HMS is then calculated and compared to the HMS obtained from the ANN-AELINK and ANN-Deflections.

4.2 Improvements

Extrapolation is neither easy to perform and nor is very accurate at times. When a data set is generated using MSC Nastran, there is a distinct possibility that the minimum, which is sought, might not be contained in the dataset. This leads to extrapolation of finding a value from the GA, which is even lower than the minimum value of the sum of absolute value of hinge moment in the dataset. The question arises when the global minimum hinge moment value is less than the minimum hinge moment present in the data set, then how best can the network be made accurate outside the bounds of the dataset, since the neural network isn't trained for that range.

Our approach is to increase the bounds of the data set gradually, as we move forward in the process. This way, the neural network does not need to go out of its bounds every time GA generates an optimum deflection combination. This process will also help in increasing the number of samples for training of the network, which may even improve the network further.

Therefore, the initial optimization schedule (Fig. 4.1) was modified to the one shown in Fig. 4.2, where there are some extra steps. The minimum we obtain from the genetic algorithm is just a local minimum that is highly dependent on the nature of the initial population, number of generations and the type of modifications GA makes to the initial population. This local minimum is then validated using MSC Nastran and then those specific control surface deflections and their respective sum of absolute of hinge moments are added into the original data set. So now the new data set has $(n+1)$ samples, where n is number of samples in the original data set. Then using this new data set, the neural network is trained again. This training will improve the bounds of the network, so that it can be more accurate in calculating the hinge moment for the given control surface deflections.

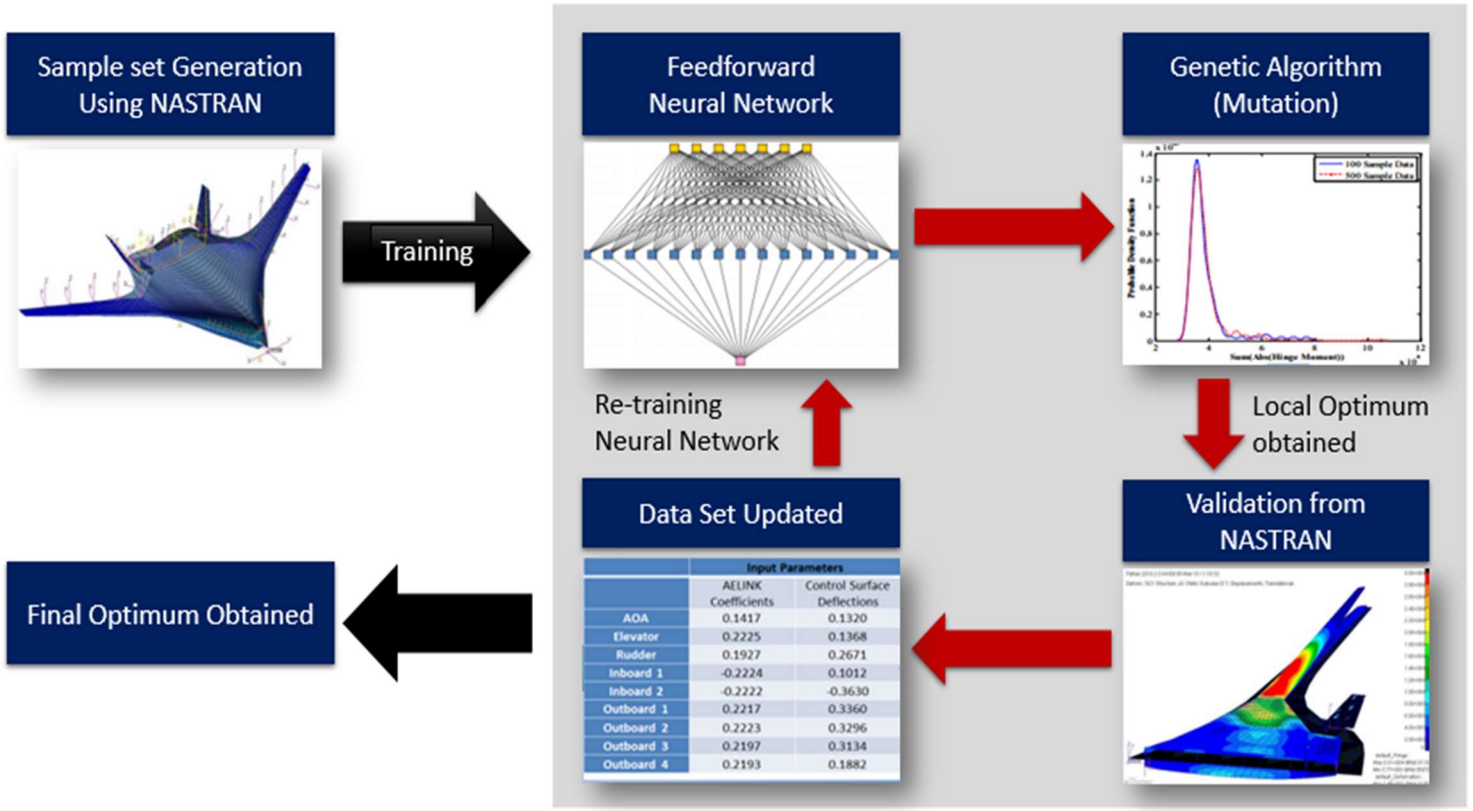


Figure 4.2 New Improved Optimization Schedule. This optimization schedule includes Data Set Updating and Retraining of Neural Network

GA is then re-run using the solution from the last iteration as the new initial values of the variables.

Figure 4.2 shows a complete iterative loop of optimization process. The user controls the number of iterations of this loop. Also, the optimization process greatly depends on this number of iterations, described subsequently.

The optimization schedule known as “Adapt and Train” was adopted. This style has two components, exploration and exploitation. Exploration is a time consuming process, but it maintains the quality of the optimization process by taking care of all the aspects, i.e. it maintains the diversity in the values of hinge moments by working globally. Exploitation deals with static efficiency, i.e. refining the existing procedures by doing the same things in a better way and reaping value from what is already known. One can make it work in the areas, which are possibly near the required answer (if the answer is known), so the outcome is certain.

This training style was adopted in a way that both exploration and exploitation were balanced in their usage. Balancing the training was accomplished by combining the generation of a broad data set, training of neural network, optimization by genetic algorithm, validation using MSC Nastran and feeding back that local optimum into the initial data set for subsequent analysis.

Chapter 5

Results

5.1 Control Optimization of Half-Span X-48B HWB Aircraft Model

The initial studies were conducted with a half-span finite element model (Fig. 5.1) of the X-48B HWB OREIO aircraft. Since, the focus of this research is on a 2.5g pitch maneuver, this configuration was deemed a good starting point. This model was provided to us by NASA Langley. The half-span model had 1 rudder, 2 inboards, 4 outboards and half of the elevator (see Fig. 5.1). But because of the huge size of the elevator, half of the control surface can also be considered as a whole with respect to that specific model. Thus, there were 8 control surfaces. The step-by-step process of analysis is shown in Fig 4.2.

5.1.1 Dataset Generation.

As mentioned in section 4.1.1, the initial data set was generated using AELINK coefficients in MSC Nastran with SOL 144. These AELINK coefficients were generated using Latin Hypercube sampling in MATLAB and then mapped between -1.2 and 1.2. Three files were created as a result of this step, one for AELINK coefficients, another one for the respective

control surface deflections and the third one for the HMS for the respective set of deflections.

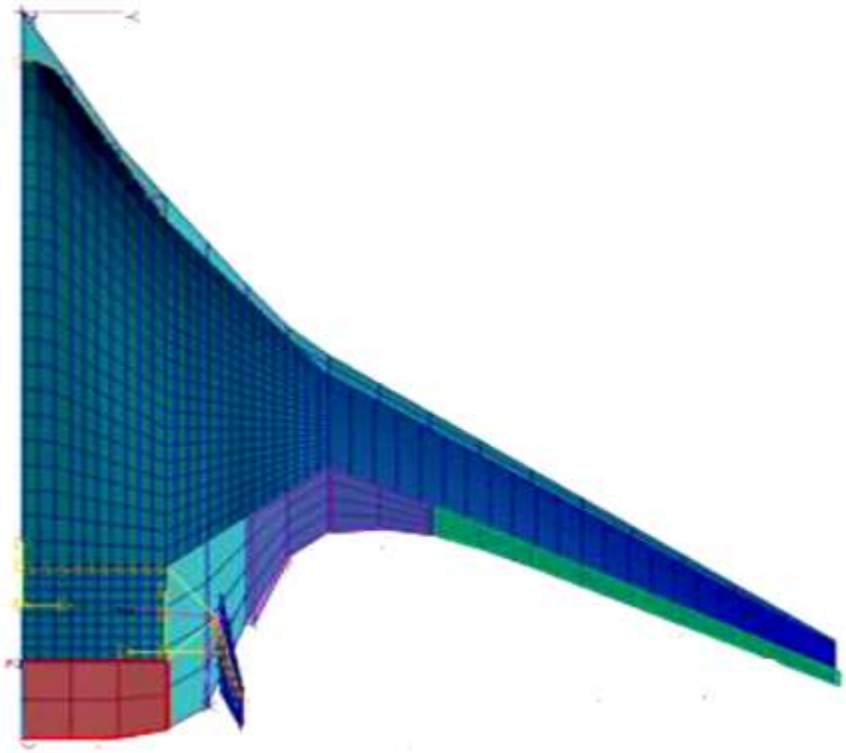


Figure 5.1 Half-span FEM model of X-48B HWB OREIO model

5.1.2 Setting up the Neural Network

A neural network with neurons between the ranges *50-500* were tried for SBC Error calculation. Figure 5.2 and 5.3 show the SBC error calculation results for ANN-AELINK and ANN-Deflections, respectively.

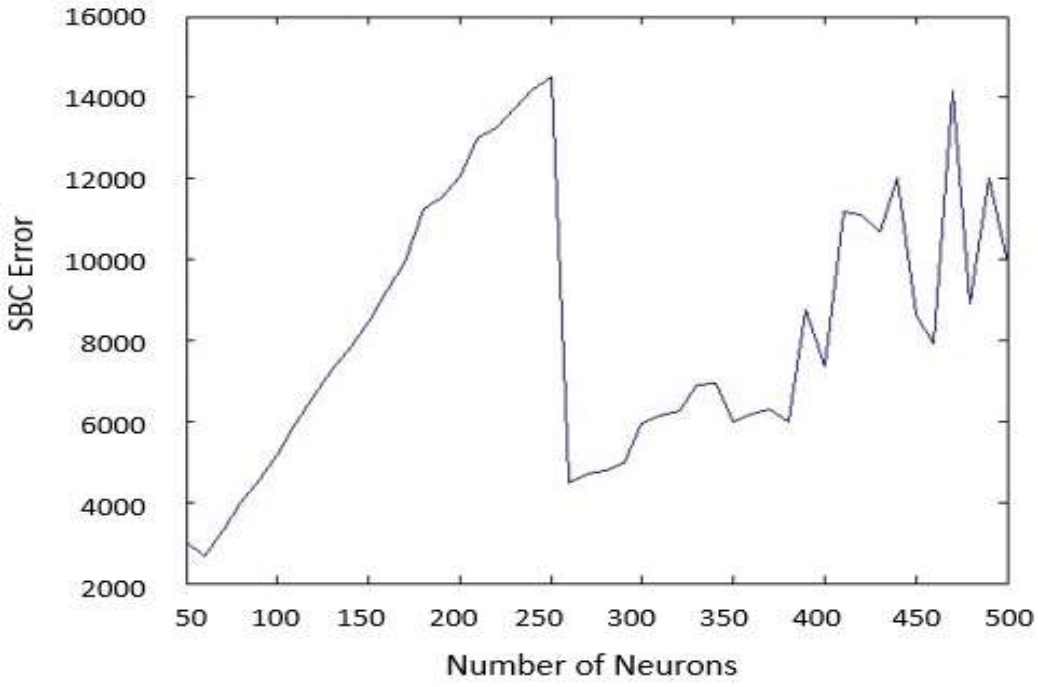


Figure 5.2 SBC Error plot for ANN-AELINK for Half-span model

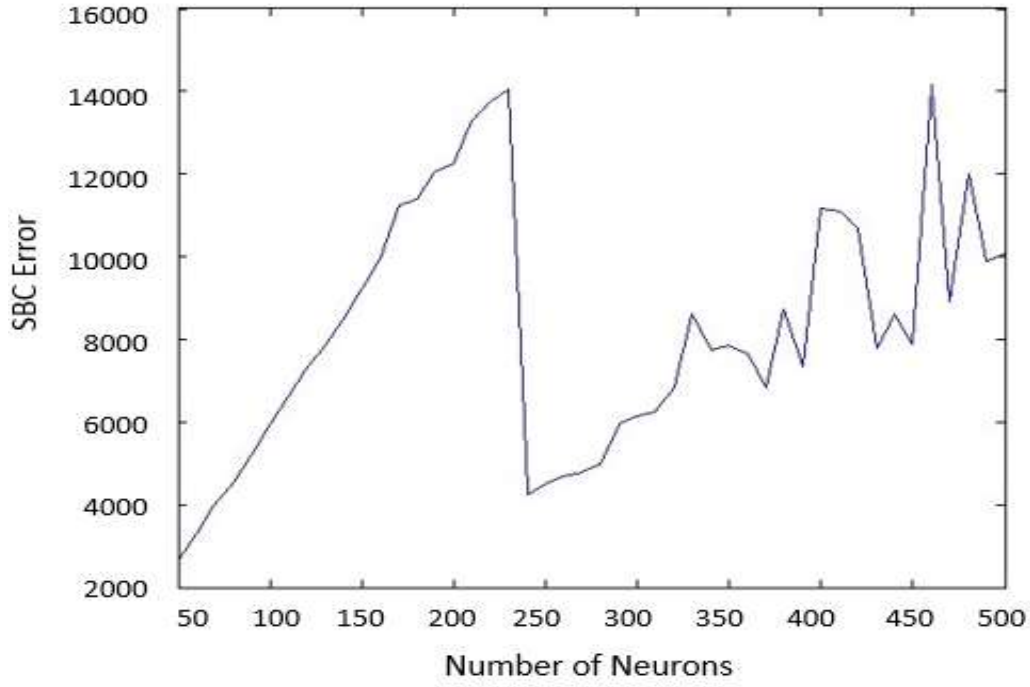


Figure 5.3 SBC Error plot for ANN-Deflections for Half-span model

On looking at Fig 5.2 and 5.3, when *50* to *100* neurons were chosen for these ANNs, a clear case of underfitting was seen. Therefore, a range of *250-400* neurons for the hidden layer, was selected for the trial-and-error approach for ANN-AELINK and a range of *250-350* neurons was chosen for ANN-Deflections.

ANN-AELINK had an input layer with 7 neurons and an output layer with 1 neuron. After the trial-and-error process was complete, *350* neurons seemed appropriate for the hidden layer for this network. 90% of the samples were used for training and 10% of the samples for testing the network. Log-sigmoid *(logsig)* was used as the transfer function for the hidden layer, while the other two layers used a linear *(purelin)* transfer function.

ANN-Deflections had an input layer with 8 neurons (7 dependent elevons and 1 elevator), an output layer with 1 neuron and a hidden layer with *300* neurons to perform the actuation power optimization. The performance errors (MSE) for training of both the ANNs were *4.5e-20* and *3.4e-20,* respectively. The regression plots for both the networks are shown in Figures 5.4 and 5.5. As explained in section 2.1.3.4, the network can be validated by creating a regression plot. If the training were perfect, the network outputs and the targets would be exactly equal, but the relationship is rarely perfect in practice. The R value is an indication of the relationship between the outputs and targets. If R = 1, this indicates that there is an exact linear relationship between outputs and targets and indicates a good fit.

5.1.3 Setting up GA and Validation from MSC Nastran

The setting for initial population and generations affects the behavior of the algorithm the most. Therefore, an initial population between 20 and 30, and a total of 16 generations, a mutation type genetic algorithm shows the kind of behavior desired for this problem.

As a validity check, both control surface deflection data sets were fed back into the Nastran SOL 144 aeroelastic trim solution to check whether Nastran would reproduce the results when using the predictions from the optimizer. Table 5.1 shows that the exact Nastran FEM analysis for the actuation power proxy matched the neural network prediction almost identically, proving the robustness of the neural network and genetic algorithm.

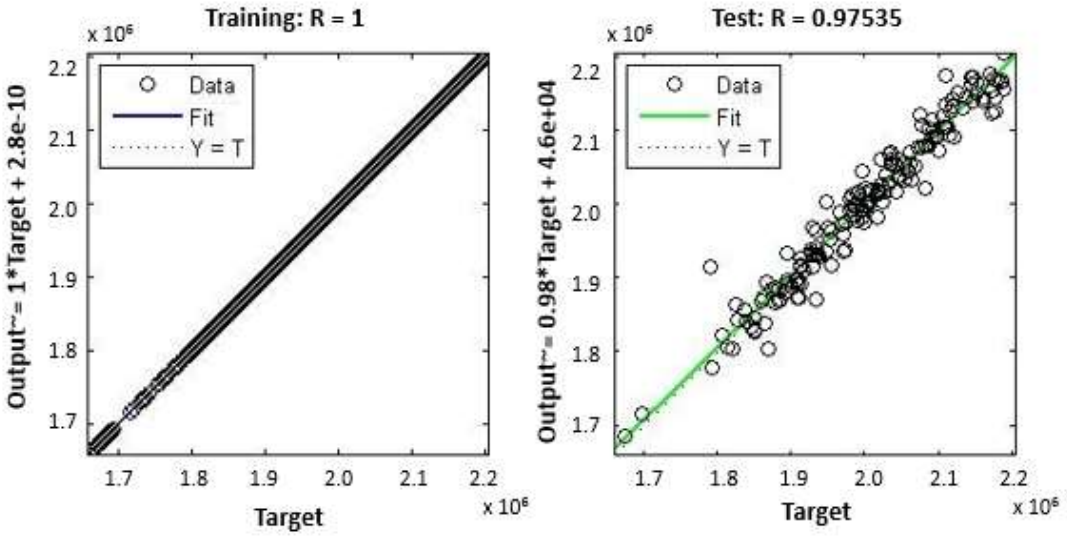


Figure 5.4 Regression plots for ANN-AELINK

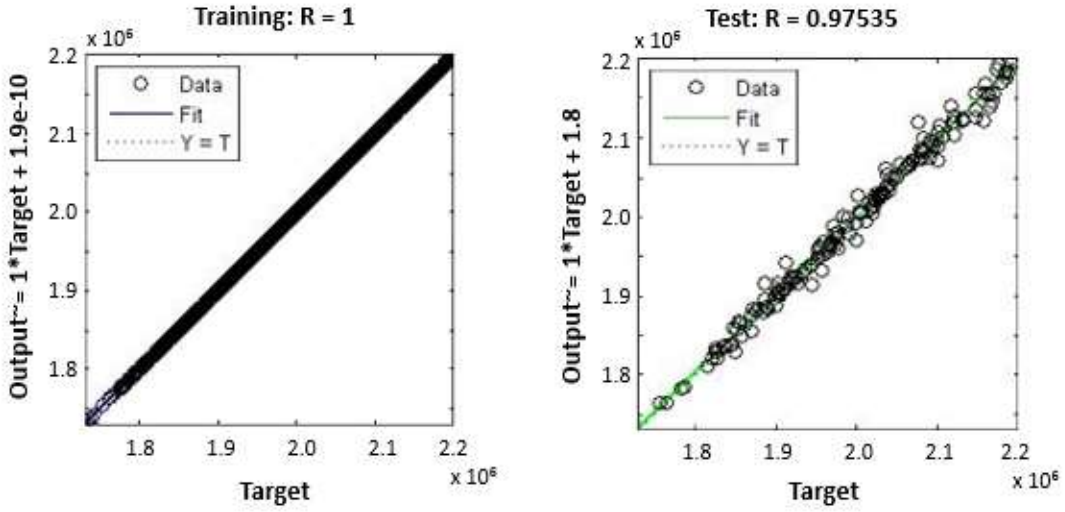


Figure 5.5 Regression plots for ANN-Deflections

5.1.4 Control Optimization, 2.5g Load Case : Results

The accomplishment of this optimization procedure as related to the semi-span model can be summarized into Table 5.1, showing the results generated by neural network over the minimum sum of absolute of hinge moment found in MSC Nastran generated trim data set. It also shows the comparison between the conventional way of using only the elevator to trim the aircraft and using all the control surfaces for the load case of 2.5g.

Table 5.1. 2.5g Maneuver optimization results for the HMS as a proxy for actuation power for the half-span HWB model

Sum of Absolute of Hinge Moment, lb.in.		
	AELINK	Control Deflections
Minimum from the Aeroelastic Trim Dataset	1.73e+06	1.7309e+06
When only Elevator is used in case of 2.5G	2.01e+06	2.0104e+06
Optimized Hinge Moment (NN + GA)	**1.66e+06**	**1.54e+06**
NASTRAN Validation	1.66e+06	1.54e+06
%ERROR (Between Optimized value and NASTRAN)	0.1242%	5.727e-14%
Improvement over best NASTRAN Case in the Initial Dataset	**4%**	**11%**
Improvement over using only Elevator	**17%**	**23%**

In addition, Table 5.1 shows the comparison between the usage of AELINK coefficients and control surface deflections as inputs to the neural network. In the case of AELINK coefficients being the input, only 7 coefficients could be included since, the elevator is an independent control surface. Due to this reason, the optimization is not as good as when the actual control surface deflections are used as the inputs to the neural network, where all 8 deflections are used in the optimization process.

It can be seen, not only that the optimization schedule improved the sum of absolute of hinge moments, but also when the control surface deflections were given as the input to the network, the results improved by 11% compared to the best in the MSC Nastran generated trim dataset. From Table 5.1, it can also be seen that the conventional scheme of using only elevator to trim the aircraft for 2.5g load case is not an effective way to trim the aircraft in terms of hinge moment. If all the control surfaces are deflected to obtain an optimum value of HMS, this HMS value is 23% lower than the one needed for employing only elevator to trim the aircraft.

It makes sense to ask, why it is better to use the other control surfaces to trim the aircraft in such flight condition. It seems, that the deflection of the other control surfaces helps the elevator to trim the aircraft better, without deflecting much (see Table 5.2). Also, since the surface area of the elevator is much greater than the other control surfaces, even a small deflection of the elevator generates a large hinge moment. Therefore, reducing the work of the elevator actually reduces the HMS. And, this reduction in the work of elevator is performed by deflecting other control surfaces.

Now, the number of iterations in this optimization schedule also plays a very important role. The results in Tables 5.1 and 5.2 were generated with *50* iterations. The optimization process in case of *50* iterations itself takes about *500* minutes. The optimization iterations could be increased by compromising the computational time. .

Table 5.2. 2.5g Maneuver optimization results for the control surface deflections for minimum HMS for the half-span HWB model

Control Surface Deflections, deg.		
	When used AELINK	When used Control Deflections
AOA	8.11	7.55
Elevon	12.75	7.83
Inboard 1	-12.74	5.79
Inboard 2	-12.73	-20.75
Outboard 1	12.70	19.26
Outboard 2	12.73	18.87
Outboard 3	12.58	17.93
Outboard 4	12.56	10.80
Rudder	11.04	15.27

Table 5.3 shows the results when the number of iterations was increased to *500*. At the expense of the increased computational time, the improvement over the best NASTRAN case has gone up from 11% to 30%, and the improvement over the case of using only the elevator has gone up to 40% from the earlier 23% (see Table 5.4). These results look encouraging, also indicating that if we further increase the number of iterations for optimization, we might have an even better result.

But the increase in computational time from apporximately 500 minutes to 5000 minutes, is a huge drawback to this process. Therefore, in order to form and validate this concept of using artificial intelligence for reducing the actuation power in an efficient manner, *50* iterations seem adequate, as it will save a lot of computational time while still showing greatly improved HMS results.

The control surface deflections in Tables 5.2 and 5.4 show that by decreasing the elevator deflection as well as Inboard1 deflection and keeping Inboard2, rudder and outboards well above 10 degrees gives a lesser hinge moment than any other scenario.

Table 5.3. 2.5g maneuver optimization results for the half-span HWB model with 500 iterations

Sum of Absolute of Hinge Moment, lb.in.	
Minimum from the Aeroelastic Trim Dataset	1.73e+06
When only Elevator is used in case of 2.5G	2.01e+06
Optimized Hinge Moment (NN + GA)	**1.20e+06**
NASTRAN Validation	1.20e+06
%ERROR (Between Optimized value and NASTRAN)	2.3e-12%
Improvement over best NASTRAN Case in the initial Dataset	**30%**
Improvement over using only Elevator	**40%**

Table 5.4. 2.5g maneuver control surfaces for minimum HMS for the half-span HWB model with 500 iterations

Control Surface Deflections, deg.	
AOA	8.03
Elevon	2.29
Inboard 1	-10.98
Inboard 2	-11.45
Outboard 1	30.00
Outboard 2	29.08
Outboard 3	30.02
Outboard 4	30.03
Rudder	25.47

5.1.5 Stability of the Optimum Result

For a trimmed flight, if the flight conditions suddenly change and we still want the aircraft to fly at the optimum HMS for that new flight condition, then it would be preferred if a little change in the allocation of deflections could give a close optimum HMS for that flight condition. For this purpose, it is important to understand, if the optimum HMS is stable about the original design point

The optimization performed in section 5.1.5 is for 2.5g load case. The stability of the optimum HMS was tested by varying the load case. The load case was sequentially reduced by a factor of *0.1* until it reached 1.5g.

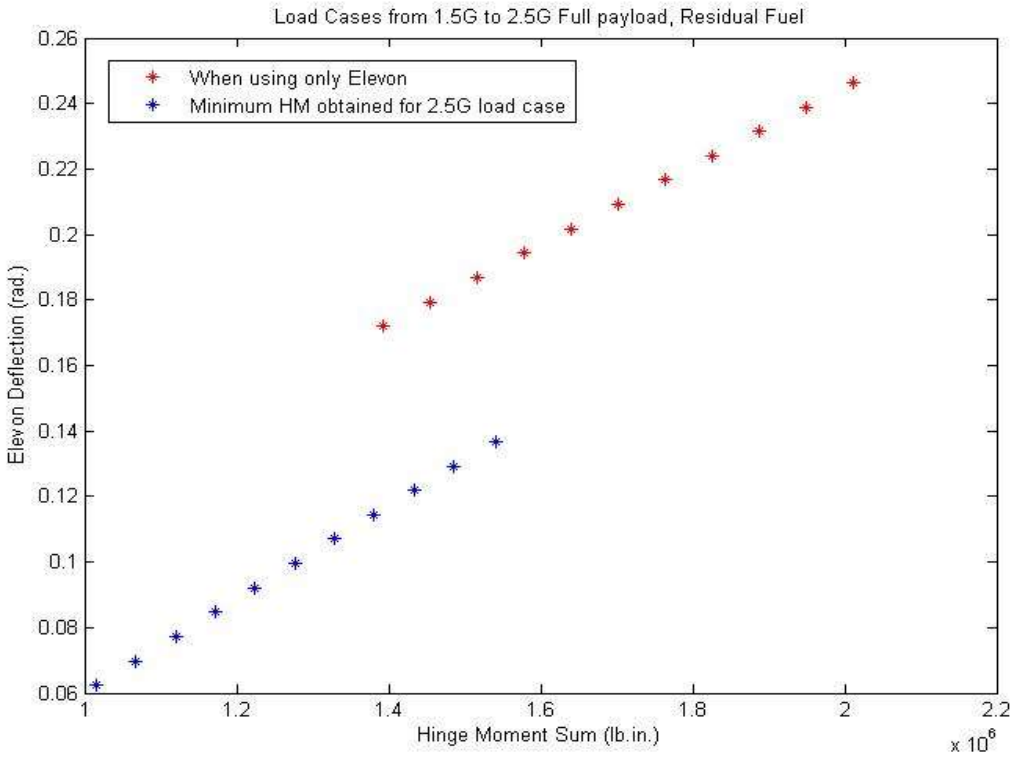


Figure 5.6 Half-Span Result Stability plot for 2.5g. The trend shows the decrease in HMS as the load factor decreases starting with the optimum control surface schedule for 2.5g

Two different cases were tested for stability: one, where all the control surfaces were fixed at 0-degree deflection and only the free elevator was deflected to trim the aircraft and second, where all the control surfaces were fixed to the deflections obtained in Table 5.2. However, in second case, in order to trim the aircraft, at least one control surface needs to be free along with the AOA; therefore, the elevator was left free.

It can be seen from Fig. 5.6, that for the same alignment case of the control surface deflections, HMS decreases as the load factor decreases. It is also noteworthy, that when all the control surfaces are fixed at some deflections, the elevator deflection also decreases as the load factor decreases. The same can be expected for full-span model.

The points shown in *blue* in the stability plot (Fig. 5.6) also need to be validated in order to understand how close those points are to the real optimum values of HMS for their respective load cases. Therefore, the same optimization schedule (Fig. 4.2) was performed for 2.0g and 1.5g load cases.

The results for 2.0g and 1.7g load cases, presented in Table 5.5, show that in terms of HMS, the points in Fig. 5.6 are very close to the optimum HMS obtained using the optimization schedule (Fig. 4.2). This implies that the optimization on one load case may give an overall picture of how the optimum HMS would behave if the load case is changed.

On the other hand, if the elevator deflections are to be observed for the close optimum HMS obtained in Fig. 5.6 compared to the optimum HMS obtained from the optimization schedule in Table 5.5, there seems to be some differences. This is shown in Fig. 5.7.

Although, the optimum obtained from the optimization schedule is lower than the one deduced from Fig. 5.6, the elevator deflection seems to have increased when compared to the 2.0g load case point in Fig. 5.6. The explanation here is that, when the optimization is performed on the model with the 2.0g load factor, all the other control surfaces would also be aligned differently, contributing to the increase of the elevator deflection.

Table 5.5. 2.0g and 1.7g Maneuver optimization results for the HMS as a proxy for actuation power for the half-span HWB model with 50 iterations

Sum of Absolute of Hinge Moment, lb.in.		
Load Case	**2.0G**	**1.7G**
Minimum from the Aeroelastic Trim Dataset	1.43e+06	1.29e+06
When only Elevator is used in case of 2.5G	1.72e+06	1.53e+06
Optimized Hinge Moment (NN + GA)	**1.21e+06**	**1.16e+06**
NASTRAN Validation	1.21e+06	1.16e+06
%ERROR (Between Optimized value and NASTRAN)	0.03%	0.08%
Improvement over best NASTRAN Case in the Initial Dataset	**16%**	**10%**
Improvement over using only Elevator	**29%**	**24%**

Whereas, as mentioned in the beginning of section 5.1.6, in order to check for the stability of the optimum result for 2.5g load case (Fig. 5.6), all the control surfaces were fixed to the deflections obtained for the optimum HMS obtained for 2.5g load case, except for elevator. This in turn, led to the other control surfaces doing more work to trim the aircraft, instead of the elevator, when compared to the deflections of the optimum HMS obtained from the optimization. Even though the other control surfaces deflected more, the additional hinge moment generated by all of them was almost equal to the additional hinge moment by elevator alone in case of optimization, which ultimately, gave the same HMS values in both cases.

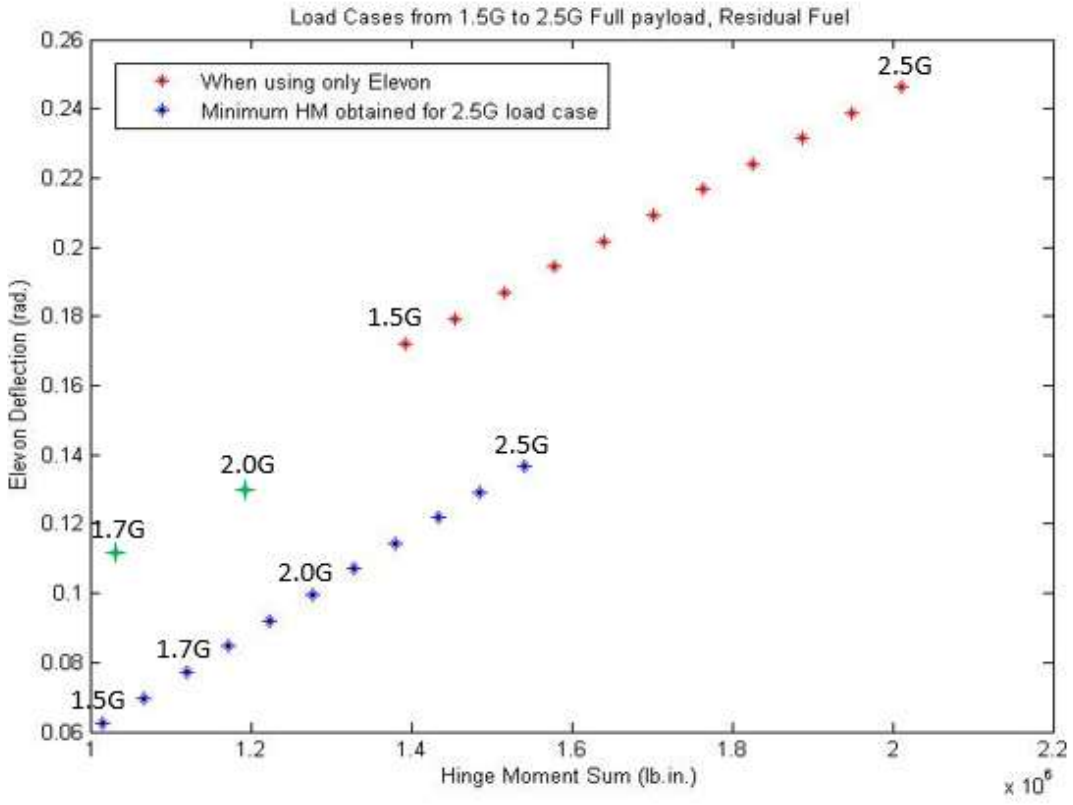


Figure 5.7 Stability of the Optimum Result obtained at 2.5g

5.2 Control Optimization of Full-span X-48B HWB Aircraft Model

The full-span aircraft model is necessary to perform full aeroservoelastic analysis which includes analysis of both symmetric and asymmetric flight maneuvers. With only a semi-span aircraft model the analysis is restricted to symmetric flight conditions.

Therefore, the next step was to generate the full-span model (Fig. 5.8) from the semi-span model. In order to do that, the process termed *mirroring* was adopted. A C++ code was developed by Nathan Love, another member of the Virginia Tech team. Using that code, structural and aerodynamic grid points and elements, element material properties, loads and boundary conditions were mirrored.

The symmetry of the full-span model was checked using NASTRAN. For this problem, only hinge moments and control surface deflections are being used for optimization, therefore the check for the symmetry of the model was also performed using the same variables.

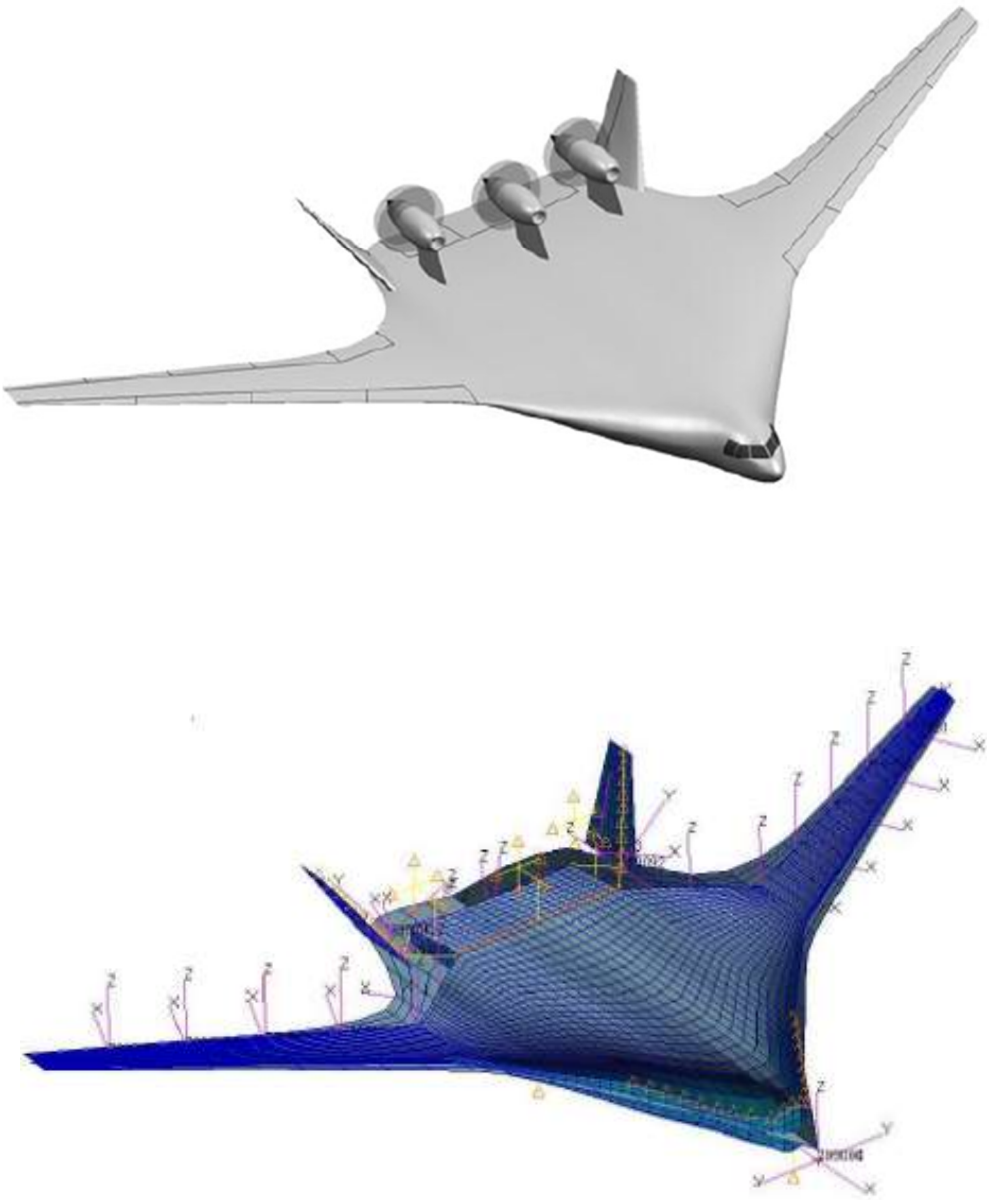


Figure 5.8 Full-Span X-48B OREIO HWB aircraft model

5.2.1 Dataset Generation

Using the same AELINK cards as those used for the half-span model, a data set of *3000* samples were created for the full-span model as well, so the hinge moments and control surface deflections could be compared.

5.2.2 Setting up the Neural Network

A neural network with neurons between the ranges *50-500* were used to compute SBC error (Fig. 5.9) and tried for *320* neurons in the hidden layer with 90% samples for training and 10% for testing the network (Fig. 5.10). Using the lessons learnt from the half-span model, the number of neurons for the hidden layer for the ANN were estimated to be between *270* and *370*, based on the coarse SBC error calculation (Fig. 5.9). With MSE of *5e-10* and training regression as *1* (Fig. 5.10), the network with *320* neurons seemed suitable for the

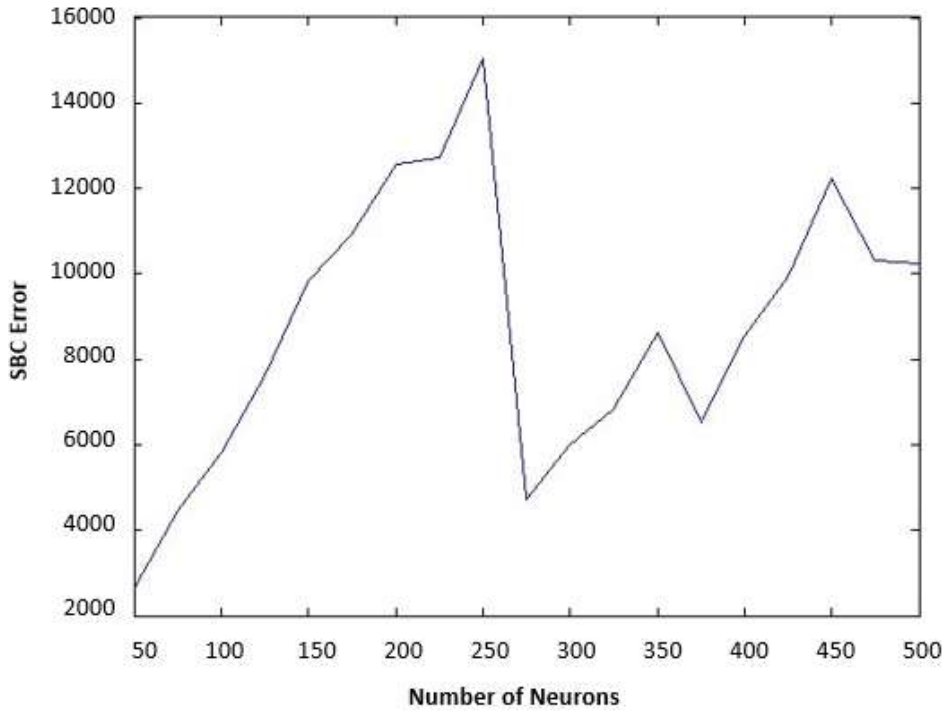


Figure 5.9 SBC Error plot for full-span model

optimization process. In addition, for *320* neurons, the gradient for the network (Fig. 5.11) was much less.

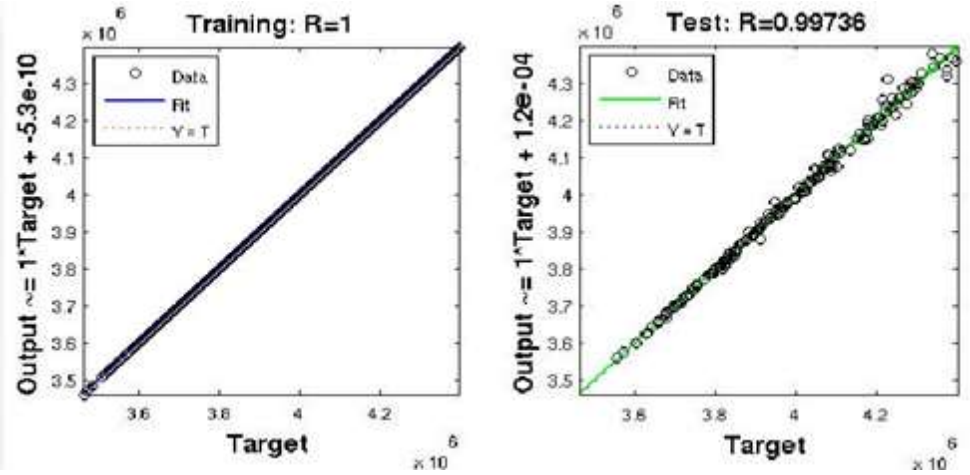


Figure 5.10 Regression plot for 320 neurons for full-span model

The size of the network had to be increased to account for the increased number of control surfaces as well as for the new relationship between the control surface deflections and the HMS, since now the HMS is exactly double the HMS for half-span model.

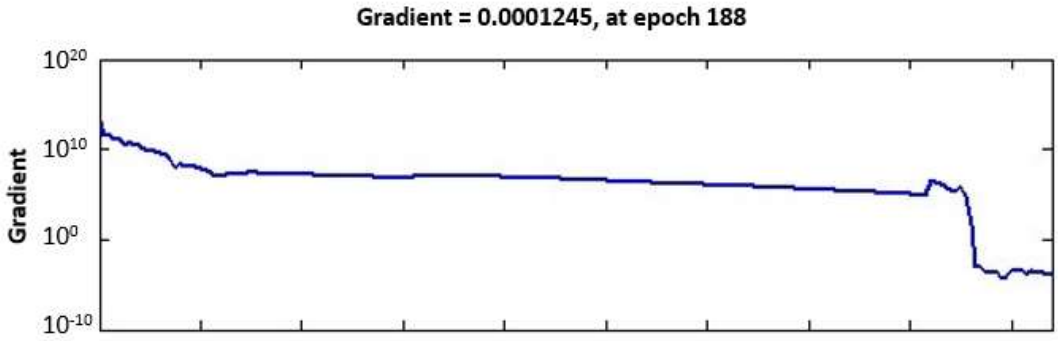


Figure 5.11 Gradient for the ANN for 320 neurons for full span HWB model

5.2.3 Optimization using Genetic Algorithm

Using the prior knowledge obtained from the half-span HWB OREIO model, a mutation type genetic algorithm was setup, with an initial population of *10* to *15* individuals, and a total of 5 generations. The minimum HMS from the process obtained is as shown in Table 5.6. Since using AELINK for optimization did not prove to be very fruitful in the case of half-span model, the AELINK optimization was not performed for the full-span model.

Table 5.6. 2.5g Maneuver optimization results for the HMS as a proxy for actuation power for the full-span HWB model for 50 iterations

Sum of Absolute of Hinge Moment, lb.in.	
Minimum from the Aeroelastic Trim Data Set	3.42e+06
When only Elevator is used	4.02e+06
Optimized Hinge Moment (NN + GA)	**2.93e+06**
NASTRAN Validation	2.93e+06
%ERROR (Between Optimized value and NASTRAN)	3.18e-14%
Improvement over best NASTRAN Case	**14%**
Improvement over using only Elevator	**27%**

Instead, control surface deflections were employed directly. Also, as mentioned before, not to increase the computational time considerably, only *50* iterations of the optimization process were performed. From Table 5.6, we can see that the HMS, when only the elevator is used, is indeed higher than the minimum in the initial dataset. The improvement in the

optimized result when all control surfaces are active compared to the minimum HMS in the aeroelastic dataset is 14%.

Table 5.7. 2.5g Maneuver control surface deflections for minimum HMS for the full-span HWB model for 50 iterations.

Control Surface Deflections, deg.	
AOA	8.00
Elevator	8.77
Inboard 1	-4.48
Inboard 2	-19.13
Outboard 1	22.26
Outboard 2	24.68
Outboard 3	20.19
Outboard 4	20.68
Rudder	9.13

If the number of iterations of the optimization schedule are increased, one can be hopeful to see even better results than those shown in Tables 5.6 and 5.7.

5.3 Lift Distributions

From the lifting line theory, an elliptic lift distribution is proven to produce minimum induced drag for a given lift. This kind of distribution is targeted to minimize induced drag produced by the wing. However, if the whole aircraft is treated as an integrated system, such an elliptic spanwise load distribution on the wing is no longer the optimum for minimum induced drag.

For a HWB, it is essential to treat the whole aircraft as an integrated system. Unlike the conventional aircraft, the spanwise lift distribution of a HWB includes the central body and

the wing as a whole (Fig. 5.12). Obviously, there is no simple answer to what should be the best lift distribution for a HWB. More on sectional lift distributional for HWB may be found in Ref. [69].

Figure 5.12 shows the sectional coefficient of lift distributions of the half-span HWB aircraft model at trimmed conditions on half of the body. It may be assumed that this distribution is symmetrical for the full body. The blue line is the sectional coefficient of lift distribution for when only elevator was used to trim the aircraft. This distribution gives the coefficient of lift at each span location of the wing.

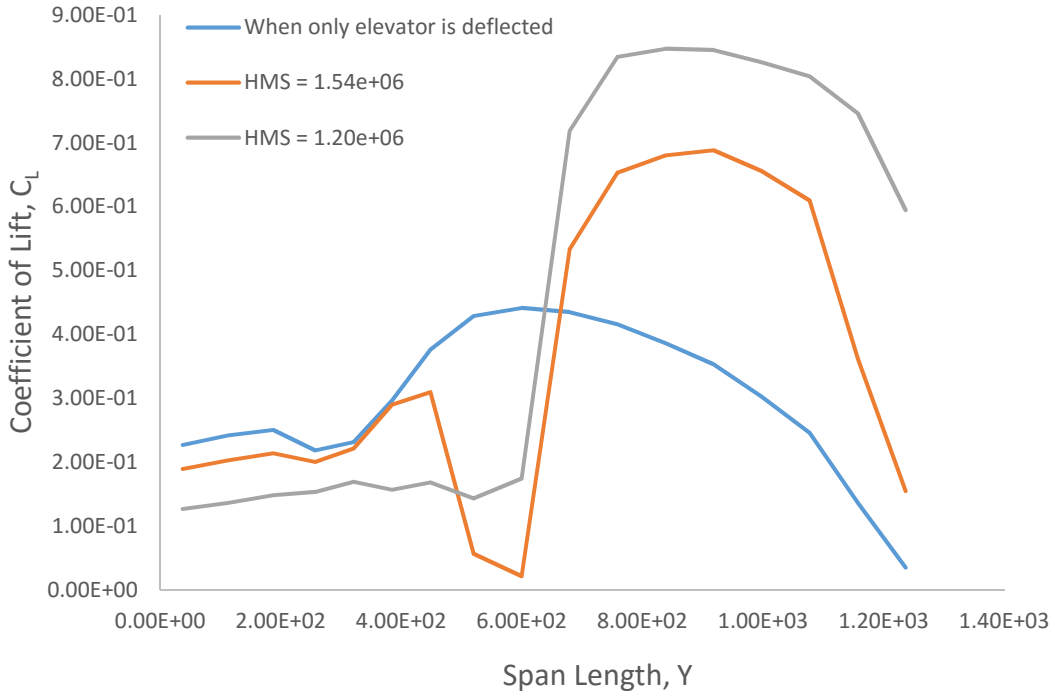


Figure 5.12 Sectional Lift Distribution plot for Half Span model. This plot shows Sectional Lift Coefficient distribution vs. the span length for using only elevator, HMS obtained by 50 iterations of optimization schedule and HMS obtained by 500 iterations of optimization schedule

In case 1, the conventional way of deflecting only the elevator to trim the aircraft in pitching maneuver has a smooth curve along the wing, with sufficient amount of lift produced by the body as well very similar to the plots obtained in Ref. [69].

Case 2 and 3 are for the optimum HMS obtained after the optimization. For case 2 and 3, the dips in the curves are very close to the root of the wing, which may be because of the elevator not producing much lift. The large peaks are at the wing area as the ailerons work harder to trim the aircraft. [71].

Chapter 6

Concluding Remarks

6.1 Hypotheses

Accomplishing improvements of 40% and 27% in HMS for the half-span and full-span HWB OREIO models respectively, have proven that artificial intelligence procedure (NN and GA) as employed here has the potential to work for determining control-surface deflection schedule that would minimize the sum of absolute value of hinge moments. These improvements also suggest that the optimization schedule was successfully customized for this work.

The modification of the data-set after each iteration, the re-training of the neural network and the usage of a genetic algorithm proved to be very fruitful in obtaining the control surface deflections. Also, the proper training of the neural network seems to have produced an accurate surrogate model. This can be confirmed by the percentage errors between the HMS from ANN and the HMS from MSC Nastran (validation) that are shown in the Tables 5.1, 5.3, 5.5 and 5.7.

The optimum HMS value obtained from the optimization schedule was found to depend upon the number of iterations which were varied from *50-500*. Going beyond *500* iterations of this schedule is exceedingly time consuming and computationally expensive. It is clear, that the GA successfully found the control allocation for at least a local optimum HMS. This local optimum HMS obtained from GA may or may not be the true global optimum. Further study on this issue is required.

In summary, all three hypotheses were proven [1].

6.2 Project Goals

Reducing actuation power is an enabler for ultra-efficient commercial transport aircraft. This optimization schedule reducing the sum of absolute value of hinge moments will directly affect the power requirements, structural loads and therefore overall vehicle weight. The biggest advantage of this approach is that it is independent of the configuration of the vehicle. It can be easily applied to other innovative and unconventional configurations.

Even though the minimization was performed on the sum of absolute value of hinge moments, the actuation power is a linear function of the hinge moments. Therefore, it can be stated that a proof-of-concept process has been developed to apply artificial intelligence to minimize the actuation power. One can also easily apply this approach with other objectives such as to minimize stress, deflections etc.

6.3 Future Work

It is planned to include actuator dynamics in terms of stiffness, damping, and power consumption into the analysis. With the full aeroservoelastic model, arbitrary asymmetric maneuvers (engine out, dynamic overswing, and sideslip) will also be analyzed. This approach will also be applied to the dynamic state space model. The plan is also to apply

this process to other different maneuvers, compute actuation energy and compare it with the conventional control surface schedule.

Finally, develop most of the neurocomputing process into a full user friendly tool in compliance with NASA software development process and prepare a complete manual, since it can easily be leveraged into other projects.

END REMARKS

"By far the greatest danger of Artificial Intelligence is that people conclude too early that they understand it."

— Eliezer Yudkowsky

REFERENCES

[1] Chhabra, R., Mulani, S. B., Kapania, R.K. and Schetz, J.A., "Control Power Optimization Using Artificial Intelligence for Hybrid Wing Body Aircraft", AIAA 2015-2323, *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization, Aviation*, June 2015.

[2] Jonsson, A., Morris, R.A. and Pedersen, L., "Autonomy in Space, Current Capabilities and Future Challenges", AAAI, *AI Magazine*, Vol.28, No. 4, 2007.

[3] "From Toys to Tools: Unmanned Aerial Vehicles", GEO Informatics, 2012.

[4] Pavone, M., Acikmese, B., Nesnas, I.A. and Starek, J., "Spacecraft Autonomy Challenges for Next Generation Space Missions", 2013, Stanford University.

[5] Eriksen, C.C., Osse, T.J., Light, R.D., Wen, T., Lehman, T.W., Sabin, P.L., Ballard, J.W. and Chiodi, A.M., "Seaglider: A Long-Range Autonomous Underwater Vehicle for Oceanographic Research", Oceanic Engineering, IEEE, Vol. 26, Issue 4, 2001.

[6] Shroff, G., The Intelligent Web: Search, Smart Algorithms and Big Data, 2013, Oxford University Press, pp. 195-198.

[7] Tsourdos A., White B. and Shanmugavel M., Cooperative Path Planning of Unmanned Aerial Vehicles, John Wiley and Sons, 2010, pp. 7-20.

[8] *X-48 Research: All Good Things Must Come to an End*, NASA.gov, April 2013.

[9] Gern, F.H., Vicroy, D.D., Mulani, S.B., Chhabra, R., Kapania, R.K., Schetz, J.A., Brown, D., and Princen, N.H., "Artificial Intelligence Based Control Power Optimization on Tailless Aircraft", ARMD Seedling Fund Phase I Final Report, *NASA Technical Memorandum*, NASA/TM-2014-218671, November, 2014.

[10] Liebeck R.H., "Design of the Blended Wing Body Subsonic Transport", *Journal of Aircraft*, Vol.41, No. 1, 2004.

[11] Lehmkuehler K., Wong K.C. and Verstraete D., "Design and Test of a UAV Blended Wing Body Configuration", 2012, *ICAS*, UK.

[12] Ikeda, T., "Aerodynamic Analysis of a Blended-Wing-body Aircraft Configuration", RMIT University, March 2006.

[13] Garrison, P., "Batplane", *Air and Space Magazine,* September 2009.

[14] Potsdam, M.A., Page, M.A. and Liebeck, R.H., "Blended Wing Body Analysis and Design", AIAA-97-2317, June 1997, *Applied Aero Proceedings*, Vol. 2, pp. 799-805.

[15] Mizrahi, J., "Flight to the Future, Which Way the World's Next Generation Airliner? A Look at the Past, the Present and the Possible!" *Sentry Magazines,* Vol. 29, No. 2, April 1999.

[16] Creech, G., Braukus, M. and Koehler, T., *Transformed X-48C Makes Successful First Flight*, NASA.gov, 2012.

[17] Cameron, D. and Princen, N., "Control Allocation Challenges and Requirements for the Blended Wing Body," AIAA Paper 2000-4539, *AIAA Guidance*, Navigation, and Control Conference and Exhibit, August 14-17, 2000, Denver, CO.

[18] Hambrick, E. H. and Thomason, N. M., "Conceptual Aircraft Hinge Moment Measurement System", California Polytechnic State University, June 2010, San Luis Obispo.

[19] Etkin, B. and Reid, L.D., Dynamics of Flight Stability and Controls, Third Edition, John Wiley and Sons, Inc., 1996, pp. 18-60.

[20] Phillips, W.H., "Journey in Aeronautical Research: Problems Encountered as a Result of Wartime Developments", *NASA Langley*, 1945.

[21] Mehrotra, K., Chilukuri M., and Sanjay R., Elements of Artificial Neural Networks. Boston: MIT Press, 1997, pp. 1-106.

[22] "Clever Computers: The Dawn of Artificial Intelligence", *The Economist*, May 2015.

[23] Taigman, Y., Yang, M., Ranzato, M.A. and Wolf, L., "DeepFace: Closing the Gap to Human-Level Performance in Face Verification", *Facebook AI Research*, 2014.

[24] Sadun, E. and Sande, S., Talking to Siri: Mastering the Language of Apple's Intelligent Assistant, 2014, Apple, Inc, pp. 1-38.

[25] Lee, N., Digital Da Vinci: Computers in the Arts and Sciences, 2014, Springer New York Heidelberg Dordrecht London, pp. 1-8.

[26] "Artificial Intelligence is Creeping into our Everyday Lives", *Marketwatch*, September 2015.

[27] Cellan-Jones, R., "Stephen Hawking Warns Artificial Intelligence could end Mankind", *BBC News*, December 2014.

[28] Gurney, K., An Introduction to Neural Networks, 1997, UCL Press Limited, pp. 12-37.

[29] Haykin, S., Neural Networks - A Comprehensive Foundation, 2nd Edition, Prentice Hall, July 1998, pp. 23-71.

[30] Ansari, N. and Hou, E., Computational Intelligence for Optimization, Kluwer Academic Publishing Group, Norwell, MA, 1997, pp. 83-98.

[31] Tagliarini, G.A., Christ, J.F., and Page, E.W., "Optimization Using Neural Networks," *IEEE Transactions on Computers*, Vol. 40, No. 12, Dec. 1991, pp. 1347-1358.

[32] Boger, Z., and Guterman, H., "Knowledge Extraction from Artificial Neural Network Models," *IEEE Systems, Man, and Cybernetics Conference*, Orlando, FL, 1997.

[33] Graupe, D., Principles of Artificial Neural Networks, 2nd Edition, 2007, Advanced Series on Circuits and Systems, Vol. 6, pp. 59-112, World Scientific Publishing Co. Pte. Ltd.

[34] Beale, M.H., Hagan, M.T. and Demuth, H.B., *Neural Network Toolbox User's Guide*, 2015, The MathWorks, Inc., Natick, MA

[35] Hornik, K., Stinchcombe, M. and White, H., "Multilayer Feedforward Networks Are Universal Approximators", *Neural Networks*, Vol.2, pp. 359-366, 1989.

[36] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation, *Parallel distributed processing: Explorations in the microstructure of cognition,* Vol. 1, pp. 318-362, The PDP Research Group, Cambridge, MA, MIT Press.

[37] MacLeod, C., "An Introduction to Practical Neural Networks and Genetic Algorithms", *Institutional Repository National Mining University of Ukraine*, 2013.

[38] Kriesel, D., A Brief Introduction to Neural Networks, Zeta version, 2007, pp. 37-124, Dkriesel.com.

[39] Sarle, W. S., "Donoho-Johnstone Benchmarks: Neural Net Results", SAS Institute Inc., 1999.

[40] Sarle, W. S., "Stopped Training and Other Remedies for Overfitting," *Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics,* 1995.

[41] Sarle, W. S., *Neural Network FAQ, part 3 of 7: Generalization*, 1997 (ftp://ftp.sas.com/pub/neural/FAQ3.html).

[42] Blum, A., *Neural Networks in C++*, 1992, NY, Wiley.

[43] Swingler, K., *Applying Neural Networks: A Practical Guide,* 1996, pp. 3-50, London: Academic Press.

[44] Yu, W. and Sanchez, E. N., Advances in Computational Intelligence, 2009, pp. 11-38, Springer-Verlag Berlin Heidelberg.

[45] Berry, M .J. A., and Linoff, G., Data Mining Techniques, 1997, pp. 281-320, NY, John Wiley & Sons.

[46] Schwarz, G., "Estimating the Dimension of a Model," *Annals of Statistics*, Vol. 6, pp. 461-464, 1978.

[47] Akaike, H., "Fitting Autoregressive Models for Prediction," *Annals of the Institute of Statistical Mathematics,* Vol. 21, pp. 243-247, 1969.

[48] Hurvich, C.M., and Tsai, C.L. (1989), "Regression and Time-Series Model Selection in Small Samples," *Biometrika,* Vol. 76, pp. 297-307, 1989.

[49] Moody, J. E., "The Effective Number of Parameters: An Analysis of Generalization and Regularization in Nonlinear Learning Systems", in Moody, J. E., Hanson, S. J., and Lippmann, R. P., *Advances in Neural Information Processing Systems 4,* 1992, pp. 847-854.

[50] Weiss, S.M. and Kulikowski, C. A., Computer Systems That Learn, 1991, Morgan Kaufmann.

[51] Efron, B. and Tibshirani, R. J., An Introduction to the Bootstrap, 1993, pp. 45-59, Chapman & Hall, London.

[52] Sumathi S., Hamsapriya T., and Surekha P., Evolutionary Intelligence, 2008, pp. 31-84, Springer-Verlag Berlink Heidelberg.

[53] Sumathi S. and Surekha P., Computational Intelligence Paradigms, 2010, pp. 29-108, Taylor and Francis Group, CRC Press.

[54] Global Optimization Toolbox User's Guide, 2015, The MathWorks Inc.

[55] MATLAB 2015a, The MathWorks, Inc. 3 Apple Hill Drive, Natick, MA 01760-2098.

[56] Schaffer, J. D., Whitley, D. and Eshelman, L. J., "Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art", Combinations of Genetic Algorithms and Neural Networks, *IEEE*, 1992, COGANN-92, pp.1-37.

[57] MSC Nastran 2013, Multidisciplinary Structural Analysis, MSC Software Corporation, Santa Ana, CA 92707.

[58] MSC Nastran 2008, Static Aeroelastic Analysis, MSC Software Corporation, 2 MacArthur Place, Santa Ana, CA 92707, USA, 2008.

[59] Neubacher, C., "Flight Dynamic Investigations of a Blended Wing Body Aircraft", 2008, *Hamburg University of Applied Sciences*.

[60] Nielsen, J. N. and Goodwin, F. K., "Preliminary Method for Estimating Hinge Moments of All-Movable Controls", *Office of Naval Research, Arlington*, March 1982.

[61] Giesing, J. P., Kalman, T. P., and Rodden, W. P., "Application of the Doublet-Lattice Method to Nonplanar Configurations in Subsonic Flow", *Journal of Aircraft*, vol.8, No. 6, 1971, pp. 406-413.

[62] Rodden, W.P. and Johnson, E.H. (Editors), MSC. Nastran Version 68, Aeroelastic Analysis User's Guide, 2013, MSC Software.

[63] Cummings, R. M., Mason, W. H., Morton, S. A. and Mcdaniel, D. R., Applied Computational Aerodynamics, 1998, pp. 208-220, Cambridge University Press.

[64] Prandtl. L, "Applications of Modern Hydrodynamics to Aeronautics" NACA-TR-116, NASA, 1923.

[65] Anderson, J.D., Introduction to Flight, 3rd Ed., New York: McGraw-Hill, 1989, pp. 357-409.

[66] Drela, M., Extended Vortex Lattice Method – AVL, *MIT,* 2004.

[67] Bisplinghoff, R. L., Ashley, H., and Halfman, R. L., Aeroelasticity, Dover, 1955, pp. 1-13, pp. 421-526.

[68] Keramat, M. and Kielbasa, R., "Latin Hypercube Sampling Monte Carlo Estimation of Average Quality Index for Integrated Circuits", *Analog Integrated Circuits and Signal Processing*, Vol. 14, No. 1/2, pp. 131-142, 1997.

[69] MSC Nastran 2008, Quick Reference Guide, MSC Software Corporation, 2 MacArthur Place, Santa Ana, CA 92707, USA, 2008.

[70] Angelova, A., Abu-Mostafa, Y., Perona, P., "Pruning Training Sets for Learning of Object Catergories", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, CA, USA, June 2005.

[71] Qin, N., Vavalle, A., Le Moigne, A., Laban, M., Hackett, K. and Weinerfelt, P., "Aerodynamic Studies for Blended Wing Body", AIAA 2002-5448, *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Sept. 2002.

[72] Caughey, D.A., *Introduction to Aircraft Stability and Control*, Cornell University, 2011, pp.25-44.