FEATURE-BASED NEURO-SYMBOLIC NETWORKS
FOR GLOBAL DIAGNOSTICS

A Thesis
Presented to
the Faculty of Engineering and Computer Science
University of Denver

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
Tracy L. Schantz
June 2012
Advisor: Dr. Rahmat Shoureshi
ABSTRACT

Engineered system diagnostics have been researched over the years with many successful results. From transportation systems to office technologies, many have been equipped with self-diagnostic capabilities and are called Smart Machines. In spite of these advances, current diagnostic systems are driven by direct sensory information without much concern for patterns of the system behavior or features associated with them. For large-scale systems with complex dynamics, global as well as local diagnostics become of great importance, where sensory information is used as input for the local diagnostics, and patterns of behavior or features are utilized for global diagnostics.

The main objective of this research is to develop a bio-inspired data/information architecture for feature-based global diagnostics of a large-scale system. In order to accomplish this goal, information from local diagnostic systems is integrated with physical and engineering principles (e.g. conservation of momentum) to create a feature-based neuro-symbolic network. This network is very similar to a neural network, except that it is based on a physical equation, and it uses features instead of raw data. Results from this network identify patterns of behavior that display whether the system has any faulty states. The architecture of this network includes two feature-based neuro-symbolic networks, one for the x-axis and one for the z-axis. Each of these networks has four inputs and four outputs, and includes one hidden layer.
In order to verify performance of this feature-based diagnostic system, a working aircraft model has been equipped with pressure sensors on its wing surface, and data has been taken during flight. By using computational fluid dynamic analysis, pressure contours along the wing have been developed. These pressure patterns have been used as input for our developed feature-based neuro-symbolic networks to predict the state of the model aircraft in real-time. Based on these research results, we have developed a neuro-symbolic diagnostic technique which can be applied to fault detection of large-scale systems, using local diagnostic information.
ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor, Dr. Rahmat Shoureshi, and the members of my thesis defense committee: Dr. Corinne Lengsfeld, Dr. Yun-Bo Yi, and Dr. David Gao. I would also like to express my gratitude to Dr. Chris Aasted for his help with the model aircraft and knowledge of neural networks, Dr. Michael Kontitsis and Dr. Sun-Wook Lim for their assistance with neural networks, and Scott Ferdinand for his help with setting up the pressure sensor network.

I would like to thank the Air Force Office of Scientific Research for their funding which made this research possible. I would also like to thank all of the professors and students included in the MURI project, from the following universities: Stanford University, UCLA, CU Boulder, UBC, and Johns Hopkins University.

Most of all, I would like to thank my friends and family for their unwavering support and assistance during this very busy and stressful time. I would especially like to thank my mother, her partner Charlie, and my boyfriend Rob, for their support and help with reviewing and editing this thesis.
# TABLE OF CONTENTS

Abstract ............................................................................................................................... ii

Acknowledgements ............................................................................................................ iv

Table of Contents ................................................................................................................ v

Nomenclature .................................................................................................................... vii

Subscripts .......................................................................................................................... vii

Acronyms ......................................................................................................................... viii

List of Figures ..................................................................................................................... x

List of Tables ................................................................................................................... xiii

Chapter One: Introduction .................................................................................................. 1
  1.1 - Background and Motivation ........................................................................... 1
  1.2 - Overview of Aircraft Structural Health Monitoring ....................................... 3
  1.3 - Research Goals and Approach ........................................................................ 5
  1.4 - Thesis Outline ................................................................................................. 6

Chapter Two: Literature Review ........................................................................................ 7
  2.1 General Fault Diagnostics ................................................................................ 7
  2.2 Feature-Based Fault Diagnostics ..................................................................... 9
  2.3 Feature Extraction .......................................................................................... 17
  2.4 Artificial Neural Networks ............................................................................ 19
  2.5 Neuro-Symbolic Networks ............................................................................ 23

Chapter Three: Analysis and Design ................................................................................ 29
  3.1 Momentum Equation for Inertial Control Volume ........................................ 29
  3.2 Testbed Aircraft ............................................................................................. 33
  3.3 Computational Fluid Dynamics Model .......................................................... 34
  3.4 Grid Validation .............................................................................................. 39
  3.5 Computational Fluid Dynamics Optimization ............................................... 40
  3.6 Computational Fluid Dynamics Simulation ................................................... 42
  3.7 Data Collection .............................................................................................. 43

Chapter Four: Design of Networks ................................................................................... 49
  4.1 Feed-Forward Neural Network ...................................................................... 49
  4.2 Feature-Based Neuro-Symbolic Network ...................................................... 51
  4.3 Matlab Code for the Networks ....................................................................... 52
# NOMENCLATURE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOA</td>
<td>Angle of Attack, [°]</td>
<td></td>
</tr>
<tr>
<td>CS</td>
<td>Control Surface, [m$^2$]</td>
<td></td>
</tr>
<tr>
<td>CV</td>
<td>Control Volume, [m$^3$]</td>
<td></td>
</tr>
<tr>
<td>$d\vec{A}$</td>
<td>Change in Area, [m$^2$]</td>
<td></td>
</tr>
<tr>
<td>$dV$</td>
<td>Change in Volume, [m$^3$]</td>
<td></td>
</tr>
<tr>
<td>$\vec{F}$</td>
<td>Total Force, [N]</td>
<td></td>
</tr>
<tr>
<td>$\vec{F}_B$</td>
<td>Body Force, [N]</td>
<td></td>
</tr>
<tr>
<td>$\overrightarrow{F}_S$</td>
<td>Surface Force, [N]</td>
<td></td>
</tr>
<tr>
<td>$\rho$</td>
<td>Air Density, [kg/m$^3$]</td>
<td></td>
</tr>
<tr>
<td>$t$</td>
<td>Time, [s]</td>
<td></td>
</tr>
<tr>
<td>$u$</td>
<td>X-axis Velocity, [m/s]</td>
<td></td>
</tr>
<tr>
<td>$\vec{v}$</td>
<td>Velocity, [m/s]</td>
<td></td>
</tr>
<tr>
<td>$\omega$</td>
<td>Z-axis Velocity, [m/s]</td>
<td></td>
</tr>
</tbody>
</table>

# SUBSCRIPTS

<table>
<thead>
<tr>
<th>Subscript</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>With respect to the x-axis, or in the x-direction</td>
</tr>
<tr>
<td>Y</td>
<td>With respect to the y-axis, or in the y-direction</td>
</tr>
<tr>
<td>Z</td>
<td>With respect to the z-axis, or in the z-direction</td>
</tr>
<tr>
<td>ACRONYMS</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>AFOSR</td>
<td>Air Force Office of Scientific Research</td>
</tr>
<tr>
<td>AFRL</td>
<td>Air Force Research Lab</td>
</tr>
<tr>
<td>AGL</td>
<td>Above Ground Level</td>
</tr>
<tr>
<td>AJETS</td>
<td>Automated Jet Engine Test Strategy</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>AOA</td>
<td>Angle of Attack</td>
</tr>
<tr>
<td>ASHM</td>
<td>Aircraft Structural Health Monitoring</td>
</tr>
<tr>
<td>BNN</td>
<td>Biological Neural Network</td>
</tr>
<tr>
<td>CFD</td>
<td>Computational Fluid Dynamics</td>
</tr>
<tr>
<td>CLIP</td>
<td>Connectionist Inductive Learning and Logic Programming</td>
</tr>
<tr>
<td>CM</td>
<td>Confusion Matrix</td>
</tr>
<tr>
<td>FBFD</td>
<td>Feature-Based Fault Diagnostics</td>
</tr>
<tr>
<td>FFNN</td>
<td>Feed-Forward Neural Network</td>
</tr>
<tr>
<td>GA</td>
<td>Genetic Algorithm</td>
</tr>
<tr>
<td>HNS</td>
<td>Hybrid Neural System</td>
</tr>
<tr>
<td>KBANN</td>
<td>Knowledge-Based Artificial Neural Network</td>
</tr>
<tr>
<td>MS</td>
<td>Microsoft</td>
</tr>
<tr>
<td>MURI</td>
<td>Multidisciplinary University Research Initiative</td>
</tr>
<tr>
<td>NACA</td>
<td>National Advisory Committee for Aeronautics</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>NN</td>
<td>Neural Network</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>NSN</td>
<td>Neuro-Symbolic Network or Neural-Symbolic Network</td>
</tr>
<tr>
<td>RC</td>
<td>Radio Controlled</td>
</tr>
<tr>
<td>RMS</td>
<td>Root Mean Square</td>
</tr>
<tr>
<td>SHM</td>
<td>Structural Health Monitoring</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>USAF</td>
<td>United States Air Force</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1. Survey of fault-diagnosis methods. Source: Data adapted from: Isermann and Rolf 2006. ................................................................. 9

Figure 2. Diagram of diagnostic/prognostic system. Source: Data adapted from: Orsagh, et al. 2003. ................................................................. 12

Figure 3. Block diagram of the diagnostic system. Source: Data adapted from: Han, Yang and Yin 2007. .......................................................... 14

Figure 4. Diagram of biological neurons. Source: Data adapted from: Hagan, Demuth, and Beale 1996. ............................................................... 19

Figure 5. Diagram of a single-input neuron. Source: Data adapted from: Hagan, Demuth, and Beale 1996. ......................................................... 21

Figure 6. Diagram of a feed-forward neural network. ........................................ 22

Figure 7. Flow chart of KBANN. Source: Data adapted from: Towell and Shavlik 1994. ................................................................. 25

Figure 8. Organization of the Perceptual System. Source: Data adapted from: Velik and Bruckner 2008. .............................................................. 27

Figure 9. Picture of the Hobbico Hobbistar 60 Select testbed. ......................... 33

Figure 10. Cross-section of the Hobbistar airfoil. Source: Figure adapted from: Hobbico Hobbistar. ................................................................. 35

Figure 11. Cross-section of 4 semi-symmetrical airfoils. Source: Figure adapted from: FlyingFoam.com. ............................................................. 36

Figure 12. Figure showing Gambit airfoil model. ............................................. 36

Figure 13. Figure showing major vertices in Gambit airfoil grid. ....................... 37

Figure 14. Figure showing major faces and lettered edges in Gambit airfoil grid. .... 38

Figure 15. Figure showing final mesh of the Gambit airfoil grid. ....................... 39

Figure 16. Figure showing the location of the minimum pressure versus the angle of attack. ................................................................. 41
Figure 17. Figure showing the location of the four pressure sensors on the airfoil.............. 42
Figure 18. Photo of pressure data logger. ......................................................................... 44
Figure 19. Block diagram of pressure data logger. Source: Figure by Scott Ferdinand... 44
Figure 20. Photo of the top pressure sensor placement. ................................................... 45
Figure 21. Photo of the front, bottom, and rear pressure sensor placement. ..................... 45
Figure 22. Close-up photo of the top sensor and how it is attached to the wing. .......... 46
Figure 23. Photo of pressure sensor ............................................................................... 46
Figure 24. Ruby system by uThere. ................................................................................. 48
Figure 25. Diagram of a feed-forward neural network .................................................... 50
Figure 26. Diagram of two parallel neuro-symbolic networks......................................... 52
Figure 27. Magnitude of pressures and forces versus velocity at zero angle of attack..... 56
Figure 28. Magnitude of pressure on top of airfoil versus angle of attack. ..................... 57
Figure 29. Magnitude of pressure on front of airfoil versus angle of attack. ................. 58
Figure 30. Magnitude of pressure on rear of airfoil versus angle of attack. .................... 59
Figure 31. Magnitude of pressure on bottom of airfoil versus angle of attack............. 60
Figure 32. Magnitude of force in the x-direction versus angle of attack. ....................... 61
Figure 33. Magnitude of force in the y-direction versus angle of attack. ....................... 62
Figure 34. Pressure in absolute psi versus time. ............................................................. 63
Figure 35. Pressure in gauge Pascal versus time. ............................................................ 64
Figure 36. Temperature in °C versus time. ..................................................................... 65
Figure 37. Altitude above ground level versus time. ...................................................... 66
Figure 38. Velocity versus time. .................................................................................... 67
Figure 39. Force in the x and z-directions versus time. ............................................... 68
Figure 40. Confusion matrix for CFD neural network results. ........................................ 71

Figure 41. Confusion matrix for CFD z-axis neuro-symbolic network results. .............. 72
LIST OF TABLES

Table 1. Strengths of neural and symbolic networks. Source: Data from Wermter and Sun 2000. .......................................................................................................................... 24

Table 2. Comparison between results of the original grid and the validation grid. ........ 40

Table 3. Results from CFD simulation. ................................................................. 55

Table 4. Percent accuracy of neural network for different numbers of hidden nodes. ..... 70

Table 5. Number of hidden nodes used for each network. ........................................ 70

Table 6. RMS, mean, and standard deviation error for neuro-symbolic networks. ........ 73

Table 7. Confusion matrix percent accuracy for all networks. ................................. 75

Table 8. Train and test time and number of iterations for each network. ..................... 76
CHAPTER ONE: INTRODUCTION

In order for the US to maintain its technological leadership in the aerospace industry, new and advanced technologies need to be developed. An area of great interest is the addition of intelligence to aerospace vehicles, especially in the self-diagnostics area. Even though aircraft travel is one of the safest forms of transportation in the US, the government and the airline industry are focused on making it even safer and more reliable. One of the primary concerns is that commercial aircraft are starting to reach or exceed their original design life, which is making the ability to monitor the health of these aircraft even more important. Also the airline industry is experiencing some hard economic times, and the ability to conduct maintenance as “needed” rather than as “scheduled” would have a profound impact both economically, as well as operationally.

This project has been designed and funded by the Air Force Office of Scientific Research (AFOSR) with the goal of developing a naturally inspired neurobiological system for aerospace vehicles, e.g. an aircraft, that can self-diagnose and eventually self-heal itself.

1.1 - Background and Motivation

This research project is conducted in conjunction with complementary projects at the following partner universities: Stanford University, UCLA, CU Boulder, University of British Columbia, and Johns Hopkins University. This is a Multidisciplinary
The goal of this MURI is:

… to design the next generation of intelligent sensing materials inspired by the parallelism, fault tolerance, and adaptability of neurobiological system. We will employ multi-scale design, synthesis, and fabrication techniques to create lightweight, intelligent aerospace materials that can sense their state automatically with high resolution and can communicate important high-level information to control and service/management systems. (Stanford University, MURI website)

This MURI has further been divided up into four main tasks:

1. Design of bio-inspired “stretchable” sensory network.
2. Sensing, diagnostics, recognition and state awareness.
3. Multi-functional material design and analysis.
4. Integration, prototype and validation.

This project specifically focuses on task 2. The main objective of this research is to develop a bio-inspired data/information architecture for feature-based global diagnostics of a large-scale dynamic system with applications to aerospace vehicles. The large-scale system in this case is an aircraft. Each sensor system on the aircraft will gather data and complete diagnostics on a local level. The goal of this research is to create a system that will collect all of the local diagnostic information and analyze it to create global diagnostics for the entire aircraft.

The motivation behind this research is that with this level of intelligence built into the global diagnostic system, the health state of the aircraft, or other system on
which it is implemented, would be known. This will make all of these systems, including aircraft, much safer and provide cost savings on parts and repairs.

1.2 - Overview of Aircraft Structural Health Monitoring

The area of Aircraft Structural Health Monitoring (ASHM) is a subset of a larger field, Structural Health Monitoring (SHM). SHM entails employing a damage detection and classification system on a structure, in order to monitor the health of the structure in real-time. Research has been done in the area of SHM by (Byington, Watson, and Edwards 2004; Caglayan, Allen, and Wehmuller 1988; Gomes, Crowther, and Wood 2008; Zhao et al. 2007). The idea of SHM has been used since the early 19th Century when train conductors would knock a hammer against the train wheels to hear if the sound had changed, which would indicate damage. The field of SHM started to grow 10 to 15 years ago, due to technological developments and an increased interest in probable life-safety and monetary benefits (Farrar and Worden 2006, 303-6).

Another reason the SHM industry had recently grown is because of the aging of commercial aircraft. Statistics show that the number of aging aircraft (over 15 years in age) is continuously growing. The flight envelopes and safe operating conditions of these aging aircraft become more difficult to predict, and more testing and maintenance is required to keep them operational. Also electronics, avionics, and engines are typically replaced, while the body (fuselage) of the aircraft is almost never replaced, which is where most of the structural damage occurs (Boller 2001, 432-33).

SHM is the use of a system to identify damage in structural systems. This type of system is typically used on aerospace, civil, and mechanical engineering structures.
Damage in SHM “can be defined as changes introduced into a system that adversely affect its current or future performance” (Farrar and Worden 2006, 303). However, this damage cannot be detected without original, undamaged, information about the structure. This undamaged information is then regularly compared to the current state in order to detect any changes in real-time.

Damage can occur over many different periods of time, and SHM needs to be able to deal with all speeds of damage propagation. For instance, fatigue and corrosion damage can accumulate over a long period of time; however earthquakes and hail storms can cause significant damage over a very short period of time (Farrar and Worden 2006, 304). SHM systems need to be able to account for these differences so that there is warning before the structure fails.

There are two main motivations for developing SHM technology: life-safety and economic impact (Farrar and Worden 2006, 305). It would save lives by alerting a pilot to damage to the aircraft, so that the pilot could safely land the plane before the damage progresses to failure. Money could be saved by a change in maintenance procedures. Currently aircraft maintenance is completed at regular intervals (as scheduled) and parts are replaced before the end of their design life. This type of system would be able to alert the maintenance team when maintenance or a part replacement is needed. This would save money used for regular maintenance and avoid replacing parts that are still in good, working condition.

Currently there are a few major issues that are creating challenges for the SHM field. The first challenge is that damage typically occurs locally and may not alter the entire system; this can cause the SHM system to miss the damage. The second challenge
is that the identification of damage must occur “in an unsupervised learning mode” (Farrar and Worden 2006, 312). The third challenge is that the SHM system needs to not be damaged during use. The fourth challenge is convincing people that the SHM system will operate correctly and provide economic and life-safety benefits (Farrar and Worden 2006, 312).

1.3 - Research Goals and Approach

The main objective of this research is to develop a bio-inspired data/information architecture for feature-based global diagnostics of a large-scale system. In order to accomplish this goal, information from local diagnostic systems is integrated with physical and engineering principles (e.g. conservation of momentum) to create a feature-based neuro-symbolic network. This network is very similar to a neural network, except that it is based on a physical equation, and it uses features instead of raw data.

To achieve this goal, four steps were completed. First, a basic neural network was created as a baseline for comparison. Second, a model aircraft was chosen and pressure sensors were set up on the front, back, top, and bottom of the wing. Next, a computational fluid dynamics (CFD) model was created of the model aircraft wing to use as the feature input into the neuro-symbolic network. Finally, the neuro-symbolic network was built, using the momentum equation for inertial control volume, and the results were compared with the basic neural network.
1.4 - Thesis Outline

This thesis contains seven chapters including Chapter 1 which discussed the necessary background, motivation, and the overall research goals and approach of the research. Chapter 2 includes a presentation of the state-of-the-art in SHM and feature-based diagnostics, and a review of recent literature from other researchers in the same area, as well as a discussion of neural and neuro-symbolic networks. Chapter 3 discusses the analysis of the momentum equation and the testbed aircraft, and the design of the computational fluid dynamics (CFD) simulations. Chapter 4 introduces the design of the neural and neuro-symbolic networks for the determination of the aircraft diagnostic. Chapter 5 comprises the results from the CFD simulations and how they were used, and the results from flying the testbed aircraft. Chapter 6 covers the simulation of the two networks as well as the comparison to the physical model used to implement these results. Finally, Chapter 7 summarizes the results, the contributions this project has made to the field, and discusses possible future research in this area.
CHAPTER TWO: LITERATURE REVIEW

Diagnostics is an integral part of many engineered systems and processes. Most diagnostics are accomplished by investigating the cause and effect of failed systems. While many such systems rely on raw data from the failed machine or process, there has been little effort in feature-based diagnostics that utilize patterns embedded within the raw data. As with most industries, the engineering community is always looking for new and better diagnostic systems. Some of the issues and shortcomings of current diagnostic systems include: slow response, systems that don’t learn or operate on their own, systems that can only operate on one part of an entire system, and systems that do not operate based on physical laws.

In this literature review, current and new fault diagnostic systems will be investigated, in five parts: (1) general fault diagnostics will be reviewed, (2) feature-based fault diagnostics will be discussed in detail, (3) feature extraction will be described, (4) artificial neural networks will be explained, and (5) neuro-symbolic networks will be discussed.

2.1 General Fault Diagnostics

Research has been done in the area of fault diagnostics by (Deuker, Perrier, and Amy 1998; Frank 1990; Garcia and Frank 1997; Isermann 2006). Fault diagnostics, or fault diagnosis, is a process comprised of the following steps: supervision, fault detection,
and fault diagnosis (Isermann 1997, 640). In the case of an automated technical process, supervision involves recording data on the current operational status. Fault detection then completes the comparison between the collected data and normal operation data. If a fault is detected, the comparison is sent to the fault diagnosis system.

Engineering fault diagnostics started to advance as the operation of technical processes became more automated. This created a need for supervision to specify when a process condition that was not wanted or allowed was present. These systems were developed to act in the necessary fashion to keep the system operating correctly and to evade damage or accidents (Isermann 1997, 639). Through these actions, fault diagnostics are able to improve reliability, safety, and the economic output of an automated process.

The purpose of a fault diagnosis system is to determine “the type, size and location of the fault as well as its time of detection” (Isermann 1997, 641). According to Isermann, there are four requirements that “advanced methods of supervision and fault diagnosis” (Isermann 1997, 640) must meet. These requirements are:

… (i) Early detection of small faults with abrupt or incipient time behaviour [sic]. (ii) Diagnosis of faults in the actuator, process components or sensors. (iii) Detection of faults in closed loops. (iv) Supervision of processes in transient states (Isermann 1997, 640).

There are many different methods to achieve fault diagnosis. Figure 1 shows an outline of some of the different methods of fault diagnosis. The method of fault diagnosis used in this research is classification using a neural network, which will then be altered to
classification using a neuro-symbolic network. More information on neural networks and neuro-symbolic networks can be found in sections four and five of this chapter.

Most current diagnostic systems are driven by direct sensory information without much concern for patterns of system behavior or features associated with them. For large-scale systems with complex dynamics, global as well as local diagnostics become of great importance, where sensory information is used as input for the local diagnostics, and patterns of behavior or features are utilized for global diagnostics.

### 2.2 Feature-Based Fault Diagnostics

Feature-based fault diagnostics (FBFD) is a new (within the last ten years) area of research within fault diagnostics. Research has been done in the area of feature-based fault diagnostics by (Huang et al. 2004; Pechenizkiy, Tsymbal, and Puuronen 2004).
Instead of using direct sensory information, as most fault diagnostic systems do, FBFD uses features, or patterns of behavior. The use of features, instead of direct sensory information, is becoming even more important as systems using fault diagnostics are getting larger in scale.

Some of the earliest projects in this area were completed in the early 2000’s by Impact Technologies, LLC and the Air Force Research Lab (AFRL). Within a few years they developed many systems that used feature-based fault diagnostics. The main systems were gas turbine engines, engine test cells, and gas turbine engine bearings.

In 2001, Impact Tech and the AFRL developed “improved test cell diagnostics capable of detecting and classifying engine mechanical and performance faults as well as instrumentation problems” (Roemer et al. 2001, 2915). The goal of this project was to lessen engine operating and maintenance costs and to optimize the effectiveness of test cells. They developed a system “of advanced diagnostic and troubleshooting tools” (Roemer et al. 2001, 2915), that were implemented on engine test cells. This system was developed as part of the Automated Jet Engine Test Strategy (AGETS) program, which is run by the US Air Force (USAF).

This system is interesting because it has anomaly detection and diagnostic software that was implemented in both real-time and with a mode to analyze the data afterwards. There are two main diagnostic parts to this system: the engine vibration diagnostics and the engine performance diagnostics. The engine vibration diagnostics acquire their data from “two accelerometer-based transducers” (Roemer et al. 2001, 2919) placed inside the test cell. Vibrations come from different parts of the test cell,
“such as engine core rotor, fan rotor, hydraulic pump, etc.” (Roemer et al. 2001, 2919).

The transducers record these vibrations and change them to electrical signals.

Using this data, “feature-based diagnostic techniques” (Roemer et al. 2001, 2919) can produce a “real-time assessment of mechanical faults (i.e. bearing, rotordynamic, and structural)” (Roemer et al. 2001, 2919). The features used in this system are: engine orders, subharmonics, sidebands, fixed frequencies, resonances, jump ups/kick downs, and noise floor. If the vibration data has multiple traces of these features then these traces are added up to determine what is wrong. Faults that have been diagnosed in this manner include: unbalance, shaft interaction, eccentricity, squeeze film malfunction, blade rush, rotor instability, oil in rotor, flange/joint slip, looseness, misalignment, and swashed track (Roemer et al. 2001, 2920). This project was updated in 2004 with another article, which described the same system in a bit more detail, as well as the improvements that had been made. The article also included the math behind how the diagnostic system worked, as well as more and better results (Modgil, Orsagh, and Roemer 2004).

Another study was done by Impact Technologies, LLC and the AFRL, in 2003, to create “Prognostics/Diagnostics for Gas Turbine Engine Bearings.” The goal was to increase “aircraft engine reliability and maintainability … [by] predicting and detecting bearing and gear failures” (Orsagh, Sheldon, and Klenke 2003, 1). Existing airplane maintenance is based on very conservative life estimates; so most parts are replaced long before they are near the end of their useful life. This process creates waste and cannot account for unpredictable circumstances, such as total failure during the early stages of the useful life (Orsagh, Sheldon, and Klenke 2003, 1).
In order to create a robust in-flight prognostic and diagnostic system for oil-wetted components, Orsagh, Sheldon, and Klenke used “existing … technologies such as advanced oil debris/condition monitors, high/low frequency vibration analysis, thermal trend analysis and empirical/physics-based modeling” (Orsagh, Sheldon, and Klenke 2003, 2). The data from these sources is then “combined in model-based and feature-based prognostic integration algorithms for the specific bearing under investigation” (Orsagh, Sheldon, and Klenke 2003, 2); historical data on the bearing is also included. In the end, all of this information plus the remaining useful life and associated risk is sent to operations and maintenance. Figure 2 shows a block diagram of the total “integrated diagnostic and prognostic capability” (Orsagh, Sheldon, and Klenke 2003, 2). At the time that this article was released the system had not yet been tested.

Figure 2. Diagram of diagnostic/prognostic system. Source: Data adapted from: Orsagh, et al. 2003.
One of the most promising developments in feature-based diagnostics was completed by Han, Yang, and Yin in 2007. In their study they discuss a system that they developed to diagnose faults in induction motors using vibration signals and feature-based diagnosis. They proposed and developed a “fault diagnosis system … for induction motors based on feature recognition through combination of feature extraction, genetic algorithm and neural network techniques” (Han, Yang and Yin 2007, 163). The goal of this project was to find a way to use vibration signals to diagnose faults in induction motors. Originally the plan was to use artificial neural networks (ANNs) because they are effective at finding connections amid large amounts of data. Also, promising results have developed from research that has been done with ANNs used for fault diagnosis. However, with ANNs, it is difficult to determine the best inputs and parameters in order to make the network compact and highly accurate (Han, Yang and Yin 2007, 164).

In order to get a compact and accurate system, Han, Yang, and Yin used feature extraction and calculation, genetic algorithms (GAs), and ANNs. In this system, the group collected vibration signals from test induction motors using accelerometers, then the features were extracted (calculated) from this data using statistical parameters. Next, in order to have a compact system, a genetic algorithm was used “as a feature selector and network optimizer” (Han, Yang and Yin 2007, 165); this process picked the features which could provide the most important information for the ANN inputs. This information was then fed into a neural network to determine if the induction motor had any faults. Figure 3 shows the architecture of this diagnostic system. The ANN that was
used is called the ART-Kohonen neural network (ART-KNN) created by Yang et al. (Yang et al. 2004).

![Block diagram of the diagnostic system. Source: Data adapted from: Han, Yang and Yin 2007.](image)

This system was tested in an experiment using a test rig composed of a motor, pulleys, a belt, a shaft, and a fan with changeable pitch blades. This fan was used as the load on the motor. Each motor used was a 0.5 kW, 60 Hz, 4-pole induction motor. One motor was normal while the other six motors were faulty. The six kinds of motor faults used were: broken rotor bar, bowed rotor, bearing outer race fault, rotor unbalance, phase unbalance, and adjustable eccentricity motor (misalignment).

Three AC current probes and three accelerometers were used to measure the stator current of three phase power supply and vibration signals of horizontal, vertical and axial directions for evaluating the fault diagnosis system (Han, Yang and Yin 2007, 163-75).
Using this method, they collected 16384 data samples, which was too much for
the neural network. In order to reduce the number of data samples, they put this data
through a GA for feature selection, and compared the results of the vibration features and
current features with GA to the features without GA. They found that the use of vibration
signals with GA was the best system.

After this experiment, the system was tested in a real environment. They tested
four motors on a student training ship. The motors were used for: cooling waste pump,
lubricating oil pump, cooling sea water pump, and mooring hydraulic pump. In this test,
vibration signals were measured by accelerometers in three directions. In order to check
for results, manual data analysis was completed. This data analysis matched the proposed
system, which showed that this system can be used for fault diagnosis of induction
motors. The main differences between Han, Yang, and Yin’s work and the work
completed for this project are; the project will be implementing the system on an entire
airplane instead of a small part, such as a motor, this project’s system will not use GA,
and this projects system will use a neuro-symbolic network which is a different kind of
ANN than what was used by Han, Yang, and Yin.

Yang then took the feature-based diagnostic research from the previous study and
used it in another area of research, Support Vector Machines. He wrote this study on,
“Support Vector Machine in Machine Condition Monitoring and Fault Diagnosis.” The
purpose of this research was to use a support vector machine (SVM) for machine
condition monitoring and fault diagnosis as part of a maintenance system in order to
reduce maintenance costs, improve productivity, and increase machine availability. In
this article the authors discuss the many different researchers who have used features for fault diagnosis and how each was different (Widodo and Yang 2007, 2560-74). Some of these will be discussed below.

A more recent study was done by Xu et al in 2010. It was a “Study of Fault Diagnosis Based on Probabilistic Neural Network for Turbine Generator Unit.” In this study the group was looking at ways to create a fault diagnosis system for turbine generator units because they play an important role in electricity production. They were doing this project because turbine generator unit faults have been occurring frequently, and these faults affect the safety and stability of operation, cause significant economic losses, can destroy the machine, and can cause casualties. There are two main ways to do fault diagnosis: model-based methods and feature-based methods. Model-based methods include parameter estimation, while feature-based methods include expert systems and neural networks (Xu et al. 2010.).

In this study Xu et al. used a probabilistic neural network (PNN), which is a four-layer neural network model, based on Parzen windows classifier and its application to Bayes statistics. The features they used for this model were the frequency of the vibrations from the turbine generator units, which were divided into six different frequency ranges. Next, the team chose three kinds of common turbine generator unit faults: oil-membrane oscillation, unbalance, and no orderliness. They used some of the data to train the PNN and then tested the network with the other data. They showed that this system can effectively diagnose the vibration faults of turbine generators.
In this literature review the current and new systems of feature-based diagnostics were summarized and discussed. As was shown, feature-based fault diagnostics is a new field and little research has been done in the area. In this project, a fully functioning feature-based fault diagnostic system has been developed. The main differences between the systems in these articles and the one being developed in this research are: this research focuses on a large scale system, e.g. an entire aircraft instead of a single motor or other part, this research is able to decide whether the aircraft is damaged, what area of the aircraft is damaged, and how badly it is damaged. This research utilizes a neuro-symbolic network, and is based on the physical system of the momentum equation for inertial control volume.

2.3 Feature Extraction

Another key component of a feature-based diagnostic system is feature extraction. Feature extraction is required to reduce the amount of on line data processing required for a diagnostic neural network. Most of such data contains either no useful information or repetitive data that is already included in another feature. Feature extraction is done to identify the features that have no data or repetitive data, which then leaves a smaller group of features that can more effectively be put through a neural network.

In 2000, Jin and Shi developed a new method for feature extraction. They explained that in most feature extraction techniques, the interaction between variables has been ignored, so they developed a new way to do feature extraction that takes into consideration these interactions. They created this new system to monitor stamping
process control. Many groups had determined processes to diagnose faults in the stamping process; however because the stamping process is so complicated, previous systems were never able to determine what was wrong and causing the change in the tonnage signal. This new system considers the many different variable interactions and because of this, can find the root cause of the problem (Jin and Shi 2000, 360).

This new diagnostic feature-extraction methodology was completed using the fractional factorial design of experiments. This system includes three steps. “In the first step, the important process variables are determined and the experimental design is made based on the complexity of the relationship between the process variables and the tonnage signals” (Jin and Shi 2000, 360). In this example the important process variables were: lubrication, material thickness, outer shut-height, inner shut-height, punch speed, and blank washer pressure. In the second step, principal component analysis (PCA) is employed in order to reduce the amount of data. In the final step, the relationship between the process variables and the tonnage signal features is determined by the use of a regression model. The system they developed can also be used on any other processes that have waveform signals. This shows that the system they designed has a broad range of applications in feature extraction and diagnostic system development. Much research has been done in the area of feature extraction (including SVM) for this project; however one of the articles was not directly used in this section (Byun and Lee 2002).
2.4 Artificial Neural Networks

An artificial neural network (ANN) or neural network (NN) is a mathematical or computational model designed to mimic the function of a biological neural network (BNN) or the brain. Much research has been done in the area of NN’s for this project; however a few of the articles were not directly used in this section. These articles include: (Byington, Watson, and Edwards 2004; Meireles, Almeida, and Simoes 2003). Neural networks are used to find complex relationships between data and for pattern recognition. A neural network is a grouping of many artificial neurons, which together operate to solve a problem. These artificial neurons are based on the structure and operation of the biological neurons in the brain. Figure 4 shows a schematic drawing of a biological neuron.

![Diagram of biological neurons. Source: Data adapted from: Hagan, Demuth, and Beale 1996.](image-url)
Biological neurons make up most of the structure of the brain, and it is known that the connections between the neurons are what make thoughts and memories. It is estimated that the human brain has approximately 100 million neurons and that each of these neurons has approximately 1,000 to 10,000 connections (Hagan, Demuth, and Beale 1996, Ch.1).

A biological neuron has three main parts: the dendrites, the cell body, and the axon. Information is brought into the neuron through the dendrites, the cell body then computes all of the inputs and decides whether to take any action or not based on thresholds. If the threshold is reached the cell body sends out a signal, which travels down the axon to the axon terminals. The axon terminals then connect to a dendrite on another neuron and pass the signal on through a synapse (Hagan, Demuth, and Beale 1996, Ch.1). A synapse is the connection between an axon terminal of one neuron and a dendrite of another neuron; this is where the information or signal is passed on.

An artificial neural network is designed to act in a similar method. Figure 5 shows a diagram of a basic single-input neuron. In a single-input neuron the input (p) is multiplied by the weight (w). This and the bias (b) are then sent to the summer (Σ). This summed total is then put through an activation/transfer function (f), which produces the output (a) (Hagan, Demuth, and Beale 1996, Ch.2).
Equation 1 is the equation for the output of a single-input neuron (Hagan, Demuth, and Beale 1996, Ch.2). This equation becomes more complicated as more neurons are combined to create a neural network.

\[ a = f(wp + b) \]  

In order to create a neural network, many neurons are connected. There are also multi-input neurons, where many inputs are brought into a single neuron and these are all multiplied by their respective weights before being summed together. As a problem becomes more complicated, a network can include more than one layer of neurons, and each layer can have many neurons as well. In this case, the outputs of the first layer become the inputs to the second layer, and this continues until the output layer is reached (Hagan, Demuth, and Beale 1996, Ch.2).

The type of neural network used in this research is a feed-forward backpropagation neural network. A feed-forward neural network (FFNN) is a simple NN
where information is only fed forward through the network; there is no feedback and no loops to go back. Figure 6 shows a diagram of a simple feed-forward neural network.

![Diagram of a feed-forward neural network.](image)

Figure 6. Diagram of a feed-forward neural network.

A feed-forward neural network has an input layer, an output layer, and one or more hidden layers. The number of input and output neurons is decided by how many inputs and outputs a network has, while the number of hidden layers and the number of neurons in each hidden layer is decided by the designer of the network and are optimized to obtain the best solution.

A feed-forward backpropagation neural network uses backpropagation as the training method. This method is a supervised learning method. It calculates the error between the actual output of the network and the desired output of the network, and then alters the network weights to reduce the error (Remelhart, Hinton, and Williams 1986, 533). This is one of the most common training methods used with FFNNs.
2.5 **Neuro-Symbolic Networks**

Neuro-symbolic networks or neural-symbolic networks (NSN) have become a very active area of research in the last 10 years. Much research has been done in the area of NSN’s for this project; however a few of the articles were not directly used in this section. These articles include: (Deuker, Perrier, and Amy 1998; Grasso et al. 2010; Kamruzzaman and Islam 2005; Melin and Castillo 2005; Qadeer et al. 2009; Wichert 2011). The basis of NSNs is in artificial neural networks. Stefan Wermter and Ron Sun are current experts in the field of hybrid neural systems (HNS), which is a larger field including NSNs. Wermter and Sun believe that, “hybrid neural systems are computational systems which are based mainly on artificial neural networks but also allow a symbolic interpretation or interaction with symbolic components” (Wermter and Sun 2000, 1).

There are two main motivations for researching hybrid neural systems. One, if networks are supposed to mimic the function of the brain, then a symbolic section is necessary because “the brain has not only a neuronal structure but has the capability to perform symbolic reasoning” (Wermter and Sun 2000, 1). Two, is the understanding that neural networks and symbolic networks have strengths and weaknesses that are almost exactly opposite, so that when they are combined, the strengths of each network cancel out the corresponding weaknesses in the other network. Table 1 shows the different strengths of neural and symbolic networks.

<table>
<thead>
<tr>
<th>Neural Networks</th>
<th>Symbolic Networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradual analog plausibility</td>
<td>Easy interpretation</td>
</tr>
<tr>
<td>Ability to learn</td>
<td>Explicit control</td>
</tr>
<tr>
<td>Robust fault-tolerant processing</td>
<td>Fast initial coding</td>
</tr>
<tr>
<td>Ability to generalize</td>
<td>Dynamic variable binding</td>
</tr>
<tr>
<td></td>
<td>Knowledge abstraction</td>
</tr>
</tbody>
</table>

There are many types of hybrid neural systems; however this research has focused on two types: knowledge-based systems and neuro-symbol based systems. Parts of these two methods have been combined to create the method used in this research. Knowledge-based systems are neural networks with some form of prior symbolic knowledge added to them. Neuro-symbol systems are very similar to neural networks, except that they use neuro-symbols, instead of neurons as the basic processing unit.

Knowledge-based systems got started when Geoffrey Towell and Jude Shavlik created KBANN (Knowledge-Based Artificial Neural Network).

KBANN is a hybrid learning system built on top of connectionist learning techniques. It maps problem-specific “domain theories”, represented in propositional logic, into neural networks and then refines this reformulated knowledge using backpropagation. …

… the idea is to insert a set of hand-constructed, symbolic rules (i.e., a hand-built classifier) into a neural network. The network is then refined using standard neural learning algorithms and a set of classified training examples. (Towell and Shavlik 1994, 119-20)
Towell and Shavlik proved that this network worked better than many other systems, including basic neural networks. Figure 7 shows the flow chart of the KBANN system. There are two algorithms that make up the system. The first, rules to network, “inserts approximately-correct, symbolic rules into a neural network” (Towell and Shavlik 1994, 123). The second, neural learning, “refines networks using the backpropagation learning algorithm” (Towell and Shavlik 1994, 123).

![Figure 7. Flow chart of KBANN. Source: Data adapted from: Towell and Shavlik 1994.](image)

This model worked so well that it was the inspiration for another system, CLIP (Connectionist Inductive Learning and Logic Programming).

CLIP builds upon KBANN so as to provide a sound theoretical foundation for reasoning in artificial neural networks, … CLIP is a massively parallel computational model based on a feedforward artificial neural network that integrates inductive learning from examples and background knowledge with deductive reasoning using logic programming. (d’Avila Garcez, Lamb, and Gabbay 2009, 35)
CLIP builds upon KBANN, by adding feature extraction and logic programming. Both of these methods are excellent to insert symbolic information into a neural network. CLIP is also very accurate, with an accuracy rate of 97.2% (d’Avila Garcez, Lamb, and Gabbay 2009, 54). While very accurate, these types of systems might benefit from the inclusion of principles from other types of methods.

Another type of NSN is one based not on artificial neurons, but neuro-symbols. This work was done by Rosemarie Velik and many of her colleagues. Velik wondered what would happen if, instead of inserting symbolic information into a neural network, the symbolic information would be built into the network, creating a neuro-symbolic network using neuro-symbols as the basic information processing unit. Neuro-symbols are similar to artificial neurons, in that they are designed from biological neurons. They are different because the “neuro-symbols combine characteristics of neural as well as symbolic information processing” (Velik and Bruckner 2008, 1042). These neuro-symbols are then connected to create a neuro-symbolic network.

This type of network has been tested by Velik and Bruckner, and they found that the network worked very well for perceptual tasks. In this case they tested the network by using it to determine when a person is present or not present in a room. It operated very well and could determine correctly if the room was occupied or not (Velik and Bruckner 2008, 1047).

Velik originally built the network for perceptive tasks and as such, she used the organization of the brains perceptual system as the basis for the organization of her network. Figure 8 shows the organization of the human perceptual system. On the bottom
layer are the receptors or sensors, which pass the sensory information to the primary cortex. In the primary cortex, simple features are extracted from the sensory information. These features are then passed to the secondary cortex, where the information is combined to create a unified perception for each of the senses.

As can been seen, each of the sensory modalities processes information separately in the first two cortices. Then, in the tertiary cortex, the information from each sensory system is combined to create a “unified multimodal perception” (Velik and Bruckner 2008, 1043).

This kind of organization is similar to the organization of the entire MURI project. Other teams on the project are creating sensors, which will act as the receptors. Each of these types of sensors will also have a local diagnostic system, which will act as the primary and secondary cortex. Then the global feature-based neuro-symbolic diagnostic network will act as the tertiary cortex. The global diagnostic network will
collect features from all of the local diagnostic systems, combine all of the features, and
determine the diagnostic state of the system.
CHAPTER THREE: ANALYSIS AND DESIGN

The purpose of this chapter is to discuss in detail the momentum equation, why it was chosen, and how it is used, as well as the radio controlled (RC) aircraft being used. This chapter also describes the computational fluid dynamics (CFD) optimization and simulation, which were used to determine the features and the best locations for the sensors; as well as the validation of the grid used in these simulations. The CFD simulation was used to determine the airflow over the airfoil. This information was crucial to determining the features to be used in the feature-based neuro-symbolic networks. Also discussed are the sensors and data acquisition systems used on the RC aircraft for real data collection.

3.1 Momentum Equation for Inertial Control Volume

In order to create a feature-based neuro-symbolic network that was based on a physical principle, an equation was chosen that is most relevant to the system. There were two main equations that could have been used for this purpose: the momentum equation for inertial control volume equation and the Navier-Stokes equation. The Navier-Stokes equation is shown in its general form as equation 2, where i equals the directions x, y, or z and j equals the directional velocities u, v, or w based on the current Cartesian coordinate (Fox, McDonald, and Pritchard 2006, 213-215).
The momentum equation for inertial control volume is shown as equation 3. This equation was chosen over the Navier-Stokes equation for many reasons. The first reason is that the momentum equation is used for macro problems while the Navier-Stokes equation is used for micro problems. The second reason is that the Navier-Stokes equation is used in many situations because it compensates for the differences that occur in airflow over an airfoil at high angles of attack: fluid separation and the viscosity effect. The simulations that were conducted in this research only used an angle of attack of between -1° and 5°. These angles of attack are not high enough for fluid separation and the viscosity effect to affect the solution.

\[
\rho \left( \frac{\partial j}{\partial t} + u \frac{\partial j}{\partial x} + v \frac{\partial j}{\partial y} + w \frac{\partial j}{\partial z} \right) = \rho g_i - \frac{\partial p}{\partial i} + \mu \left( \frac{\partial^2 j}{\partial x^2} + \frac{\partial^2 j}{\partial y^2} + \frac{\partial^2 j}{\partial z^2} \right) \tag{2}
\]

\[
\bar{F} = \bar{F}_S + \bar{F}_B = \frac{\partial}{\partial t} \int_{cv} \bar{V} \rho dV + \int_{cs} \bar{V} \rho \bar{V} \cdot d\bar{A} \tag{3}
\]

The momentum equation states that the total force on the system is equal to the sum of the surface forces and body forces, which is equal to the partial derivative with respect to time of the integral over the control volume of the velocity, air density, and change in volume, summed with the integral over the control surface of the velocity, air density, velocity, and change in area. In this case, the control volume is the entire volume of the aircraft, and the control surface is the area of the surface of the wing (Fox, McDonald, and Pritchard 2006, 112-14).
The momentum equation can also be separated into its x, y, and z components. Equations 4 and 5 show the x and z components of equation 3. The y component equation is not shown, since the y component adds complexity that is beyond the scope of this research. There are four main forces on an aircraft: lift, gravity, thrust, and drag. All of these main forces act in the x and z directions, which is another reason for ignoring the y component.

\[ \vec{F}_x = \vec{F}_{sx} + \vec{F}_{bx} = \frac{\partial}{\partial t} \int_{CV} u \rho dV + \int_{CS} u \rho \vec{V} \cdot d\vec{A} \]  
\[ \text{(4)} \]

\[ \vec{F}_z = \vec{F}_{sz} + \vec{F}_{bz} = \frac{\partial}{\partial t} \int_{CV} w \rho dV + \int_{CS} w \rho \vec{V} \cdot d\vec{A} \]  
\[ \text{(5)} \]

As can be seen in equations 4 and 5, the sum of the forces is replaced with the sum of the forces in the x and z directions respectively. Also the first velocity term in each integral has been changed from the total velocity (\( \vec{V} \)) to the velocity in the x and z directions (\( \vec{u} \) and \( \vec{w} \)) respectively.

The momentum equation is the sum of two terms. The two terms can then be independently broken down by their basic units to determine their meaning. This process can be seen in equation 6 for part one, and equation 7 for part two of the equations. As can be seen in equation 6, part one of the equation can be broken down to: the product of
the velocity, air density, and the volume, divided by the time. This then reduces to:
kilograms times meters divided by seconds squared, which is equal to a Newton or Force.

\[
\frac{\partial}{\partial t} \int_C \vec{V} \rho dV = \frac{\vec{V} \rho V}{t} = \frac{m k g}{s m^3} = \frac{k g \cdot m}{s^2} = N = F
\]  

(6)

As can be seen in equation 7, part two of the equation can be broken down to: the
product of the velocity, air density, velocity, and area. This then reduces to kilograms
divided by meters times seconds squared, all times meters squared. This is equal to a
Pascal times meters squared, or pressure time area, which is equal to force. Because this
part of the equation can be reduced to the pressure over the area of the wing, pressure was
chosen as the main variable to be used in the data collection as well as the neural and
neuro-symbolic networks.

\[
\int_{C_s} \vec{V} \rho \vec{V} \cdot d\vec{A} = \vec{V} \rho \vec{V} \cdot \vec{A} = \frac{m k g}{s m^3} \cdot \frac{m}{s} \cdot \frac{m^2}{s^2} = \frac{k g}{m \cdot s^2} \cdot m^2 = Pa \cdot m^2 = P \cdot A = F
\]  

(7)

Both equations 6 and 7 show that the right side of the momentum equation
reduces down to a force, which is equal to the left side or the total sum of forces. This
momentum equation for inertial control volume was used to design the neuro-symbolic
network, which will be discussed more in Chapter 4. It was also used to determine which
properties to solve for in the CFD simulation, and which properties to collect in the
testbed data collection, which will be discussed in the next two sections.
3.2 Testbed Aircraft

In order to test this research in a close to reality situation, a radio controlled (RC) airplane was chosen. The Hobbico Hobbistar 60 Select RTF 71” is the RC airplane currently in use for this research (see Figure 9).

![Figure 9. Picture of the Hobbico Hobbistar 60 Select testbed.](image)

This RC model was selected because it is a beginner/trainer aircraft, as well as being very simple and quick to assemble. Also it is made entirely of balsa and plywood, so it is lightweight, and parts are easy to replace. The wing consists of two half wings that are screwed together. It has a 71” wingspan and the wing is mounted to the fuselage with rubber bands. The covering of the aircraft is a factory applied adhesive backed covering. The engine is a pre-installed O.S. Max .65 LA sport with a muffler and a glow plug. The radio used is a Futaba 6EXAP with four servos. The plane has wire tricycle style landing
gear with three 2.75” diameter foam wheels. It also uses 5% - 15% nitro model airplane fuel.

The specifications of the model aircraft include:

- Wingspan: 71”
- Wing Area: 888 in²
- Wing Chord: 12.5”
- Weight: 7.5 – 8.5 lbs.
- Wing Loading: 20 – 22 oz. /sq. ft.
- Length: 55”
- Width: 3-3/8”
- Airfoil: Semi-symmetrical, high-wing
- Center of Gravity: 3” back from the wing’s leading edge

This RC model aircraft was then instrumented and flown multiple times to obtain data required for this research. Section 3.4 discusses in detail the instrumentation that was used on the model.

3.3 Computational Fluid Dynamics Model

In order to determine the features for the neural and neuro-symbolic networks, a Computational Fluid Dynamics (CFD) simulation was completed to determine the pressure distribution and contours around the aircraft wing. As described in section 3.1, pressure was chosen as the variable because the momentum equation is equal to the sum of the forces on the aircraft, which is equal to the sum of the pressures over the area of
the wing. Also because the sum of the pressures over the area of the wing is equal to the forces on the wing, the CFD simulation was used to determine the forces on the wing.

Before completing the CFD simulation, a computer model had to be built. This model was built in Gambit. It was decided that a 2D model of the aircraft wing would be made and that the model should look as close to the real testbed wing as possible. In order to do this, the specifications of the testbed wing were used to find a NACA (National Advisory Committee for Aeronautics) wing that closely matched. The airfoil on the testbed is semi-symmetrical, so four NACA semi-symmetrical airfoils were compared to the testbed airfoil. It was determined that the NACA 2412 was the closest match to the testbed airfoil. The coordinates for the NACA 2412 airfoil that were used to build the model can be found in Appendix A. Figure 10 shows a 2D cross-section of the testbed Hobbistar airfoil. Figure 11 shows the four 2D cross-sections of the NACA semi-symmetrical airfoils that were compared with the testbed airfoil.

![Figure 10. Cross-section of the Hobbistar airfoil. Source: Figure adapted from: Hobbico Hobbistar.](image)

Next the airfoil model was built in Gambit. The coordinates for the NACA 2412 that can be found in Appendix A are between zero and one on the x-axis. Figure 12 shows the airfoil model that was created in Gambit and a small part of the grid around it.
Figure 11. Cross-section of 4 semi-symmetrical airfoils. *Source:* Figure adapted from: FlyingFoam.com.

Figure 12. Figure showing Gambit airfoil model.
In order to run a CFD simulation on the model, a grid had to be built around the airfoil model. The airfoil goes from (0, 0) on the x-axis to (1, 0), so the grid was built from -11.5 to 21 on the x-axis and from -12.5 to 12.5 on the y-axis, with the part in front of the airfoil making a semi-circle and the part behind the airfoil making a rectangle. Figure 13 shows the major vertices created in Gambit to create the grid around the airfoil. The actual airfoil is shown in the center where the large white spot is between (0, 0) and (1, 0).

![Figure 13. Figure showing major vertices in Gambit airfoil grid.](image)

After these vertices were created, they were connected by edges to create faces. Figure 14 shows these edges and faces. There were four faces created: the airfoil, the front semi-circle, and the two rectangles behind the airfoil. After the four faces were
created, they needed to be combined to create just one face. In order to do this the airfoil face was Boolean subtracted from the Boolean addition of the other three faces. Each edge within this one face was then meshed separately.

Figure 14. Figure showing major faces and lettered edges in Gambit airfoil grid.

Figure 14 also shows the lettered edges in the grid. Edges A, B, and C were meshed with 160 intervals and a successive ratio of 1.05 and the smaller sections on the left side. Edges D, E, F, and G were meshed with 160 intervals and a successive ratio of 1.05 and the smaller sections toward the middle. Edges H and I were meshed with 214 intervals and a successive ratio of 1.05 with the smaller sections on the right side; these also matched up with the mesh on the airfoil edges. Figure 15 shows the final meshed grid.
The Gambit journal file for this mesh can be found in Appendix B. This model was then exported in a 2D mesh file, which could then be opened in Fluent.

### 3.4 Grid Validation

In order to ensure that the simulation was solved accurately, grid validation was completed. The plan was to compare the results from the grid to the NASA airfoil simulator, FoilSim III; however the NASA airfoil simulator does not have an airfoil shape that is close to the model used in this project. So, instead a new grid was created with a mesh half the size as the original grid. The results of maximum and minimum pressure, and the x and y force were compared between the original grid and the grid that was half the size. The results from this comparison can be seen in Table 2.
Table 2. Comparison between results of the original grid and the validation grid.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>15.2005</td>
<td>15.3274</td>
<td>-8.6138</td>
<td>-8.6169</td>
<td>0.1399</td>
<td>0.1377</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>9.5441</td>
<td>9.1883</td>
<td>-5.0124</td>
<td>-4.8575</td>
<td>0.4551</td>
<td>0.4523</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>13.2574</td>
<td>13.1943</td>
<td>-7.8845</td>
<td>-8.0206</td>
<td>0.9533</td>
<td>0.9556</td>
</tr>
<tr>
<td>40</td>
<td>2</td>
<td>25.4244</td>
<td>23.0701</td>
<td>-14.0685</td>
<td>-14.5678</td>
<td>1.6214</td>
<td>1.5322</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>36.4865</td>
<td>36.5849</td>
<td>-21.9427</td>
<td>-22.4325</td>
<td>2.4583</td>
<td>2.2671</td>
</tr>
<tr>
<td>60</td>
<td>2</td>
<td>523.2882</td>
<td>519.9829</td>
<td>-316.0842</td>
<td>-319.1879</td>
<td>3.4501</td>
<td>3.5050</td>
</tr>
</tbody>
</table>

As can be seen in Table 2 the results from the validation grid are very similar to the results from the original grid. These results show that using smaller mesh elements, which requires more computing power and time, yields minimal improvement in the accuracy of the results. Therefore the grid used in this research is adequate to achieve accurate results.

### 3.5 Computational Fluid Dynamics Optimization

In order to determine the optimal positions to place the sensors on the aircraft wing, a CFD optimization was completed. The plan was to find the minimum and maximum pressures at four different locations on the airfoil: the front, back, top, and bottom. This would enable the determination of lift and drag on the wing. The optimization was needed because at different angles of attack the location of the minimum and maximum pressures changes. The optimization was completed on the top pressure sensor.

In order to complete this optimization, a Fluent journal file was built and two Matlab programs were developed. The code for these programs can be found in Appendix
C. These programs were built to determine the location of the minimum pressure on the top of the airfoil, for different angles of attack.

To do this the program starts at a certain location and finds the pressure at that location, this is then compared with the minimum pressure. The “fmincon” function in Matlab is used to minimize the difference between the pressure at the current location and the minimum pressure. When this value is minimized the location of the minimum pressure is located.

Optimization was completed for the following angles of attack: -1°, 0°, 1°, 2°, 5°, and 10°. Figure 16 shows the location of the minimum pressure, as a distance from the front of the airfoil, versus the angle of attack. It shows how as the angle of attack increases, the distance from the front of the airfoil to the minimum pressure decreases.

Figure 16. Figure showing the location of the minimum pressure versus the angle of attack.
A similar process was repeated for the other pressure sensor locations. Because of this optimization, the sensors have been placed in the following locations:

- Top Sensor – Located 2.0079” back from the front of the airfoil.
- Front Sensor – Located 0.3” back from the front of the airfoil on the bottom of the airfoil.
- Rear Sensor – Located 0.2559” forward from the back of the airfoil on the bottom of the airfoil.
- Bottom Sensor – Located 3.374” back from the front of the airfoil.

Figure 17 shows these four sensor locations on the airfoil. These four pressure sensor locations were used both in the simulation, as well as on the actual testbed during data collection.

![Figure 17](image)

Figure 17. Figure showing the location of the four pressure sensors on the airfoil.

### 3.6 Computational Fluid Dynamics Simulation

After determining the placement of the four sensors, CFD simulations were run in order to determine the pressures at the four different sensor locations, as well as the forces on the airfoil. These simulations were completed for different airspeeds and angles.
of attack. The airspeeds used were: 10 knots, 20 knots, 30 knots, 40 knots, 50 knots, and 60 knots. The angles of attacks used were: -1°, 0°, 1°, 2°, and 5°.

In order to complete these 30 simulations, a Fluent journal file was built which was run by a Matlab program. This journal file and Matlab program can be found in Appendix D. The Matlab program was designed to run through the simulations for all of the airspeeds and angles of attack. The fluent journal file was designed to print out the following data from each simulation: the minimum and maximum pressures, the pressures at the four sensor locations, and the forces in the x and y axes.

3.7 Data Collection

A testbed was developed in order to test the global diagnostic system by collecting data that is used as the input to the neuro-symbolic network. This testbed was described in detail in section 3.2. In this section, the implementation of sensors and sensor recorders on the aircraft will be discussed.

In order to record the data that was needed, two data acquisition systems were installed on the testbed aircraft. One system was built in-house by Scott Ferdinand, which records pressure and temperature from four sensors, as well as acceleration from a 3-axis accelerometer. The second system, the Ruby, was purchased from uThere, LLC.

Figure 18 shows a photo of the data acquisition system built in-house. Figure 19 shows a block diagram of this same data acquisition system.
Figure 18. Photo of pressure data logger.

Figure 19. Block diagram of pressure data logger. Source: Figure by Scott Ferdinand.
There are a total of six sensors which record air pressure and temperature, and can be attached to the microcontroller; however only four were used. These four sensors were attached to the wing in the same locations as the sampling points in the CFD simulation. Figure 20 shows a photo of the RC aircraft wing upright and the location of the top sensor. Figure 21 shows a photo of the RC aircraft wing upside-down and the location of the front, bottom, and rear sensors.

Each sensor was placed facing forward and perpendicular to the surface of the wing. To do this, the sensors were pushed through holes in the wing surface, and small triangular pieces of wood were glued behind the sensors in order to keep them upright.
Figure 22 shows a close up of one of these sensors and how it was attached. The air pressure and temperature sensors that were used are the Bosch BMP085 digital pressure sensors. Figure 23 shows a photo of one of these pressure and temperature sensors. Appendix E includes features and electrical characteristics for this sensor.

![Figure 22. Close-up photo of the top sensor and how it is attached to the wing.](image1)

![Figure 23. Photo of pressure sensor.](image2)
Each of the four sensors is attached to the microcontroller with four wires. Two of these wires supply power to the sensor, while the other two wires are used to read the digital data from the sensor. In order to determine the pressure and temperature from the sensors, first the data must be changed from hexadecimal to decimal numbers, and then the eleven coefficient data values for each sensor are combined with the temperature and pressure readings to determine the real pressure and temperature. A detailed diagram of this process with equations can be found in Appendix E.

Other than the sensors, the following are also attached to the microcontroller: lights and switches, a 3-axis accelerometer, a serial port interface, and flash memory. These are all powered by a 7.4 volt battery. The 3-axis accelerometer that was used for this data acquisition system was the ADXL330 3-axis accelerometer built by Analog Devices. Appendix E also includes features, specifications, and a functional block diagram for this sensor.

The other data acquisition system which was used is the Ruby built by uThere, LLC. This system includes a Pitot tube for airspeed measurements, an altimeter, an inertial measurement unit, a Global Positioning System (GPS), and a control input recorder. This system is still in beta testing, so documentation is unavailable. Figure 24 shows a photo of the Ruby system installed into the testbed aircraft.
Figure 24. Ruby system by uThere.
CHAPTER FOUR: DESIGN OF NETWORKS

The purpose of this chapter is to describe the design of both the feed-forward neural networks and the feature-based neuro-symbolic networks. In all, a total of nine networks were created, three neural networks and six neuro-symbolic networks, one neural network and two neuro-symbolic networks for each set of data. The neuro-symbolic networks were specifically designed to run faster and be more accurate than the neural networks. Also in this chapter is a description of how the Matlab code works for the networks.

4.1 Feed-Forward Neural Network

The feed-forward neural networks were designed to operate as very simple neural networks. The inputs are basic flight data such as velocity, angle of attack, phase of flight, and the pressures at the four locations on the wing. The outputs are basic information required for diagnostics: whether the system is normal or not, the location of the damage, and the severity of the damage. This creates a feed-forward neural network with seven inputs and three outputs.

The phase of flight is a variable that was created in order to allow the network to determine the typical pressures during a certain phase of flight. This would prevent the network from getting confused between correct pressures during a certain phase of flight and incorrect pressures during a different phase of flight. The phase of flight is defined
as: 1=Taxi, 2=Takeoff, 3=Climb, 4=Level Flight or Cruise, 5=Descent, 6=Flare, and 7=Landing. This set of numbers was used in the networks that were based on the CFD simulations; however for the actual testbed data only the data from level flight was used, so the phase of flight input was removed. Because of this the neural network was reduced to six inputs and three outputs.

Figure 25 shows a diagram of a feed-forward neural network used in this project. In this figure the network is shown with one hidden layer and six hidden neurons; however, the actual networks that were used had between 10 and 60 hidden neurons. This will be discussed in detail in Chapter 6.

Figure 25. Diagram of a feed-forward neural network.

The design of this neural network was then used to design, in combination with the momentum equation, the neuro-symbolic networks.
4.2 Feature-Based Neuro-Symbolic Network

The design of this neuro-symbolic network was based mainly on the momentum equation for inertial control volume, which was discussed in detail in Chapter 3. In order to use the equation, it was decided that the network would be separated into two neuro-symbolic networks, one each for the x and z axes. Because of this, the networks each have four inputs and four outputs. The four inputs are: the velocity in the respective direction (x or z), the air density, the pressure differential in the respective direction, and the phase of flight. The four outputs are: the total force in the respective direction, the normality (whether the system is in a normal state or not), the location of damage, and the severity of damage.

These inputs and outputs almost fully constitute the momentum equation. The only variables that are missing are the area of the wing surface and the volume of the aircraft. These are included in the neuro-symbolic network because they will be accounted for in the value of the weights that are calculated for the networks.

Figure 26 shows a diagram of two parallel feature-based neuro-symbolic networks that were used in this project. In this figure the networks are shown with one hidden layer and six hidden neurons each; however, the actual networks that were used had between 10 and 60 hidden neurons. This will be discussed in detail in Chapter 6.
Due to the smaller number of inputs for each neuro-symbolic network, as compared to the neural network, these networks should run faster.

4.3 Matlab Code for the Networks

In order to create, run, and test these neural and neuro-symbolic networks a code was created in Matlab, using the Neural Network Toolbox. A generic version of this code can be seen in Appendix F. This code loads the files with the inputs and outputs for the network into Matlab, initializes variables, and records the time using the “tic toc” function. It then randomly chooses which lines of data to use for training versus testing.
The network is then created with the number of hidden nodes that is input by the user, and then the network is trained.

Once the network is trained, it is tested using the randomly chosen test data. This test then outputs the results. For the neural networks, the three outputs are compared using confusion matrices. For the neuro-symbolic networks, the force output is compared using root mean square (RMS) error. Also the mean and standard deviation of the error is determined. Then, the other three outputs are compared using confusion matrices. Finally, the time and number of epochs (iterations) to convergence is reported. This process is then repeated for a set number of times (typically 100). The results from all of the networks are shown and discussed in detail in Chapter 6.
CHAPTER FIVE: SIMULATION RESULTS

This chapter includes results from the computational fluid dynamics (CFD) simulation and how these results were used to generate features for the networks. Also included are the results from the RC aircraft data collection.

5.1 Computational Fluid Dynamics Simulation Results

As was discussed in section 3.5, CFD simulations were completed to determine: the minimum and maximum pressures, the pressures at the four sensor locations, and the forces in the x and y axes for different airspeeds and angles of attack. The airspeeds used were: 10 knots, 20 knots, 30 knots, 40 knots, 50 knots, and 60 knots. The angles of attacks used were: -1°, 0°, 1°, 2°, and 5°.

Table 3 shows the results from this simulation. All pressures were recorded in gauge pressure. As can be seen, the pressures and forces increase in magnitude as the velocity increases. Also the maximum and rear pressures and the forces are always positive. The minimum, top, and bottom pressures are always negative, and the front pressure is mostly negative (except at high angles of attack). Also the force in the y-direction is always greater than the force in the x-direction.
Table 3. Results from CFD simulation.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>-1</td>
<td>14.506</td>
<td>-6.690</td>
<td>-6.3188</td>
<td>-5.3522</td>
<td>1.4475</td>
<td>-3.0573</td>
<td>0.1185</td>
<td>0.8611</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>14.974</td>
<td>-7.244</td>
<td>-7.1066</td>
<td>-3.3794</td>
<td>1.6522</td>
<td>-2.6135</td>
<td>0.1237</td>
<td>0.9017</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>14.183</td>
<td>-7.929</td>
<td>-7.8898</td>
<td>-3.6157</td>
<td>1.3482</td>
<td>-2.1839</td>
<td>0.1383</td>
<td>1.1518</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>14.244</td>
<td>-8.675</td>
<td>-8.6724</td>
<td>1.0185</td>
<td>1.4026</td>
<td>-1.7594</td>
<td>0.1471</td>
<td>1.1998</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>14.506</td>
<td>-9.343</td>
<td>-9.0573</td>
<td>1.3482</td>
<td>1.2526</td>
<td>-1.3984</td>
<td>0.2047</td>
<td>1.2847</td>
</tr>
<tr>
<td>20</td>
<td>-1</td>
<td>56.407</td>
<td>-27.064</td>
<td>-25.6643</td>
<td>-21.5423</td>
<td>6.9651</td>
<td>-12.0592</td>
<td>0.4255</td>
<td>2.7687</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>55.114</td>
<td>-25.024</td>
<td>-32.0345</td>
<td>-16.660</td>
<td>6.8354</td>
<td>-8.5299</td>
<td>0.4855</td>
<td>4.7580</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>55.407</td>
<td>-23.955</td>
<td>-35.2527</td>
<td>0.9258</td>
<td>6.7314</td>
<td>-6.7886</td>
<td>0.5420</td>
<td>5.7390</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>56.350</td>
<td>-28.102</td>
<td>-44.9247</td>
<td>15.4719</td>
<td>6.2987</td>
<td>-1.6727</td>
<td>0.8139</td>
<td>6.5618</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
<td>124.269</td>
<td>-66.697</td>
<td>-65.3158</td>
<td>-30.6382</td>
<td>15.2215</td>
<td>-22.0676</td>
<td>0.9404</td>
<td>8.6844</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>123.200</td>
<td>-60.153</td>
<td>-75.8904</td>
<td>2.3364</td>
<td>15.8795</td>
<td>-15.0304</td>
<td>1.1582</td>
<td>13.1107</td>
</tr>
<tr>
<td>50</td>
<td>3</td>
<td>125.159</td>
<td>-109.332</td>
<td>-161.850</td>
<td>44.1828</td>
<td>14.9544</td>
<td>-5.4808</td>
<td>1.7745</td>
<td>19.5042</td>
</tr>
<tr>
<td>50</td>
<td>4</td>
<td>221.465</td>
<td>-109.127</td>
<td>-103.602</td>
<td>-86.2857</td>
<td>29.6219</td>
<td>-47.8390</td>
<td>1.5103</td>
<td>11.4426</td>
</tr>
<tr>
<td>50</td>
<td>6</td>
<td>216.266</td>
<td>-120.943</td>
<td>-129.517</td>
<td>-24.1213</td>
<td>29.2493</td>
<td>-83.5575</td>
<td>1.7579</td>
<td>19.5228</td>
</tr>
<tr>
<td>50</td>
<td>7</td>
<td>217.656</td>
<td>-143.090</td>
<td>-145.592</td>
<td>4.4425</td>
<td>28.9265</td>
<td>-35.4997</td>
<td>1.6884</td>
<td>23.5228</td>
</tr>
<tr>
<td>50</td>
<td>8</td>
<td>220.694</td>
<td>-195.599</td>
<td>-181.700</td>
<td>76.8829</td>
<td>27.2772</td>
<td>-5.9364</td>
<td>3.0879</td>
<td>24.8526</td>
</tr>
<tr>
<td>50</td>
<td>9</td>
<td>344.578</td>
<td>-170.861</td>
<td>-162.4310</td>
<td>-134.7850</td>
<td>46.9885</td>
<td>-74.5966</td>
<td>2.7771</td>
<td>18.0275</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>341.194</td>
<td>-182.291</td>
<td>-182.5820</td>
<td>-84.8089</td>
<td>46.7618</td>
<td>-63.3678</td>
<td>2.4156</td>
<td>24.3929</td>
</tr>
<tr>
<td>50</td>
<td>12</td>
<td>338.558</td>
<td>-224.216</td>
<td>-223.4806</td>
<td>7.2818</td>
<td>45.9934</td>
<td>-41.1758</td>
<td>3.0284</td>
<td>36.9401</td>
</tr>
<tr>
<td>50</td>
<td>13</td>
<td>343.579</td>
<td>-265.381</td>
<td>-284.6550</td>
<td>123.6170</td>
<td>45.2728</td>
<td>-9.0165</td>
<td>4.7509</td>
<td>54.7552</td>
</tr>
<tr>
<td>50</td>
<td>15</td>
<td>489.759</td>
<td>-267.729</td>
<td>-263.440</td>
<td>-121.8960</td>
<td>68.1447</td>
<td>-91.5577</td>
<td>3.5984</td>
<td>35.3171</td>
</tr>
<tr>
<td>50</td>
<td>16</td>
<td>482.768</td>
<td>-293.648</td>
<td>-292.7850</td>
<td>-53.6039</td>
<td>67.6475</td>
<td>-74.9936</td>
<td>3.7528</td>
<td>44.4167</td>
</tr>
<tr>
<td>50</td>
<td>18</td>
<td>492.774</td>
<td>-442.333</td>
<td>-410.720</td>
<td>178.4230</td>
<td>63.0787</td>
<td>-12.7130</td>
<td>6.7807</td>
<td>79.0564</td>
</tr>
</tbody>
</table>

These results were then used to determine the features, which are used as a basis for the networks.

### 5.2 Feature Generation

In order to generate the features for the networks, equations were determined for each of the variables in Table 3 based solely on the velocity and angle of attack. These equations were determined with help from the trendline feature in MS Excel. The first step was to develop a baseline equation with velocity as the only variable, for each of the...
pressures and forces. Figure 27 shows the magnitudes of each of the pressures and forces at zero angle of attack versus velocity, as well as the equations of the trendlines.

![Figure 27](image)

**Figure 27.** Magnitude of pressures and forces versus velocity at zero angle of attack.

The trendline equations shown in figure 27 were used as the baseline equations for each of the pressures and forces, the x in the trendline equations was replaced with the velocity. The next step was to add in the effect of the angle of attack. In order to do this, graphs similar to figure 27 were created for each of the pressures and forces at different angles of attack, and the trendlines were determined for each velocity.
Figure 28 shows a graph of the pressures on the top of the airfoil for different velocities and angles of attack, as well as the trendline equations for each velocity.

Figure 28. Magnitude of pressure on top of airfoil versus angle of attack.

To develop a total equation for the pressure on the top of the airfoil, the coefficients in each of the trendline equations, except the y-intercept, were individually divided by the square of the velocity pertaining to that trendline. The resulting values are very similar. The average of these numbers was then placed into the total equation and multiplied by the square of the velocity and either the angle of attack or the square of the angle of attack. Equation 8 is the total equation for the pressure on the top of the airfoil.
\[ TopP = -(0.0712 \times V^2) - (0.0028 \times V) - (0.00786287 \times V^2 \times AOA) \] (8)

Figure 29 shows the pressures on the front of the airfoil for different velocities and angles of attack, as well as the trendline equations for each velocity. Equation 9 is the total equation for the pressure on the front of the airfoil.

\[ FrontP = -(0.0455 \times V^2) + (0.0259 \times V) - (0.0004826204 \times V^2 \times AOA^2) \]
\[ + (0.0191316764 \times V^2 \times AOA) \] (9)
Figure 30 shows the pressures on the rear of the airfoil for different velocities and angles of attack, as well as the trendline equations for each velocity. Equation 10 is the total equation for the pressure on the rear of the airfoil.

\[
\text{RearP} = (0.019 \cdot V^2) - (0.0755 \cdot V) - (0.0000259138 \cdot V^2 \cdot \text{AOA}^2) - (0.0002870823 \cdot V^2 \cdot \text{AOA})
\]

(10)
Figure 31 shows the pressures on the bottom of the airfoil for different velocities and angles of attack, as well as the trendline equations for each velocity. Equation 11 is the total equation for the pressure on the bottom of the airfoil.

\[ \text{Bottom}P = -(0.0275 V^2) - (0.0092 V) + (0.00000227 V^2 \cdot AOA^2) + (0.00451387 V^2 \cdot AOA) \]  

(11)
Figure 32 shows the forces in the x-direction on the airfoil for different velocities and angles of attack, as well as the trendline equations for each velocity. Equation 12 is the total equation for the forces in the x-direction on the airfoil.

\[ X_{\text{Force}} = (0.0007 \cdot V^2) + (0.0058 \cdot V) + (0.0002061 \cdot V^2 \cdot AOA^2) \\
+ (0.0006743 \cdot V^2 \cdot AOA) \]  \hspace{1cm} (12)
Figure 33 shows the forces in the y-direction on the airfoil for different velocities and angles of attack, as well as the trendline equations for each velocity. Equation 13 is the total equation for the forces in the y-direction on the airfoil.

\[
Y_{\text{Force}} = (0.0085 \times V^2) - (0.0026 \times V) - (0.00001547 \times V^2 \times AOA^2)
\]

\[
+ (0.00247294 \times V^2 \times AOA)
\]  

(13)

These equations were then used to fill in the data for velocities and angles of attack that were not simulated, for the CFD based neural and neuro-symbolic networks.
In this context, they were used as features to determine the data that the CFD simulation networks would be based upon. This will be discussed in detail in Chapter 6.

5.3 Pressure and Temperature Data Acquisition System Results

To obtain data, the RC aircraft was flown in three separate flights. The pressure and temperature data acquisition system, which was built in-house, recorded pressure and temperature from four different sensors. As discussed in section 3.5, the sensors were placed in the same four locations as in the CFD simulation. These sensors measured the raw data for pressure and temperature at each location. Figure 34 shows the raw pressure data recorded from the pressure sensors.

Figure 34. Pressure in absolute psi versus time.
These sensors recorded pressure in absolute psi; however, the pressure was changed from absolute psi to gauge Pascal because gauge pressure is a more meaningful value. This was done by first changing the pressures to absolute Pascal, and second determining the average ambient pressure and subtracting that to get gauge Pascal. Then to get the pressure solely from the airflow around the airfoil, the average of the four pressures was determined at each time step and this was subtracted from the pressure at each sensor. Figure 35 shows the four pressures after this final step.

Figure 35. Pressure in gauge Pascal versus time.

These four sensors also recorded temperature in degrees Celsius over the duration of the flights. Figure 36 shows the temperature records during the flights. As can be seen the RC aircraft was flown on a chilly day, as the ambient temperature was around 8-10
°C. It can also be seen that during the three flights the temperature dropped by 6 to 10 degrees because of the wind chill.

This temperature data was not formally used in any calculations. The pressure data from these sensors was combined with data from the other data acquisition system in order to build the neural and neuro-symbolic networks.

5.4 Ruby Data Acquisition System Results

The Ruby data acquisition system records data about the RC aircraft flight. A list of some of the data it records is: angle of attack, altitude AGL (Above Ground Level),
estimated groundspeed, heading, pitch, roll, airspeed from pressure, x, y, and z acceleration, and throttle, aileron, elevator, and rudder control. Out of these many variables, five were the most applicable for this project: altitude, velocity, angle of attack, and x and z acceleration. Figure 37 shows the altitude above ground level for the three recorded flights. This data was used to determine which flight phase the aircraft was in at a certain time.

![Altitude above ground level versus time.](image)

While altitude was never used as an input to any of the networks, the information was critically important. Velocity and angle of attack were two variables that were used both as direct inputs to the neural network, and to determine the inputs for the neuro-
symbolic networks. Figure 38 shows the velocity in meters per second during the three recorded flights.

![Figure 38. Velocity versus time.](image)

The X and Z accelerations were also used. The accelerations were reported as a factor of g (acceleration due to gravity), so they were first changed to meters per seconds squared, and then they were multiplied by the mass of the aircraft (~8.6 lbs.) in order to get the forces in the x and z directions. These forces were used as one of the outputs of the neuro-symbolic networks. Figure 39 shows the x and z forces versus time for the
three recorded flights. In this figure, the z-axis force is much lower due to the effect of gravity.

Figure 39. Force in the x and z-directions versus time.
CHAPTER SIX: NEURAL NETWORK IMPLEMENTATION AND RESULTS

The purpose of this chapter is to present the results of the feed-forward neural networks and the feature-based neuro-symbolic networks, as well as to compare the results for the different kinds of networks.

There were three different sets of data that were used to create the networks. The first data set included undamaged data from the features that were derived from the CFD simulation results, and damaged data using random numbers. The second set of data was based on the data from the first of the three flights done with the testbed. During this flight, the physical aircraft was not damaged. The undamaged data was from typical flight patterns, and the damaged data was from atypical flight patterns, such as: loops, rolls, stalls, and sideways and upside-down flying. The third data set included data from all three of the flights done with the test bed. The undamaged data included the first flight. The first layer of damaged data was from the second flight, where only the top side of the wing was physically damaged. The second layer of damaged data was from the third flight, where both the top and bottom of the wing were physically damaged.

6.1 Feed-Forward Neural Network

The feed-forward neural network that was built for the first set of data (from the CFD simulation) was very accurate. In order to determine how many hidden neurons
should be used in each network, a quick version of each network was run with different numbers of hidden neurons. Whichever number of hidden neurons had the highest percent accuracy was used for the network. Table 4 shows this process. As can be seen in Table 4, 30 hidden nodes were chosen because it has the highest total accuracy.

Table 4. Percent accuracy of neural network for different numbers of hidden nodes.

<table>
<thead>
<tr>
<th># of Hidden Neurons</th>
<th>Output 1</th>
<th>Output 2</th>
<th>Output 3</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>77.67%</td>
<td>77.25%</td>
<td>55.33%</td>
<td>70.08%</td>
</tr>
<tr>
<td>10</td>
<td>95.44%</td>
<td>95.44%</td>
<td>81.28%</td>
<td>90.72%</td>
</tr>
<tr>
<td>14</td>
<td>97.10%</td>
<td>97.10%</td>
<td>86.17%</td>
<td>93.46%</td>
</tr>
<tr>
<td>20</td>
<td>98.51%</td>
<td>98.56%</td>
<td>87.21%</td>
<td>94.76%</td>
</tr>
<tr>
<td>24</td>
<td>98.42%</td>
<td>98.40%</td>
<td>87.40%</td>
<td>94.74%</td>
</tr>
<tr>
<td>30</td>
<td>98.57%</td>
<td>98.57%</td>
<td>87.57%</td>
<td>94.90%</td>
</tr>
<tr>
<td>40</td>
<td>97.60%</td>
<td>97.60%</td>
<td>86.20%</td>
<td>93.80%</td>
</tr>
</tbody>
</table>

This process was repeated for each of the neural and neuro-symbolic networks. Table 5 shows the number of hidden neurons that was chosen for each network.

Table 5. Number of hidden nodes used for each network.

<table>
<thead>
<tr>
<th>CFD Results</th>
<th># of Hidden Neurons</th>
</tr>
</thead>
<tbody>
<tr>
<td>X Neuro-Symbolic</td>
<td>18</td>
</tr>
<tr>
<td>Z Neuro-Symbolic</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1st Flight Only</th>
<th># of Hidden Neurons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural Network</td>
<td>50</td>
</tr>
<tr>
<td>X Neuro-Symbolic</td>
<td>40</td>
</tr>
<tr>
<td>Z Neuro-Symbolic</td>
<td>50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>All 3 Flights</th>
<th># of Hidden Neurons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural Network</td>
<td>50</td>
</tr>
<tr>
<td>X Neuro-Symbolic</td>
<td>60</td>
</tr>
<tr>
<td>Z Neuro-Symbolic</td>
<td>40</td>
</tr>
</tbody>
</table>
6.2 Feature-Based Neuro-Symbolic Network

Two methods were used to determine how accurate a specific network was. The first method, confusion matrices (CM), was used for both neural and neuro-symbolic networks. The second, third, and fourth outputs of the neuro-symbolic networks were the same as the three outputs from the neural networks, so confusion matrices were a good method for comparison. Figure 40 shows the confusion matrix for output three for the neural network using the CFD data set. Figure 41 shows the confusion matrix for output three for the z-axis neuro-symbolic network using the CFD data set.

![Confusion matrix for CFD neural network results.](image)

Figure 40. Confusion matrix for CFD neural network results.
As can be seen from these two confusion matrices, the total accuracy of the z-axis neuro-symbolic network is almost 2% better than the total accuracy of the neural network, meaning in this instance that more of the outputs were classified correctly using the neuro-symbolic network. The percent accuracy results for all of the networks are presented in section 6.3.

The second method used to determine the accuracy of the networks, root mean square (RMS) error, was only used for the neuro-symbolic networks. The first output from the neuro-symbolic networks is the total force in the respective direction (x or z). This is not a classification problem, so the use of a confusion matrix does not apply. Instead, the error was determined between the output and the target, and the RMS error, mean error, and standard deviation of the error were found. Table 6 shows these values for each of the neuro-symbolic networks.
As can be seen in Table 6, the error for the neuro-symbolic networks based on the CFD results is much less in the z-axis, while the error is less in the x-axis for the neuro-symbolic networks based on the real testbed data. One reason why the error is much greater in the x-axis for the networks based on the CFD results is that the force in the z-axis is directly related (via a linear relationship) to the pressure differential in the z-axis. In the x-axis, most of the data between the force and pressure differential is linearly related; however this is different at high angles of attack.

<table>
<thead>
<tr>
<th></th>
<th>RMS Error</th>
<th>Mean Error</th>
<th>Standard Deviation of Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFD Results</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X Neuro-Symbolic</td>
<td>7.436</td>
<td>4.017</td>
<td>6.260</td>
</tr>
<tr>
<td>Z Neuro-Symbolic</td>
<td>0.277</td>
<td>0.136</td>
<td>0.240</td>
</tr>
<tr>
<td>1st Flight Only</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X Neuro-Symbolic</td>
<td>7.899</td>
<td>4.879</td>
<td>6.012</td>
</tr>
<tr>
<td>Z Neuro-Symbolic</td>
<td>45.747</td>
<td>27.193</td>
<td>36.383</td>
</tr>
<tr>
<td>All 3 Flights</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X Neuro-Symbolic</td>
<td>5.351</td>
<td>3.636</td>
<td>3.911</td>
</tr>
<tr>
<td>Z Neuro-Symbolic</td>
<td>11.913</td>
<td>7.634</td>
<td>9.048</td>
</tr>
</tbody>
</table>
CHAPTER SEVEN: SUMMARY OF RESULTS AND FUTURE RESEARCH

The purpose of this chapter is to summarize the results obtained from the neural and neuro-symbolic networks based on the CFD simulations and the RC aircraft flights, as well as to explain the contributions this research made to the field, and to discuss changes that could be made for future research.

7.1 Summary of Results

The main goal of this project was to build a neuro-symbolic network that is more accurate than its respective neural network. Table 7 shows an overview of the total percent accuracies for each network. As can be seen the CFD based z-axis NSN is more accurate than the neural network by more than 2%, however the x-axis NSN is less accurate by more than 9%. It can also be seen that the neural networks based on the real testbed data are more accurate than the neuro-symbolic networks.

Table 7 shows that the network results from the CFD data were more accurate than the results from the testbed data. This is due to the fact that CFD simulations are completed using perfect conditions, whereas the testbed aircraft data was not collected during perfect conditions. The testbed data was collected on a cold and windy day which may have altered some of the results. Also, on the aircraft, the pressure cannot be determined on the surface of the wing because the pressure sensors are a few centimeters off the surface of this wing. Another reason why the results from the testbed are less
accurate is that there is no real failure data for the testbed because the pilot has continuous control of the plane and is compensating for the failure.

Table 7. Confusion matrix percent accuracy for all networks.

<table>
<thead>
<tr>
<th></th>
<th>Percent Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Output 1</td>
</tr>
<tr>
<td>CFD Results</td>
<td></td>
</tr>
<tr>
<td>Neural Network</td>
<td>97.79%</td>
</tr>
<tr>
<td>X Neuro-Symbolic</td>
<td>87.24%</td>
</tr>
<tr>
<td>Z Neuro-Symbolic</td>
<td>98.22%</td>
</tr>
<tr>
<td>1st Flight Only</td>
<td></td>
</tr>
<tr>
<td>Neural Network</td>
<td>80.66%</td>
</tr>
<tr>
<td>X Neuro-Symbolic</td>
<td>73.54%</td>
</tr>
<tr>
<td>Z Neuro-Symbolic</td>
<td>65.07%</td>
</tr>
<tr>
<td>All 3 Flights</td>
<td></td>
</tr>
<tr>
<td>Neural Network</td>
<td>77.79%</td>
</tr>
<tr>
<td>X Neuro-Symbolic</td>
<td>63.04%</td>
</tr>
<tr>
<td>Z Neuro-Symbolic</td>
<td>64.29%</td>
</tr>
</tbody>
</table>

The other part of the overall goal of this project was to make a neuro-symbolic network that takes less time to operate than its respective neural network. Table 8 shows the average time to train and test each of the networks, as well as the average number of iterations or epochs before convergence.

As can be seen, for the networks based on the CFD results, the neuro-symbolic networks take a negligible 0.0012 extra seconds to test. Also the neural network based on the real first set of data is faster to test. On the other hand, the neuro-symbolic networks based on the real testbed data from all three flights are quicker to test than their respective neural network. It can also be seen that even with a large increase in train time and the number of iterations before convergence, the time to test takes almost no additional time.
This project was successful at creating feature-based neuro-symbolic networks. It was somewhat successful at creating more accurate networks, and was somewhat successful at creating networks that take less time to test.

### 7.2 Contributions of this Research

This research made many contributions to the fields of mechanical engineering, neural and neuro-symbolic networks, and aircraft diagnostic systems. It demonstrated how scientific & engineering principles, e.g. conservation laws, can be incorporated into a neural network. Also, by using engineering principles this research has demonstrated how a NSN can be designed. This project also developed a methodology for formulating a global diagnostic system using inputs from local diagnostics. The experimental testbed that was designed provides a platform for investigation feasibility of research results from the MURI teams.
7.3 Future Research

While the results from this research were not ideal, a great deal was learned in the process. This project only took into account one model, and the neuro-symbolic networks were based on only one equation. It would be beneficial to test different models (different shaped aircraft wings, more realistic aircraft wings) as well as using different equations to base the neuro-symbolic networks on. The momentum equation describes the system relatively well; however, an equation that more directly relates to the problem at hand might yield better results.

One condition that might have altered the results of the networks based on real test data is that the data was recorded on a cold and windy day. The effect of this wind on the pressure results was somewhat accounted for by subtracting the average of the pressures from each of the pressures at each time step; however it still could have had an impact. Another condition that was not optimal was that the GPS on the Ruby data acquisition system was not fully functional during flight data recordings. This might have altered some of the altitude data which was used to determine which phase of flight the aircraft was in.

Another process that could have been altered was the lack of trials used to determine how many hidden neurons were optimal. For most of these test, a total of two to ten network runs were used. This lack of information resulted in small errors. Also for all of the data that was collected a total of 100 network runs were used, and the results were averaged. A larger number of trials would be beneficial because potential irregularities are averaged over a larger sample size.
REFERENCES


diagnostics for engine test cells. Paper presented at the Aerospace Conference,
March 6-13: 3361-71.


diagnostics for gas turbine engine bearings. Paper presented at the Aerospace
Conference, March 8-15: 3095-103.

Pechenizkiy, Mykola, Alexey Tsymbal, and Seppo Puuronen. 2004. PCA-based feature
the 17th IEEE Symposium on Computer-Based Medical Systems.

Knowledge representation for a neuro-symbolic network in home care risk
identification. Paper presented at the 7th IEEE International Conference on
Industrial Informatics, Cardiff, Wales. June 24-26: 277-82.


Roemer, Michael J., Gregory J. Kacprzynski, Michael Schoeller, Ron Howe, and Sqrd.

Stanford University. MURI. http://structure.stanford.edu/MURI/index.htm (accessed July
2010-May 2012).


UIUC Applied Aerodynamics Group. Naca2412.dat. In the UIUC Airfoil Coordinates
Database, http://www.ae.illinois.edu/m-selig/ads/coord_database.html#N

to a new information processing principle. Paper presented at the International

Wermter, Stefan and Ron Sun, eds. 2000. Lecture notes in artificial intelligence:
Subseries of lecture notes in computer science. Berlin: Springer-Verlag.


APPENDICES

The purpose of these appendices is to add extra information that is important to this research, but that was not included in the previous chapters. These appendices include information and data on: (A) the coordinates for the airfoil used in the simulations, (B) the journal files used to create the CFD mesh, (C) the journal files and programs used in the CFD optimization, (D) the journal files and programs used in the CFD simulation, (E) information about the sensors used in the research, and (F) the generic Matlab programs used for the neural and neuro-symbolic networks.
Appendix A: X and Y Coordinates for NACA 2412 Airfoil

<table>
<thead>
<tr>
<th>x-coordinates*</th>
<th>y-coordinates*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.9500</td>
<td>0.0114</td>
</tr>
<tr>
<td>0.9000</td>
<td>0.0208</td>
</tr>
<tr>
<td>0.8000</td>
<td>0.0375</td>
</tr>
<tr>
<td>0.7000</td>
<td>0.0518</td>
</tr>
<tr>
<td>0.6000</td>
<td>0.0636</td>
</tr>
<tr>
<td>0.5000</td>
<td>0.0724</td>
</tr>
<tr>
<td>0.4000</td>
<td>0.0780</td>
</tr>
<tr>
<td>0.3000</td>
<td>0.0788</td>
</tr>
<tr>
<td>0.2500</td>
<td>0.0767</td>
</tr>
<tr>
<td>0.2000</td>
<td>0.0726</td>
</tr>
<tr>
<td>0.1500</td>
<td>0.0661</td>
</tr>
<tr>
<td>0.1000</td>
<td>0.0563</td>
</tr>
<tr>
<td>0.0750</td>
<td>0.0496</td>
</tr>
<tr>
<td>0.0500</td>
<td>0.0413</td>
</tr>
<tr>
<td>0.0250</td>
<td>0.0299</td>
</tr>
<tr>
<td>0.0125</td>
<td>0.0215</td>
</tr>
<tr>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.0125</td>
<td>-0.0165</td>
</tr>
<tr>
<td>0.0250</td>
<td>-0.0227</td>
</tr>
<tr>
<td>0.0500</td>
<td>-0.0301</td>
</tr>
<tr>
<td>0.0750</td>
<td>-0.0346</td>
</tr>
<tr>
<td>0.1000</td>
<td>-0.0375</td>
</tr>
<tr>
<td>0.1500</td>
<td>-0.0410</td>
</tr>
<tr>
<td>0.2000</td>
<td>-0.0423</td>
</tr>
<tr>
<td>0.2500</td>
<td>-0.0422</td>
</tr>
<tr>
<td>0.3000</td>
<td>-0.0412</td>
</tr>
<tr>
<td>0.4000</td>
<td>-0.0380</td>
</tr>
<tr>
<td>0.5000</td>
<td>-0.0334</td>
</tr>
<tr>
<td>0.6000</td>
<td>-0.0276</td>
</tr>
<tr>
<td>0.7000</td>
<td>-0.0214</td>
</tr>
<tr>
<td>0.8000</td>
<td>-0.0150</td>
</tr>
<tr>
<td>0.9000</td>
<td>-0.0082</td>
</tr>
<tr>
<td>0.9500</td>
<td>-0.0048</td>
</tr>
<tr>
<td>1.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

*Source: Data adapted from: UIUC Applied Aerodynamics Group.
Appendix B: Gambit Journal File for Airfoil Mesh

/ Journal File for GAMBIT 2.4.6, Database 2.4.4, ntx86 SP2007051421
/ Identifier "default_id6304"
/ File opened for write Fri May 04 11:59:56 2012.
vertex create coordinates 1 0 0
vertex create coordinates 0.95 0.0114 0
vertex create coordinates 0.9 0.0208 0
vertex create coordinates 0.8 0.0375 0
vertex create coordinates 0.7 0.0518 0
vertex create coordinates 0.6 0.0636 0
vertex create coordinates 0.5 0.0724 0
vertex create coordinates 0.4 0.078 0
vertex create coordinates 0.3 0.0788 0
vertex create coordinates 0.25 0.0767 0
vertex create coordinates 0.2 0.0726 0
vertex create coordinates 0.15 0.0661 0
vertex create coordinates 0.1 0.0563 0
vertex create coordinates 0.075 0.0496 0
vertex create coordinates 0.05 0.0413 0
vertex create coordinates 0.025 0.0299 0
vertex create coordinates 0.0125 0.0215 0
vertex create coordinates 0 0 0
vertex create coordinates 0.0125 -0.0165 0
vertex create coordinates 0.025 -0.0227 0
vertex create coordinates 0.05 -0.0301 0
vertex create coordinates 0.075 -0.0346 0
vertex create coordinates 0.1 -0.0375 0
vertex create coordinates 0.15 -0.041 0
vertex create coordinates 0.2 -0.0423 0
vertex create coordinates 0.25 -0.0422 0
vertex create coordinates 0.3 -0.0412 0
vertex create coordinates 0.4 -0.038 0
vertex create coordinates 0.5 -0.0334 0
vertex create coordinates 0.6 -0.0276 0
vertex create coordinates 0.7 -0.0214 0
vertex create coordinates 0.8 -0.015 0
vertex create coordinates 0.9 -0.0082 0
vertex create coordinates 0.95 -0.0048 0
save name "NACA2412validate.dbs"
edge create nurbs "vertex.1" "vertex.2" "vertex.3" "vertex.4" "vertex.5" "vertex.6" "vertex.7" "vertex.8" "vertex.9" "vertex.10" "vertex.11"
"vertex.12" "vertex.13" "vertex.14" "vertex.15" "vertex.16" "vertex.17" "vertex.18" interpolate
edge create nurbs "vertex.18" "vertex.19" "vertex.20" "vertex.21" "vertex.22" "vertex.23" "vertex.24" "vertex.25" "vertex.26" "vertex.27" "vertex.28" "vertex.29" "vertex.30" "vertex.31" "vertex.32" "vertex.33" "vertex.34" "vertex.1" interpolate
vertex create coordinates 1 12.5 0
vertex create coordinates 21 12.5 0
vertex create coordinates 21 0 0
vertex create coordinates 21 -12.5 0
vertex create coordinates 1 -12.5 0
vertex create coordinates -11.5 0 0
dge create straight "vertex.35" "vertex.36"
dge create straight "vertex.36" "vertex.37"
dge create straight "vertex.37" "vertex.38"
dge create straight "vertex.38" "vertex.39"
dge create straight "vertex.39" "vertex.1"
dge create straight "vertex.1" "vertex.35"
dge create straight "vertex.1" "vertex.37"
dge create center2points "vertex.1" "vertex.35" "vertex.40" minarc arc
dge create center2points "vertex.1" "vertex.40" "vertex.39" minarc arc
dface create "rect1" wireframe "edge.3" "edge.4" "edge.9" "edge.8" real
dface create "rect2" wireframe "edge.6" "edge.5" "edge.9" "edge.7" real
dface create "circ1" wireframe "edge.8" "edge.10" "edge.11" "edge.7" real
dface create "airfoil" wireframe "edge.1" "edge.2" real
dface subtract "circ1" faces "airfoil"
undo begingroup
dge modify "edge.4" "edge.7" backward
edge picklink "edge.4" "edge.7" "edge.5" "edge.8"
dge mesh "edge.8" "edge.4" "edge.7" "edge.5" successive ratio1 1.05
   intervals 160
undo endgroup
undo begingroup
dge modify "edge.6" backward
edge picklink "edge.6" "edge.9" "edge.3"
dge mesh "edge.3" "edge.9" "edge.6" successive ratio1 1.05 intervals 160
undo endgroup
dge split "edge.13" percentarclength 0.31294328 connected
dge split "edge.12" percentarclength 0.69231581 connected
undo begingroup
dge modify "edge.15" backward
edge picklink "edge.15" "edge.13"
dge mesh "edge.13" "edge.15" successive ratio1 1.03 intervals 120
undo endgroup
undo begingroup
edge modify "edge.12" backward
edge picklink "edge.12" "edge.14"
edge mesh "edge.14" "edge.12" successive ratio1 1 size 0.0075
undo endgroup
undo begingroup
edge modify "edge.11" backward
edge picklink "edge.11" "edge.10"
edge mesh "edge.10" "edge.11" successive ratio1 1.05 intervals 214
undo endgroup
face mesh "circ1" map size 1
face mesh "rect1" map size 1
face mesh "rect2" map size 1
save
physics create "Front" btype "VELOCITY_INLET" edge "edge.10" "edge.11"
physics create "TopBot" btype "SYMMETRY" edge "edge.3" "edge.6"
physics create "Back" btype "OUTFLOW" edge "edge.4" "edge.5"
physics create "Airfoil" btype "WALL" edge "edge.14" "edge.13" "edge.15" "edge.12"
export fluent5 "NACA2412validate.msh" nozval
save
/ File closed at Fri May 04 12:02:24 2012, 21.19 cpu second(s), 47583552 maximum memory.
Appendix C: Fluent and Matlab Programs for CFD Optimization with Zero Angle of Attack

Fluent Journal File

chdir C:\users\10081\desktop\murioptim\0AoA
file read-case NACA2412.msh
grid check
grid scale 0.3175 0.3175
define models viscous spalart-allmaras y
define materials change-create
  air
  air
  y
  constant
  1.048
  n
  n
  n
  n
  n
  n
  n
  n
  n
  define boundary-conditions velocity-inlet
    front
    n
    y
    y
    n
    20.6
    n
    0
    y
    n
    0.001
define operating-conditions reference-pressure-location
    6.66
    0
solve initialize compute-defaults velocity-inlet
front
solve initialize initialize-flow
solve monitors residual convergence-criteria
  0.000001
  0.000001
  0.000001
  0.000001
surface point-surface
topp
  @#%&
  &%#@
solve iterate 1000
report volume-integrals minimum fluid
  pressure
  y
  minpressure0AoA#&@%.srp
  n
  y
plot plot
  n
toppressure0AoA#&@%.srp
  n
  n
  n
  pressure
  n
  n
  q
topp

file write-case
  NACA24120AoA#&@%.cas
  y

exit
y
Matlab Fmincon Program

ub=[0.16];
lb=[0.02];
x0=[0.04];
options =
optimset('Algorithm','sqp','TolFun',0.001,'TolX',0.001,'DiffMaxChange',
0.1,'DiffMinChange',0.005);
[x, fval]=fmincon(@myfun, x0, [], [], [], [], lb, ub, [], options)

function f=myfun(x)
!cd C:\users\10081\desktop\murioptim\0AoA
b=num2str(x);
e=num2str(x*10000);
if x>=0.15875
    c=0.02299;
elseif (x>=0.14287) && (x<0.15875)
    c=0.02413;
elseif (x>=0.127) && (x<0.14287)
    c=0.02477;
elseif (x>=0.10525) && (x<0.127)
    c=0.02515;
elseif (x>=0.09525) && (x<0.10525)
    c=0.0251;
elseif (x>=0.08437) && (x<0.09525)
    c=0.02515;
elseif (x>=0.07937) && (x<0.08437)
    c=0.02435;
elseif (x>=0.07462) && (x<0.07937)
    c=0.02305;
elseif (x>=0.03175) && (x<0.04762)
    c=0.02099;
elseif (x>=0.02381) && (x<0.03175)
    c=0.01788;
elseif (x>=0.01587) && (x<0.02381)
    c=0.01575;
elseif (x>=0.00794) && (x<0.01587)
    c=0.01311;
elseif (x>=0.00397) && (x<0.00794)
    c=0.00949;
else
    c=0.00683;
end
d=num2str(c);
replaceinfile('@#%', b, 'FluentTopOptim0AoA.jou',
'Fluent1TopOptim0AoA.jou');
replaceinfile('&%#@', d, 'Fluent1TopOptim0AoA.jou', 'Fluent2TopOptim0AoA.jou');
replaceinfile('#&@%', e, 'Fluent2TopOptim0AoA.jou', 'FluentNewTopOptim0AoA.jou');
!fluent 2ddp -g -wait -i FluentNewTopOptim0AoA.jou
pause(3);
f='minpressure0AoA';
g='.*srp';
h=strcat(f, e, g);
 fid=fopen(h);
j=textscan(fid, '%*s %s %[^
]');
k=j{1,1};
l=k(5);
fclose(fid);
m=l(1,1);
n=str2double(m);
v=abs(n);
o='toppressure0AoA';
p=strcat(o, e, g);
 fid=fopen(p);
q=textscan(fid, '%*s %s %[^
]');
r=q{1,1};
s=r(4);
fclose(fid);
t=s(1,1);
u=str2double(t);
w=abs(u);
z=v-w;
f=z;
Appendix D: Fluent and Matlab Programs for CFD Simulation

Fluent Journal File

chdir C:\users\10081\desktop\muriforce
file read-case NACA2412.msh
grid check
grid scale 0.3175 0.3175
define models viscous spalart-allmaras y
define materials change-create
  air
  air
  y
  constant
  1.048
  n
  n
  n
  n
  n
  n
  n
define boundary-conditions velocity-inlet
  front
  n
  y
  y
  n
  @##&
  n
  &##@
  y
  n
  0.001
define operating-conditions reference-pressure-location
  6.66
  0
solve initialize compute-defaults velocity-inlet
front
solve initialize initialize-flow
solve monitors residual convergence-criteria
  0.000001
  0.000001
  0.000001
  0.000001
surface point-surface
topp
0.051
0.0215
surface point-surface
frontp
0.00762
-0.00724
surface point-surface
rearp
0.311
-0.00007
surface point-surface
bottomp
0.0857
-0.0133
solve iterate 1000
report forces wall-forces
n
airfoil

@%#&
&#%%
y
&&@%K@&AOAXForce.srp
report forces wall-forces
n
airfoil

&&%@
@%#&
y
&&@%K@&AOAYForce.srp
report volume-integrals maximum fluid

pressure
y
&&@%K@&AOAmaxpressure.srp
n
Matlab MURI Results Program

!cd C:\users\10081\desktop\muriforce
for i=1:6
  if i==1
    v=10;
  elseif i==2
    v=20;
  elseif i==3
    v=30;
  elseif i==4
    v=40;
  elseif i==5
    v=50;
  end

v=50;
else
  v=60;
end
for j=1:6
  if j==1
    a=-1;
  elseif j==2
    a=0;
  elseif j==3
    a=1;
  elseif j==4
    a=2;
  else
    a=5;
  end
  w=v*0.5144;
  s=sind(a);
  c=cosd(a);
  x=w*c;
  y=w*s;
  b=num2str(x);
  d=num2str(y);
  e=num2str(v);
  f=num2str(a);
  g=num2str(c);
  h=num2str(s);
  replaceinfile('@@%', b, 'MURIResults.jou', 'MURIResults1.jou');
  replaceinfile('%%@', d, 'MURIResults1.jou', 'MURIResults2.jou');
  replaceinfile('#@$', e, 'MURIResults2.jou', 'MURIResults3.jou');
  replaceinfile('%&%', f, 'MURIResults3.jou', 'MURIResults4.jou');
  replaceinfile('@%&', g, 'MURIResults4.jou', 'MURIResults5.jou');
  replaceinfile('###', h, 'MURIResults5.jou', 'MURIResultsNew.jou');
  !fluent 2ddp -g -wait -i MURIResultsNew.jou
  pause(3);
end
Appendix E: Testbed Sensor Information

*Bosch BMP085 Digital Pressure Sensors*

<table>
<thead>
<tr>
<th>Key features</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Pressure range:</td>
<td>300 ... 1100hPa (+9000m ... -500m above sea level)</td>
</tr>
<tr>
<td>Supply voltage:</td>
<td>1.8 ... 3.6V (V_{DDA})</td>
</tr>
<tr>
<td></td>
<td>1.62V ... 3.6V (V_{DDO})</td>
</tr>
<tr>
<td>LCC8 package:</td>
<td>Robust, ceramic lead-less chip carrier (LCC) package</td>
</tr>
<tr>
<td>Small footprint:</td>
<td>5.0mm x 5.0mm</td>
</tr>
<tr>
<td>Super-flat:</td>
<td>1.2mm height</td>
</tr>
<tr>
<td>Low power:</td>
<td>5µA at 1 sample / sec. in standard mode</td>
</tr>
<tr>
<td>Low noise:</td>
<td>0.06hPa (0.5m) in ultra low power mode</td>
</tr>
<tr>
<td></td>
<td>0.03hPa (0.25m) ultra high resolution mode</td>
</tr>
<tr>
<td></td>
<td>down to 0.1m (rms noise) possible</td>
</tr>
</tbody>
</table>

- Temperature measurement included
- I²C interface
- Fully calibrated
- Pb-free, halogen-free and RoHS compliant,
- MSL 1
1 Electrical characteristics

If not stated otherwise, the given values are maximum values over temperature/voltage range in the given operation mode.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Condition</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating temperature</td>
<td>$T_A$</td>
<td>operational</td>
<td>-40</td>
<td></td>
<td>+85</td>
<td>°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>full accuracy</td>
<td>0</td>
<td></td>
<td>+65</td>
<td>°C</td>
</tr>
<tr>
<td>Supply voltage</td>
<td>$V_{DD}$</td>
<td>ripple max. 50mVpp</td>
<td>1.8</td>
<td>2.5</td>
<td>3.6</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td>$V_{DDO}$</td>
<td></td>
<td>1.62</td>
<td>2.5</td>
<td>3.6</td>
<td>V</td>
</tr>
<tr>
<td>Supply current @ 1 sample/sec. 25°C</td>
<td>$I_{DDLOW}$</td>
<td>ultra low power mode</td>
<td>3</td>
<td></td>
<td></td>
<td>μA</td>
</tr>
<tr>
<td></td>
<td>$I_{DDSTD}$</td>
<td>standard mode</td>
<td>5</td>
<td></td>
<td></td>
<td>μA</td>
</tr>
<tr>
<td></td>
<td>$I_{DDHR}$</td>
<td>high resolution mode</td>
<td>7</td>
<td></td>
<td></td>
<td>μA</td>
</tr>
<tr>
<td></td>
<td>$I_{DDUHR}$</td>
<td>ultra high res. mode</td>
<td>12</td>
<td></td>
<td></td>
<td>μA</td>
</tr>
<tr>
<td>Peak current</td>
<td>$I_{peak}$</td>
<td>during conversion</td>
<td>650</td>
<td>1000</td>
<td></td>
<td>μA</td>
</tr>
<tr>
<td>Standby current</td>
<td>$I_{DDIS}$</td>
<td>at 25°C</td>
<td>0.1</td>
<td></td>
<td></td>
<td>μA</td>
</tr>
<tr>
<td>Serial data clock</td>
<td>$f_{SCL}$</td>
<td></td>
<td></td>
<td>3.4</td>
<td></td>
<td>MHz</td>
</tr>
<tr>
<td>Conversion time temperature</td>
<td>$t_{C, temp}$</td>
<td>standard mode</td>
<td>3</td>
<td>4.5</td>
<td></td>
<td>ms</td>
</tr>
<tr>
<td>Conversion time pressure</td>
<td>$t_{C, p, low}$</td>
<td>ultra low power mode</td>
<td>3</td>
<td>4.5</td>
<td></td>
<td>ms</td>
</tr>
<tr>
<td></td>
<td>$t_{C, p, std}$</td>
<td>standard mode</td>
<td>5</td>
<td>7.5</td>
<td></td>
<td>ms</td>
</tr>
<tr>
<td></td>
<td>$t_{C, p, hr}$</td>
<td>high resolution mode</td>
<td>9</td>
<td>13.5</td>
<td></td>
<td>ms</td>
</tr>
<tr>
<td></td>
<td>$t_{C, p, UHR}$</td>
<td>ultra high res. mode</td>
<td>17</td>
<td>25.5</td>
<td></td>
<td>ms</td>
</tr>
<tr>
<td>Absolute accuracy pressure</td>
<td></td>
<td>700 ... 1100 hPa 0 ... +65 °C</td>
<td>-2.5</td>
<td>±1.0</td>
<td>+2.5</td>
<td>hPa</td>
</tr>
<tr>
<td>$V_{DD} = 3.3\text{V}$</td>
<td></td>
<td>300 ... 700 hPa 0 ... +65 °C</td>
<td>-3.0</td>
<td>±1.0</td>
<td>+3.0</td>
<td>hPa</td>
</tr>
<tr>
<td></td>
<td></td>
<td>300 ... 1100 hPa -20 ... 0 °C</td>
<td>-4.0</td>
<td>±1.5</td>
<td>+4.0</td>
<td>hPa</td>
</tr>
<tr>
<td>Resolution of output data</td>
<td></td>
<td>pressure</td>
<td>0.01</td>
<td></td>
<td></td>
<td>hPa</td>
</tr>
<tr>
<td></td>
<td></td>
<td>temperature</td>
<td>0.1</td>
<td></td>
<td></td>
<td>°C</td>
</tr>
<tr>
<td>Relative accuracy pressure</td>
<td></td>
<td>700 ... 1100 hPa @ 25 °C</td>
<td>±0.2</td>
<td></td>
<td></td>
<td>hPa</td>
</tr>
<tr>
<td>$V_{DD} = 3.3\text{V}$</td>
<td></td>
<td>0 ... 65 °C @ p const.</td>
<td>±0.5</td>
<td></td>
<td></td>
<td>hPa</td>
</tr>
</tbody>
</table>
Calculating absolute altitude

With the measured pressure $p$ and the pressure at sea level $p_0$ e.g. 1013.25hPa, the altitude in meters can be calculated with the international barometric formula:

$$\text{altitude} = 44330 \times \left(1 - \left(\frac{p}{p_0}\right)^{\frac{1}{5.255}}\right)$$

Thus, a pressure change of $\Delta p = 1$hPa corresponds to 8.43m at sea level.

Calculating pressure at sea level

With the measured pressure $p$ and the absolute altitude the pressure at sea level can be calculated:

$$p_0 = \frac{p}{\left(1 - \frac{\text{altitude}}{44330}\right)^{5.255}}$$

Thus, a difference in altitude of $\Delta\text{altitude} = 10$m corresponds to 1.2hPa pressure change at sea level.
FEATURES
3-axis sensing
Small, low-profile package
   4 mm x 4 mm x 1.45 mm LFCSP
Low power
   180 µA at V₁ = 1.8 V (typical)
Single-supply operation
   1.8 V to 3.6 V
10,000 g shock survival
Excellent temperature stability
BW adjustment with a single capacitor per axis
RoHS/WEEE lead-free compliant
## Specifications

Ta = 25°C, Vcc = 3 V, Ca = Cb = 0.1 μF, acceleration = 0 g, unless otherwise noted. All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed.

### Table 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Conditions</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sensor Input</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Measurement Range</td>
<td>Each axis</td>
<td>±2</td>
<td>±3.6</td>
<td></td>
<td>g</td>
</tr>
<tr>
<td>Nonlinearity (%)</td>
<td></td>
<td>±0.3</td>
<td></td>
<td></td>
<td>%</td>
</tr>
<tr>
<td>Package Alignment Error (%)</td>
<td></td>
<td>±1</td>
<td></td>
<td></td>
<td>Degrees</td>
</tr>
<tr>
<td>Interaxis Alignment Error (%)</td>
<td></td>
<td>±0.1</td>
<td></td>
<td></td>
<td>Degrees</td>
</tr>
<tr>
<td>Cross Axis Sensitivity (%)</td>
<td></td>
<td>±1</td>
<td></td>
<td></td>
<td>%</td>
</tr>
<tr>
<td><strong>Sensitivity (Ratiometric)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensitivity at Xout, Yout, Zout</td>
<td>Vcc = 3 V</td>
<td>270</td>
<td>300</td>
<td>350</td>
<td>mV/g</td>
</tr>
<tr>
<td>Sensitivity Change Due to Temperature (%)</td>
<td>Vcc = 3 V</td>
<td>±0.015</td>
<td></td>
<td></td>
<td>%/°C</td>
</tr>
<tr>
<td><strong>Zero g Bias Level (Ratiometric)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 g Voltage at Xout, Yout, Zout</td>
<td>Vcc = 3 V</td>
<td>1.2</td>
<td>1.5</td>
<td>1.8</td>
<td>V</td>
</tr>
<tr>
<td>0 g Offset vs. Temperature</td>
<td></td>
<td>±1</td>
<td></td>
<td></td>
<td>μg/°C</td>
</tr>
<tr>
<td><strong>Noise Performance</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Noise Density Xout, Yout</td>
<td></td>
<td>280</td>
<td></td>
<td></td>
<td>μg/Hz rms</td>
</tr>
<tr>
<td>Noise Density Zout</td>
<td></td>
<td>350</td>
<td></td>
<td></td>
<td>μg/Hz rms</td>
</tr>
<tr>
<td><strong>Frequency Response</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bandwidth Xout, Yout</td>
<td></td>
<td>1600</td>
<td></td>
<td></td>
<td>Hz</td>
</tr>
<tr>
<td>Bandwidth Zout</td>
<td></td>
<td>550</td>
<td></td>
<td></td>
<td>Hz</td>
</tr>
<tr>
<td>P2T Tolerance</td>
<td></td>
<td>32 ± 15%</td>
<td></td>
<td></td>
<td>kΩ</td>
</tr>
<tr>
<td>Sensor Resonant Frequency</td>
<td></td>
<td>5.5</td>
<td></td>
<td></td>
<td>kHz</td>
</tr>
<tr>
<td><strong>Self Test</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logic Input Low</td>
<td></td>
<td>+0.6</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>Logic Input High</td>
<td></td>
<td>+2.4</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>ST Actuation Current</td>
<td></td>
<td>+60</td>
<td></td>
<td></td>
<td>μA</td>
</tr>
<tr>
<td>Output Change at Xout</td>
<td></td>
<td>-150</td>
<td></td>
<td></td>
<td>mV</td>
</tr>
<tr>
<td>Output Change at Yout</td>
<td></td>
<td>+150</td>
<td></td>
<td></td>
<td>mV</td>
</tr>
<tr>
<td>Output Change at Zout</td>
<td></td>
<td>-60</td>
<td></td>
<td></td>
<td>mV</td>
</tr>
<tr>
<td><strong>Output Amplifier</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Swing Low</td>
<td></td>
<td>0.1</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>Output Swing High</td>
<td></td>
<td>2.8</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td><strong>Power Supply</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating Voltage Range</td>
<td>Vcc = 3 V</td>
<td>1.8</td>
<td></td>
<td>3.6</td>
<td>V</td>
</tr>
<tr>
<td>Supply Current</td>
<td></td>
<td>320</td>
<td></td>
<td></td>
<td>μA</td>
</tr>
<tr>
<td>Turn-On Time</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>ms</td>
</tr>
<tr>
<td><strong>Temperature</strong></td>
<td></td>
<td>-25</td>
<td></td>
<td>70</td>
<td>°C</td>
</tr>
</tbody>
</table>
Appendix F: Generic Matlab Code for Neural and Neuro-Symbolic Networks

clear all

load InputFile.csv
load OutputFile.csv

a=InputFile';
b=OutputFile';
n=0;
TIME=0;
EPOCH=0;
CM1=zeros(2,2);
CM2=zeros(2,2);
CM3=zeros(4,4);

tic

for i=1:100;
    train_index=randi(2291,2600,1);

    inputs=a(:, train_index);
    targets=b(:, train_index);

    test_index=zeros(2291,1);
    test_index(train_index)=1;
    test_index(find(test_index==0));

    test_inputs=a(:, test_index);
    test_targets=b(:, test_index);

    % Create Network
    numHiddenNeurons = 40;  % Adjust as desired
    net = newff(inputs,targets,numHiddenNeurons);
    net.divideParam.trainRatio = 70/100;  % Adjust as desired
    net.divideParam.valRatio = 15/100;  % Adjust as desired
    net.divideParam.testRatio = 15/100;  % Adjust as desired

    % Train and Apply Network
    [net,tr] = train(net,inputs,targets);
    outputs = sim(net,test_inputs);

    outorig1=outputs(1,:);
    outorig2=outputs(2,:);
    outorig3=outputs(3,:);

    out1=outorig1;
out2=outorig2;
out3=outorig3;

size=size(out1,2);
for j=1:size
    if (out1(j)>=0.5)
        out1(j)=1;
    else
        out1(j)=0;
    end
end
clear size

size=size(out2,2);
for k=1:size
    if (out2(k)>=0.5)
        out2(k)=1;
    else
        out2(k)=0;
    end
end
clear size

size=size(out3,2);
for l=1:size
    if (out3(l)>=2.5)
        out3(l)=3;
    elseif (out3(l)>=1.5) && (out3(l)<2.5)
        out3(l)=2;
    elseif (out3(l)>=0.5) && (out3(l)<1.5)
        out3(l)=1;
    else
        out3(l)=0;
    end
end

testtarg1=test_targets(1,:);
testtarg2=test_targets(2,:);
testtarg3=test_targets(3,:);

cml1=confusionmat(out1, testtarg1);
cm2=confusionmat(out2, testtarg2);
cm3=confusionmat(out3, testtarg3);

clear size

last=size(tr.time);
last=last(2);
time=tr.time(last);
epoch=tr.epoch(last);

clear last
TIME=TIME+time;
EPOCH=EPOCH+epoch;
CM1=CM1+cm1;
CM2=CM2+cm2;
CM3=CM3+cm3;
n=n+1

end

toc
CM1
CM2
CM3
TIME
EPOCH
n