



***LOAD MEASUREMENT OF IN-SERVICE MARINE  
STRUCTURES  
TO INFLUENCE THEIR DESIGN***

**Mohammad Reza Ramazani**

A thesis submitted in partial fulfilment of the requirement of  
Bournemouth University for the degree of Doctor of Philosophy

*June 2013*

Bournemouth University

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and due acknowledgment must always be made of any material contained in, or derived from, this thesis.

## Abstract

Current methods to predict hydrodynamic loads rely either on oversimplified and semi-empirical methods or the use of numerical simulation and analysis techniques such as Finite Element Analysis (FEA) or Boundary Element Analysis (BEA). These methods are conservative which results in the over-design of these craft so they are heavier and slower than they could otherwise be. Better understanding of load intensities will inform the design process of marine structures and could result in lighter and more efficient designs. This research investigates the possibility of solving these problems employing artificial intelligence (AI) as an alternative to the current methods. Few studies have applied Artificial Intelligence to the design of marine structures. Detailed review of the past and present research shows that AI and in particular Artificial Neural Networks (ANN) can be used as an inverse problem solver when there are no closed form relationships that exist between the input and the output. An inverse approach is defined as the problem where response of the structure is known but the load that caused that response is unknown. In real problems/structures the response to a point load is experienced throughout the structure with different levels of intensities which is the link between the external load and these differential intensities. Determining this relationship will result in a unique solution without the knowledge of material constitutive laws, material properties and structure size or thickness. The aim of this investigation is to develop a real time in-service load measurement tool using an inverse approach. To achieve this, ANN, experimental techniques and FEA analysis are combined to form a hybrid inverse problem solver that can be trained to use structural response, such as strains at various locations, to predict the loads that caused them.

The main objective of this research is to investigate the suitability of the proposed methodology for real time in-service load monitoring on large marine structures. The proposed system must be able to measure both steady-state as well as transient load such as equivalent slamming load. The outcome of this investigation was successful prediction of the external loads in terms of their approximate location and load intensities. The only disadvantage of this method is that the solver requires training and can only learn from cases that it has been subjected to. However, once the system is trained it can predict both static and dynamic loads quickly and accurately.

## Publications

The peer reviewed publications below have been published in international journals and can be found in Appendix 6:

1. **Ramazani, M. R.**, Noroozi, S., Koohgilani, M., Cripps, B., and Sewell, P., 2013a. Determination of static pressure loads on a marine composite panel from strain measurements utilising artificial neural networks. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 227, 12–21.
2. **Ramazani, M. R.**, Noroozi, S., Sewell, P., Khandan, R., and Cripps, B., 2013b. Using artificial neural networks and strain gauges for the determination of static loads on a thin square fully-constrained composite marine panel subjected to a large central displacement. *Insight Journal: Non-Destructive Testing and Condition Monitoring*, 55(8), 442-448.
3. **Ramazani, M. R.**, Sewell, P., Noroozi, S., Koohgilani, M., and Cripps, B., 2013c. In-service transient load measurement on a large composite panel. *Trans Tech Publications, Switzerland: Applied Mechanics and Materials*, 248, 204-211.
4. **Ramazani, M. R.**, Sewell, P., Noroozi, S., Koohgilani, M., and Cripps, B., 2013d. Sensor optimisation for in-service load measurement of a large composite panel under small displacement. *Trans Tech Publications, Switzerland Applied Mechanics and Materials*, 248, 153-161.

## Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Networks
BEA	Boundary Element Analysis
DAQ	Data Acquisition system
ERSG	Electrical Resistance Strain Gauges
GA	General Algorithm
GRFP	Glass Reinforced Fibre Polymer
HIPE	Hybrid Inverse Problem Engine
FEA	Finite Element Analysis
FL	Fuzzy Logic
MATLAB NNT	MATLAB Neural Network Toolbox
MOD	Ministry Of Defence
MSE	Mean Squared Error
NI	National Instruments
SSE	Sum of Squared Error
RIB	Rigid Inflatable Boat
RN	Royal Navy
RNLI	Royal National Lifeboat Institution
RMS	Root Mean Square

## Contents

Abstract .....	III
Publications .....	IV
Abbreviations .....	V
Contents .....	VI
List of Figures .....	IX
List of Tables .....	XV
Acknowledgements .....	XVI
Chapter 1. Introduction to Project and Literature Review .....	1
1.1 Introduction .....	1
1.1.1 BAE Systems Surface Ships Ltd.....	2
1.1.2 Aims of the Research .....	3
1.1.3 Research Question and Objectives.....	4
1.2 Background Research and Literature Review .....	4
1.2.1 Boat Design and Structural Analysis .....	5
1.2.2 Literature Review Findings.....	19
1.3 Summary.....	24
Chapter 2. Introduction to Methodology and Artificial Neural Networks (ANN) .....	27
2.1 Introduction .....	28
2.2 Description of the Methodology.....	29
2.3 Introduction to Artificial Neural Network.....	34
2.3.1 Definition of Artificial Neural Network .....	34
2.3.2 Neuron Model .....	37
2.3.3 ANN Types Based on Learning Approaches .....	44

2.3.4	Advantages and disadvantages of ANNs over conventional computer algorithms .....	46
2.3.5	Design Principles .....	47
2.3.6	Improving Results .....	53
2.3.7	Why MATLAB Neural Network Toolbox.....	53
2.4	Equipment Set Up.....	55
2.4.1	Small Displacement Equipment Set Up.....	61
2.4.2	Sensor Optimisation Experimental Setup .....	62
2.4.3	Large Displacement Experimental Set Up.....	64
2.4.4	Drop Test Simulation Set Up .....	65
2.5	Summary.....	68
Chapter 3.	General Behaviour of Composite Panel with Attached Strain Gauges under Small Displacement .....	69
3.1	Introduction .....	70
3.2	General Behaviour of Composite Panel with Attached Strain Gauges under Small Displacement (Static Load is Applied).....	70
3.2.1	Methodology .....	70
3.2.2	Results and Discussion.....	76
3.2.3	Summary and Conclusion .....	86
3.3	Sensor Optimization .....	87
3.3.1	Methodology .....	87
3.3.2	Results and Discussion.....	89
3.4	Discussion and Conclusion.....	93
3.5	Summary.....	94
Chapter 4.	General Behaviour of Composite Panel with Attached Strain Gauges under a Large Displacement and Drop Test .....	95
4.1	Introduction .....	96

4.2	Large Displacement Experiment Setup under Static Loading condition .....	97
4.2.1	Methodology .....	97
4.2.2	Results .....	107
4.2.3	Conclusion.....	115
4.3	Predicting Impact Loads Using Artificial Neural Networks .....	117
4.3.1	Methodology .....	117
4.3.2	Results .....	125
4.3.3	Conclusion.....	130
4.4	Summary.....	130
Chapter 5. DISCUSSION, EVALUATION, FINAL CONCLUSION AND FUTURE RESEARCH .....		131
5.1	Summary and Discussion .....	132
5.2	Evaluation of the proposed In-Service Load Monitoring System .....	134
5.2.1	Experimental Set-up.....	135
5.2.2	Artificial Neural Network .....	136
5.2.3	Finite Element Analysis and Experimental Techniques .....	137
5.3	Final Conclusion.....	138
5.4	Future Research .....	138
References	.....	140
Appendices	.....	151

## List of Figures

Figure 1-1: BAE Systems Halmatic Range Pacific 950 and Pacific 24 (Shockmitigation 2013) .....	3
Figure 1-2: Hull structure of a cruiser (US Naval, 2013) .....	6
Figure 1-3: Slamming Phenomena (Petrov 2012).....	8
Figure 1-4: Green Water Phenomena (Hydrolance 2013) .....	8
Figure 1-5: The loading on a ship derives from two sources: gravity and water pressure (Tupper 2004) .....	10
Figure 1-6: Still water hogging ( Tupper 2004).....	10
Figure 1-7: Sagging on a wave ( Tupper 2004) .....	11
Figure 1-8: Typical loading, shearing force and bending moment curves for a boat/ship (Tupper 2004) .....	12
Figure 2-1: Artificial Neural Network training loop to minimise error (Sewell et al. 2010) .....	30
Figure 2-2: A two-layer tansig/purelin feed-forward network (Hagan et al. 2002).....	31
Figure 2-3: Over-fitting problem .....	32
Figure 2-4: Learning and the problem of over-fitting (Schmidt 1996).....	33
Figure 2-5: Example of ANN problem solving model (Zaknich 2003).....	34
Figure 2-6: Schematic of a single input neuron (Beale et al. 2012).....	38
Figure 2-7: Linear transfer function (Beale et al. 2012) .....	39
Figure 2-8: Log-Sigmoid transfer function (Beale et al. 2012) .....	39
Figure 2-9: Tan-Sigmoid transfer function (Beale et al. 2012).....	40
Figure 2-10: single neuron with an R-element input vector (Beale et al. 2012).....	41
Figure 2-11: Single layer neural network (Beale et al. 2012) .....	42

Figure 2-12: Multiple Layers Neural Network (Beale et al. 2012).....	43
Figure 2-13: Flowchart for building an ANN model .....	49
Figure 2-14: Schematic of the composite panel .....	57
Figure 2-15: Schematic of a strain gauge and how it work .....	58
Figure 2-16: Microlink 751 Multi-function USB Data Acquisition (Windmill 2011) ...	59
Figure 2-17: Microlink 594 Bridge Input Unit and Screw Terminals (Windmill 2011)	59
Figure 2-18: The Windmill Chart and Logger programs (Windmill 2011).....	59
Figure 2-20: Panel drawing with the location of gauges and loading for small displacement experiment .....	62
Figure 2-21: 5 Kg Weight and 0.05 Kg stand.....	62
Figure 2-22: Schematic of composite panel indicating strain gauge, loading and support location (Sensor Optimisation experiment) .....	64
Figure 2-23: Schematic of composite panel indicating strain gauge, loading and support location (Large Displacement experiment.....	65
Figure 2-24: Schematic of composite panel (Drop test) .....	67
Figure 2-25: Meshed FEA Panel.....	67
Figure 3-1: Theory of Superposition Example (Sewell et al. 2010) .....	72
Figure 3-2: Example of acquired data from experiment (Sewell et al. 2010).....	72
Figure 3-3: Generation of training input (strains) (Sewell et al. 2010).....	73
Figure 3-4: Generation of training output (load) (Sewell et al. 2010) .....	73
Figure 3-5: Example of the training file.....	74
Figure 3-6: MATLAB representation of optimum ANN architecture.....	75
Figure 3-7: Data Logger.....	78

Figure 3-8: Data Reading set for loading 10 Kg on location 10 after 18 days .....	78
Figure 3-9: Data Reading set for loading 1 Kg on location 11 after 2 days .....	79
Figure 3-10: Examples of the linear relation between the loading and the response of the structure .....	80
Figure 3-11: Net. Predicted vs. Expected Values for new data generated data by superposition .....	82
Figure 3-12: Behaviour of the system prediction over a range of loading quantities .....	83
Figure 3-13: Net. Prediction vs. Expected Values for experimental data (one area loaded) .....	84
Figure 3-14: Net. Prediction vs. Expected Values for experimental data (two areas loaded) .....	85
Figure 3-15: Comparison of the experimental data and data generated by superposition .....	85
Figure 3-16: Estimated load data by ANN vs. expected load values (from superposition at location L1) .....	91
Figure 3-17: Estimated load data by ANN vs. expected load values (from superposition at location L6) .....	91
Figure 3-18: Estimated load data by ANN Vs. expected load values (from experiment at location L6) .....	92
Figure 5-1: MATLAB representation of ANN architecture for method one .....	99
Figure 5-2: MATLAB representation of ANN architecture for method two .....	100
Figure 5-3: Flowchart of PYTHON Script of Large Displacement Experiment Description .....	101
Figure 5-4: Partitioned panel (16 rectangles and 13 circles) .....	103
Figure 5-7: distributed Pressure load (Load_Surf_1) .....	105

Figure 5-8: Schematic of a Large displacement experiment simulation result.....	106
Figure 5-9: Comparison of FEA and Experimental data selected strain gauges (S6)...	108
Figure 5-10: Comparison of FEA and Experimental data selected strain gauges (S10) .....	109
Figure 5-11: Error between FEA and Experimental data selected strain gauges (S6)..	109
Figure 5-12: Error between FEA and Experimental data selected strain gauges (S10)	109
Figure 5-13: SSE Performance of network architecture 2 .....	111
Figure 5-14: Comparison of the SSE values of the two network architectures .....	111
Figure 5-16: ANN estimation for FEA and experimental data Vs. Expected real data for L13 .....	114
Figure 5-17: ANN estimation for FEA and experimental data Vs. Expected real data for L1 .....	114
Figure 5-18: ANN estimation for FEA and experimental data Vs. Expected real data for L7 .....	115
Figure 5-19: Flowchart of PYTHON Script of Drop Test Experiment .....	119
Figure 5-21: Cylinder part in modelling of the drop test .....	121
Figure 5-22: Panel part is meshed and has 7031 elements .....	122
Figure 5-23: Cylinder part is meshed and has 40 elements .....	122
Figure 5-24: Fully Fixed Boundary condition for the panel edges .....	123
Figure 5-25: Velocity just before impact is calculated for specific free fall height of the cylinder .....	124
Figure 5-26: Schematic of impact simulation results.....	124
Figure 5-27: FEA Vs. experimental test results .....	126
Figure 5-28: Typical FEA strain data during the impact for gauge S1 .....	127

Figure 5-29: Estimated impact load data by ANN Vs. expected values from training data (L3).....	128
Figure 5-30: Estimated impact load data by ANN Vs. expected values from training data (L9).....	128
Figure 5-31: Estimated impact load data by ANN Vs. expected values from test data (L1) .....	129
Figure 5-32: Estimated impact load data by ANN Vs. expected values from test data (L6) .....	129
Figure 1-1: Flowchart of the MATLAB GUI .....	171
Figure 1-2: Developed MATLAB Programme main GUI.....	172
Figure 1-3: General Network Data Manager (nntool) GUI .....	173
Figure 1-4: number of loading locations, strain and load gauges configuration GUIs .	174
Figure 1-5: Strain data acquisition No.1 GUIs .....	175
Figure 1-6: Export the acquired data to the MATLAB workspace.....	176
Figure 1-7: Training data generation parameter configuration GUIs .....	177
Figure 1-8: Export the generated data to the MATLAB workspace.....	177
Figure 1-9: Network Parameter Configuration GUIs for ANN architecture 1 .....	178
Figure 1-10: Neural network Training tool.....	179
Figure 1-11: Export the network and its main parameters to the MATLAB workspace .....	180
Figure 1-12: Load estimation results indicated as a bar chart.....	181
Figure 1-13: Export the network output to the MATLAB workspace.....	181
Figure 1-14: Nload networks hidden layer and neuron number configuration.....	182
Figure 1-15: General Network Data Manager GUI .....	198

Figure 1-16: Import to Network/Data Manager GUI.....	199
Figure 1-17: Export from Network/Data Manager GUI .....	199
Figure 1-18: Create Network or Data GUI .....	200
Figure 1-19: Network Architecture View .....	203
Figure 1-20: Network Train window (Training Info index) .....	203
Figure 1-21: Network Train window (Training Parameters index) .....	204
Figure 1-22: Network Adapt window (Adaption Info index) .....	204
Figure 1-23: Network Network Training GUI.....	205
Figure 1-24: Network Performance Plot.....	206
Figure 1-25: Network Regression Plot.....	207
Figure 1-26: Network Reinitialize Weights Window .....	208
Figure 1-27: Network View/Edit Weights Window .....	208
Figure 1-28: Network Simulate Window.....	209

## List of Tables

Table 2-1: Composite Panel Material Specification .....	57
Table 2-2: Windmill 751-TC Unit Specifications (Windmill 2011).....	60
Table 2-3: Rosette Strain Gauges Properties (Vishay_Group 2010).....	62
Table 2-4: Strain gauge specification (Vishay_Group 2010) .....	65
Table 3-1: Architecture of the artificial neural network .....	75
Table 3-2: ANN architecture parameters (optimisation experiment) .....	88
Table 3-3: Comparison of ANN performance SSE for various sensor configurations...	90
Table 4-1: ANN Architectures .....	110
Table 4-2: Optimum ANN Architecture (method 2) .....	112
Table 1-1: List of MATLAB NNT Training Functions (Beale, et al., 2012) .....	200
Table 1-2: List of MATLAB NNT Network Types (Beale, et al., 2012) .....	201
Table 1-3: List of MATLAB NNT Performance Functions (Beale, et al., 2012).....	201

## **Acknowledgements**

I would like to express my gratitude to Dr. Philip Sewell and Prof. Siamak Noroozi who have supported me from the beginning right to the end of my study with their tremendous supervision. The author would also like to thank Prof. Bob Cripps formally of BAE Surface Ships and Dr. Mehran Koohgilani for their valuable time and guidance. This work was supported by BAE Systems Surface Ships Ltd, Portsmouth and Bournemouth University. The Author would like to thank the financial supports as well as Steve Austin, Head of Engineering Support of the Royal National Lifeboat Institution for providing the marine composite panel used in this investigation. Finally, I would like to thank my family for giving me such a wonderful life and strong will and all of their endless care and support during this course.

## Chapter 1. **Introduction to Project and Literature Review**

This chapter introduces the research project including a background to the topic, research aims and objectives. This chapter also covers brief introductions to a review of the academic literature, its relation and contexts to the research problem.

## **1.1 Introduction**

In-service Marine structural components are subjected to hydrodynamic loads that are both complex and transient in nature which makes their measurement particularly difficult. Current practices to predict these loads rely either on oversimplified and semi-empirical methods or the use of numerical simulation and analysis techniques that can be subjective. This means the methods are conservative which results in the over-design of these crafts so they are heavier and slower than they could otherwise be. Measuring the actual load history once the craft has been manufactured and put in-service can provide designers with more accurate load information which informs them at the design stage with more accurate and realistic design load data. Hence, the outcome helps progressive improvement in the design of such crafts, improving operational safety and enabling more accurate remaining life prediction. In-service load monitoring systems can also provide information relating to the damage of the structure as a result of impact that may affect the structure's integrity as well as ride quality and its dynamic characteristics/response.

The outcome of this study could result in important added value to academia and industry. Providing more accurate load information from real life load history helps designers avoid over-designing the structure by informing the design process which can also affect and improve the operational safety.

The thesis consists of six chapters. In the first chapter an initial introduction to the research project background and its original aim is covered. A literature review is conducted in this chapter as well. Having an overview of the current literature about the research topic, more detailed aims and objectives are presented at the end of first chapter.

Following the first chapter, the research plan including the proposed methodology and the equipment set up is covered in the second chapter. In this chapter, Artificial Neural Network (ANN) is introduced in more detail necessary to understand the methodology employed in this research. This chapter includes brief ANN definition and historical overview as well as common ANN models, architectures and training algorithms. Furthermore, in the second chapter some important considerations and general rules in design of a successful ANN model are presented. Finally, the ANN software employed in this research is introduced.

The third Chapter describes a simple panel test rig that was donated by the RNLI to test the inverse problem engine by predicting lateral static loads that causes small lateral displacement. Also, further investigation was conducted in order to minimise the number of sensors needed to predict unique approximate solutions on seven locations on the panel.

Chapter four concerns the use of the system to solve non-linear problems. Large lateral load can cause large displacement of marine panels. In such cases using linear elastic/static analysis fails to generate accurate training data for the ANN and ANN fails to converge. To ensure convergence, a novel methodology is proposed, presented and tested that allows convergence resulting in predicting both the magnitude and the location of the large internal load. In Chapter five, conclusions and the possible future works are described.

### **1.1.1 BAE Systems Surface Ships Ltd**

BAE Systems Surface Ships Ltd are interested in developing an in-service load monitoring system to have practical load history of specific marine structure such as their marine crafts. As a result, this project was funded by BAE Systems Surface Ships Ltd and officially started from October 2009. The outcome of this study will result in important added value to education and industry. Providing more accurate load information from real life load history, it helps designers avoid the over-design of vessels and crafts and can inform their design resulting in progressive improvement, improve operational safety and enable remaining life to be predicted.

BAE Systems is the premier global defence, security and aerospace company delivering a full range of products and services for air, land and naval forces, as well as advanced electronics, security, information technology solutions and customer support services. With approximately 106,400 employees worldwide, BAE Systems' sales exceeded £18.5 billion (US \$34.4 billion) in 2008 (Bae 2009).

BAE Systems Surface Ships was established in Nov 2009 following the formation of BVT Surface Fleet in July 2008 that bought together the surface warship building and through-life support operations of BAE Systems and VT Group, including their joint venture, Fleet Support Limited and VT Halmatic, its small boat design and manufacturing business. BAE Systems Surface Ships is the UK's leading provider of surface warships and through-life support, a world-class industrial partner for the UK Ministry of Defence

(MOD) and a leader in the global export market for warships and innovative naval surface ship support.

BAE Systems Surface Ships is also the strategic partner to Portsmouth's Naval Base Commander, offering in-service support capability to the Royal Navy and manage Portsmouth Naval Base and logistics and warehousing services to the Base and the Royal Navy surface ship fleet based in Portsmouth.

BAE Systems Surface Ships has operations in Glasgow, Portsmouth and Filton near Bristol, employing over 8,000 people and has the facilities, skills and partnerships to set the global standard as a trusted and innovative through-life surface ship partner in the UK and export markets.

BAE Systems Surface Ships Small Boat business was formed from the legacy Halmatic business, building on over 50 years of experience of design and manufacturing high performance military and paramilitary craft including its Rigid Inflatable Boats (RIBS), such as the Pacific 24, the sea boat of choice for the UK Royal Navy and the Pacific 950, the latest boat in the BAE Systems Surface Ships small boat range Figure 1-1.



Figure 1-1: BAE Systems Halmatic Range Pacific 950 and Pacific 24 (Shockmitigation 2013)

### 1.1.2 Aims of the Research

The general aim is to develop a technique that potentially can be used for real-time load monitoring of marine structures to meet the requirements of the marine industry. This can lead to the development of a tool for the study of static and dynamic characteristics of the

hull. The proposed system will potentially enable the determination of in-service, transient loads in real-time and produce a load history of the vessel. This would provide valuable information about characteristics of the hull of a vessel to inform their future design.

### **1.1.3 Research Question and Objectives**

The research question is as follows:

“Is it possible to estimate the loads applied on a marine structure by means of establishing a relationship between the applied loads and the structural responses to these loads on a marine structure?”

The following objectives will be carried out to meet the aim:

- Synthesis information from current practice and academic literature to critically analyse the state of the art in load monitoring/measurement/prediction of structures.
- Investigate and develop a methodology to accurately quantify the load on a marine structure, which may be influenced by:
  1. the loading condition (static and transient loading).
  2. the structure’s response (linear or nonlinear response to load).
  3. the structure’s size.
- Validate the methodology utilising a representative structure under known loading conditions.
- Optimise the methodology and develop a prototype tool that can be used potentially to study and/or monitoring of marine structures.
- Provide new insights in load monitoring of structures.

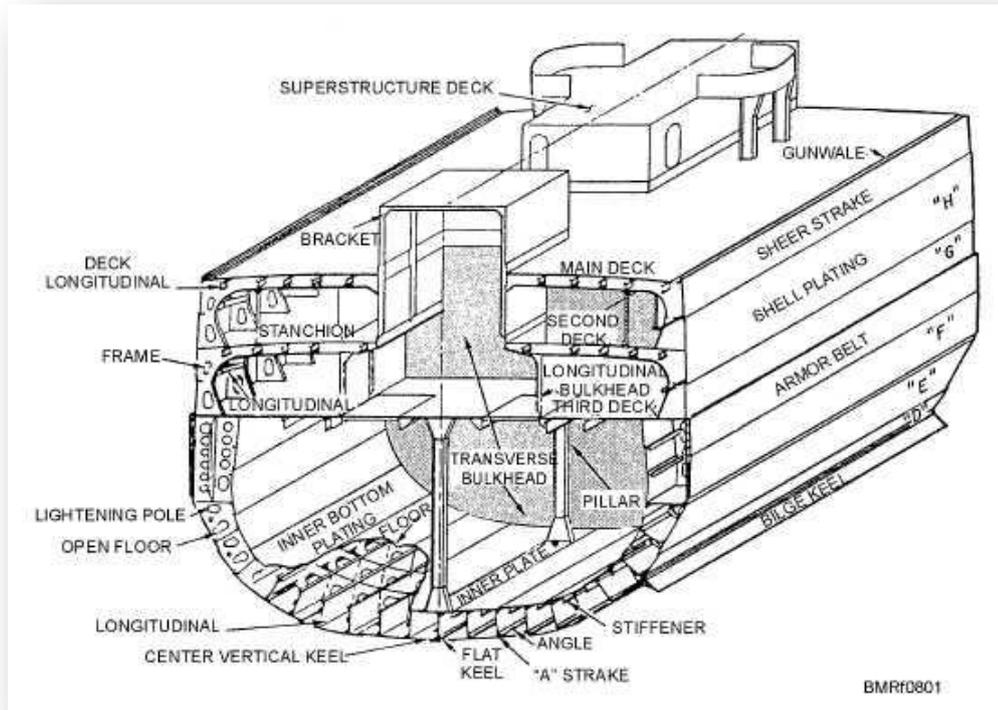
## **1.2 Background Research and Literature Review**

This section outlines some introductory information about boat design and structural analysis as well as critical analysis of the state of the art in load monitoring /measurement/ prediction of structures. Furthermore, an investigation of the potential of using Artificial Neural Networks as an inverse method to be used for load monitoring systems are presented in this section.

### **1.2.1 Boat Design and Structural Analysis**

Several key components make up the main structure of most boats. The hull is the main structural component of the boat which provides buoyancy. The hull of a boat consists of an internal network of frames that extend from side to side (transverse) and that run the length of the boat (longitudinal), covered by outer shell plating, usually made of fibreglass or metal. The roughly horizontal, but chambered structures spanning the hull of the boat are referred to as the deck. In a ship there are often several decks, but a boat is unlikely to have more than one, if any at all. Above the deck are the superstructures. The underside of a deck is the deck head.

An enclosed space on a boat is referred to as a cabin. Several structures make up a cabin: the similar but usually lighter structure which spans a raised cabin is a coach-roof. The "floor" of a cabin is properly known as the sole, but is more likely to be called the floor (a floor is properly, a structural member which ties a frame to the keelson and keel). The vertical surfaces dividing the internal space are bulkheads. The keel is a lengthwise structural member to which the frames are fixed (sometimes referred to as a backbone). The front (or forward end) of a boat is called the bow. The rear (or aft end) of the boat is called the stern. The right side (facing forward) is starboard and the left side is port. See Figure 1-2 which indicates a typical hull structure of a cruiser.



**Figure 1-2: Hull structure of a cruiser (US Naval, 2013)**

Possible loading conditions of a ship are calculated and information is supplied usually in the form of a profile of the boat indicating the positions of all loads on board, a statement of the end draughts, the trim of the boat and the meta-centric height. Stability information in the form of curves of statical stability is often supplied. The usual loading conditions covered are:

- the lightship
- fully loaded departure condition with homogeneous cargo
- fully loaded arrival condition with homogeneous cargo
- Ballast condition
- Other likely service conditions.

A booklet is prepared for the ship showing all these conditions of loading. Nowadays the supply of much of this data is compulsory and, indeed, is one of the conditions for the assignment of a freeboard (Tupper 2004).

There are different ways to classify the various loads that are exerted on a marine structure (Koelbel and Jr. 1995). As a first approach, a ship is considered along with all its equipment, cargo and fluids as the system under consideration. The loads exerted on this system could be classified into the following two categories:

(1) Standard Loads

(2) Extreme Loads

Standard or operational loads are the ones that the ship will experience during most of her lifetime. These loads act on the ship as a whole as concentrated loads. Such loads include:

- Static still water and wave buoyancy: these are the hydrostatic forces that act on the ship hull when the ship is afloat.
- Dynamic lift loads: for semi-displacement and planning hulls.
- Wind pressure: especially for ships with large superstructure area.
- Dry-dock loads: when a ship lays on the dry-dock platform.
- Mooring lines and anchors: these act as concentrated loads.

Extreme loads are a class of loads which occur when the ship sails in harsh weather conditions. The naval architect should keep in mind that these loads may occur rarely, such as slamming, ship-ship and ship-obstacle collisions.

Slamming is the impact of the bottom structure of a ship onto the sea surface. It is mainly observed while sailing in waves, when the bow rises from the water and subsequently impacts on it (see Figure 1-3). Slamming induces extremely high loads to ship structures and is taken under consideration when designing ships. Environmental loads such as wave loads have to be taken into consideration in the design process as well.



**Figure 1-3: Slamming Phenomena (Petrov 2012)**

In heavy storms, the waves and ship motions can become so large that water flows onto the deck of a ship. This problem is known as ‘green water loading’. The term ‘green water’ is used to distinguish between the spray (small amounts of water and foam) flying around and the real solid seawater on the deck. Because seawater is rather more green than blue, the term ‘green water’ is widely used (see Figure 1-4).



**Figure 1-4: Green Water Phenomena (Hydrolance 2013)**

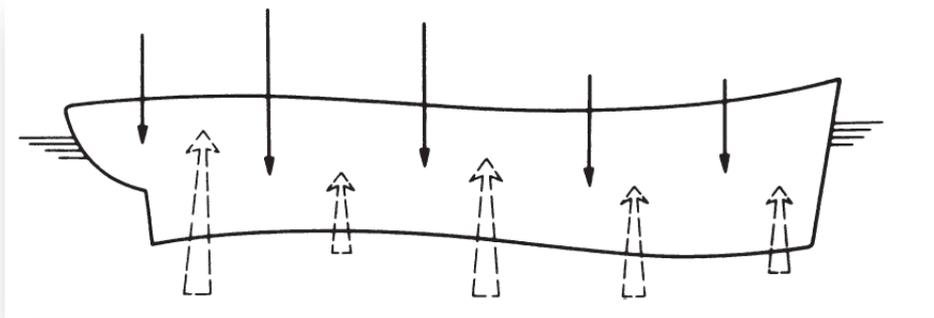
Assume that the designer has already determined the dimensions and shape of the hull, and has made the first estimate of the weight and centre of gravity of the design. In addition, assume that the designer has laid out the basic structural arrangement for the hull (locations of the plate stiffeners, frames, girders, and bulkheads). Given all this, the structural design procedure is as follows (Koelbel and Jr. 1995):

1. Select the design sea state, or significant wave height.
2. Calculate the added resistance in waves and determine the maximum speed the boat can achieve with the installed power,
3. Calculate the vertical accelerations at the centre of gravity for the design speed in the design sea state.
4. Determine the total load on the bottom, and the peak pressure.
5. Calculate the design pressure for each structural element.
6. Calculate the required thickness for plating, and the required section modulus for frames and girders.

To solve any of these complex structural engineering problems, it is necessary to reduce it to a series of unit problems which can be dealt with individually and superimposed. The smallest units of structure which have to be considered are the panels of plating and single stiffeners which are supported at their extremities by items which are very stiff in comparison; they are subject to normal and edge loads under the action of which their dishing, bowing and buckling behaviour relative to the supports may be assessed. Many of these small units together constitute of large curved surfaces of plating and sets of stiffeners called grillages, supported at their edges by bulkheads or deck edges which are very stiff in comparison. They are subject to normal and edge loading and their dishing and buckling behaviour as a unit relative to their supports may be assessed. Finally, many bulkheads, grillages and decks, together constitute a complete hollow box whose behaviour as a box girder may be assessed.

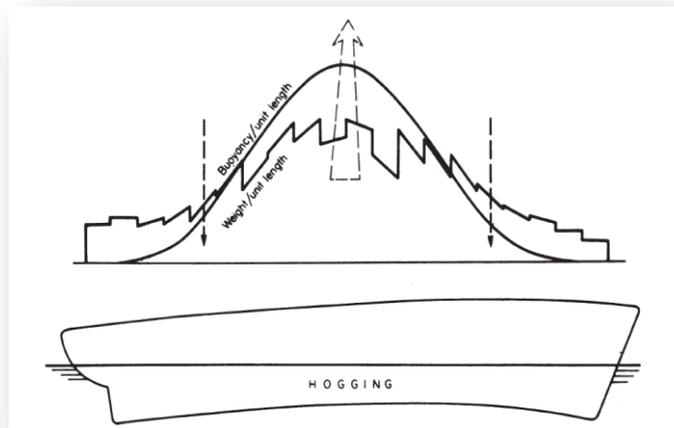
Excluding inertia loads due to ship motion, the loading on a ship derives from only two sources, gravity and water pressure. It is impossible to conceive a state of the sea whereby the loads due to gravity and water pressure exactly cancel out along the ship's length. Even in still water, this is exceedingly unlikely but in a seaway where the loading is changing continuously, it is inconceivable. There is therefore an uneven loading along the ship and,

because it is an elastic structure, it bends. It bends as a whole unit, like a girder on an elastic foundation and is called the ship girder. The ship will be examined as a floating beam subject to the laws deduced in other textbooks for the behaviour of beams.



**Figure 1-5: The loading on a ship derives from two sources: gravity and water pressure (Tupper 2004)**

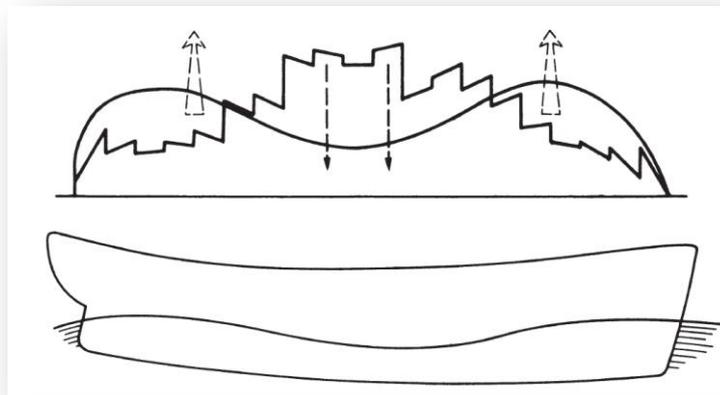
In still water, the loading due to gravity and water pressure are, of course, weight and buoyancy. The distribution of buoyancy along the length follows the curve of areas while the weight is conveniently assessed in unit lengths and might, typically, result in the block diagram of Figure 1-6 (clearly, the areas representing total weight and total buoyancy must be equal.). This figure would give the resultants dotted which would make the ship bend concave downwards or hog.



**Figure 1-6: Still water hogging ( Tupper 2004)**

The reverse condition is known as sagging. Because it is not difficult to make some of the longer cargo ships break their backs when badly loaded, consideration of the still water hogging or sagging is vital in assessing a suitable cargo disposition. It is the first mate's yardstick of structural strength.

It is not difficult to imagine that the hog or sag of a ship could be much increased by waves. A long wave with a crest amidships would increase the upward force there at the expense of the ends and the hogging of the ship would be increased. If there were a hollow amidships and crests towards the ends sagging would be increased (Figure 1-7).



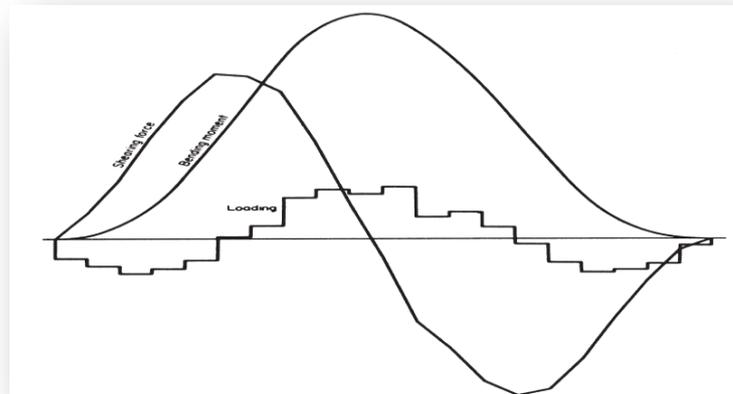
**Figure 1-7: Sagging on a wave ( Tupper 2004)**

The loads to which the complete hull girder is subject are, in fact:

- a. Those due to the differing longitudinal distribution of the downward forces of weight and the upward forces of buoyancy, the ship considered at rest in still water;
- b. The additional loads due to the passage of a train of waves, the ship remaining at rest;
- c. Loads due to the superposition on the train of the waves caused by the motion of the ship itself through still water;
- d. The variations of the weight distribution due to the accelerations caused by ship motion.

Consideration of the worst likely loading effected by (a) and (b) is the basis of the standard calculation. The effects of (c) and (d) are smaller and are not usually taken into

account. A graph for typical loading, shearing force and bending moment curves is shown Figure 1-8.



**Figure 1-8: Typical loading, shearing force and bending moment curves for a boat/ship (Tupper 2004)**

#### State of the Art in Load Monitoring, Measurement and Prediction of Structures

Recent studies outline that the measurement of in-service hydrodynamic load on marine structural components is particularly difficult (Guo-Dong and Wen-Yang 2009). This is due to the transient nature of the load intensities. Current practices to predict these loads are conservative and rely either on oversimplified and semi-empirical methods or the use of numerical simulation and analysis techniques that can be subjective (Faltinsen 2007). This results in the over-design of these craft so they are heavier and slower than they could otherwise be. Developing novel load measurement methods leads to better understanding of marine structure environmental and operating conditions. This means more realistic loads are considered in the design stage which results in lighter and faster final solutions.

A review on methods for evaluating hydrodynamic loads on ships by Phelps (1997) outlines that for given ship conditions, both hydrostatic and self-weight loads can be evaluated with a high degree of confidence in ship structural design and analysis.

Having the buoyancy distribution and the self-weight distribution over the hull of boat or ship, Finite Element (FE) programmes can be employed easily for the analysis of both hydrostatic and self-weight loads. This is due to the fact that such loads are static, well defined and are easily applied as pressure or self-weight load types available within FE programs. Furthermore, detailed documents are available for the hull giving necessary

information about its underwater shape. This enables calculation of the buoyancy distribution through Archimedes principle. The self-weight distribution can be determined employing detailed weight knowledge of stores, cargo and fuel. This means FE programmes will be able to analyse hydrostatic and self-weight loads. In contrast, the determination of hydrodynamic loads generated by sea waves is less predictable. Furthermore, reliable guidance for understanding of dynamic nature of the loading as well as transient effects such as slamming is not available.

Research has provided improved knowledge of the nature of hydrodynamic loads in the past five decades. The basis for the mathematical description of hydrodynamic loads on ships has been provided by the work of St. Denis and Pierson (1953), Korvin-Kroukovsky (1955), Korvin-Kroukovsky and Jacobs (1997) , Salvesen et al. (1970) and Hughes (1983). These methods are basically linear and two-dimensional in nature and although reasonably accurate for moderate sea conditions and ship speeds, they are less accurate for extreme loading conditions which are also the most critical when considering hull structural strength (Phelps 1997).

Non-linear theories and three-dimensional load prediction methods (panel methods) mostly developed by Ochi and Motter (1973) and Belik et al. (1983) have been introduced but these require greater computational effort and have not yet proven to be significantly more accurate than the two dimensional methods.

The measurement of hydrodynamic impact loads has an important role in the design of reliable cost and weight effective marine structures. Weight is a major factor to optimise the speed of a marine structure. Although employing stiff and light materials such as composites leads to increase in speed, structural damage is still significant (Bunting and Sheahan 2009). It is expected that the structural damage may be influenced by global hydroelastic behaviour from waves and/or local hydrodynamic impact loads from slamming (Guo-Dong and Wen-Yang 2009). Wave impact is a random nonlinear phenomenon which is very sensitive to relative motion and attack angle between the body and free surface of the water. Since the duration of wave impact loads is very short, hydroelastic effects are large. In addition, because of air trapping, the wave impact phenomenon is difficult to describe. As the new generation of high speed naval craft gets larger and faster, slamming impact loads on these vessels becomes a critical design

concern. However, the hydrodynamic impact load is one of the least understood areas of marine structure design (Lee and Wilson 2009). Wave impact has challenged many researchers for more than half century and yet more research for an accurate practical estimation method of wave impact loads is required.

Many studies have been performed so far for evaluating hydrodynamic loads on ships. Such methods are summarised in the reviews by Phelps (1997) and Guo-Dong and Wen-Yang (2009). It is indicated that the evaluation of wave generated hydrodynamic loads is less reliable than the static loads and there is less guidance as to how to handle the dynamic nature of the loading as well as transient effects such as slamming. In the past four to five decades, research has provided increased knowledge of the nature of hydrodynamic loads, which together with the improvements in computing power have greatly enhanced the capability to determine the effects of these loads on ship structures. Starting with the work of Karman (1929), various research works have been carried out to describe hydrodynamic loads in Naval Architecture, on motorboats and sailboats in parallel with rules and regulations (Abs 2011; Allen et al. 1978; Bv 2008; Dnv 2011; Faltinsen 2000; Faltinsen 2007; Heller and Jasper 1961; Hentinen and Holm 1994; Iso 2008; Joubert 1982, 1996; Kapsenberg et al. 2003; Ochi and Motter 1973; Reichard 1984; Savitsky and Brown 1976; Stavovy and Chuang 1976) . Furthermore, the introduction of composite materials in marine architecture, such as fast marine crafts, has brought new types of operational failures in panels. In contrast to metallic materials, when the design makes use of composite materials, different types of cracks can appear such as delaminations. This usually happens due to the localised impacts from service loads (such as slamming loads) often observed in composite high-speed crafts. Although, many studies have been carried out to describe the slamming phenomenon and its consequences on the design of ships made of metallic materials (Beukelman 1980; Chalmers 1993; Gerritsma and Beukelman 1964; Heller and Jasper 1961; Jones 1972; Ochi and Motter 1969; Ochi and Motter 1973; Ssc 1995; Stavovy and Chuang 1976), there are limited studies in this area for composite marine structures. In addition, as it is pointed out in various studies (Bunting and Sheahan 2009; Manganelli 2006), there is a need for more accurate knowledge of the hydrodynamic impact on marine structure problems as the knowledge on wave impact is still far from sufficient.

The work by Guo-Dong and Wen-Yang (2009), has reviewed the techniques used to predict impact loads. In summary, these techniques can be categorised into four groups: theoretical approaches, experiments, empirical formulas and numerical simulations. Theoretical approaches and numerical simulations are most flexible and full of potential. Experiments are thought to be most reliable, but it can be costly and suffers from scale effect. The measured results such as a 2-D wedge, a 2-D cross section of a ship, a 3-D drop cone or sphere, are usually used for the validation of a theoretical method or numerical simulation. There are also sea-keeping tests conducted in the towing tank in either regular or irregular waves. The goal of these experimental studies is to obtain the relationship between the pressure magnitude and the velocity at the instant of impact of a given body. The elasticity of the structure and the scale effect may play an important role in the magnitude of pressure. A number of empirical equations have been developed to estimate the impact loads in the time domain (Guo-Dong and Wen-Yang 2009).

For a more accurate prediction, a method involving complete determination of the loads based on scientific principles is needed rather than empirical procedures. These oversimplified approaches cannot give a designer the comprehensive information required. Measurement of hydrodynamic loads generated by sea waves is difficult due to the fact that the sea is highly irregular. The common practice to determine wave loads is based on applying rules and standards which often relies on conservative methods due to large uncertainties in the theoretical calculations used for wave load predictions for ships. In addition, for unconventional ships with new structural designs, it can be sometimes difficult to apply general standards and rules. Direct calculation procedures are needed specifically for complex structures and designs. However, the direct calculation procedures, especially the calculation of the wave loads, are less applied in the shipbuilding industry (Kukkanen 2010). One reason is the rather large uncertainty in the wave load predictions for ships as well as lack of experience. In addition, the theoretical basis of the calculation methods is not necessarily sufficient to achieve reliable predictions. Furthermore, uncertainties also exist in all assumptions involved in stochastic methods and prediction procedures including environmental and operational conditions. Sometimes these are difficult to determine accurately in advance and hence assumptions need to be made to estimate them (Kukkanen 2010).

The current techniques to measure hydrodynamic loads indicate that many techniques developed are either simplified or very expensive and time-consuming (Bai 2003). Many studies have been performed to date for evaluating hydrodynamic loads on ships. Various strip methods have been developed (Iwashita et al. 1993; Watanabe 1994) and used to estimate wave induced ship motions and wave loads. The conventional strip theory is successfully used in the sea-keeping analysis of normal displacement ships (Salvesen et al. 1970). However, its validity can be questioned when it is used for ships with higher maximum operating speeds (Faltinsen 2005). Nonlinearities become more significant for higher speed ships, however, the conventional strip theory is a linear theory. Strip methods are extensively used as a standard tool to predict the nonlinear loads and motions (Jensen and Pedersen 1979). Though, there are some concerns on the accuracy of the strip methods for estimation in short waves as they do not precisely consider hydrodynamic interference effects of reflected waves among strips lengthwise or three-dimensional effects.

In order to enhance the accuracy of estimation, especially in short waves, many numerical methods considering the three-dimensional effects have been proposed. Among them are the three-dimensional Green function method (Iwashita et al. 1993) and the Rankine source method based on three-dimensional potential theory (Nakos and Sclavounos 1990; Takagi 1993; Yasukawa 1990). However, nonlinear theories and three-dimensional load prediction methods require greater computational effort and have not yet proven to be significantly more accurate than the two dimensional methods (Guo-Dong and Wen-Yang 2009; Phelps 1997).

At present all the modelling is based on some rational approximations, for example, it is common that the fluid is assumed to be ideal and incompressible, and the solid body to be rigid. Even with these assumptions, the problem is still very complicated. Almost all models used to predict impact loads find it difficult to incorporate the hydrodynamic impact into the sea-keeping prediction. Most of the sea-keeping theories are linear, and the hydrodynamic impact is strongly nonlinear. Usually engineers would prefer to estimate the relative speed between the waves and the ship hull, and then the calculation of hydrodynamic force would be presented separately. Computational Fluid Dynamics (CFD) techniques give a new way to solve these problems including overturning and wave breaking (Guo-Dong and Wen-Yang 2009).

It has become popular to use CFD to solve fluid flow problems. This is due to the fact that conventional engineering tools based on linear panel methods cannot accurately describe the flow. Green water on deck and Slamming are particular examples on fluid motions that have been investigated by CFD techniques (Faltinsen 2007).

Some CFD methods based on solving the Navier-Stokes equations are thought to be effective in the violent fluid motion problems. Mixture between air and fluid may occur and viscous effects can matter (Guo-Dong and Wen-Yang 2009). There is extensive work worldwide in applying CFD to green water on deck. However, most work is related to two-dimensional flow. An attempt has been made by Greco et al. (2007) to classify how the different green water phenomena occur as a function of wave parameters in head sea conditions for stationary ships, based on experimental and theoretical studies of a restrained two-dimensional body. Slamming is of concern in many marine applications.

It seems generally accepted that CFD codes have difficulties in predicting impact loads (Faltinsen 2007). Actually, hydro-elasticity may then play an important role. The fact that the CFD methods are still time consuming makes statistical estimations of response variables in sea difficult. It is not always easy to follow the fluid particles in the simulations. A minor error at one point and at one time may cause the breakdown of the entire simulation. One of the essential steps is to use mesh regeneration after several time steps to avoid cluster or stretch of the elements. Smoothing has to be used regularly to remove saw tooth behaviour of the free surface (Guo-Dong and Wen-Yang 2009). According to Faltinsen (2007), Commercial CFD codes are generic in nature and special physical features are specific for the different application fields, one cannot necessarily trust documentation of verification and validation in other applications than those that one is interested in. Therefore, A balance between analytical methods, CFD, model tests and sea tests is recommended (Faltinsen 2007). Although CFD solutions have been greatly improved in both speed and accuracy in recent years, it is still very challenging to perform complete nonlinear three-dimensional computations for the present problem (Sun and Faltinsen 2012). As vessels and craft, in most cases, are extremely complicated structures, the mechanical properties, or relations between externally induced excitation and structural responses, are difficult to formulate. An appropriate load monitoring system and technique has to be developed for naval assets and large structures (Cao et al. 1998).

Finite Element Analysis (FEA) can be employed for ship structural strength assessment. This requires a suitable method to apply loads to the FEA model. This approach is dependent upon the accuracy of the methods used to define and calculate the loads in the first place and also the specific FEA software being used. In addition it is more difficult to apply hydrodynamic loads to finite element models than for static load cases. This is due to the dynamic nature of the problem and less clearly defined loads.

According to Phelps (1997) who has reviewed methods used for determining hydrodynamic loads on ships and their limitations, many documents have been written in relation to various hydrodynamic loading theories and likewise on finite element analysis of ship structures and components. He has outlined a review of wave load calculation methods and their application to finite element models as well as their evaluation using experimental data. This review indicates that although methods for static load measurements are generally reliable, predicting wave loads on ships in different sea states, ship speeds and headings are conservative and less accurate. Phelps (1997) concluded that the works done so far were limited and correlation between numerical predictions and test data and further evaluations were necessary. He has also recommended that further work be done to validate results obtained against alternative methods and full scale measurements. This would enable more realistic loading scenarios to be analysed with greater confidence and within a shorter period of time.

Kukkanen (2010) investigated hydrodynamic load and strength analysis of marine structures. He has categorised environmental and operational conditions as important elements in the load and strength analysis of marine structures. In other words knowing these two elements is the starting point for the structural design and analysis of the ships and marine structures.

The main operational conditions for ships are the speed, the heading with respect to waves and the sea state (i.e. height and length of the waves). The operational profile can vary considerably between different ship and boat types. Depending on the ship type, other operational conditions should also be taken into account, for instance, load condition of the vessel and time spent in harbour. Furthermore, voluntary speed reduction or possible restrictions in speed or heading in high waves should be considered in order to define the sea states the ship can operate in safety.

Most designers of high-speed craft use the rules, guidelines, and procedures of the various classification societies to calculate the bottom loads and pressures. Unfortunately there are some uncertainties in these calculations that can lead to overdesign (Koelbel and Jr. 1995).

### **1.2.2 Literature Review Findings**

Based on current limitations of load monitoring methods, there is a need for a new technique/system that is able to overcome some of these limitations. Therefore, such a system ideally should have the following specifications:

- Available methods cannot be employed in real-time therefore the system should be able to monitor applied loads in real-time for in-service purposes
- The system should be capable of monitoring the whole structure with a minimum/optimum number of low maintenance sensors
- The system should be able to be adopted for measuring in-service transient slamming loads
- The system should require as small as possible computational effort (should be fast and respond in real time)
- Both experimental and computational stress analysis requires that the material properties should obey constitutive laws and have well-defined geometrical data such as size, thickness or constraints. However, an inverse approach that can bypass the requirement for all of the above parameters may be an alternative approach

Therefore this research aims to find a solution to the problem by focus on finding a means of establishing a relationship between loads and structural responses that represents hydro-dynamically the mechanical characteristics of the boat/ship structure. In other words the research question is as follows:

“Is it possible to estimate the loads applied on a marine structure by means of establishing a relationship between the applied loads and the structural responses to these loads on a marine structure?”

It is shown that other techniques should be developed to overcome current limitations in practice used to measure loads employed in marine structure design procedures. A

literature survey is carried out to investigate the novelty of using Artificial Intelligence (AI) approach in the marine industry, employed to relate the structural responses to the hydrodynamic loads. Different approaches employing AI methods may be utilised to solve the problems. Genetic Algorithm (GA), Fuzzy Logic (FL), Artificial Neural Network (ANN) and many more are the examples of various AI techniques available (Mamdani and Assilian 1975). These methods are utilised in many research areas where problems are solved by pattern recognition, generalisation and pattern classification (Rao and Rao 1993).

As an inverse solving technique, one of the artificial intelligence mechanisms that can be used is an ANN. A connectionist theory is the basis of ANN, where this modern mathematical approach is made of neurons (units) and connections between them. According to Kohonen (1998), W.S. McCulloch and W.A. Pitts introduced the methods as the fundamentals of neural computing in 1943 for the first time. Introducing examples to the ANN, it can learn the relationships through a training process. In fact, this technique is based on simulating the way that the brain processes different data. Therefore, they are able to perform behaviours similar to brain activities, such as: learning, generalisation, categorisation, association, optimisation and feature extraction. Network topology or structure is the way neurons are connected to each other. Common developed topologies include networks with feedback connections called recurrent networks and feed-forward networks which have only direct connections (M. Raudenský 1996). An important advantage of an ANN is its ability to learn. It can learn relationships and patterns between two sets of variables. Consequently, rather than using an analytical relationship derived from mechanical principles to model a system, the ANN employs an adaptive training process to learn the relationship between variables.

The most suitable applications for ANNs have the following characteristics (Lee et al. 1999):

1. A large database is available.
2. It is difficult to find an accurate solution to the problem by existing mathematical approaches.
3. The dataset is incomplete, noisy or complex.

ANN have attracted considerable attention and shown promise for modelling complex nonlinear relationships. This technique is based on the human brain having a number of interconnected units (Neurons) (Schalkoff 1997). Artificial Neural Networks look promising and have been used extensively in many fields (Amali et al. 2000; Amali et al. 2006; Cao et al. 1998; Rao and Rao 1993; Rosenblatt 1958; Sewell et al. 2010; Shar and Palmieri 1990; Soo-Young et al. 2006; Xu et al. 2010; Ziemianski and Harpula 1999). For instance, ANN is widely used for damage identification (Ziemianski and Harpula 1999). Employment of such novel techniques allows the structural health during service to be predicted, without the need to interrupt or terminate the usage of the structure. This is due to the fact that when an ANN is trained and put in service, no additional information about current environment and structure status (material, shape, etc.) is required. In a case study on an aircraft structure, it has been proven that a strain-based damage prediction methodology employing ANN leads to an accurate crack damage pattern recognition, which has an error comparable to that of the conventional methods. The main advantages of the methodology are its accuracy, reliability, independency from external load variations or any experimental data and its easiness to be practically applied to on-line System Health Monitoring (SHM) methods (Katsikeros and Labeas 2009). Furthermore, the ANN method has indicated promising efficiency for the identification and localisation of imperfections (Efstathiades et al. 2007; Katsikeros and Labeas 2009).

According to a review on applications of ANN on composites by Zhang and Friedrich (2003), the applications of ANNs to polymer composites are still in their basic stages and the investigations of the implementation possibilities in different material research areas and the improvements of the predictive qualities are still matters for further research. Fatigue is one of the most complicated problems for fibre composites, and failure mechanisms are still not well understood. Extensive tests must be carried out because of the absence of a well-defined failure criterion that can be used to predict fatigue failure in polymer composites. ANNs offer the possibility of developing models that will predict the behaviour of composites without being linked to mechanistic arguments. They have, therefore, been introduced to predict fatigue life by Lee et al. (1999), Aymerich and Serra (1998), Al-Assaf and El Kadi (2001) and El Kadi and Al-Assaf (2002). Other publications on fatigue prediction in various metallic alloys using the ANN approach are recommended to the composite community as further references to improve this technique

for their problems (Haque and Sudhakar 2001; Pleune and Chopra 2000; Venkatesh and Rack 1999). Velten et al. (2000) and Zhang et al. (2002) were among the earliest pioneers to explore this approach in polymer composites, using ANNs to predict the wear volume of short-fiber/particle reinforced thermoplastics.

Applying artificial intelligence to the ship/boat has been limited to use of ANN as an AI technique for only the preliminary ship design techniques (Rosenblatt 1958; Shar and Palmieri 1990). Research proposed by Soo-Young et al. (2006) presents a new method to classify surface plates effectively in preliminary ship design using Neural Networks as an AI method.

Considering the literature review so far, the most relevant research performed in monitoring of various structures are the SHM methods where a direct structure response is utilised to analyse and monitor the structure status and integrity (Katsikeros and Labeas 2009; Ko and Ni 2005; Li et al. 2006; Lynch et al. 2004; Majumder et al. 2008; Zingoni 2005).

Efstathiades et al. (2007) developed a method for structural health monitoring and fault detection in curtain-wall systems. In this study, ANN was utilised and it was proved to be an efficient method for identification and localisation of imperfections in these systems.

In a study by Cao et al. (1998) an approach was developed to identify the loads acting on aircraft wings where an artificial neural network was utilised to model the load-strain relationship in structural analysis. The research demonstrated that using an artificial neural network to identify loads is feasible and a well-trained artificial neural network reveals an extremely fast convergence and a high degree of accuracy in the process of load identification for a cantilevered beam model.

In a study by Amali et al. (2000), it is illustrated that AI as a modern mathematical method can be mixed by the requirement for knowledge of material properties and component geometry to create a hybrid inverse problem analysis tool or Inverse Problem Engine. The hybrid approach can be applied to both direct and inverse problems that avoid the need for having component geometry and material properties at the end stage of problem solving (Amali et al. 2006). Furthermore, as a medical application, ANN

technique is successfully utilised as an inverse method by Sewell et al. (2010) to estimate pressure distribution at the residual limb/socket interface of a lower-limb prosthesis.

In the case of complex structures, the relation cannot easily be formulated (Tang et al. 1995). As vessels and crafts in most cases, are extremely complicated structures thus the mechanical properties or relations between external excitement and structural responses are difficult to formulate, so an appropriate load monitoring system has to be developed (Cao et al. 1998).

Different properties of the system can be monitored such as any change in system physical properties, stress, strain, local or distributed loads. Load data can be employed to locate damage, detect composite delaminating, measure residual stress recoveries. To measure each of them, the appropriate transducer should be employed.

In order to perform load monitoring of a structure, a specific sensing system is required. Sensing technology has witnessed significant progresses in the past decades and recently some types of innovative sensing systems are now commercially available such as load cells, fibre optic sensors and wireless sensors (Lynch et al. 2004; Spencer et al. 2004; Sumitro and M.L. 2003).

Fibre optic sensors have successfully been applied for long-term structural health monitoring like large scale bridges. Furthermore, Fibre Bragg grating (FBG) sensors have been employed for monitoring, diagnostics and control in structures and the performance was reliable. One of the main beneficial aspects of FBG sensors to other technologies is its versatility in the structural sensing area (J.M. Ko July 2005; Mufti Aa 1997).

FBG sensors are appropriate solutions in longitudinal strain measurement in static and dynamic strain sensing in a number of application areas. According to Mousumi Majumder (2008), this is due to the inherent properties of FBG sensors such as:

- Lightweight
- Immunity to electromagnetic interference and harsh environment
- Ability to be multiplexed for distributive measurement
- lower integration costs with new shipboard optical network backbones

- optical sensors are self-referencing enabling long-term absolute strain measurements without the need for frequent sensor calibrations and bridge balancing such as that required by Electrical Resistance Strain Gauge (ERSG) sensors

Modern composite laminate hulls have sensors integrated within them during manufacture. Stresses and strains generated due to the loads acting on a craft are used to monitor structural integrity, and also used to monitor stresses at key locations where there are geometrical or material discontinuities. This system provides real time information concerning the state of the ship structure and increases operational attainability and ship service life and can lead to reductions in on-going maintenance costs (Baldwin et al. 2002).

To reduce overall maintenance costs, ships have been equipped with strain gauges so information for estimating the structural health of ship hulls can be predicted when they are in-service. To date, some naval ships have become able to collect data with this embedded instrumentation and the data can be employed to validate computer models anticipating damage accumulation over the expected service life of the vessel without actually calculating the exerted load on the structure (Adamchak 1984; Baldwin et al. 2002; Sikora et al. 1983).

Ideally, load monitoring systems should be constructed of an array of inexpensive, spatially distributed, wirelessly powered and networked, embedded sensing devices which support frequent and on demand acquisition of real-time information about the loading and environmental effects, structural characteristics and responses (Ko and Ni 2005). Different responses of the system can be monitored directly and related to applied loads such as any change in the system's physical properties, stress, strain and local or distributed pressure. To measure structure response to a load, the appropriate transducer should be employed.

### **1.3 Summary**

In summary, current practices to predict hydrodynamic loads rely either on oversimplified and semi-empirical methods or the use of numerical simulation and analysis techniques. These methods are conservative which results in the over-design of these craft so they are heavier and slower than they could otherwise be. Experimental data acquired from sensors embedded in marine structures informs only on the structure response on load

and it does not give direct measurement of the load. To conclude, a novel technique is required to overcome current limitations in practices used to measure loads employed in marine structure design procedures.

The literature shows that there is a novel approach that has been used to relate response to loads in other engineering fields, such as system health monitoring and damage detection. The inherent properties of ANN, such as learning of the relations, makes it a suitable potential candidate to be used in an in-service load monitoring system for marine structures. Through such a method, knowledge of the component material constitutive laws or detailed knowledge of the component geometry is not required. In fact the aim of this inverse problem analysis approach is to find a function that relates the inputs to outputs considering these parameters. This technique may be able to offer a fast solution for monitoring in-service loads on marine structures when the relationship between the loading and response of the structure has been established. The final system will immediately predict the load by analysing any similar new response data introduced to the solver and no great computational effort is required. However, it is required to investigate the capabilities of such a system in the determination of in-service, transient loads in real-time and produce a load history of the vessel.

Although there have been some works for load prediction employing AI in various areas such as aircraft wing or prosthetics, the literature has shown that there has been no relevant application of AI in hydrodynamic load monitoring of marine structures. In other words based on this literature review, in-service load monitoring of marine structures to influence their design is novel research. The Author hypothesises that AI can be employed to find an inverse solution to predict hydrodynamic loads applied on marine structures which may be used to inform designers in the stage of preliminary ship design. In this research, the aim is to employ and investigate the ability of this approach to immediately predict/estimate the load by analysing any similar new response data introduced to the solver. This brings the following sub-questions to be answered from the research question:

- Can in-service loads be predicted on a vessel to inform design?
- How to measure in-service loads on a vessel or craft representative?

- Can Artificial Intelligence be used as an inverse method to relate acquired craft responses and what load caused the responses?
- Which types of loads can be measured with this method?
- How to validate the predicted loads?
- Is it possible to optimise the number of sensors and improve the accuracy of the method?

In order to investigate the applicability of the new system in online load monitoring of marine structures such as Rigid Inflatable Boats (RIB), some critical objectives are recognised and categorised to be investigated. This includes following items:

- General behaviour of a marine structure with attached strain gauges under small displacement (static load is applied)
- General behaviour of a marine structure with attached strain gauges under large displacement (static load is applied)
- Optimisation of the sensors quantity
- General behaviour of the marine structure with attached strain gauges under transient load condition (dynamic load is applied)
- Develop a Graphical User Interface (GUI)

## **Chapter 2. Introduction to Methodology and Artificial Neural Networks (ANN)**

This chapter introduces the research methodology. This chapter also covers the background to Artificial Neural Networks.

## 2.1 Introduction

Chapter one led to the hypothesis that utilising a hybrid approach that combined experimental technique, FEA, and AI can provide a general purpose inverse problem solver that is capable of solving most classes of linear problems. Once an ANN is trained (converged) for a given problem it can be used to link the input to the output without any processing time.

This chapter proposes a methodology to answer the research question which is can external forces be estimated by the ANN using the structure's response as input. This chapter also gives more detailed information about the ANN used in the proposed methodology.

Estimating in-service loads on marine crafts is highly desirable. Currently there are no tools that can estimate the intensity and location of such loads. A novel methodology is proposed that, if successful, can transform every marine structure into a transducer capable of estimating any in-service transient loads. Better understanding of load intensities will inform the design process and results in lighter and more efficient designs.

The philosophy is based on detailed investigation of different parameters used in the Hybrid Inverse Problem Engine (HIPE) and their effect on the integrity of the solution when applied to large structures.

Marine vessels, depending on their size, can generally be manufactured from a large number of flat, shaped or reinforced panels. Better understanding of their response to external load can affect the design, dynamics and structural integrity of the vessel.

This research begins with the investigation of the application of HIPE for determining external load on a large flat marine panel and investigates (as explained in section 1.2.1) the performance and suitability of using HIPE to monitor large structures. Here a large marine composite panel is instrumented and calibrated and programmed to function as a complex load cell capable of measuring both surface and reaction loads.

In summary, as a quantitative research, the following steps have been carried out:

- Design and develop a general purpose hybrid inverse problem solver suitable for use in large marine structures.

General understanding of the overall boat behaviour and its critical points is needed. The appropriate data acquisition should be investigated. The material behaviour in terms of linearity or nonlinearity needs to be investigated. Also, the most suitable AI method and its parameters need to be investigated. Based on the literature review in Chapter 1, it is important to consider the prediction of transient loads.

- Validate and Modify the method:

The suitability of the proposed methodology will be indicated by successful prediction of the applied loads. Finally, a real time GUI should be developed allowing better communication with the system and to change different parameters used for data acquisition and to control ANN convergence rate etc. The performance of the load monitoring system will be evaluated by performing various tests by means of applying this system on a marine structure representative such as a cross section of the boat hull.

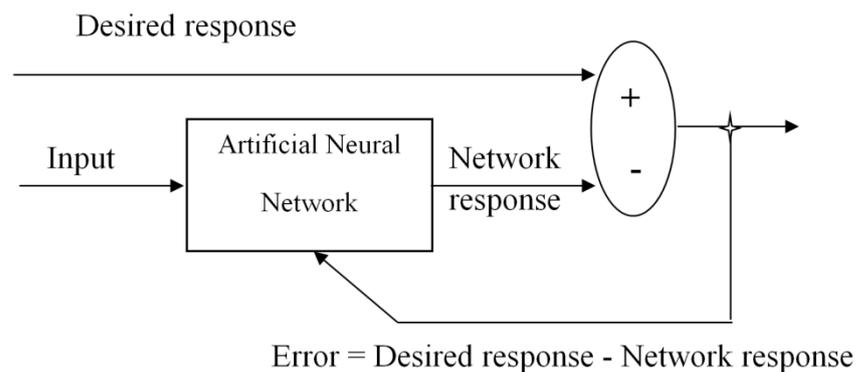
## **2.2 Description of the Methodology**

The basis of the inverse problem analysis is to relate external load or boundary conditions at specified locations to the amount of structural response at pre-determined locations on the structure. In other words, loads are to be identified from the known structural responses. Efstathiades et al. (2007) outline that throughout the past decade, AI presented techniques are useful for solving inverse problems. In a study by Xu et al. (2010), it is indicated that many fields of science and industry employ inverse analysis such as material property estimation, radar tracking, medical tomography, residual stress determination, oil reservoir identification and non-destructive testing. The basis of the inverse technique is upon determining a relationship between causes and their effects. In this study, finding the static or hydrodynamic loads (the cause) on marine structures such as small craft is the main aim of the inverse problem. This is achievable by acquiring repeatable structural response to the load such as strain (the effect) at one or more interior points.

The ANN has to be trained so a specific input (strain as the structural response) leads to a particular output (load). There are two basic paradigms of learning, supervised and unsupervised, both of which have their models in biology. Supervised learning is a process where both input vectors, and the desired outputs are given. Unsupervised learning works

only on the input vectors, and the desired output is not specified. This learning method is commonly employed for adaptation to specific features, process of categorisation, or discovering regularities.

In an ANN context, supervised learning is a process of memorising vector pairs. The input vector together with the desired output vector is known. Training is based on adjusting some weights and biases. This is achieved by comparison of the network response and the desired response. This adjustment continues as a loop (see Figure 2-1) until the network response matches the desired output or at least the error function becomes an acceptably small value. Generalisation is a feature of the trained network working similar to approximation. In other words, a well-trained network estimates appropriate output data even for untrained patterns. Although training the network is time consuming, the trained network operates quickly in an application process.



**Figure 2-1: Artificial Neural Network training loop to minimise error (Sewell et al. 2010)**

A network is constructed of one or multiple simple neurons. Some neurons can be put together in a layer, and a typical network may have one or more such layers. The response of a network is calculated from each individual element input. In a common feed-forward network, inputs are multiplied by weights (a bias may be added here as well) and a transfer function acts on it.

A simple and popular network is a two-layer feed-forward network with three layers named as: input layer, hidden layer and output layer, as shown in Figure 2-2. In feed-forward networks normally one or more hidden layers whose transfer function is log-sig type is connected to an output layer with pure-lin transfer function (see section 2.3.2.2 for detailed description of transfer functions). Having multiple hidden layers of neuron with

nonlinear transfer functions enables the network to understand both nonlinear and linear relationships between input and output vectors.

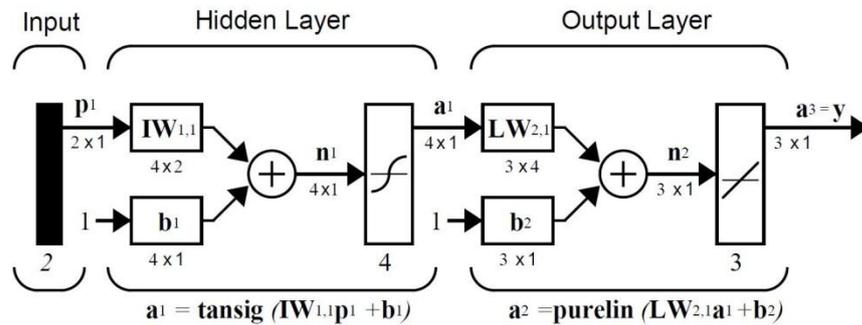


Figure 2-2: A two-layer tansig/purelin feed-forward network (Hagan et al. 2002)

Back-propagation is one of the most popular ANN in use today. It is simple to understand and it works well generally as a classifier or for non-linear continuous multivariate function mapping. Mean Squared Error (MSE) is normally employed as the back-propagation error function. The squared errors are calculated as differences between the desired and network response. As the network weights approach a minimum solution in the training stage of the network, the gradient becomes small and the step size reduces as well. This results in slow convergence. Usually a factor called ‘momentum factor’ is added that makes this process faster. This also normally reduces the possibilities of getting stuck in a local minimum. Pre-processing the input data before training can reduce the training time.

A sufficiently trained ANN can be employed as an analytical tool on other data that it has not previously seen or been trained with. This stage is called simulation and at this point the output is retained and no back-propagation occurs. A properly trained ANN will perform well as an interpolation tool, but usually has a very poor extrapolation performance. Depending on how well the ANN is trained, it may have very good or poor approximation capabilities. Usually, a well-trained ANN with good generalisation capabilities performs well for both interpolation and approximation. Unfortunately there is no general rule to find out if a new pattern is within the interpolation space (Helliwell et al. 1995) and to have a measure for the generalisation ability is very difficult. Each data set determines how good the generalisation may be and what for one purpose can be satisfactory may be useless for another (Holden and Rayner 1995).

Deciding on training stoppage time is difficult. Most of the time, having a network with a small MSE, does not necessarily indicate acceptable predictions on new data from the same domain. This problem in training is called over-fitting (overtraining). In general, this problem happens if the trained ANN is more accurate in fitting known data but less accurate in predicting new data. As it is indicated in Figure 2-3, Training error is shown in blue and validation error in red where both of them are a function of the number of training cycles. If the validation error upsurges (positive slope) while the training error steadily decreases (negative slope) then an over-fitting situation may have occurred. In this case, the training process goes on too long and the network is biased towards the training set. This practically reduces the generalisation ability of the ANN. In contrast, stopping the training process too early causes the decision to be very rough. Figure 2-4 illustrates the various possibilities. In order to improve the generalisation of the ANN a set of noisy data generated from the original data can be introduced in the training process. This also helps to prevent the possibilities of over-fitting (Mackay 1992).

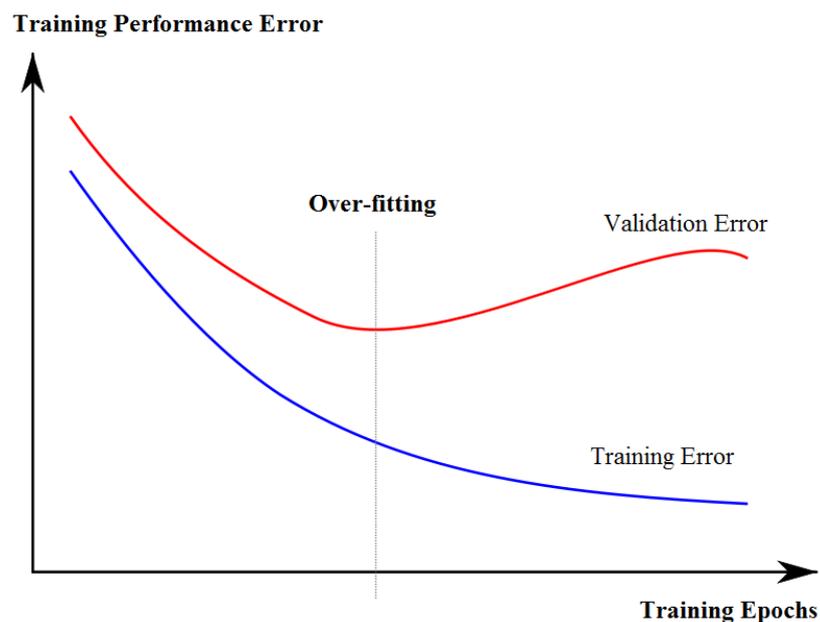
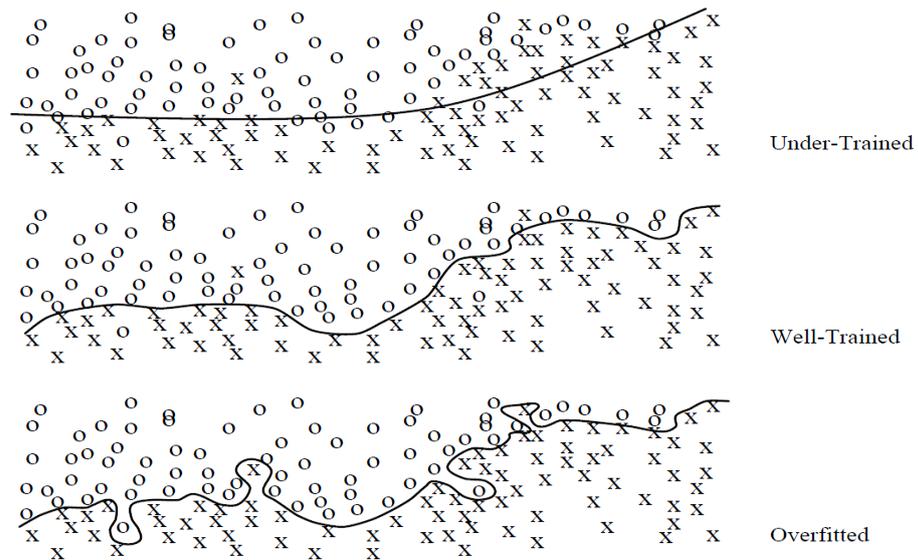


Figure 2-3: Over-fitting problem



**Figure 2-4: Learning and the problem of over-fitting (Schmidt 1996)**

For the proposed inverse problem solver to work efficiently the response of the structure to load, must remain linear. However, nonlinear structures are also considered in later chapters. The success of a good ANN as depends on the following key points:

- Accurate data collection
- Pre-processing acquired data
- Selection of a suitable ANN type and topology
- Selection of optimised ANN training parameters
- Testing and validation of ANN with new data set

When dealing with ANN, it is possible that the output may not be adequate on the first design pass. In such situations, the process steps have to be repeated one or more times until the problems are solved. On some occasions the ANN does not exhibit the satisfactory performance. This can be due to a wide range of reasons needing to be investigated such as:

- Unsuitable ANN architecture or learning method
- Insufficient representative data
- Inadequate pre processing
- Unsuitable ANN training parameters

However most of the time this is not the case and the ANN will be well trained and performs satisfactory even on a new untrained data set. Figure 2-5 indicates an ANN problem solving flowchart.

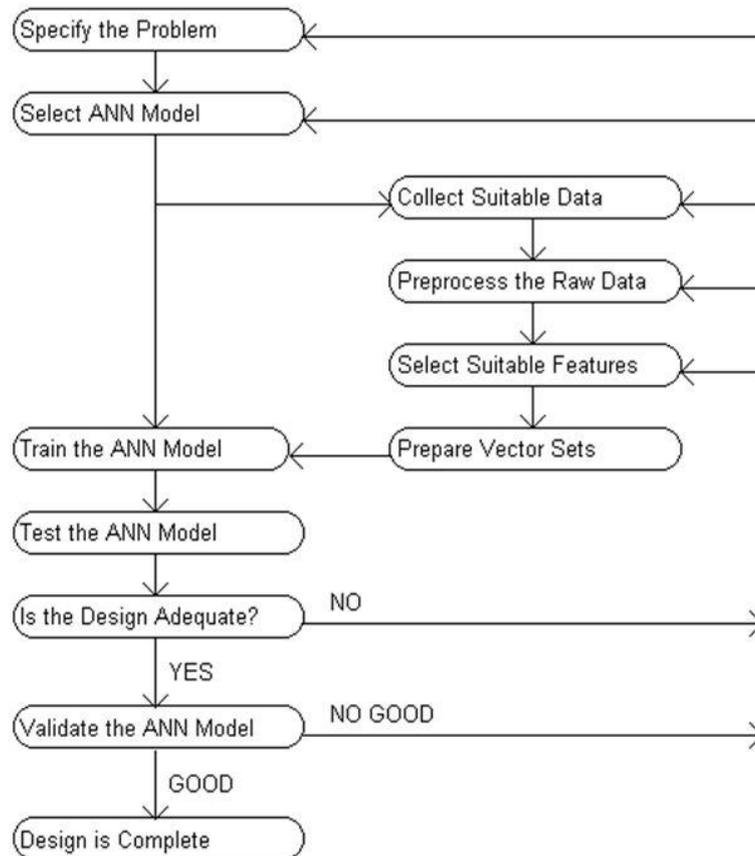


Figure 2-5: Example of ANN problem solving model (Zaknich 2003)

## 2.3 Introduction to Artificial Neural Network

In this section, after defining ANN, a brief historical overview of ANN is described. Afterwards, simple neuron models as well as common ANN architectures and training algorithms are introduced. Furthermore, in this section some important hints and general rules in design of a successful ANN model are presented. Finally, the ANN software employed in this research is introduced.

### 2.3.1 Definition of Artificial Neural Network

Artificial Neural Networks is one of the most popular methods of Artificial Intelligence composed of simple elements inspired by brain biological nervous systems operating in

parallel. In fact, the function of the network is principally determined by the connections between these elements. A neural network can be trained to achieve a specific purpose by modifying some weights (connection values) of the neurons (elements). In general, the ANN is trained in order to relate a specific input to a particular target output (Figure 2-1). During the training process, output values are compared with the target ones in an iterative manner until the target and the network output values are matched as much as possible. Generally, in order to train a network enough pairs of input and target values (examples) are required.

ANN can be considered as a black box which is employed to process information in the way that inputs are taken and outputs are produced. Once an ANN is properly trained, it can be utilised to do even complex functions in various fields such as learning, generalisation, pattern recognition, non-linear system modelling or inverse modelling, feature extraction, categorisation and optimisation. Indeed, a trained ANN is normally used in applications which are challenging for conventional mathematical methods and the problem data exhibit nonlinearities, high dimensionality and more often noisy, complex, imprecise and imperfect sensor data are available.

Haykin (1994) has a more official definition of an ANN as follows:

"A neural network is a massively parallel-distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1. Knowledge is acquired by the network through a learning process.
2. Inter-neurone connection strengths known as synaptic weights are used to store knowledge."

Another definition according to Zurada (1992) is:

"Artificial Neural systems or Neural Networks (NN) are physical cellular systems which can acquire, store, and utilise experiential knowledge."

A Brief Historical Overview

The history of ANN from their roots starting in 1890 to 1987 is described in a work by Anderson and Rosenfeld (1988). In summary, the first person to publish about the brain structure and its function was James (1890) which was more related with neuropsychological research and psychological theories. However, According to Kohonen (1988), the fundamentals of neural computing were shaped from the theories of Mcculloch and Pitts (1943). Hebb (1949) defined a technique to update synaptic weights, which is now called the Hebbian learning method. The ANN was further developed by works of Farelly and Clark (1954), Rosenblatt (1958) and Caianiello (1961). In fact, it was Rosenblatt who defined one of the most common ANN structures called the perceptron. His model was simulated employing an IBM 704 computer which became famous as the "learning machine".

The growth in computer technology in the 1960s was influential in the overall development of artificial neural models. Nilson (1965) published a book in which he organised various ANN research of the time. It is still believed that the ADALINE (ADaptive LInear NEuron) introduced by Widrow and Hoff (1960) is one of the most important ANN developments from an engineering point of view. This is due to the fact that ADALINE employed a much faster learning algorithm called the least mean squares algorithm. This algorithm is still being used in today's multi-layer perceptron.

In 1972, researches by Kohonen (1972) and Anderson (1972) on associative memory, laid the groundwork for the development of the unsupervised learning or self-organising networks. In 1982, Kohonen published his self-organising map method (Kohonen 1982). In the late 1970s, a model for the visual pattern recognition mechanism emerged called the NEOCOGNITRON (Fukushima 1980, 1982, 1986; Fukushima et al. 1983).

Several noteworthy publications, introduced from 1982 to 1986, had a major effect in the development of ANN technology. Hopfield (1982) published a paper in 1982 and a follow-on paper in 1984. These papers introduced ANN structures with the capability of being generalised with a high degree of robustness. The first International Joint Conference on Neural Networks was held in 1987 and since then there has been many research interests in ANN theory and its applications. This era was named as the "Age of Neoconnectionism" by Cowan and Sharp (1988). From then, a number of regular journals about ANN started such as Neural Computation by MIT Press, the IEEE Transactions on

Neural Networks and the Neural Networks Journal of the International Neural Networks Society.

Nowadays ANNs are used to solve a wide variety of complex scientific and engineering problems. ANNs can learn from examples, and therefore can be trained to find solutions of the complex non-linear, multi-dimensional functional relationships without any prior assumptions about their nature; further, the network is built directly from experimental data by its self-organizing capabilities.

Various applications of ANN are outlined in many recent papers (Capecchi 2010; Coello Coello and Lamont 2004; Liu and Li 2004; Livingstone 2008; Pan 2012; Panigrahi et al. 2010; Rudas et al. 2010; Schumann and Liu 2010; Sivakumar et al. 2010; Xu and Wunsch 2009; Yang 2010; Yu 2013; Zainun 2011; Zeng and Wang 2010).

Details about the theory and mathematics behind the neural networks can be found in several textbooks (Aleksander and Morton 1990; Beale et al. 2012; Bishop 1995; Fausett 1994; Haykin 1994; Swingler 1996). An introduction to the principles of ANN is outlined below.

### **2.3.2 Neuron Model**

In this section a simple neuron model is described.

#### **2.3.2.1 The Basic Model of the Neuron**

A single input neuron is the most basic building block in an ANN. Figure 2-6 indicates a schematic of a single input neuron. Three different functional operations exist in a single neuron. First, there is a multiplication between two scalar values, input  $p$  and the scalar weight  $w$ , to form the product  $wp$ , which will be a scalar value as well. This process is called the weight function. Second is a summation between two scalar values, the weighted input  $wp$  and bias  $b$ , to form the net input  $n$ . Generally, the bias can be considered as a shift of the function  $f$  to the left by an amount  $b$ . The name given to this process is the net input function. Finally, the net input is passed through the transfer function  $f$ , to form the scalar output  $a$  (Equation 1). The last process is called the transfer function.

$$a = f(wp + b) \quad (\text{Eq. 1})$$

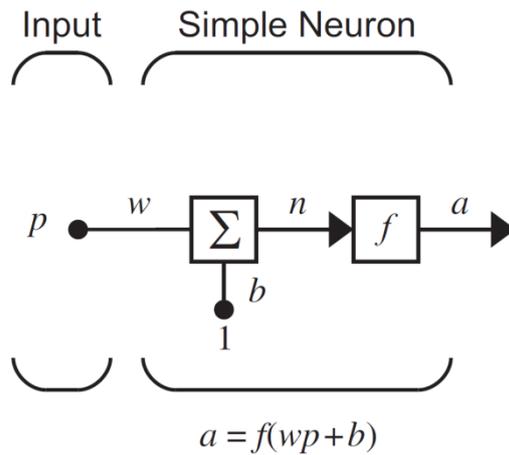


Figure 2-6: Schematic of a single input neuron (Beale et al. 2012)

### 2.3.2.2 Transfer Functions

A linear or a nonlinear function of  $n$  can be used as the transfer function in Figure 2-6. According to the particular specifications of the problem, each neuron may have its individual specific transfer function. Many transfer functions are employed in various neurons; however, the following three are the most commonly utilised:

1. The linear transfer function

In the final layer of multilayer networks, usually a linear transfer function is employed. This is mostly due to the fact that the output of such a transfer function is the same as its input. In other words, considering Figure 2-6, the relation is calculated from Equation 2. This transfer function is generally used in many applications however it is particularly common in function fitting problems. Figure 2-7 illustrates the graphical representation of this transfer function. Most of the time, a transfer function is represented using the symbol which it is indicated in the square to the right of the graph. The general  $f$  in Figure 2-6 can be replaced by this symbol in the network diagram blocks to allow better representation of the specific transfer function being used for each neuron.

$$a = n \tag{Eq. 2}$$

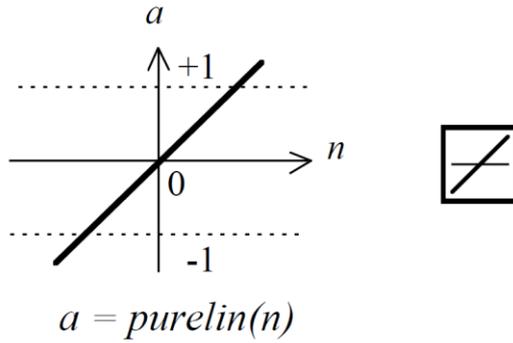


Figure 2-7: Linear transfer function (Beale et al. 2012)

## 2. The Log-sigmoid transfer function

Another common transfer function is the log-sigmoid type which is often employed in the hidden layers of multilayer networks. Due to the fact that this function is differentiable, usually, the training algorithm used to train the neurons having log-sigmoid transfer function is a powerful algorithm called back-propagation.

The argument of this transfer function can be any value between plus and minus infinity, and the output will be scaled to a specific value in the range of 0 to 1 according to Equation 4. The graphical representation of this function is indicated in Figure 2-8.

$$a = \frac{1}{1+e^{-n}} \quad (\text{Eq. 4})$$

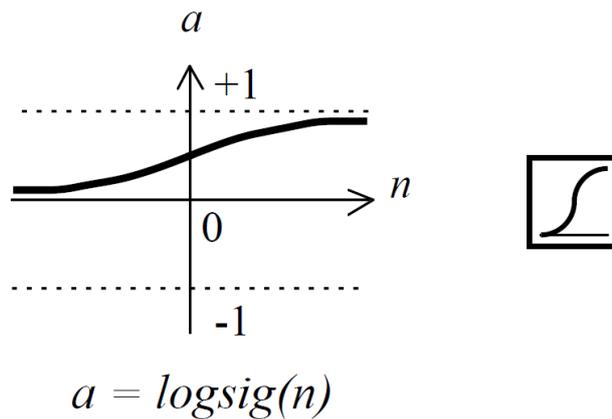


Figure 2-8: Log-Sigmoid transfer function (Beale et al. 2012)

## 3. The tan-sigmoid transfer function

Instead of the above functions, a tan-sigmoid transfer function may be used as the transfer function of the neurons. This function is often employed in the output layer of

multilayer networks designed for pattern recognition problems. The tan-sigmoid transfer function is very similar to the log-sigmoid one and its argument can be any value between plus and minus infinity. However, the output range is from -1 to +1 according to Equation 5. The graphical representation of this function is indicated in Figure 2-9.

$$a = \frac{e^n - e^{-n}}{e^n + e^{-n}} \quad (\text{Eq. 5})$$

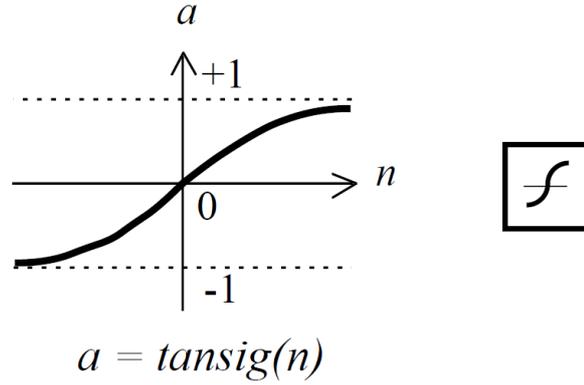


Figure 2-9: Tan-Sigmoid transfer function (Beale et al. 2012)

### 2.3.2.3 Neuron with Vector Input

The simple neuron indicated in Figure 2-6 can be easily extended in order to handle vector inputs. Figure 2-10 indicates a single neuron with an R-element input vector. Having a vector input, each input element is multiplied by its individual weight and the weighted values are fed to the summing junction. The multiplication is the dot product of the single row matrix  $\mathbf{W}$  and the vector  $\mathbf{p}$  which will be simply  $\mathbf{Wp}$  (Equation 6). As in single input neuron with single input, one scalar is achieved from this dot product. This result is then added to the bias  $b$  of the neuron to form the net input  $n$ . As before, the transfer function  $f$  acquires the net input  $n$  as its argument and the neuron output  $a$  will be the output of this transfer function (Equation 7).

$$n = \mathbf{Wp} + b = \begin{bmatrix} P_1 \\ P_1 \\ \vdots \\ P_R \end{bmatrix} [W_{1,1} \quad W_{1,2} \quad \dots \quad W_{1,R}] + b = \text{scalar} \quad (\text{Eq. 6})$$

$$a = f(\mathbf{Wp} + b) = f(n) \quad (\text{Eq. 7})$$

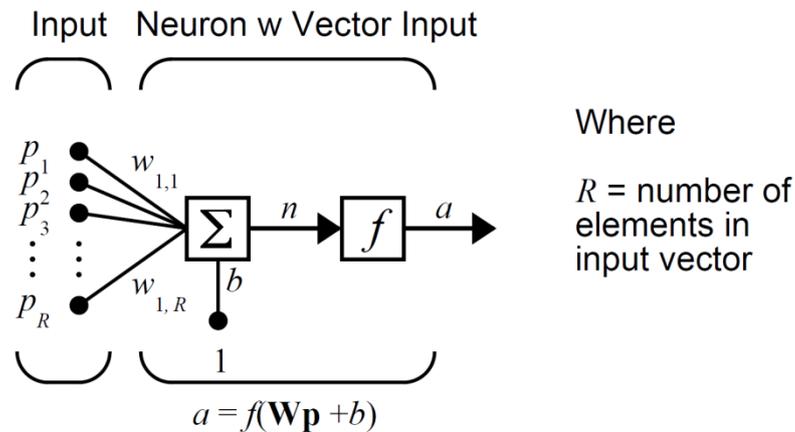


Figure 2-10: single neuron with an R-element input vector (Beale et al. 2012)

### 2.3.2.4 Feedforward Neural Network Architectures

The multilayer feedforward neural network is the workhorse of many ANN problems and is used commonly for general function approximation, pattern recognition and function fitting problems. In fact, a fairly simple feedforward neural network can fit any practical function. In this section, the architecture of the feedforward neural network as one of the most common neural network architectures is described. A layer of neurons is created by combination of two or more single neurons. A specific neural network may have one or many more of such layers. In the following, a single layer feedforward neural network and a multiple layer feedforward network will be described.

### 2.3.2.5 Single Layer Feedforward Neural Network

A single layer neural network which has  $R$  input elements and  $S$  neurons is illustrated in Figure 2-11. Here, the same input vector  $\mathbf{p}$  is introduced to a layer of neurons and therefore each element of it will be connected to each neuron input through the weight matrix  $\mathbf{W}$ . The multiplication of  $\mathbf{W}\mathbf{p}$  would be a vector matrix. Each neuron has a bias  $b_i$  which will be added to  $i_{th}$  row of the vector matrix result of the multiplication  $\mathbf{W}\mathbf{p}$  to form its own scalar output  $n_i$ . The various  $n_i$  taken together form an  $S$ -element net input vector  $\mathbf{n}$  (Equation 8).

$$\mathbf{n} = \mathbf{W}\mathbf{p} + \mathbf{b} = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_R \end{bmatrix} \begin{bmatrix} W_{1,1} & W_{1,2} & \dots & W_{1,R} \\ W_{2,1} & W_{2,2} & \dots & W_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ W_{S,1} & W_{S,2} & \dots & W_{S,R} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_s \end{bmatrix} = \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_s \end{bmatrix} \quad (\text{Eq. 8})$$

At the end, the transfer function vector  $\mathbf{f}$  acquires the net input vector  $\mathbf{n}$  as its argument and forms the neuron layer output to be a column vector  $\mathbf{a}$  (Equation 9).

$$\mathbf{a} = \mathbf{f}(\mathbf{W}\mathbf{p} + \mathbf{b}) = \mathbf{f}(\mathbf{n}) \quad (\text{Eq. 9})$$

It should be noted that most of the time the number of inputs to a layer is different from the number of neurons. In other words  $R$  is not necessarily equal to  $S$ .

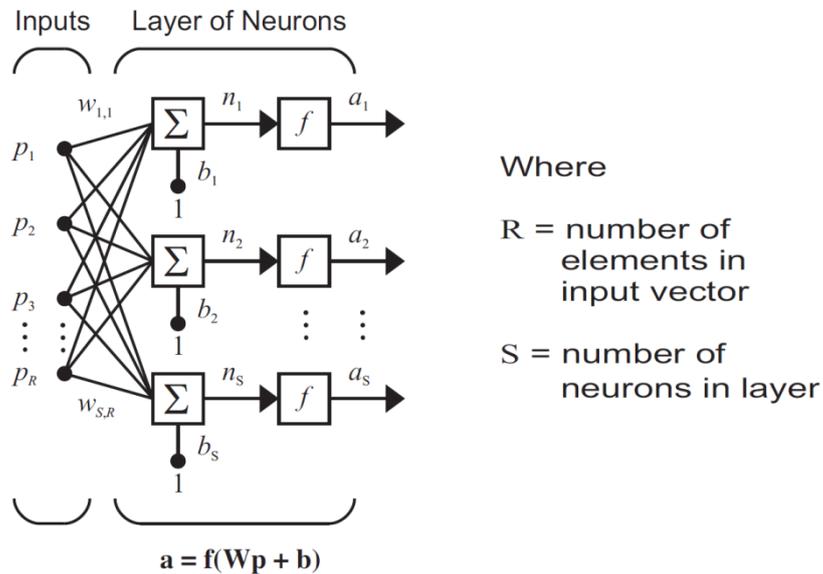


Figure 2-11: Single layer neural network (Beale et al. 2012)

### 2.3.2.6 Multiple Layers Feedforward Neural Network

A neural network can be designed to have more than one layer. In this case, each layer is like a single layer network and has a weight matrix  $\mathbf{W}$ , a bias vector  $\mathbf{b}$ , and an output vector  $\mathbf{a}$ . However, in order to distinguish  $\mathbf{W}$ ,  $\mathbf{b}$ ,  $\mathbf{a}$ , etc. for each of these layers, a particular superscript is added to the notation of each variable. The utilisations of notations for a three-layer network are indicated in Figure 2-12 and equations (Equation 10-13).

$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{I}\mathbf{W}^{1,1}\mathbf{p} + \mathbf{b}^1) = \mathbf{f}^1(\mathbf{n}^1) \quad (\text{Eq. 10})$$

$$\mathbf{a}^2 = \mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{a}^1 + \mathbf{b}^2) = \mathbf{f}^2(\mathbf{n}^2) \quad (\text{Eq. 11})$$

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\mathbf{a}^2 + \mathbf{b}^3) = \mathbf{f}^3(\mathbf{n}^3) \quad (\text{Eq. 12})$$

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3) \quad (\text{Eq. 13})$$

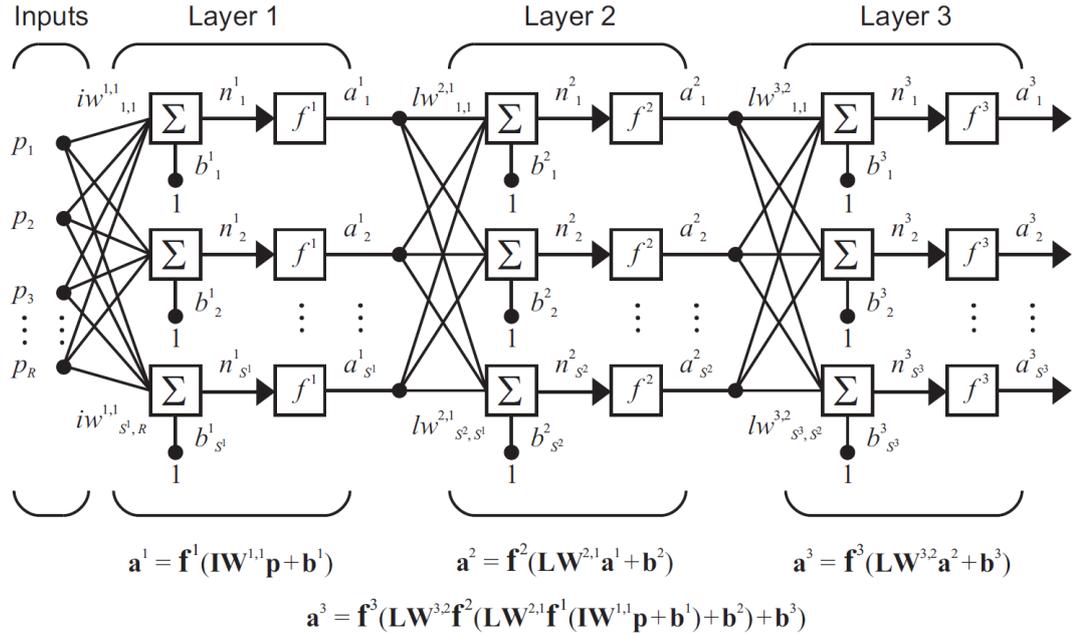


Figure 2-12: Multiple Layers Neural Network (Beale et al. 2012)

The layers of a multilayer network have different names. The first layer is called the input layer. A layer that produces the network output is named an output layer. All other layers are called hidden layers. The network illustrated in Figure 2-12 has one input layer which has  $R$  inputs, two hidden layers and one output layer. The first hidden layer has  $S^1$  neurons, the second hidden layer has  $S_2$  neurons and the third hidden layer has  $S_3$  neurons. It is common for different layers to have different numbers of neurons. In a multiple layer neural network the outputs of each intermediate layer are the inputs to the following layer. Therefore, hidden layer 2 can be considered as a one-layer network with  $S^1$  inputs,  $S^2$  neurons, and a  $S^1 \times S^2$  weight matrix  $W^2$ . The input to layer 2 is  $\mathbf{a}^1$  and the output is  $\mathbf{a}^2$ . The output layer can be treated in the same way. This means that the input is  $\mathbf{a}^2$  and the output would be  $\mathbf{a}^3$ .

Hidden layers in a feedforward network usually have log-sigmoid or tan-sigmoid transfer functions. However, most of the time the output layer of the feedforward architectures has the linear transfer function (Darpa 1988). Having multiple layers of neurons with nonlinear transfer functions (sigmoid type) allows the network to learn nonlinear relationships between input and output vectors. The transfer function of the output layer is most often set to be linear for function fitting (or nonlinear regression) problems. Alternatively, in the case of pattern recognition problems it is more common to use the sigmoid transfer function as the output layer transfer function (Lippman 1987).

### **2.3.3 ANN Types Based on Learning Approaches**

The three main learning approaches used in various artificial neural networks are: Supervised learning type, Reinforcement learning type and Self-organising (Unsupervised learning) type (Kohonen 1987, 1997; Moller 1993). The first and most common learning method for ANN is supervised learning. In this type, the ANN is trained using some representative examples in the form of paired inputs/outputs. In this type of learning process, the error between the actual network response and the desired one is computed during each of the training iterations and the magnitude of this error is used to make modifications to the network's weights according to a specific learning algorithm. Weights are updated in each of the learning iterations so that the error value is gradually reduced until it reaches a minimum or at least a reasonably small value. A schematic representation of such learning process is indicated in Figure 2-1.

Another learning method is the reinforcement learning type. In this method, computing the precise error between the actual network output and the desired response is not needed. Instead, the teacher gives a pass or fail signal for each of the training iterations. In the case of the fail signal, the network is designed to continue the modification of its parameters until it reaches a pass signal or to continue for a fixed number of attempts, whichever comes first. Some have categorised this reinforcement learning method as a special case of the supervised learning method (Murray et al. 1992).

Finally, in the self-organising type, examples of the inputs are introduced to the ANN and it will arrange them to form automatic various clustering or inter groupings according to some previously defined measure of similarity or closeness. This will make it possible to

assign specific information to those clusters based on the nature of the problem (Lippman 1987).

### **2.3.3.1 Multilayer Networks and Back-propagation Training**

One of the most common and popular ANN models among many specific ones is the Back-propagation Neural Network. The popularity is due to its simplicity to understand as well as its ability to solve many problems in general (Zaknich 2003). However, the ANN users should bear in mind that back-propagation is a supervised training approach and it needs enough examples to train. Previously, the term “back-propagation” was used to refer to the gradient descent algorithm applied to neural network training. However, nowadays, due to the fact that the process of computing the gradient and Jacobian by performing calculations backward through the network is applied in most of the training functions, this terminology is not used most often (Caudill and Butler 1992). Instead of utilising this term alone, it is clearer to use the name of the specific optimisation algorithm that is being used. Likewise, the multilayer network is sometimes recognised as a back-propagation network. However, the back-propagation technique that is used to compute gradients and Jacobians in a multilayer network can also be applied to many different network architectures (Rumelhart et al. 1986).

In order to illustrate how the back-propagation training works, the simplest optimisation algorithm, gradient descent, is considered here. Employing this algorithm, the network weights and biases are updated in the direction in which the performance function decreases most rapidly (Beale et al. 2012). One iteration of training using this algorithm is according to the Equation 14:

$$\mathbf{X}_{k+1} = \mathbf{X}_k - \alpha_k \mathbf{g}_k \quad (\text{Eq. 14})$$

In Equation 14,  $\mathbf{X}_k$  is a vector of current weights and biases,  $\mathbf{g}_k$  is the current gradient, and  $\alpha_k$  is the learning rate. This equation is iterated until the network converges. However, for large problems this training algorithm can be slow. To solve this, the training speed can be improved by new learning laws or by putting constraints on some of the weights during learning (Vogl et al. 1988).

Use of an Artificial Neural Network

In this section the principles of an artificial neural network design and usage is discussed. In order to clarify when use of ANN is preferable over other conventional methods, advantages and disadvantages of using ANN over conventional computer algorithms are described.

#### **2.3.4 Advantages and disadvantages of ANNs over conventional computer algorithms**

Although for many applications ANN can provide comparable answers to other classical techniques, some attractive features of ANN makes it appealing. The two main advantages of ANNs over conventional computer algorithms are:

1. Ability to handle some noise and error in data.

In reality, most of the signals have some inbuilt noise or errors which make the patterns slightly different each time. Conventional computer algorithms cannot easily deal with the error or noise in input data and may result in different output whereas an appropriate and well-trained ANN can handle the noise in the input data.

2. Desired knowledge based on experience can directly be put into function.

The ANN's non-linear nature enables it to handle functions more than the capability of even the best linear or conventional rule based processing methods. There are many occasions where conventional methods are too hard. In contrast, a simple ANN can determine appropriate rules employing its inherent self-organisation process. Due to the fact that ANN is sensitive to statistical regularities in large data sets, it can derive important information from actual relationships implicit in the data. Furthermore, as a result of adaptive nature of ANN, changes and the characteristics of input data can be adapted and learned in real-time (Caudill 1989).

Conversely, there are some disadvantages of ANN over conventional computer algorithms as well. Some major ones are introduced as follows:

1. No unique solution or general design theory.
2. Generally it is not guaranteed that the ANN will converge to its global minimum or occasionally even will converge at all.
3. May be too slow in the case of large-scale problems when common serial processing digital computers are employed.

4. The number of inputs to the ANN needs to be equal or greater than the number of outputs.

### **2.3.5 Design Principles**

While each ANN problem and application is unique and no general rule or solution exists for all the applications, some broad principles can be used to guide the design process and ensure better results. In fact the design of the specification and parameters of a particular ANN should be targeted to provide the most appropriate system and best overall performance according to the specific problem it is dealt with. It is important to not expect the ANN to do too much on its own without enough related guidance and learning examples. The ANN designer should perform preliminary feature extraction and data pre-processing as much as possible to reduce the task of the network.

Various parts of a system or problem may employ an ANN. It is advised to fully understand the system and what it will do. The input and output data formats should be identified precisely according to the problem. When the data is collected or generated in a suitable format, it should be pre-processed and the related features be chosen according to the specific problem. Features should be selected based on their correlation to the desired output. Often, it is wise to remove redundant or useless data from data sets. An ANN generally needs sets of independent input/output vector pairs descriptive of the process. Three vector sets are needed for training, testing and validation. Alternatively one data set can be divided in a systematic or random manner to generate the required three vector sets. The training set is used to train the network weights and biases. Test and validation data sets are used to ensure better training of the ANN which avoids overfitting and ensures a more general ANN.

Furthermore, appropriate approaches should be designed to verify and validate the system performance. In general, it is not easy to measure the characteristics of performance and limits of ANNs solely with analytical techniques. Hence, it is highly recommended that a testing regime be defined to ensure its satisfactory performance. For this purpose, one way is to present the ANN with selected extreme inputs within the possible limits of system operation.

The design of an ANN application should aim to produce the best system and performance overall. The general neural network design process has the following principal steps (Hagan et al. 1996):

1. Collect and Prepare the Data.
2. Raw data pre-processing.
3. Network Creation and Configuration.
4. Network Training.
5. Network Validation and post-processing.
6. Use the network.

A typical flowchart of ANN designing is indicated in Figure 2-13. Each of the above steps is described more as follows.

#### ***2.3.5.1 Collect and Prepare the Data***

The first step to design and use a successful ANN is to collect and prepare sample data sets. This step is very important due to the fact that the ANN can only be as accurate as the data that are used to train it. Data sets should cover the whole range of inputs of the problem (Hagan and Menhaj 1999). This is due to the fact that whilst ANNs are generally capable of being trained to generalise well within the range of training data sets, they cannot correctly extrapolate beyond this range. In addition, redundant or useless data should be removed from data sets.

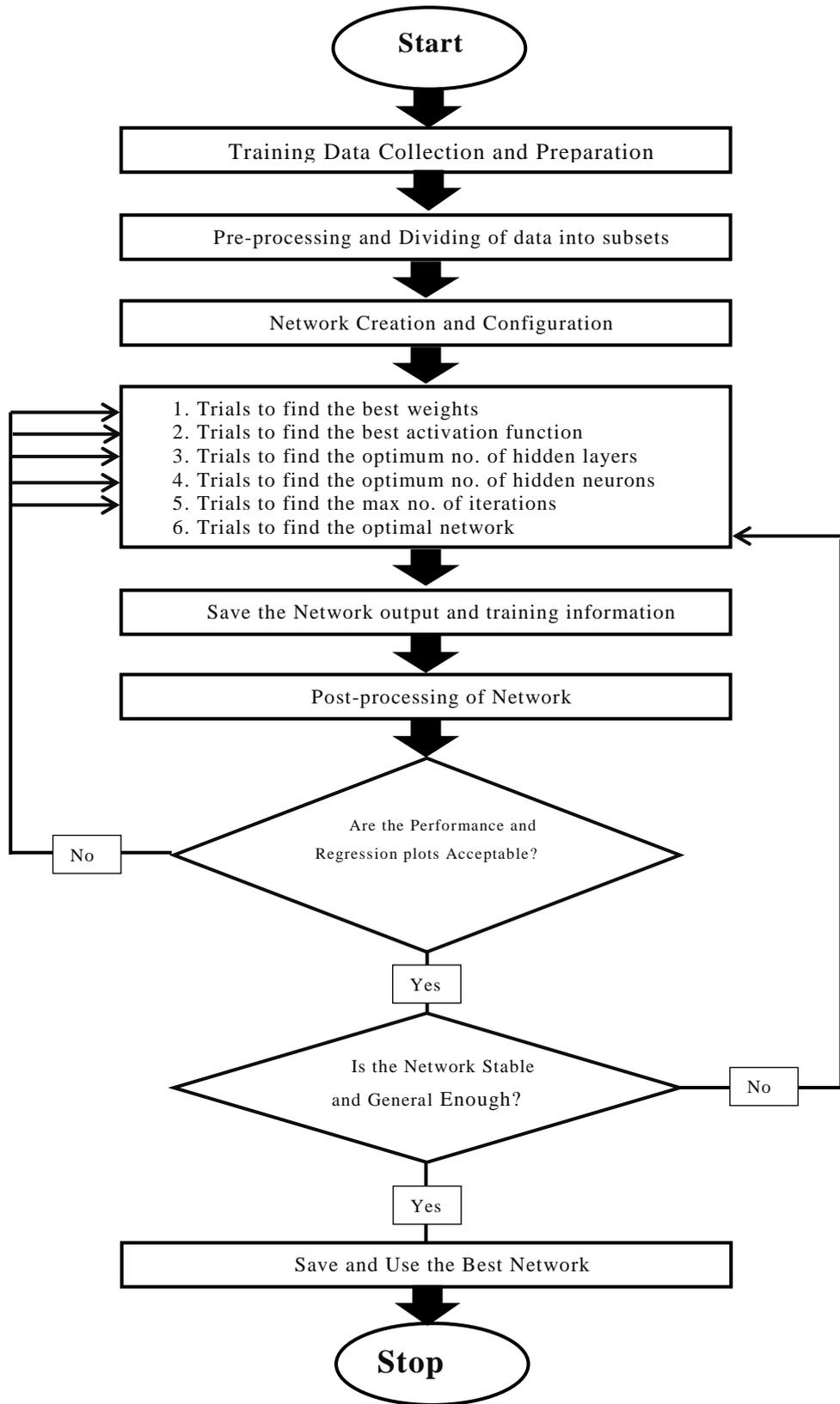


Figure 2-13: Flowchart for building an ANN model

### ***2.3.5.2 Pre-processing and Dividing Data into Subsets***

Once enough data is collected, in order to have more efficient training, two major steps have to be carried out before they can be used for training the network: pre-processing and dividing data into subsets.

It is best to build the pre-processing as a part of the network objective, so whenever the ANN is utilised, data coming into the network is pre-processed in the same way. One of the standard pre-processing practices that help to improve training is normalisation of the input data before applying them to the network. Normalisation should be applied to both the input vectors and the target vectors in the data set. As a result of normalisation, the network output will be in a normalised range. Reverse transformation of the network outputs back into the units of the original target data must be done when the network is put to use in the field.

Furthermore, the prepared and pre-processed data should be divided into three subsets. The first subset is the training set, which is used for computing the gradient and updating the network weights and biases. The others are the test and validation sets.

### ***2.3.5.3 Network Creation and Configuration***

Before training the network, the network should be created based on the desired architecture and its weights and biases initiated according to the data sets. The weights and biases are initiated according to the network type randomly or can be fixed values. However, sometimes it may be necessary to reinitialise them. In most software, network configuration is only possible after it is created. The network configuration includes examining input and target data sets, setting the network's input and output sizes to match the data, and choosing settings for processing inputs and outputs that will enable best network performance. The configuration is normally done automatically, when the training function is called and before the training process starts.

### ***2.3.5.4 Train the Network***

Various mathematical techniques can be employed as training and learning functions where their duty is to perform the adjustment of the ANN weights and biases automatically. As a matter of fact, the global algorithm affecting all the weights and biases

of the ANN is defined by the training function. In the training process, training data sets prepared in previous steps are applied to the network as inputs  $p$  and target outputs  $t$ .

The training process of an ANN involves adjusting the weights and biases of the network with the aim of network performance optimisation which is defined by a function called the network performance function. Regularly, Mean Squared Error (MSE) or Sum of Squared Error (SSE) is used as the performance function for a typical feedforward network. MSE is the average squared error between the network outputs  $a$  and the target outputs  $t$  (Equation 15). Similarly, SSE is the sum of squared error between the network outputs  $a$  and the target outputs  $t$  (Equation 16) (Beale et al. 2012).

$$MSE = \frac{1}{N} \sum_{i=1}^N (e_i)^2 = \frac{1}{N} \sum_{i=1}^N (t_i - a_i)^2 \quad (\text{Eq. 15})$$

$$SSE = \frac{1}{2} \sum_{i=1}^N (e_i)^2 = \frac{1}{2} \sum_{i=1}^N (t_i - a_i)^2 \quad (\text{Eq. 16})$$

Generally two main training styles exist: incremental and batch training. In incremental training the weights and biases of the network are updated each time an input is presented to the network. In this case the inputs and targets are presented as sequences. In batch training the weights and biases are only updated after all the inputs are presented. The batch training methods are generally more efficient (Beale et al. 2012) .

#### ***2.3.5.5 Network Validation and Post-processing (post-training analysis)***

Once the network is trained, its performance should be evaluated to see if any of the training process parameters, the data sets or even network architecture need alteration. To evaluate the performance of the network, all of the network information during the training (which is normally saved during the training) should be checked. Most of the new developed software for ANN contains algorithms that the training information is saved and can be accessed numerically and graphically. One of the best ways to validate network performance is to check the performance plot and regression plot.

The performance plot indicates three curves: training, validation and test performance curves over certain training steps (epochs). Usually when training is performed successfully all the curves are very similar and the performance values are decreased as the training process proceeds. However, if the difference between test curve and training curve

increases significantly then the training should be stopped. This is due to the fact that the training process is not converging anymore and it starts to diverge from the minimum (best) performance value. Furthermore, if the test curve is increased considerably before the increase of the validation curve, then most often some overfitting problem may happen and network generalisation capability will largely decrease.

The other plot helpful in validating the network performance is the regression plot. In this plot the relationship between the outputs of the network and the targets are indicated by fitting a linear regression line between outputs and targets. In theory, the ideal is to have a perfect network whose outputs and targets are exactly equal. However, in practice this is not the case and the relationship is rarely perfect. As an indication of the relationship between the outputs and targets, an index normally called R is calculated for each particular regression plot which can be anything between 0 and 1. Considering  $R=1$ , there will be an exact linear relationship between outputs and targets. In contrast, if it becomes close or equal to zero, then it clearly means that there is no linear relationship between them. Generally, validation test results with R value greater than 0.9 indicate a linear relation between targets and network outputs and the training process can be accepted. However, a decision should be made according to the particular regression plot of each data set. The scatter plot of data on the regression graph will be very helpful as well to indicate which certain data points have a poor fit (Beale et al. 2012).

#### ***2.3.5.6 Use the Network***

After the network is trained and validated, the network object can be used to calculate the network response to any input in the range of training data. The network can be called to calculate the outputs for a set of input vectors. The network has its latest weights and biases that are updated according to the best performance of the network during its training with training data sets. As a batch mode form of simulation, all the input vectors can be placed in one matrix which is much more efficient than presenting the vectors one at a time. A properly trained network will provide reasonable outputs when presented with inputs that they have never seen. Normally, presenting to the network a new input which is similar to inputs employed in the training set results in an accurate output. This is known as network generalisation capability and makes training a network possible with a sample

set of input/target pairs and the ANN achieves acceptable results without training with all possible input/output pairs.

### **2.3.6 Improving Results**

Since there is no general training rule for an ANN and the training process is a relative process according to the training data set of a particular problem, it is very probable that the network results may not be satisfactory on the first design pass. In this situation, it is necessary to redo one or more of the process steps repeatedly until better results are achieved. When the training process results in a network which is not accurate enough, several approaches are highly recommended. One way is to initialise the network again and perform the training again. Initialising a network leads to new initial weights and biases. When the network parameters are changed, it might result in different solutions. Another solution is to increase the number of hidden neurons, usually 20 or more. The reason is that the network will have more flexibility having a larger numbers of neurons in the hidden layer. It is recommended to increase the layer size gradually. Training a network with too large a hidden layer is more time consuming and may cause the network to be under-characterised as well. This means that the network has more parameters to optimise than there are data vectors to constrain them. The third popular approach is to try a different training function. Furthermore, different training data can be tried to train the network. Additional data sets can be used in the training data set or in contrast redundant or challenging data may be removed from the training data set. Presenting network additional data is expected to produce a more general network that can handle new data well (Beale et al. 2012).

#### Software Implementation

Nowadays, several general artificial neural work software packages are commercially available and researchers use them in various engineering and science fields. One of the most comprehensive and all-purpose ANN packages developed so far is the MATLAB Neural Network Toolbox (MATLAB NNT).

### **2.3.7 Why MATLAB Neural Network Toolbox**

MATLAB (MathWorks, Natick, Massachusetts, USA) a programming language developed by MathWorks is used widely worldwide and is popular with researchers due to its easily understood GUI and general programming language (Beale et al. 2012). Indeed,

the product families of MATLAB including its specific developed toolboxes are major computational tools at educational institutions worldwide. This is due to the fact that MATLAB is an interactive environment which enables the user to access its high-level language for programming, numerical computation and visualisation of data through well-developed graphical user interfaces. Various MATLAB toolboxes can be employed at the same time in a script to perform a variety of functions such as data acquisition, numerical computation and analysis of data, develop algorithms, and create models and applications. Its general programming language, built-in maths functions and toolboxes enables the user to achieve a solution much faster than with spreadsheets or even old-style programming languages, like C/C++ or Java. MATLAB can be used to develop customised programmes that can perform a range of functions simultaneously, including data acquisition and signal processing, test and measurement, numerical computation, as well as code generation and verification. Furthermore, MATLAB can communicate with most of the standard commercial hardware and software easily to exchange data (Beale et al. 2012).

MATLAB NNT is popular due to the fact that a variety of architectures including supervised and unsupervised networks as well as most of the common training algorithms are supported. Researchers can use a modular approach of MATLAB NNT to build different networks or even to develop custom network architectures for a specific problem. Generally, pre-processing of the network inputs and targets improves the efficiency of neural network training and post-processing enables detailed analysis of network performance. MATLAB NNT has built-in pre-processing and post-processing functions that can automatically reduce the dimensions of the input vectors and perform regression analysis between the network response and the corresponding targets. Furthermore, at the time of network creation MATLAB NNT performs an automatic data preprocessing and data division. In addition, during the training process the input and target values are automatically scaled to the range  $[-1,1]$  to improve the training speed (Beale et al. 2012).

Overfitting is a common problem in the training stage of the neural network. An overfitted network cannot handle new data or noise in data efficiently. In other words when overfitting happens, the training set is memorised by the network very well but it cannot generalise its knowledge to new inputs. MATLAB NNT improves generalisation through two methods: regularisation and early stopping. The regularisation method automatically

modifies the network's measure of error and produces a network that performs well with the training data and exhibits smoother behaviour when presented with new data. In the early stopping method two different data sets are used. First is the training data set which is employed to update the weights and biases. Second is the validation data set which is used to stop training when the network begins to overfit the data (Beale et al. 2012).

The described introductory information about ANN helps to better understand the methodology employed in this research. In the following chapters, the experiments designed and performed in this research to fulfil the project objectives are presented.

## 2.4 Equipment Set Up

In order to investigate the behaviour of marine structures and the application of the methodology, a 1 m<sup>2</sup> Glass Reinforced Fibre Polymer (GRFP) composite panel (as explained in section 1.2.1) as a representative of panels used in a boat is employed to set up a test rig. The panel is deemed as a suitable representation of a marine structure due to the fact that most marine crafts are manufactured from a number of flat panels attached to the frame structure.

The marine composite panel employed in this study is made of 7 layers of stitched biaxial  $\pm 45$  E-glass cloth and with Ampreg 22 epoxy resin system (all provided by SP Gurit Systems), hand laid up with a total thickness of  $5 \times 10^{-3}$  m (Figure 2-14). The panel was divided into a four-by-four grid producing sixteen equal regions of area  $0.25 \times 0.25$  m<sup>2</sup> as shown in Figure 2-14. Table 2-1 shows the mechanical properties of the glass fibre as provided by SP Gurit Systems. Whilst static loading is achieved by applying weights normal to the panel surface, transient loading is achieved by free fall of a cylinder from various heights normal to the panel surface. Since strain has direct relation with load, as a suitable panel response to the load, strain gauges are used to measure structural response. A strain gauge is made of strip of conductive metal which if it is stretched, it will become skinnier and longer, both changes resulting in an increase of electrical resistance end-to-end. Conversely, if a strip of conductive metal is placed under compressive force, it will broaden and shorten. If these stresses are kept within the elastic limit of the metal strip (so that the strip does not permanently deform), measuring the difference in electrical voltage and resistance of strip end-to-end can be used as a measuring element for physical force.

Strain gauges are frequently used in various engineering research projects. Figure 2-15 indicates a strain gauge schematic.

In fact, marine craft are manufactured from a number of flat panels that are structural members. Their characteristics are such that their thickness is small compared to their other dimensions. It is important to note that depending on the loading condition and the panel aspect ratio, the panel shows different behaviour under loading. Panels can be classified according to their thickness and their lateral deflection compared to their thickness (Boresi et al. 1993). They can be classified as: 1) thick plate small deflection; 2) thin plate and small deflection; 3) thin plate large deflection or 4) very thin plate (membranes) with either small deflection or large deflection. In all cases the solutions are approximate, not exact or closed form. The deflection at the centre of a plate subject to pressure is offered by Westergaard and Salter (1921) and is based on modified flexure theory of plates. Where, depending on the plate aspect ratio, edge boundary conditions and load, different approximate empirical solutions are found. In such cases a small displacement is defined as a displacement less than or equal to half the thickness of the plate. If the displacement exceeds this limit then the problem is treated as a non-linear problem where the displacement can no longer be accurately predicted using the above theory. This is due to highly nonlinear double curvature deformation unlike the displacement function stated above. In large displacement analysis the transverse shear can also no longer be ignored and if the panel is composite then transverse shear requires further special treatment. In brief, small displacement leads to a linear relationship between the increase of load and the panel deflection. In contrast, in the case of large displacement, the relation is no longer linear and should be treated as a non-linear problem where the displacement can no longer be accurately predicted using the linear theories.

In this research the linear problem is defined as a panel where the maximum displacement due to lateral load is around  $1/2$  to  $2/3$  of the total panel thickness. Larger loads causing large displacement are treated as nonlinear displacement, where displacements is a nonlinear function of the applied load. In such a situation the theory of superposition used so far can no longer be used to generate the large number of training data that are required by the ANN. In such cases, alternative methods are needed to

generate the necessary training data. The methodology for solving nonlinear inverse problems will be discussed in later chapters.



Figure 2-14: Schematic of the composite panel

Table 2-1: Composite Panel Material Specification

<b>Material name</b>	<b>XE905</b>			
<b>Material type</b>	Stitched biaxial			
<b>Fibre Volume Fraction</b>	0.46			
<b>Longitudinal PROPERTY</b>	$N/mm^2$			Units
<b>Longitudinal Tensile Modulus</b>	21220	Poissons Ratio (Longitudinal Strain)		0.120
<b>Longitudinal Tensile Strength</b>	318.3	Poissons Ratio (Transverse Strain)		0.120
<b>Longitudinal Compressive Modulus</b>	21220	Longitudinal Coeff. of Thermal Expansion	10-6/°K	14.62
<b>Longitudinal Compressive Strength</b>	254.6	Transverse Coeff. of Thermal Expansion	10-6/°K	14.62
<b>Transverse PROPERTY</b>		Density	$kg/m^3$	1786
<b>Transverse Tensile Modulus</b>	21220	Structural Ply Thickness	mm	0.75
<b>Transverse Tensile Strength</b>	318.3	Actual Ply Weight	$g/m^2$	1364
<b>Transverse Compressive Modulus</b>	21220	Shear thickness	mm	0.75
<b>Transverse Compressive Strength</b>	254.6			
<b>SHEAR PROPERTIES</b>		<b>DERIVED SHEAR PROPERTIES @ ±45°</b>		
<b>InterLaminar Shear Modulus</b>	3050	Shear material name:	1 x XE905 @ ±45°	
<b>InterLaminar Shear Strength</b>	36.6	Axial modulus with fibres @±45°	$N/mm^2$	9737
<b>In-Plane Shear Modulus</b>	3050	Shear modulus with fibres @45°	$N/mm^2$	9471
<b>In-Plane Shear Strength</b>	46.1	Poisson's ratio with fibres @±45°		0.596

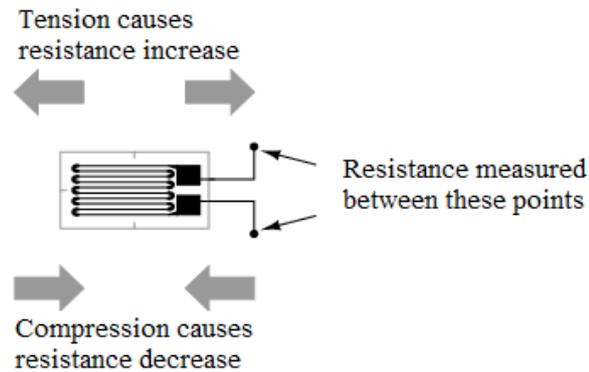


Figure 2-15: Schematic of a strain gauge and how it work

Depending on the required number of analogue input channels for each experiment, one of two Data Acquisition (DAQ) systems were used to acquire data in this project: Microlink (up to 128 analogue input channels) and National Instruments Equipment (up to 16 analogue input channels).

### 1. Microlink Equipment

Generally, a DAQ system is constructed from two main parts: data logger and conditioner. Microlink 751 Multi-function USB Data Acquisition (Figure 2-16) and Microlink 594 conditioning units (Figure 2-17) with Windmill 7 Software (Figure 2-18) (all by Biodata, Manchester, UK) are employed each featuring 16 analogue inputs for strain measurement. Eight USB units can be connected to one computer to monitor up to 128 strain gauges. The Microlink 751-SG is supplied with Windmill data acquisition and control software. This modular suite of software offers data logging, charting, alarm indication, output control and real-time data transfer to other applications such as EXCEL, MATLAB and FORTRAN. Specifications for Microlink 751 Multi-function USB Data Acquisition are illustrated in Table 2-2. Using this data acquisition system, all the strain gauges were wired and connected to the DAQ system in a quarter bridge configuration. Although, both half-bridge and full-bridge configurations grant greater sensitivity over the quarter-bridge circuit, often (specifically in marine structures) it is not possible to bond complementary pairs of strain gauges to the test specimen. As a result, the quarter-bridge circuit is frequently used in strain measurement systems. The strain data can be collected

directly via Windmill 7 Software or alternatively through bespoke data acquisition/ANN software linked to the Windmill data acquisition system.



Figure 2-16: Microlink 751 Multi-function USB Data Acquisition (Windmill 2011)

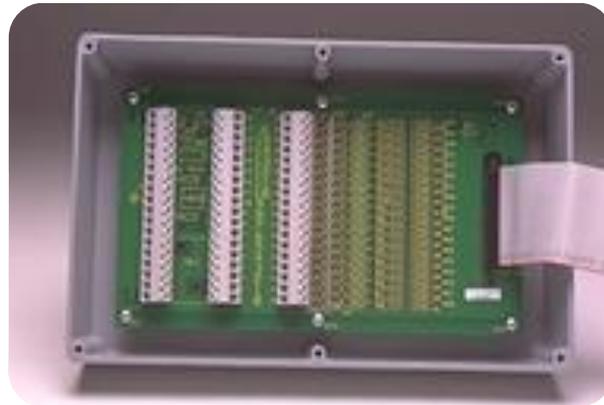


Figure 2-17: Microlink 594 Bridge Input Unit and Screw Terminals (Windmill 2011)

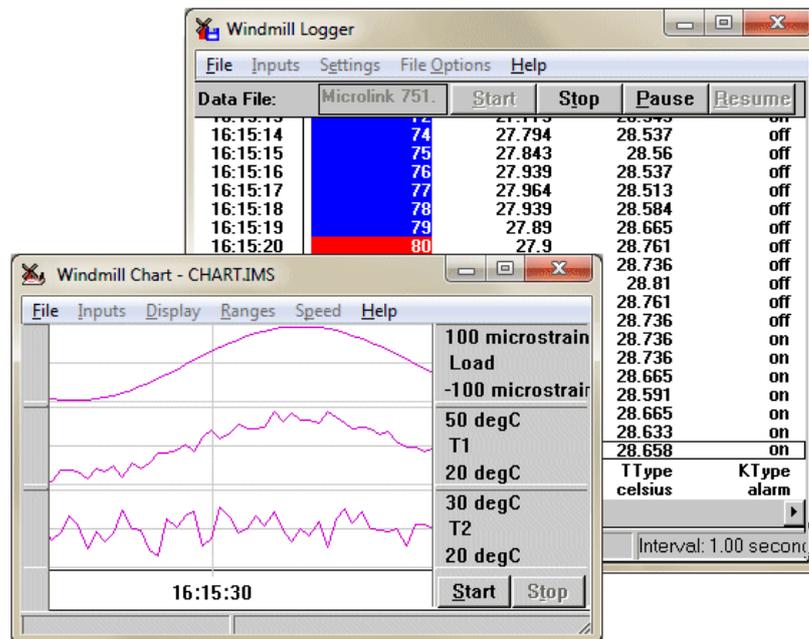


Figure 2-18: The Windmill Chart and Logger programs (Windmill 2011)

Table 2-2: Windmill 751-TC Unit Specifications (Windmill 2011)

Microlink Equipment Specification	
<b>Analogue Inputs</b>	16
<b>Common mode range</b>	±13 V
<b>Resolution</b>	Samples/Second
<b>12 bits</b>	2.5 msec 80
<b>15 bits</b>	20 msec 32
<b>16 bits</b>	40 msec 16
<b>18 bits</b>	160 msec 6
<b>Input voltage range</b>	0 to 5 V
<b>Maximum count speed with low resolution</b>	160 counts per second

## 2. National Instruments (NI) Equipment

CDAQ system provided by NI consists of a chassis, CDAQ modules, and software. The chassis can connect to a host computer over USB. The CDAQ system employed in this study includes NI CDAQ 9236 (350 Ohm) modules mounted on NI CDAQ 9174 Chassis (4 slots) which can provide the strain monitoring and control data acquisition system with resolution of +/- 0.1 microstrain (Figure 2-19). The CDAQ 9236 module is specifically designed for quarter-bridge strain gauge measurement and contains the signal converter, connectivity, and conditioning circuitry in a single rugged package which provides differential inputs to monitor eight strain gauges at up to 10000 samples per second. The strain data can be collected directly via NI Measurement & Automation Explorer (MAX) or alternatively through bespoke data acquisition/ANN software linked to the NI-DAQmx Driver.



Figure 2-19: NI CDAQ Equipment (Ni 2013)

Various experiments are performed employing the composite panel and data acquisition systems during this research. The individual experiments set ups used for each of the experiments are explained in following sections. Since the NI DAQ system is much more

reliable and faster than the Microlink one, the preference in DAQ selection was NI system. As the drawback of the NI system was its maximum 16 analogue channels, should the experiment need more than 8 channels, the Microlink DAQ system was employed.

#### **2.4.1 Small Displacement Equipment Set Up**

In order to investigate the general behaviour of composite a simple test rig is designed in the form of a flat composite panel. The bottom surface of the composite panel is supported at four points using rectangular section blocks, each 0.035 m wide, 0.009 m deep and 0.025 m long (Figure 2-20). Loading is achieved by applying masses normal to the panel surface at the 12 locations indicated in (L<sub>1-12</sub>). Loading is achieved using masses lying on a 0.05 Kg circular section stand (see Figure 2-21). When one mass is located on an area, a small distributed load is applied on the surface.

The panel is divided into sixteen equal areas of  $0.25 \times 0.25 \text{ m}^2$  and a rectangular strain gauge rosette is attached to the top surface of the panel at the centre of each region S<sub>L1-16</sub>. From sixteen rosette strain gauges attached, 48 strain readings can be acquired. The specifications of the strain gauges are indicated in the Table 2-3. The Microlink DAQ system is employed to acquire 48 strain readings for this experiment constructed from three Microlink 751 Multi-function USB Data Acquisition and three 594 conditioning units with Windmill 7 Software.

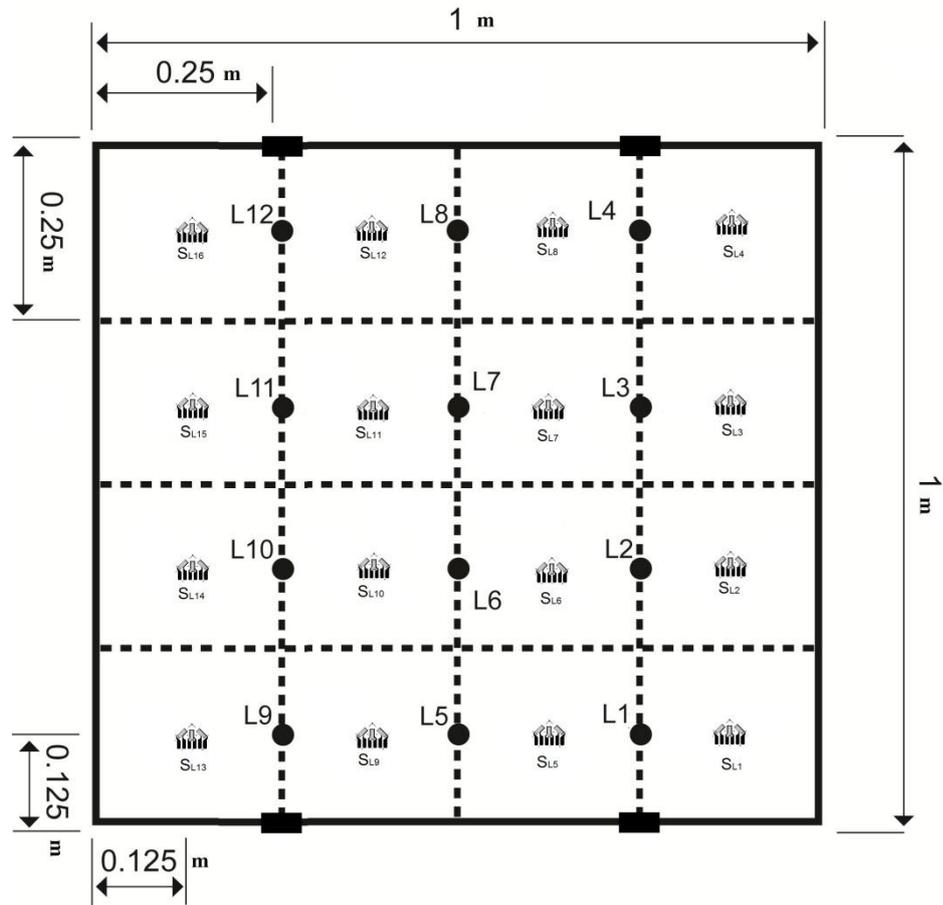


Figure 2-20: Panel drawing with the location of gauges and loading for small displacement experiment



Figure 2-21: 5 Kg Weight and 0.05 Kg stand

Table 2-3: Rosette Strain Gauges Properties (Vishay\_Group 2010)

General purpose Rosette Strain Gauge Properties				
Grid Resistance (OHM)	Gauge Factor TC (%/100c°)	Gauge Factor @24c°	Gauge length	Gauge width
120.0 ±0.6%	+1.3±0.2	2.100±0.5%	3.18 mm	1.78 mm

### 2.4.2 Sensor Optimisation Experimental Setup

In order to optimise the number of strain gauges utilised the same equipment set up is used as described in Section 2.4.1. However, to investigate the effect of size sensitivity by

optimisation of the sensor quantity four more rectangular strain gauge rosettes were also placed in the middle of the panel ( $S_{L17-20}$ ) attached at the corners of a  $64 \times 10^{-6} \text{ m}^2$  square enabling 12 more strain readings (Figure 2-22). The additional strain gauges had the same properties as the original gauges (Table 2-3).

The sensor optimisation is based on convenience sampling and some trial and error. A simple FEA analysis helped to define critical edge distance so that mechanical cross talks would be avoided and sensors will remain independent. What can cause problem is when two sensors respond the same to the same load or vice versa. So the sensor optimisation also helps to find locations to attach strain gauges.

Convenience sampling is a non-probability sampling technique where subjects are selected because of their convenient accessibility and proximity to the researcher (Frederick et al. 2011). The sensor locations are selected just because they are the easiest places to attach the sensors for the study and the researcher did not consider selecting subjects that are representative of the entire structure. Researchers use convenience sampling not just because it is easy to use, but because it also has other research advantages. In pilot studies, convenience sample is usually used because it allows the researcher to obtain basic data and trends regarding his study without the complications of using a randomised sample. In all forms of research, it would be ideal to test the entire population, but in most cases, the population is just too large that it is impossible to include every individual. This is the reason why most researchers rely on sampling techniques like convenience sampling, the most common of all sampling techniques. Many researchers prefer this sampling technique because it is fast, inexpensive, easy and the subjects are readily available. The most obvious criticism about convenience sampling is sampling bias and that the sample is not representative of the entire population (Frederick et al. 2011).

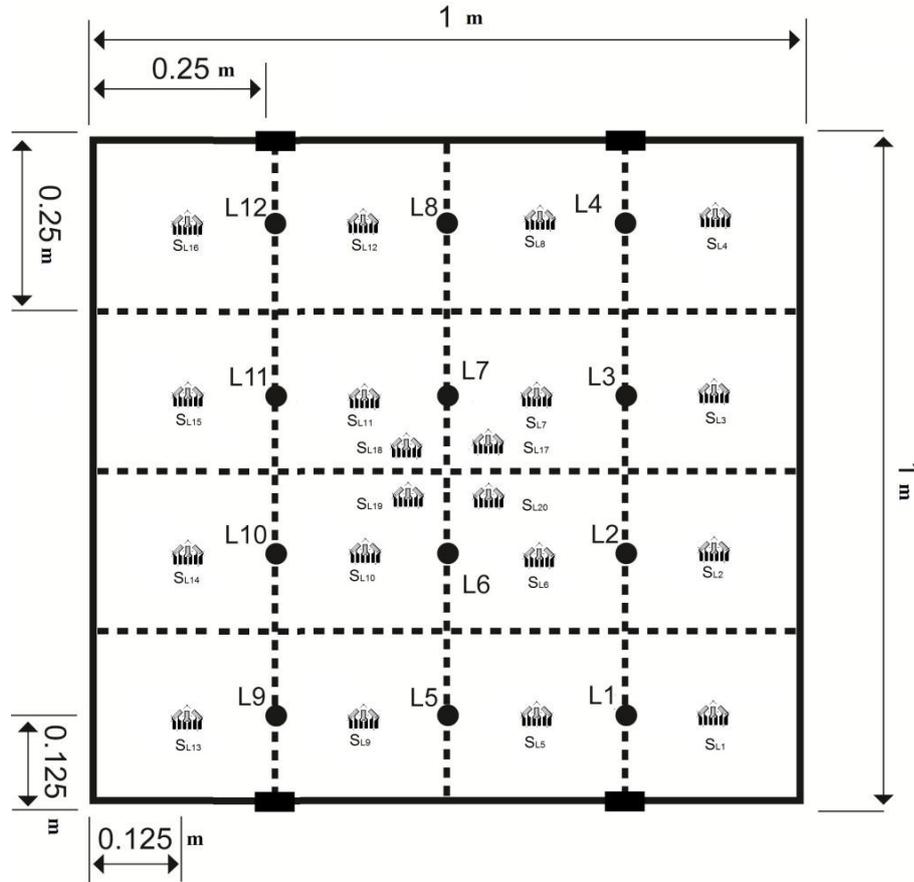


Figure 2-22: Schematic of composite panel indicating strain gauge, loading and support location (Sensor Optimisation experiment)

### 2.4.3 Large Displacement Experimental Set Up

In order to investigate the general behaviour of the composite panel the previous test rig was modified so it was fully fixed and higher loads could be applied. In fact, the previous test set up could not handle high loads and would fail and change the boundary conditions. Hence, the new test rig was designed to be fully fixed and the bottom surface of the GRFP marine composite panel was supported on all four edges using aluminium bars, each 0.0381 m high, 0.01905 m wide and 1m long (Figure 2-23). Sixteen linear strain gauges (S1-16) were bonded to the centre of each region (specification in Table 2-4: Strain gauge specification). Normal loads were randomly applied to the top surface of the panel at thirteen grid intersections (L1-13). NI Equipment is used to capture the response of the structure to the applied loads. For this purpose, Two eight Channel NI cDAQ 9236 (350 Ohm) modules mounted on NI cDAQ 9174 Chassis were used to acquire data (see Figure 2-19) using MATLAB (MathWorks, Natick, Massachusetts, USA) and its Data

Acquisition Toolbox capabilities. The gauge resistance is increased to 350 Ohm so better strain readings can be achieved from composite materials.

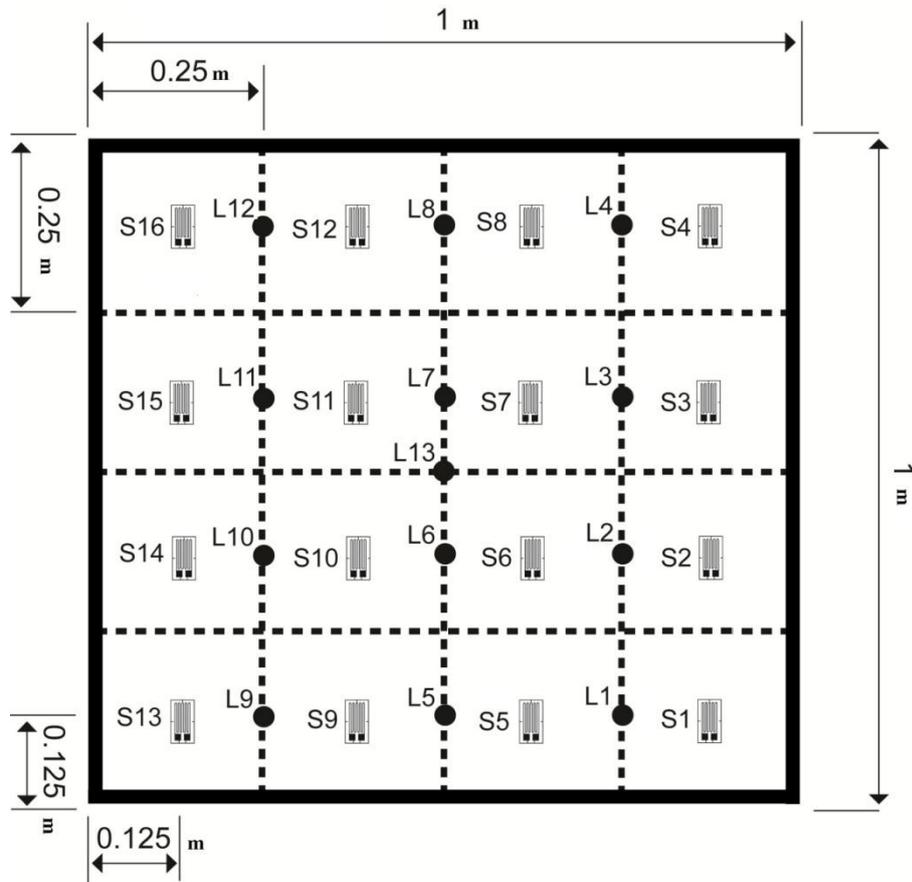


Figure 2-23: Schematic of composite panel indicating strain gauge, loading and support location (Large Displacement experiment)

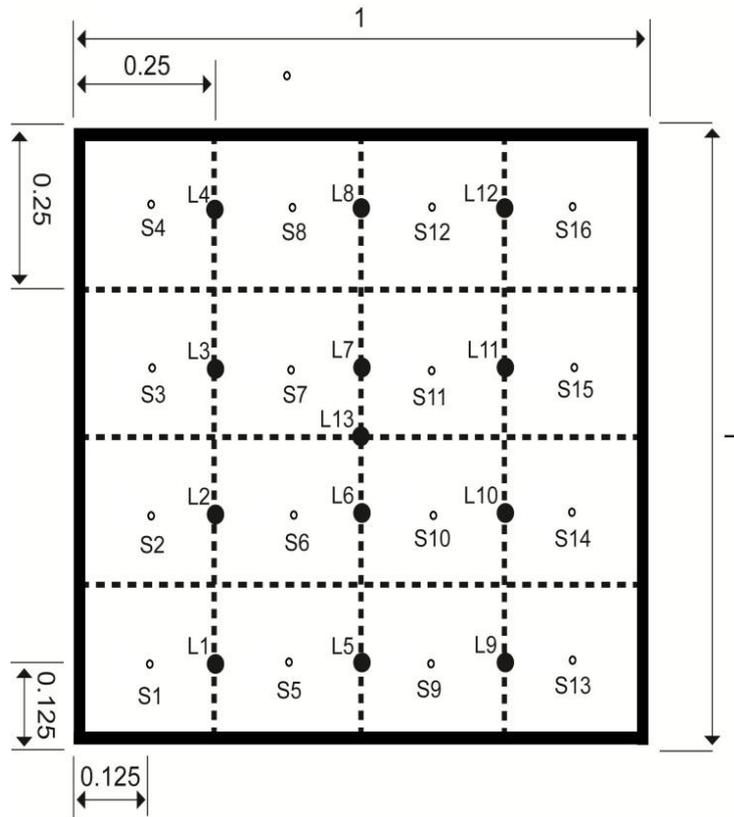
Table 2-4: Strain gauge specification (Vishay\_Group 2010)

Type	General purpose linear gauge
Resistance	350 ohms $\pm$ 0.6%
Gauge factor	2.100 $\pm$ 0.5%
Gauge length	6.35 mm
Gauge width	2.54 mm

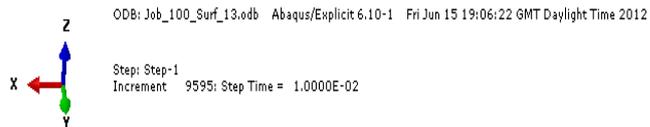
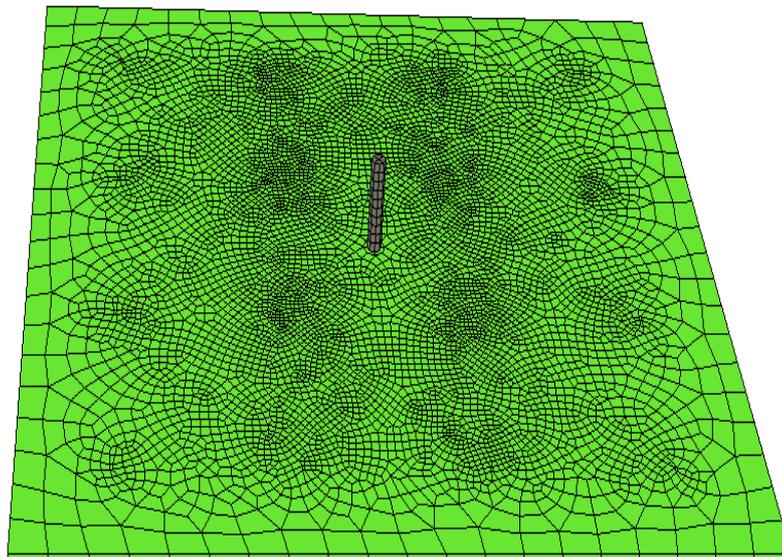
#### 2.4.4 Drop Test Simulation Set Up

The structure under consideration is the same as the large displacement experiment (Section 2.4.3) and strain readings (S1-16) are captured from the centre of each of the 16 regions on the top surface of the panel. Loading was achieved by simulating a free fall

impact of a rigid mild steel cylinder (length 0.103 m, diameter of 0.02 m and mass of 0.254 kg) normal to the panel surface at thirteen locations (L1-13) from various heights. Figure 2-24 indicates the schematic of the drop test experiment. For this study, the finite element models are developed and simulated in ABAQUS 6.10-1 (SIMULIA). The ABAQUS element library provides a complete geometric modelling capability. For this reason any combination of elements can be used to make up the model. All elements use numerical integration to allow complete generality in material behaviour. In almost all elements, primary vector quantities (such as displacements and rotations) are defined in terms of nodal values with scalar interpolation functions where the interpolation functions are written in terms of the parametric coordinates. Such isoparametric elements are guaranteed to be able to represent all rigid body modes and homogeneous deformation modes exactly, a necessary condition for convergence to the exact solution as the mesh is refined. ABAQUS will use either "full" or "reduced" integration. For full integration the number of integration points is sufficient to integrate the virtual work expression exactly, at least for linear material behaviour. All triangular and tetrahedral elements in ABAQUS use full integration. Reduced integration can be used for quadrilateral and hexahedral elements; in this procedure the number of integration points is sufficient to integrate exactly the contributions of the strain field. In this study, the panel has 7031 elements and the cylinder has 40 elements. The mesh element type used is hexahedral isoparametric. The advantage of the reduced integration elements is that the strains and stresses are calculated at the locations that provide optimal accuracy. A second advantage is that the reduced number of integration points decreases CPU time and storage requirements. In finite element modelling, a finer mesh typically results in a more accurate solution. However, as a mesh is made finer, the computation time increases. Generally, in order to select the optimum number of mesh elements, a simple mesh convergence study is carried out. In brief, to perform a mesh convergence study manually, a mesh is created using the fewest, reasonable number of elements and the model is analysed. The mesh is then recreated with a denser element distribution and reanalysed and the new results are compared to those of the previous mesh. Increasing the mesh density and re-analysing the model continued until the results converge satisfactorily. Figure 2-25 indicates a typical Finite Element Analysis (FEA) model output employed in this study.



**Figure 2-24: Schematic of composite panel (Drop test)**



**Figure 2-25: Meshed FEA Panel**

## 2.5 Summary

In order to answer the research question, the proposed methodology is described in this chapter. This chapter also gives more detailed information about the ANN used in the proposed methodology as well as the experiment set ups employed in this research. In brief, this research will explore relevant aspects of the application of hybrid combinations of ANN, FEA and experimental techniques both in direct and inverse studies. The hybrid technique will be used to produce solutions to problems that existing measurement or computing techniques cannot solve. This technique can provide more information faster than standard existing techniques without the need for knowledge of material/geometrical properties. The suitability of the proposed methodology will be indicated by successful prediction of the applied loads. Finally, a real time GUI should be developed allowing better communication with the system and to change different parameters used for data acquisition and to control ANN convergence rate etc. The performance of the load monitoring system will be evaluated by performing various tests by means of applying this system on a marine structure representative such as a cross section of the boat hull.

### **Chapter 3. General Behaviour of Composite Panel with Attached Strain Gauges under Small Displacement**

This chapter introduces the research undertaken to investigate the general behaviour of a composite panel with attached strain gauges under small displacement as well as optimisation of the number of sensors required for accurate load prediction.

### **3.1 Introduction**

This chapter reports on the research undertaken to achieve two main objectives of this project established in Chapter one. The first objective was to develop an ANN methodology for quantifying static pressure loads on a marine composite panel from strain measurements collected from the panel under small displacement. The suitability and performance of utilising an ANN for this experiment is presented in the first section of this chapter. The aim of the first experiment is to investigate the ANN's ability to accurately estimate static pressure loads applied to up to 12 locations on the structure using 48 strain readings from 16 strain gauge rosettes when the panel is under small displacement. The second objective covered in this chapter is to investigate the effect of size sensitivity by optimisation of the sensor quantity. Therefore, in the second section of this chapter, the research undertaken to optimise and reduce the minimum possible required number of strain gauges to accurately estimate 12 loads is described.

### **3.2 General Behaviour of Composite Panel with Attached Strain Gauges under Small Displacement (Static Load is Applied)**

In order to investigate the applicability of the ANN methodology for in-service load monitoring of marine structures, the first objective is to investigate the general behaviour of a composite panel as a representative of a marine structure under static load condition. The static pressure load condition represents the hydrostatic loads applied on a panel of a marine structure in the water. The following sections describe the methodology and results of this experiment employed to evaluate the proposed load monitoring system. The constant load equivalent to the Root Mean Square (RMS) value of the hydrostatic load on the panel causes a small deflection at the centre of the plate.

#### **3.2.1 Methodology**

The methodology and results employed to evaluate the suitability and performance of utilising an ANN as an inverse problem solver for quantifying the load applied to the composite panel is presented in this section. The first stage of the investigation was to design a load quantification methodology for the panel utilising an ANN. In the second stage the load quantification methodology was validated by comparing loads estimated by the ANN with the actual loads applied to the panel.

### ***3.2.1.1 Collecting the Training Data Using Superposition Theorem***

In order to investigate the behaviour of a composite panel used in marine structures and the application of the methodology, a simple composite panel is employed to set up a simple test rig (see section 2.4.1 for equipment set up). The initial aim was to understand the material behaviour and strain measurement sensitivity to various applied loads. Moreover, the aim is to experimentally capture data which can be used to relate known loading conditions to the response of the panel employing the proposed methodology. For this purpose, the GRFP composite panel is loaded by applying weights normal to the panel surface at the 12 locations indicated in Figure 2-20 ( $L_{1-12}$ ). Since strain has a direct relation with load, it is deemed a suitable panel response to the load and is used in this research to measure structural response. When one weight is located on an area, a small distributed load is applied on the surface. Known loads were applied to the predetermined load locations and the corresponding voltages from the strain gauges were recorded. It is worth noting that the strain gauge measurement is based on measuring induced difference in voltage in the circuit for even small displacements. Employing an ANN the exact value of the strain or any value proportional to it will be handled the same. Therefore, it is also possible to determine the applied load from the value of voltage measured from the gauge instead of the exact strain values.

In real circumstances the number of sensors needs to be optimised to reduce the cost and weight as well as computation efforts. This means that the locations used to measure the structural response must be selected carefully. These locations have to be sensitive to change in applied load. Furthermore, the responses collected must be unique for each set of applied loads. The load locations also should be fixed indicating the locations where load data are to be predicted by the ANN. If strain is collected from non-sensitive regions of the panel and/or the strain data collected is not unique for each load distribution the ANN is less likely to be able to find a function relating the input and output.

In order to train a network at least two sets of data as inputs and the desired response pairs are required. These sets of data (training data) should be enough to cover the whole range of applied loads. Gathering all of this data can be time consuming especially when many data sets are required and collecting each set individually should be avoided when possible. There are other

ways to create data sets based on manipulating only a small amount of the acquired data such as curve fitting or superposition methods. In this investigation, should the structural response have a linear behaviour within the range of applied loads, data collection can be achieved more quickly employing the theory of superposition. In other words, if the material and structure of this test behave linearly within the range of applied loads, the theory of superposition can be used to generate data sets. This theory states that the strain at a point on a structure due to a series of loads is equal to the sum of the strains from each individual load case (Sewell et al. 2010). For example, applying loads L1 and L2 to a hanging plate causes the three strains (S1–S3). This can alternatively be found by applying each load separately and summing the strains at each location (Figure 3-1). This theory enables the generation of an infinite number of training patterns from data collected by applying one known load to each location on the structure individually.

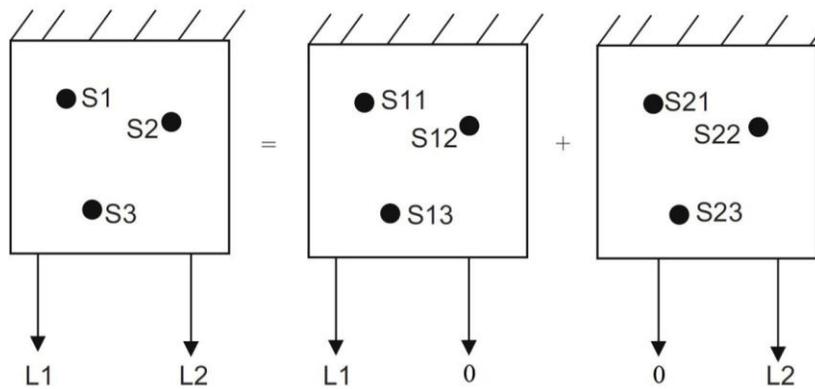


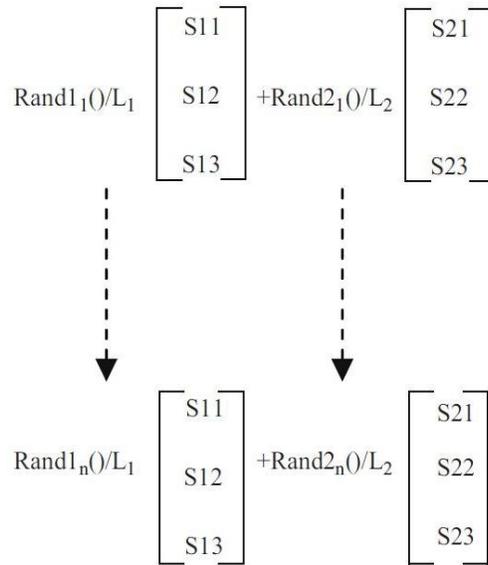
Figure 3-1: Theory of Superposition Example (Sewell et al. 2010)

Applying a load at each required position separately, a data matrix is generated where the columns represent each load case and here the first three rows represent the strains captured due to each load (Figure 3-2).

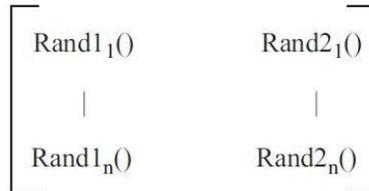
S11	S21
S12	S22
S13	S23
L1	0
0	L2

Figure 3-2: Example of acquired data from experiment (Sewell et al. 2010)

Using superposition any number of training patterns (n) can be generated using a random number generator that produces values between the minimum and a maximum load value specified. The maximum and minimum values should be the limits of the loads that can occur on the component. Figure 3-3 and Figure 3-4 show the training input and target data sets generation method using the example training file in Figure 3-2.



**Figure 3-3: Generation of training input (strains) (Sewell et al. 2010)**



**Figure 3-4: Generation of training output (load) (Sewell et al. 2010)**

Employing the superposition theorem, a training data matrix can be generated, where the columns represent each load case having M strain readings and N load values for pre-determined loading locations (see Figure 3-5). Using the superposition theorem, it is possible to have the desired number (K) of data vectors using a random number generator that produces values between the minimum and a maximum load value specified. Setting the limits of data according to the maximum and minimum of possible applied loads domain enables the coverage of all load scenarios for a specific application. In order to improve the generalisation of an ANN a set of

noisy data generated from the original data can be introduced in the training data. The noise value can be either added or subtracted from a predefined number of the data in a random manner.

$$\begin{bmatrix} S_{11} & \cdots & S_{1K} \\ \vdots & \ddots & \vdots \\ S_{M1} & \cdots & S_{MK} \\ L_{(M+1)1} & \cdots & L_{(M+1)K} \\ \vdots & \ddots & \vdots \\ L_{(M+N)1} & \cdots & L_{(M+N)K} \end{bmatrix}_{(M+N) \times K}$$

**Figure 3-5: Example of the training file**

In this study, the panel was divided by 16 equal patches and in the middle of each patch one strain gauge rosette was attached (see Figure 2-14). This means that 48 strain readings from 16 rosettes (inputs) and 12 load readings from 12 locations (outputs) were used with the superposition method to generate 1396 sets of training data from which 700 data sets are for cases when all 12 locations are loaded randomly and 696 data sets are for cases when only one loading location is loaded randomly within the loading envelope (58 data sets for each loading location). In fact, for each data set 12 random loads (zero or non zero) are applied at locations on the panel and the resultant 48 strains caused by these random loads were acquired to find the relationship between the input/output data. 280 data sets are used for test and validation during training to ensure convergence. This represented almost twenty percent of the original 1396 total patterns generated. 200 noisy patterns were also added to the training data based on the level of noise in the data acquisition system (+/- 1 microstrain). It should be noted that if the structure responses in terms of displacement indicate much bigger values than the noise and accuracy of the DAQ system (which is the case), the small noise of the DAQ system can even be neglected.

In this experiment, utilising the superposition theorem to generate training data from experimental data and employing the inverse approach to relate strain response to the applied loads, practically the material properties of the composite panel are not required at the training stage. Furthermore, due to the major advantage of the inverse approach, the material properties

are not required to quantify the load applied to the structure after ANN training. ANN training is achieved using MATLAB NNT capabilities.

### 3.2.1.2 ANN Architecture/Topology

An iterative process was used to determine the optimum network architecture for the panel based on the value of the final MSE of each network tested. The number of layers chosen can have an effect on the efficiency of the network. However, due to the relatively low number of inputs and outputs involved in this example this was not deemed to be an important factor to consider when selecting the architecture.

It was determined, through the testing of some random network architectures, that an ANN network having one hidden layer with 22 neurons that each has tan-sig as transfer function indicates acceptable training performance. The output layer had 12 neurons (representing the 12 loads to be estimated) and used a pure-lin transfer function (see Figure 3-6). The final specification of the ANN is shown in Table 3-1.

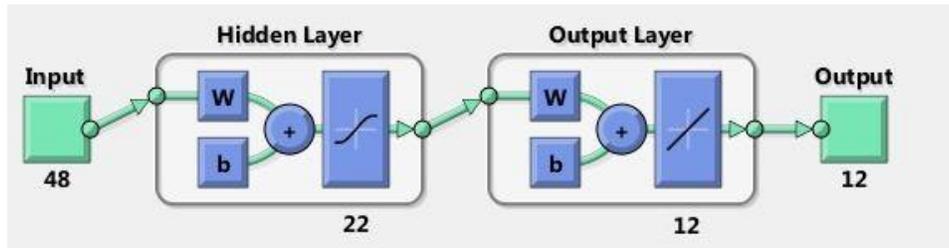


Figure 3-6: MATLAB representation of optimum ANN architecture

Table 3-1: Architecture of the artificial neural network

Feature	Details
Architecture	Feed forward back-propagation
Number of layers	2
Data process	Normalisation
Noise generator	+/-1 microstrain on 200 of training patterns
Range of loads	0 to 196.2N
Number of inputs (surface strains)	48
Number neurons in output layer	12
Number of neurons in hidden layer	22
Number of training patterns	1116
Number of testing patterns	280
Number of problem patterns	Depends on the number of patterns collected

It is always possible to over-train the ANN if not careful, which means that the ANN has been trained to respond to only one type of input. To ensure the ANN was not over-trained the training was supervised to ensure that the MSE of the testing data did not increase, which is an indication of over-training. The ANN was trained to minimise the MSE. The final training MSE was approximately  $4 \times 10^{-3}$ , which took under three minutes to reach. In other words applying the same data set employed in training to the network as inputs, the mean of squared errors between the ANN estimated loads and desired ones are as small as  $4 \times 10^{-3}$  N.

### ***3.2.1.3 ANN Validation and Performance***

The validity and performance of the ANN method was evaluated by comparing the load estimated by the ANN with known loads applied to the panel (problem data). What is referred as experimental problem data is the captured strain data from the 48 strain gauges attached to the panel while it is being loaded.

The first validation study utilised load and strain data generated from the original superposition data collected to produce the training data. This meant that any issues with the repeatability of the strains collected for a given load were removed. In the second study, problem strain data was captured directly from the panel under different loading conditions (i.e. one or two random loads were placed at random locations on the panel) and again the estimated loads compared with the actual applied loads.

## **3.2.2 Results and Discussion**

This section analyses and discusses the experimental results. In order to investigate the general behaviour of the composite panel with attached strain gauges under small displacement, a specific test rig and equipment setup is employed. This experiment is performed under static loading condition. Loading is achieved by applying gravity force of various dead weights. In this test, raw data is the strain readings from 16 rosette strain measurement sensors placed in the centre of the composite panel. In total 48 strain readings are achieved in one data acquisition call back. Three strain readings are set to be zero due to unreliable readings which was in result of manual error at the time of soldering the wire to these small strain gauges. The collected data

then needs to be checked for reliability of results. This will be in terms of having stable, repeatable data each time that the loading condition is the same.

### ***3.2.2.1 Validity of Data Collection and Analysis***

In this section, to ensure that data acquisition system reading is acceptable, validity of assumptions is checked. For this purpose, a set of experiments were performed over several days to investigate repeatability of the system as well as the amount of drift it may experience in a normal room condition where it is located.

### ***3.2.2.2 Reliability of Data readings***

The strain data was collected through bespoke data acquisition/ANN software linked to the Windmill data acquisition system. This software was developed by the author in MATLAB utilising Windmill Direct Data Exchange (DDE) protocols to acquire the strain data and the MATLAB Artificial Neural Network Toolbox capabilities. It is found that the data was experiencing a very small drift over the time (usually in hours). These drifts in values are not a big issue because they all experience the same drift. This can be easily removed by zeroing the data acquisition system or using the difference in strain readings for loaded and unloaded situations. In this study, to eliminate the effect of drift after some hours of running the device, it is decided to introduce extra data acquisition when the system is unloaded before each loading. This data set is then employed as a reference for zeroing the strain readings for the unloaded system. The results indicated that this removed the effect of drift from the strain readings. Although the system was assumed to have almost no effect of drift, there have been other difficulties such as noise in data. Most of the readings experienced a maximum sudden change of up to 7 microstrain (see Figure 3-7). This happened when DAQ system setting's resolution was set to 15 bits. The noise issue was reduced by introducing an averaging algorithm in the data acquisition function in the program.

Another important aspect is the repeatability of the data reading for the same loading condition. This is confirmed by having the same pattern of data for the same loading condition. For this purpose several data sets are acquired over a relatively long period. For instance, Figure 3-8 depicts acquired data set for loading 10 kgf on location 10, eighteen days apart.

Furthermore, Figure 3-9 indicates acquired data set for loading 1 kgf on location 11, two days apart. These figures show that the data acquisition gives acceptable repeatable readings over time.

Windmill Logger - PANEL3.IMS												
File Inputs Settings File Options Help												
Data File:	test.wl	Start	Stop	Pause	Resume	Log						
13:42:14	-802.946	-1300.65	-193.937	-1681.27	-1393.21	-1128.67	-1184.33	-1399.35	-1656.72	-1632.18	-1214.67	
13:42:17	-803.201	-1306.8	-193.937	-1681.27	-1392.77	-1134.81	-1177.81	-1405.05	-1656.72	-1638.31	-1208.53	-1
13:42:20	-803.201	-1306.8	-193.937	-1681.27	-1392.77	-1134.81	-1177.81	-1405.05	-1656.72	-1638.31	-1208.53	-1
13:42:23	-802.946	-1301.07	-187.781	-1681.27	-1393.21	-1134.81	-1177.81	-1405.05	-1650.59	-1632.18	-1208.53	-1
13:42:29	-802.946	-1300.65	-187.841	-1675.67	-1392.77	-1134.81	-1184.33	-1405.05	-1650.59	-1638.31	-1208.53	-1
13:42:32	-796.798	-1300.65	-193.937	-1675.13	-1398.91	-1128.67	-1177.81	-1405.05	-1656.72	-1638.31	-1215.05	-1
13:42:35	-802.946	-1300.65	-193.999	-1681.27	-1398.91	-1134.81	-1184.33	-1398.91	-1656.72	-1632.18	-1208.91	-1
13:42:38	-796.798	-1300.65	-187.781	-1681.27	-1393.21	-1134.81	-1177.81	-1405.05	-1656.72	-1632.7	-1214.67	
13:42:42	-803.201	-1301.07	-187.781	-1681.27	-1398.91	-1134.81	-1183.95	-1405.05	-1656.72	-1638.31	-1208.53	-1
13:42:45	-796.798	-1300.65	-187.781	-1675.13	-1392.77	-1128.67	-1184.33	-1405.05	-1650.59	-1632.18	-1208.53	
13:42:48	-802.946	-1306.8	-187.781	-1681.81	-1392.77	-1128.67	-1183.95	-1399.35	-1657.25	-1632.18	-1208.53	-1
Time	00001	00002	00003	00004	00005	00006	00007	00008	00009	00010	00011	mic
13:42:51	microstrain	microstrain	microstrain	microstrain	microstrain	microstrain	microstrain	microstrain	microstrain	microstrain	microstrain	mic
Running		Started at 13:30:53			Interval: 1.00 seconds							

Figure 3-7: Data Logger

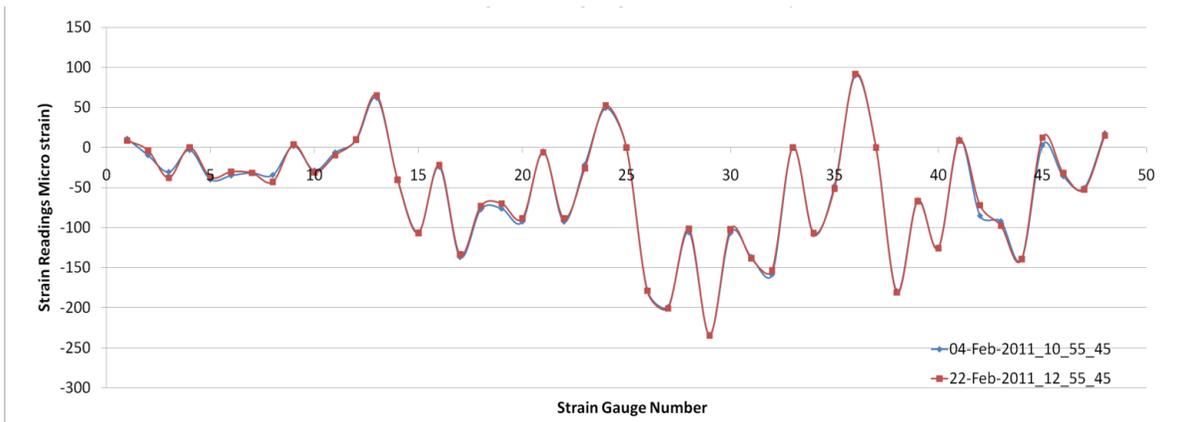


Figure 3-8: Data Reading set for loading 10 Kg on location 10 after 18 days

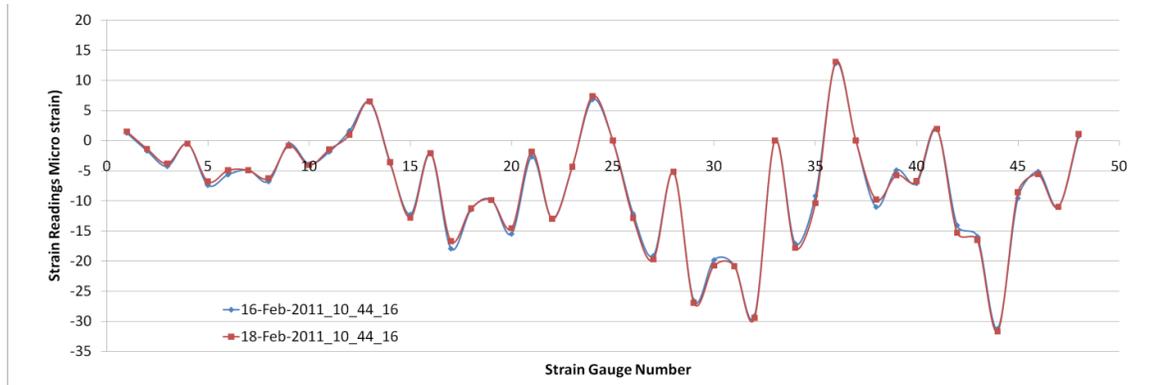


Figure 3-9: Data Reading set for loading 1 Kg on location 11 after 2 days

### 3.2.2.3 Fixed Supports to Rule out Twist Factor

It is important to note that the inverse problem relates one set of data to another set of data. In other words, the network acts as a transfer function of a specific structure which recognises the patterns of this specific structure response to certain loading conditions and employs them to predict almost the same loading conditions another time for the same structure. This is done without the need of having any information about the material properties and geometry of the structure. This implies that the panel has to have the same geometry configuration all the time. For this purpose it is reminded that the panel geometry in this test rig, which is constructed of the panel itself and its supports, has to have the same geometry configuration. This can be achieved by having fixed supports acting as negative loading in this configuration. In order to eliminate the chance of lost contact between the supports and the table, the loading is limited to the gravity force of masses up to 12 kg (experimentally tested). This is in fact due to the limitation of experimental set up where the panel only rests on four supports and extra loading will change the boundary condition and the testing condition may no longer be the same.

### 3.2.2.4 Linearity of Results

In this experiment, the acquired data is employed, as a set of known experimental data, for generating two data sets: ANN training and experimental test data sets. In fact, acquired data is used in a programme utilising random number generation and superposition theorem functions to

generate the required number of loading and response data sets. But before that, it is needed to make sure that the superposition assumption of linearity relation is the case here.

As was described earlier, to utilise the superposition technique for data generation, data should have a linear relation inherently. In order to check this, in the range of the loading (0-12Kg), some experiments are performed and their data are compared. Investigations indicated a linear relation between the loading from gravity force of the weights and the response of the structure to that loading condition in the range of the loading (0-12Kg). Some examples of this linear relation are illustrated in Figure 3-10 for randomly selected strain sensors and loading locations. This confirms that superposition can be used.

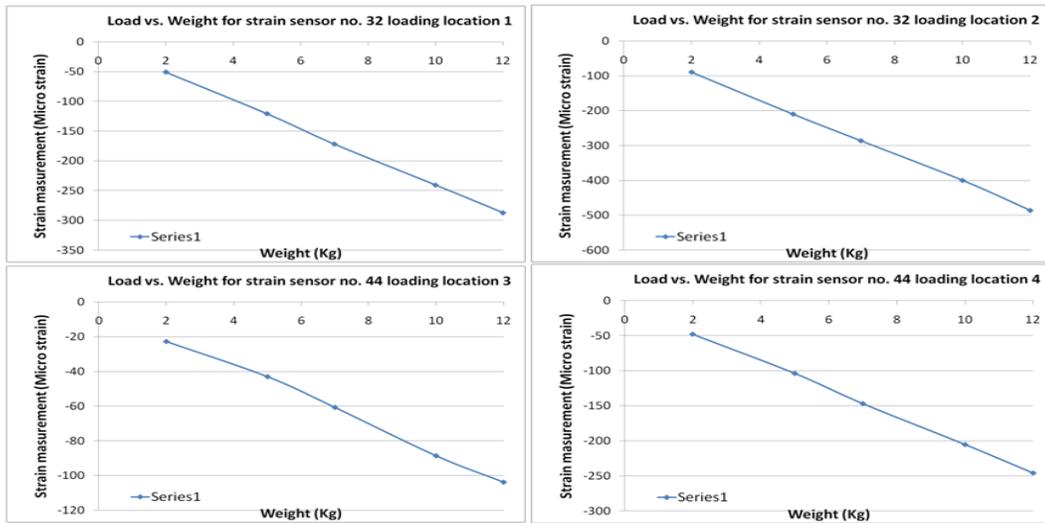


Figure 3-10: Examples of the linear relation between the loading and the response of the structure

### 3.2.2.5 Small Displacement Test results (Network Prediction)

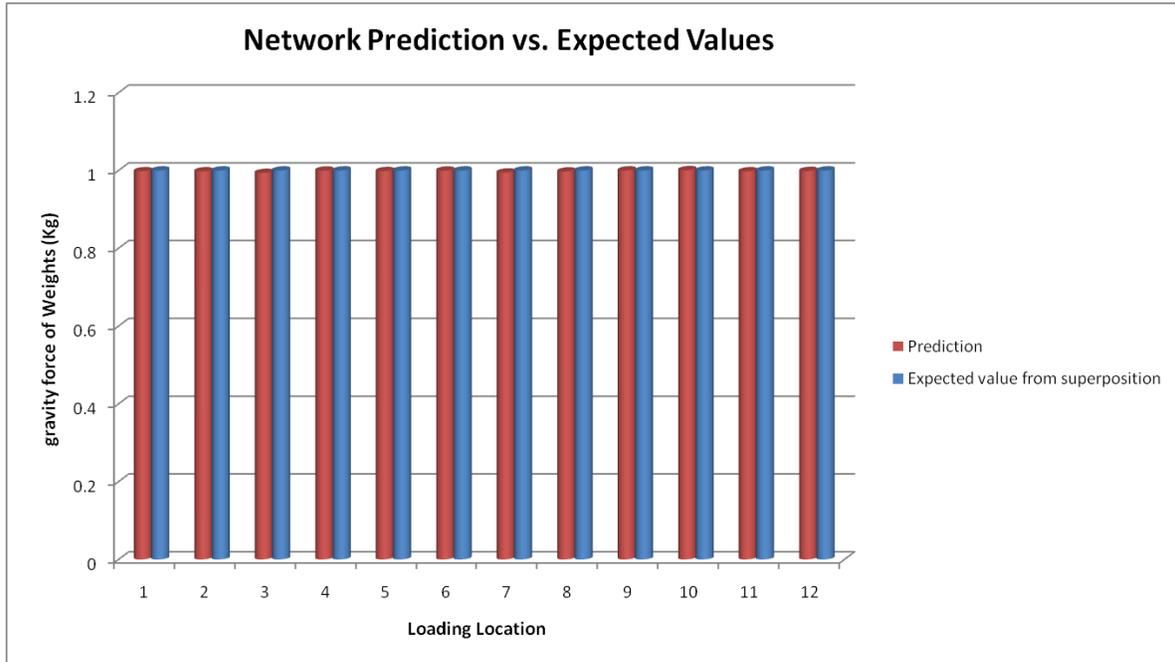
In order to generate the required number of loading and response data sets employing the superposition method (the system shows linear behaviour), some loading and response data sets are experimentally generated and normalised to generate a unique loading set. In this loading data set, the structure strain readings as the response to specific loading conditions are required. The specific loading conditions are when all twelve positions are individually loaded by the same gravity force of mass at the time. Due to the fact that this data set is the resource for generating all other different random load data sets employing the superposition theorem, a series of tests is

performed and then the average values are used to reduce the effect of human error. For this purpose, each of the twelve loading points is loaded by the gravity force of five various masses (selected in the range of 0-12Kg as 2Kg, 5Kg, 7Kg, 10Kg, and 12 Kg). The data acquisition system acquired and saved the response of the system. All the data sets are normalised and averaged to have twelve data sets for the load cases when one load at a time is applied to the panel and the rest are zero. Each data set for loading cases is saved as a column vector with 60 rows. The first 48 rows are strain readings and rows 49-60 are the dead weight values applying the loads. The complete one at a time loading cases will be a matrix of data set having 60 rows and twelve Columns. In this thesis, this matrix is named: “original reference data matrix”. Using the superposition theorem, this original reference data matrix is manipulated and the number of desired training data sets are produced. Each training data set consists of a matrix with size of  $48 \times K$  strain readings as the training input data to network and a matrix with size of  $12 \times K$  weight values as representative of the loads as the training target data. A back-propagation network with a hidden layer is employed initially to be trained and tested.

In order to test the network, a set of data was generated employing the experimental set up and test rig. Since this set of data is not seen by the network in the training stage, it is used to evaluate the prediction of the system. In order to test the network, a set of experimental data was generated employing the experimental set up and test rig. The training stage of the network is a time consuming procedure and several characteristics of the network and data sets can be configured to have a generalised trained network capable of good prediction. Figure 3-11 shows the comparison between actual and predicted loads. The data for this graph is from a data set in the form of a matrix of  $60 \times 1396$  having  $K=1396$  loading cases, each containing 60 data (48 Strain and 12 load representatives). This data was generated by the superposition method randomly for gravity force of the dead weights between 0-20 Kg. In order to enhance the system performance some noisy patterns ( $\pm 1$  micro strain) are introduced to randomly chosen strain data.

It is not possible generally to quantify the performance characteristics and bounds of an ANN system by any analytical techniques. Therefore, it is very important to subject it to a thorough testing regime to ensure it will perform adequately. Therefore, the performance accuracy,

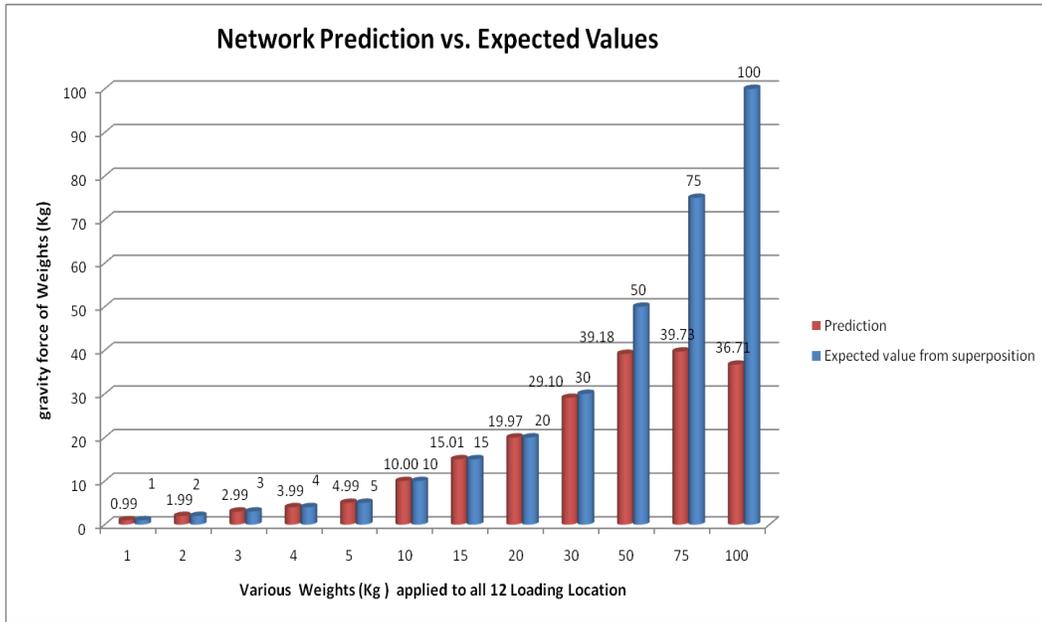
reliability and robustness of the system are investigated by presenting it with some extreme inputs which are within the possible bounds of system operation. For this purpose, the network is presented with new data to check its functionality. The results in Figure 3-11 indicate an error of less than 0.0000062 % between the predicted and expected values.



**Figure 3-11: Net. Predicted vs. Expected Values for new data generated data by superposition**

In order to check the prediction of the network, the same test is performed for different loading conditions. The behaviour of the system prediction over a range of loading quantities was studied. As it is illustrated in the Figure 3-12, the prediction is almost perfect for the range that the system is trained with its data (0-20 Kg). This figure indicates behaviour of the system prediction over a range of loading quantities where each time, all the 12 positions are loaded equally. For instance once all the 12 positions are loaded with 5 Kg masses and the structural responses are read through DAQ system. This set of strain readings are introduced to the previously trained network. The network ideally has to predict 5 Kg for all the 12 loading positions. With the same error margin discussed earlier, predictions agreed with what were expected. Although the network was trained with data for load cases ranging 0-20 Kg, the results indicate that the network is capable of prediction approximately up to 30 Kg. It can be

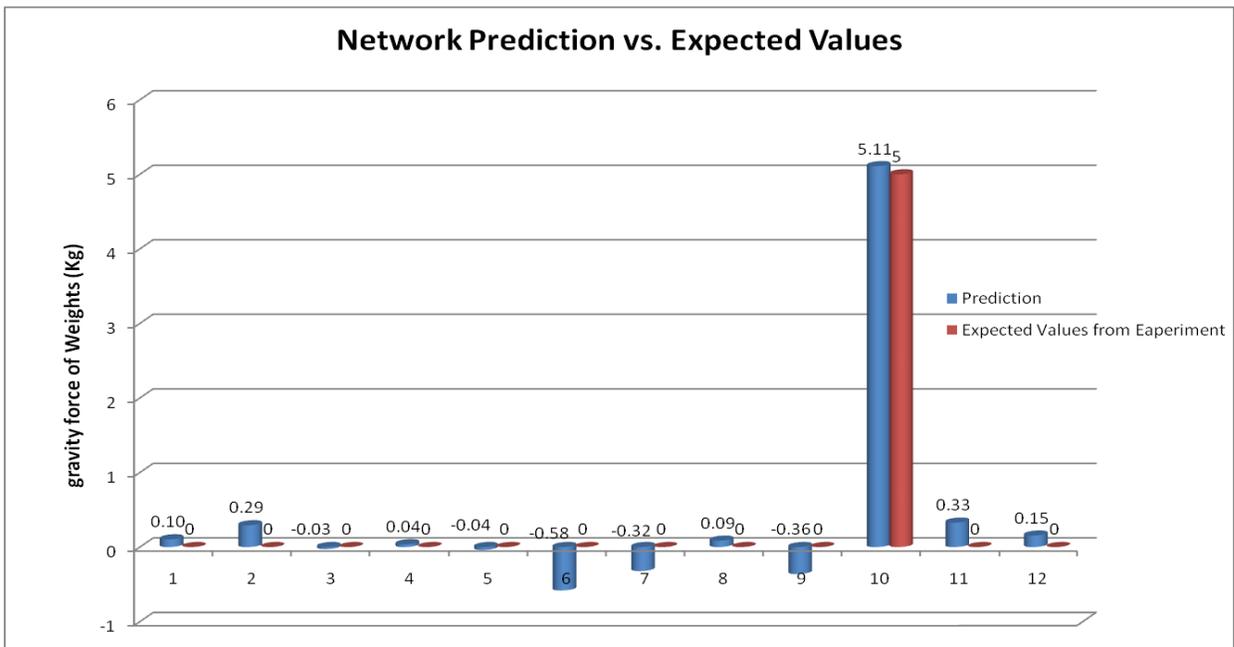
noted that as range of the load cases exceed that of which the network is trained with, the prediction loses its accuracy. This is due to the inherent lack of extrapolation properties of ANN. In other words, the network is able to predict best in the same criteria that it has been trained with and is subject to upper and lower bound of the loading envelope. To subject the system to extreme upper and lower bound loads of the working load envelope of the operating conditions, the system should be trained with a bigger working load envelop to predict realistic loads.



**Figure 3-12: Behaviour of the system prediction over a range of loading quantities**

The results to date indicate the suitability of this method for load prediction. In the next investigation, the ANN is experimentally validated. A MATLAB script is developed in the way that it is able to acquire strain data directly from the panel and introduce it to the network as new input data (see appendix 3-5 for the developed MATLAB program). The trained network is now able to simulate and predict loading data in terms of the applied mass in Kg in real time and get the panel responses as the input and relate them to the load that has caused them. As a testing regime to ensure network performs adequately, the performance accuracy, reliability and robustness of the system are investigated by presenting it with some extreme inputs which are within the possible bounds of system operation. This can be having one load at one location and the others are assumed zero. Figure 3-13 is a random example of the system performance for

load prediction. Figure 3-14 is another random example of the system performance for load prediction where this time two loading locations are loaded and the rest are not loaded. The negative values are due to the network approximation for the new set of inputs. In fact, due to the errors in the data acquisition stage, a slightly different pattern is introduced to the system and the network approximated the output with some error (see Figure 3-15). Further improvements in terms of more accurate data acquisition and optimised training parameters for the network may solve this issue.



**Figure 3-13: Net. Prediction vs. Expected Values for experimental data (one area loaded)**

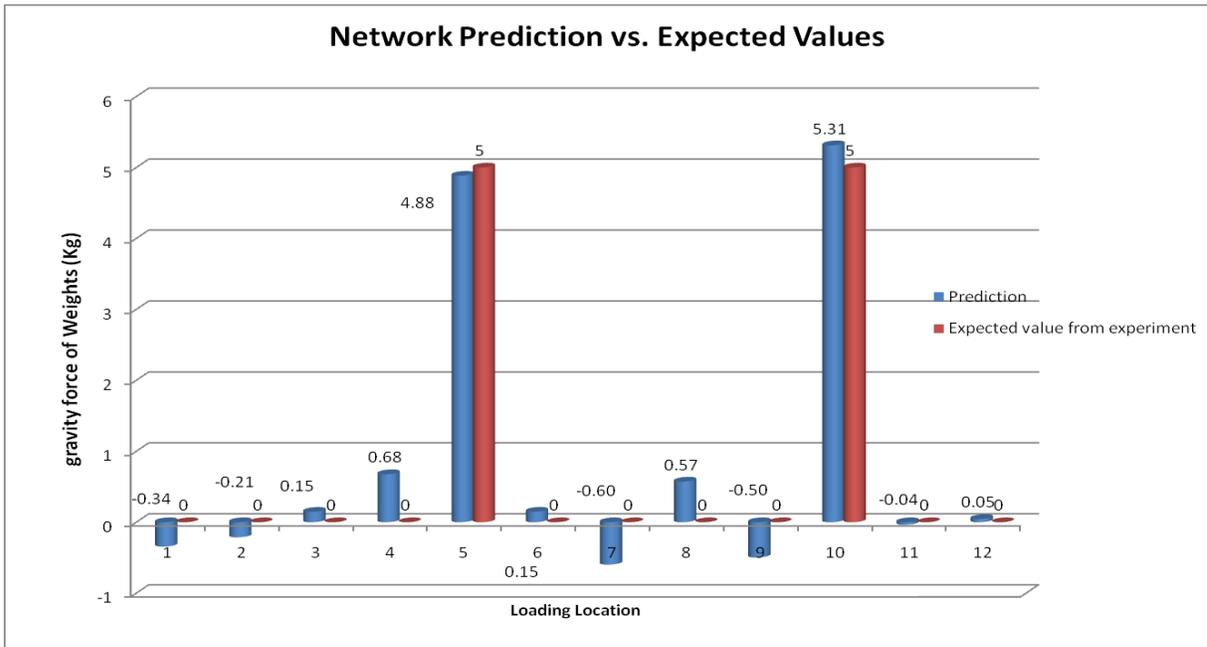


Figure 3-14: Net. Prediction vs. Expected Values for experimental data (two areas loaded)

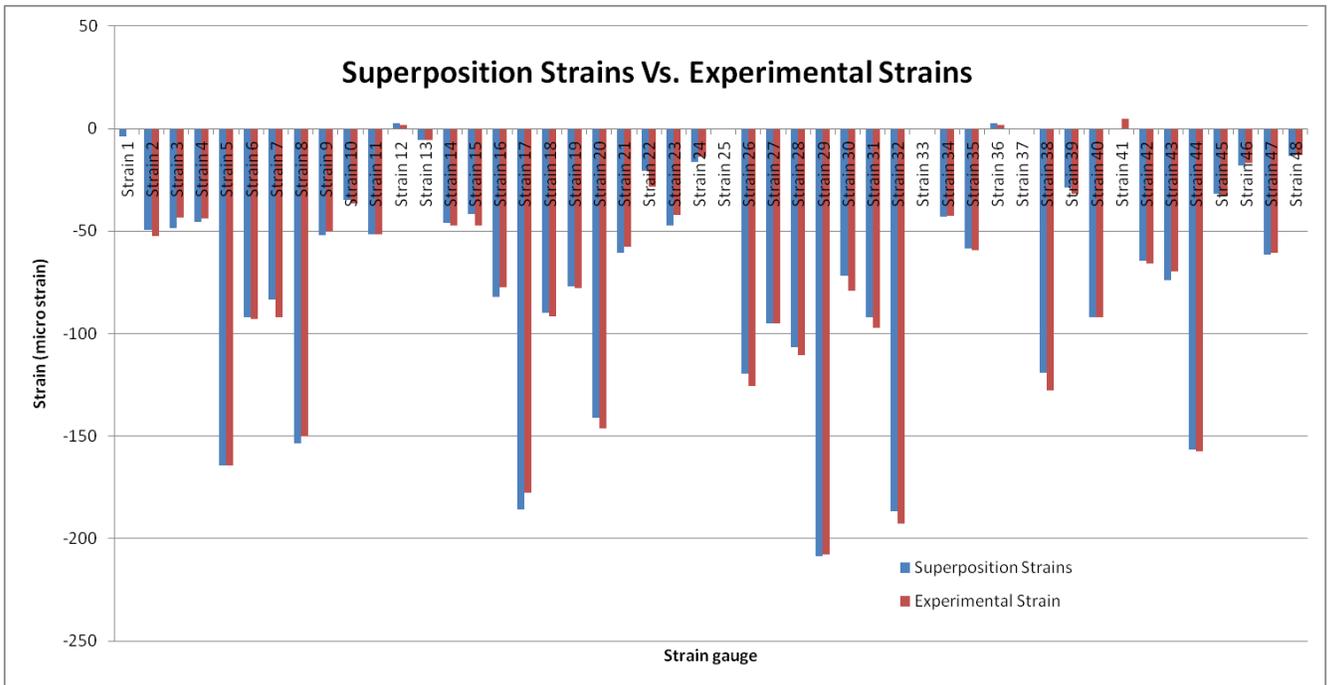


Figure 3-15: Comparison of the experimental data and data generated by superposition

### 3.2.3 Summary and Conclusion

It is shown that the inverse problem approach can be used to predict the loads applied on a marine structure composite panel. Results from a small displacement experiment setup indicated very good performance of the methodology in load prediction which can be achieved in real time, providing an accurate load history for a component. This potentially makes the system ideal for solving many classes of engineering problem that require load monitoring and/or structural health monitoring. The results of this study can be summarised as follows:

- An ANN can be trained using experimental based data for direct and inverse problems.
- A real time experimental and an inverse problem engine can be created using a combination of ANN and experimental data.
- It is shown that a mathematical relationship exist between the applied loads and the surface strains and can be related by employing the ANN system which always converges and the MSE is in the range of acceptable error.
- The system is effectively capable of predicting the static loads.
- Although there is no general rule to select the training parameters, changing various parameters in the training stage of a network (structured or randomly) may result in improvements in the network's learning ability.
- When obtaining response data from the composite panel, generally the higher the load the higher structure response in terms of displacement values (in microstrain) are experienced. Since the noise of the DAQ system is a fixed value (in microstrain), with higher loads the effect of the noise in DAQ system is reduced and the system will indicate more accurate results.
- The main source of error was found to be the difference between the experimental strains and the strains acquired by the data acquisition system which is caused by human and device error.

Furthermore, this experiment is performed employing a large number of strain readings. However, it is not very practical and cost effective to attach many sensors to the panel. Therefore, more research is needed to optimise the number of sensors and investigate the effect of geometry size in predictions.

### **3.3 Sensor Optimization**

In Section 3.2, the capability of the system to estimate the loads applied on a marine structure composite panel was demonstrated. Furthermore, it was also concluded from the previous study that more research is needed to optimise the number of sensors and investigate the effect of geometry size in predictions. This is due to the fact that for an ideal system, the number of sensors should be minimised to reduce the overall training time, cost and weight of the system. In fact, the aim of this optimisation is to use the minimum number of sensors with maximum coverage with respect to the load. In this section, the research undertaken to investigate the number of sensors required for accurate load estimation by optimising the method is presented. The methodology of the experiment, results and discussions as well as this particular experiment conclusion are described.

#### **3.3.1 Methodology**

The methodology employed to generate training data sets and optimise the quantity of sensors based on the performance of various trained ANN as an inverse problem solver for quantifying the load applied to the composite panel is presented in this section. The first stage of the investigation was to design a load quantification methodology for the panel utilising an ANN. In the second stage the load quantification methodology was validated by comparing loads estimated by the ANN with the known loading cases of the panel.

##### ***3.3.1.1 Generation of ANN Training Data***

The efficiency of the training data collection process can be increased by reducing the amount of data collected. This is achieved again in this experiment by using the theory of superposition (test is still small displacement loading and data are linear) to generate training and testing patterns from the independent parent patterns, as discussed in Section 3.2.1.1. The theory of superposition states that the strain at a point on a structure due to a series of loads is equal to the sum of the strains from each individual load case. Using this theory, an infinite number of training patterns can be generated by applying one known load to each location on the structure individually.

### 3.3.1.2 ANN Architecture/Topology.

In this study a common Back-propagation ANN architecture is used and trained employing MATLAB Artificial Neural Network toolbox capabilities. An iterative process was used to determine the optimum network architecture for the panel based on the value of the final Sum of the Squared Errors (SSE) of each network tested. As described before, SSE is a measure of the discrepancy between the data and an estimation model. A small SSE indicates a tight fit of the model to the data. Table 3-2 lists the major parameters of the network architecture selected based on convenience sampling used in this study.

Table 3-2: ANN architecture parameters (optimisation experiment)

Architecture	Feed Forward Back-propagation
Number of layers in each network	2
Range of load estimation	0-196.2 [N]
No. of inputs (surface strains)	48,18,15,12
No. of output layer neurons (loads)	12
No. of each hidden layer neurons	[20 20]
Number of training patterns	1116
Number of testing patterns	280

### 3.3.1.3 Optimisation

The equipment set up of this experiment is described in Section 2.4.2. In order to optimise the number of gauges, various sensor configurations are employed to acquire and generate training data sets and the corresponding ANN is trained. The performance of each ANN in terms of SSE is used for comparison. The aim of this optimisation is to minimise the number of strain gauges needed to reasonably estimate the magnitude of load at twelve loading locations on the composite panel.

In this study, the number of rosettes are reduced/optimised in several steps and the effect on the performance of the load estimation methodology is investigated. Due to the inherent characteristic of an ANN, reduction of the number of inputs is limited by the number of outputs. Hence, a minimum of twelve strain inputs are required to be able to successfully estimate twelve load outputs. Therefore, the minimum number of rosettes that could be used to successfully predict twelve loads is four. In order to optimise the number of the strain gauges, the number of gauges are reduced step by step from 16 (48 readings) to 6 (18 readings), 5 (15 readings) and 4 rosettes (12 readings) respectively. There are multiple permutations for selecting 6, 5 or 4 rosettes from the 16 rosettes attached to the panel. Since the aim of this study was first to find the optimum number of gauges, random

permutations of gauges were based on convenience sampling and the ANN performance using the SSE values for each ANN were compared. It was then possible to determine the optimum number of gauges required to achieve a high quality estimation of the 12 loads from this study. A further strategy for optimising the sensor locations was then also investigated.

#### **3.3.1.4 ANN Validation and Performance**

Finally, the validity of the ANN using the optimised strain gauges was evaluated by comparing the load estimated by the ANN with known loads applied to the panel (problem data). Experimental problem data is the captured strain data from the optimised strain gauges attached to the panel while it is being loaded. It is essential that the strain data is captured at identical locations for both the training and problem data. The first validation study utilised load and strain data generated from the original superposition data collected to produce the training data of the optimised sensor configuration. This meant that any issues with the repeatability of the strains collected for a given load were removed. In the second study, problem strain data for the same sensor configuration was captured directly from the panel under different loading conditions (i.e. one or two random loads were placed at random locations on the panel) and again the estimated loads compared with the actual applied loads.

### **3.3.2 Results and Discussion**

Having introduced the proposed methodology, this section analyses and discusses the results of this optimisation study.

#### **3.3.2.1 Optimisation**

A selection of the optimisation tests and the ANN performance results are presented in Table 3-3. The results show that as the number of the rosettes is reduced, the SSE values as performance indicators of various random tests are reasonably small values. The highest SSE performance value is 2.963 which itself is a very low, even negligible error. This implies that even for cases with only 6, 5 or 4 rosettes, the ANN is trained well and it is capable of estimating the magnitude and position of the applied loads. The results indicated that utilising only 4 rosettes (12 strain readings) it is possible to accurately estimate applied loads on all 12 locations ( $L_{1-12}$ ).

**Table 3-3: Comparison of ANN performance SSE for various sensor configurations**

	<b>16 Rosette Locations (<math>S_L</math>)</b>	<b>Performance SSE</b>
<b>Random Test 1</b>	1-16	1.520
	<b>6 Rosette Locations (<math>S_L</math>)</b>	<b>Performance SSE</b>
<b>Random Test 1</b>	2, 3, 6, 11, 14, 15	1.670
<b>Random Test 2</b>	3, 6, 8, 9, 11, 14	0.842
<b>Random Test 3</b>	5, 6, 7, 10, 11, 12	2.376
<b>Random Test 4</b>	6, 7, 8, 9, 10, 11	2.038
	<b>5 Rosette Locations (<math>S_L</math>)</b>	<b>Performance SSE</b>
<b>Random Test 1</b>	2, 5, 7, 10, 15,	1.442
<b>Random Test 2</b>	3, 6, 7, 10, 14	2.963
	<b>4 Rosette Locations (<math>S_L</math>)</b>	<b>Performance SSE</b>
<b>Random Test 1</b>	3, 6, 11, 14	2.643
<b>Random Test 2</b>	4, 7, 10, 13	2.185
<b>Random Test 3</b>	5, 8, 9, 12	2.642
<b>Random Test 4</b>	5, 8, 9, 12	1.208

Furthermore, in practice, it is more desirable to have the minimum number of strain gauges attached only in one small area on the structure. Hence, 4 new rosettes were attached much closer together in the middle of the panel (Figure 2-22,  $S_{L17-20}$ ) and the performance of the ANN was investigated. Achieving reasonable results enables the gauges to be placed in just a small portion of the panel and yet be able to accurately estimate the position and the value of 12 externally applied loads ( $S_{L17-20}$ ). The SSE performance value of this test was 8.259 which is still a small value and indicates a tight fit of the model to the data.

### **3.3.2.2 Validation using Superposition Data**

In order to validate the trained network, a set of problem strain data was generated from the panel using the 12 strain readings from the optimised patch ( $S_{49-60}$ ). Since this set of data had not been seen by the network during the training stage, it can be used to evaluate the accuracy of the ANN's load estimation. The theory of superposition was used to generate problem data (12 loads and 12 strains generated on the panel surface from these loads) for loads between 0 N and 196.2 N. These were introduced to the ANN and the estimated loads calculated by the ANN compared to the expected load profile. Figure 3-16 and Figure 3-17 indicate typical examples where the estimated load value by ANN can be compared with expected values from superposition. The figures indicate that ANN estimates the load data very well from superimposed strain data.

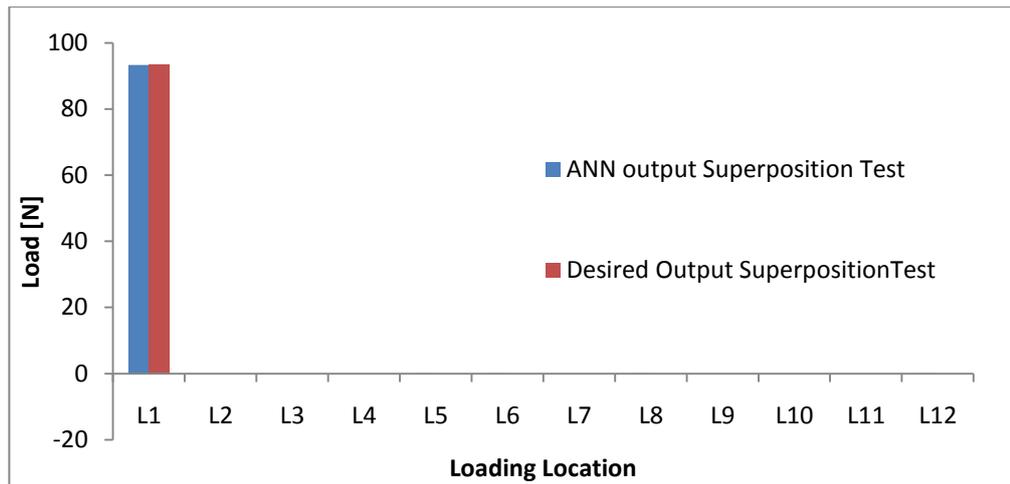


Figure 3-16: Estimated load data by ANN vs. expected load values (from superposition at location L1)

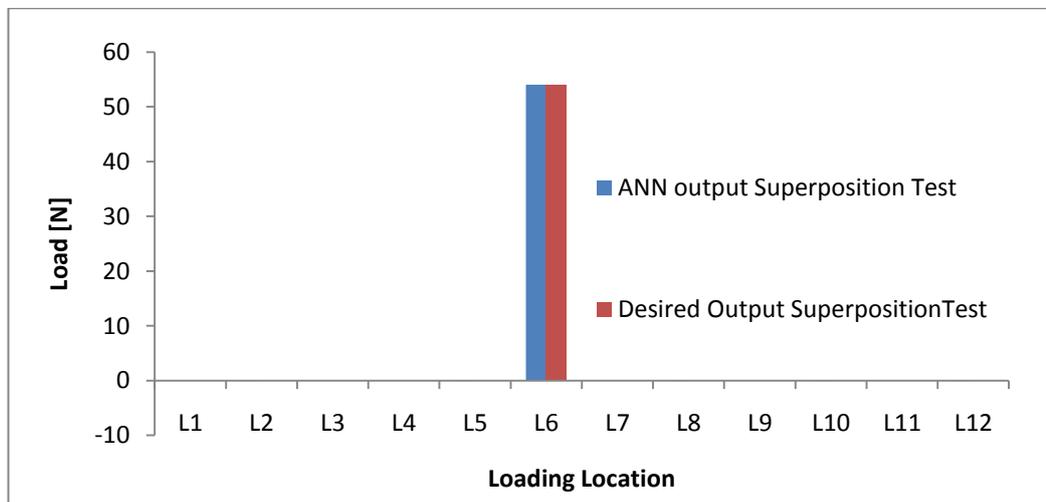


Figure 3-17: Estimated load data by ANN vs. expected load values (from superposition at location L6)

### 3.3.2.3 Validation using Direct Data

In order to validate the generalisation performance of the trained ANN system, new acquired strain data directly from the experiment (which ANN was not trained with) are introduced to the trained network. This strain data was gathered by applying loads at the extremes of the range that the network was trained to estimate (0-196.2 N). Figure 3-18 and Figure 3-19 indicate typical examples of the comparison between the actual loads applied to the panel with the ANN estimated loads generated from the introduced problem data. In these examples, one load is applied to the panel. For both sets of problem data it can be seen that the ANN can again estimate the load at the loaded locations with a high degree of accuracy. However, slightly different values (for instance in Figure 3-18: 51.16

N Vs. 49.59 N instead of 50 N and in Figure 3-19: 153.56 N Vs. 147.79 N instead of 150 N) are achieved. These small errors are related to the network generalisation capabilities to handle a new set of inputs. In fact, due to the errors in the data acquisition stage, a slightly different pattern is introduced to the system and the network estimated the output with some error. Further improvements in terms of better data acquisition and optimised training parameters for the network may solve or improve this issue.

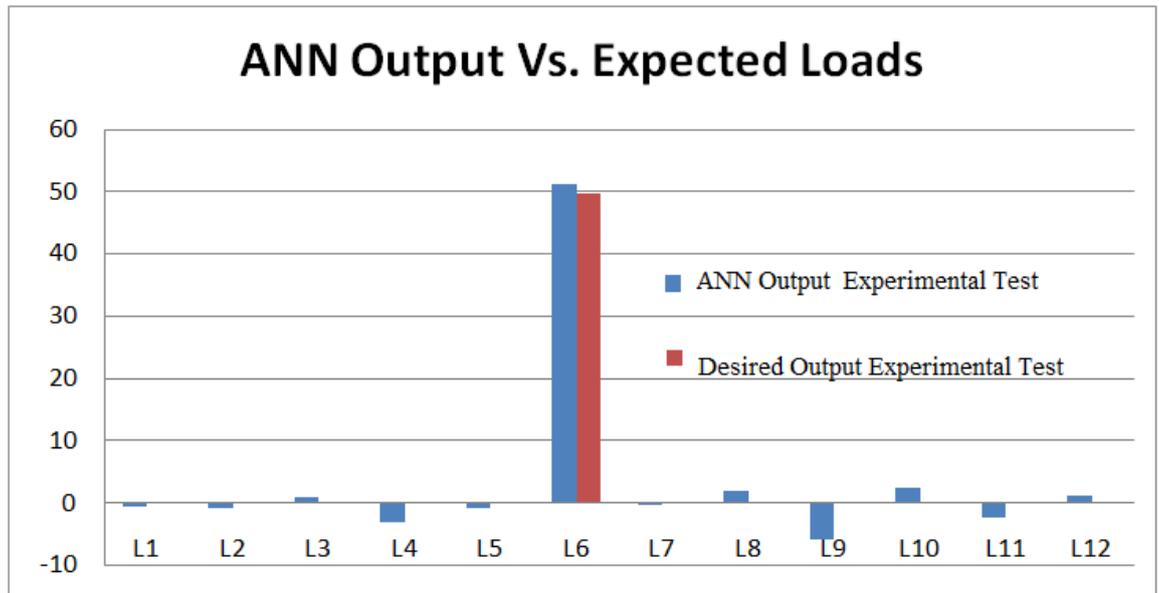


Figure 3-18: Estimated load data by ANN Vs. expected load values (from experiment at location L6)

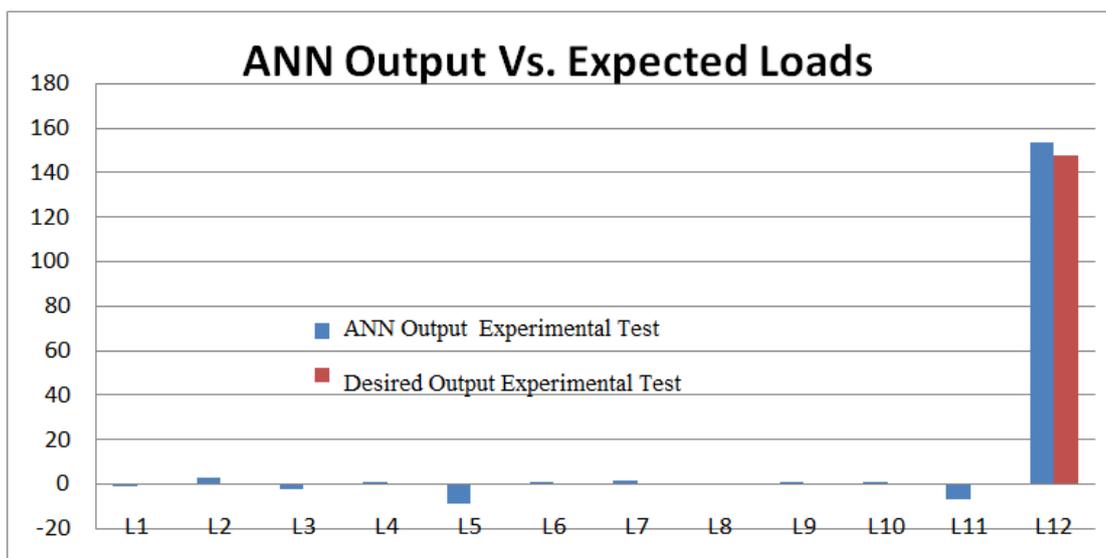


Figure 3-19: Estimated load data by ANN Vs. expected load values (from experiment at location L12)

### 3.4 Discussion and Conclusion

Establishing an inverse problem analysis approach for structural analysis can result in important advantages over simulation, numerical or theoretical methods. Through utilising such a method, knowledge of the component material constitutive laws and component geometry are not required. In contrast, they are necessary for valid and accurate simulation, numerical or theoretical analysis. The results presented in this section show that the inverse problem method, utilising an ANN, can accurately estimate the position and magnitude of 12 static loads applied to the composite panel from 4 strain gauge rosettes placed close to each other in the centre of the panel. The results indicate that the system always converges, the SSE is small and in the range of acceptable error. This means that an ANN can be trained using experimental data to solve inverse problems and accurately estimate the static loads. Although, the first validation study indicated that the estimated load data by the ANN almost perfectly fits the expected load values from superposition, small error values were seen in the second validation study. The main source of error was found to be in the reliability of the data acquisition system utilised due to the large variance in the strain data collected at different time intervals. However, the noise to strain ratio decreases as the load increases which results in less variances in strain data patterns. Having more similar strain data patterns to those employed to train the network leads to a better load estimation output.

The ability to measure the actual load history of a craft in-service would enable the designer to validate the load estimation and structural design tools used during the design stage of a craft. This would lead to the development of more optimal structure designs for this type of craft. The operational safety of the craft can also be improved by having a real-time load monitoring system that is able to detect any degradation of the structural integrity and defects within the structure.

The aim of this experiment was to establish an inverse load monitoring approach based on directly acquired structural response from an optimised set of sensors. It has been shown that the inverse problem approach can be used to estimate 12 loads applied on a composite marine panel from the strain measurements from 4 strain gauge rosettes. A comparison of the optimised ANN loads with the actual applied loads indicated a very good performance of the methodology. This was achieved in real-time, providing an

accurate load history for a component without requiring knowledge of the material properties or component geometry. This potentially makes the system ideal for solving many classes of complex engineering problem that require load monitoring. The results of this study can be summarised as follows:

- An ANN can be trained using experimental based data from only 4 strain gauge rosettes
- A real time experimental and an inverse problem engine can be created using a combination of ANN and experimental data all from the middle of the panel close together
- The system is effectively capable of predicting the static loads with the minimum possible strain gauge readings
- The main source of error was found to be the difference between the experimental strains and the strains acquired by data acquisition system which is caused by human and device error.

### **3.5 Summary**

The findings from both experiments of this chapter indicate that should the structural response have a linear behaviour within the range of applied loads, the ANN can be trained employing experimental based data. However, non-linear behaviour is inevitable for higher load ranges and needs further investigations. In the next chapter, the general behaviour of a composite panel with attached strain gauges under a large displacement and drop test will be investigated.

## **Chapter 4. General Behaviour of Composite Panel with Attached Strain Gauges under a Large Displacement and Drop Test**

This chapter introduces the research undertaken to investigate the general behaviour of a composite panel with attached strain gauges under large displacement as well as transient loading condition.

## 4.1 Introduction

In the previous chapter the proposed methodology for quantifying static pressure loads on a marine composite panel from strain measurements collected from the panel under small displacement was investigated. The performance of utilising an ANN was acceptable and the system proved to be satisfactory and is effectively capable of predicting the static loads. Furthermore, the quantities of the sensors were optimised and acceptable load estimation was still achieved. However, the proposed methodology can be used for situations when a linear relation exists between the load and the structural response. In this chapter, the research undertaken to further develop the methodology is presented. The ideal load monitoring system should be able to successfully estimate applied loads on the structure even if the relation between the load and structural response is not linear. Normally, nonlinear behaviour is exhibited from marine panels. This is due to the fact that the panel deflections exceed the linear limit. Therefore, the problem is treated as a non-linear problem where the displacement can no longer be accurately predicted using the linear theories. In such cases, the classical inverse approach used previously, based on utilising data generated from superposition, can no longer be employed due to the complexity of the displacement function.

In this section, a new experiment is designed and performed having a larger load range. In the first section of this chapter, the proposed load monitoring system is modified to handle the nonlinear loading conditions. The aim of the first experiment is to investigate the ANN ability to accurately estimate static pressure loads applied to up to 13 locations on the structure using 16 strain readings from 16 unidirectional strain gauges when the panel is under large displacement.

In addition to static loading conditions, it is necessary to perform more investigations on the suitability of the proposed methodology for in-service load monitoring of marine structures under transient load conditions such as slamming. For this purpose, an experiment is designed and performed to further develop the ANN methodology for quantifying pressure loads on a marine composite panel under transient load conditions from strain measurements.

Therefore, in the second section of this chapter, the research undertaken to develop a specific methodology and to modify the proposed system for in-service load monitoring of a composite panel under transient load conditions such as drop test are described.

## **4.2 Large Displacement Experiment Setup under Static Loading condition**

This section reports on the research undertaken to further develop the ANN methodology to quantify static pressure/central load on a composite marine panel from its non-linear displacements.

### **4.2.1 Methodology**

The methodology employed to evaluate the suitability of ANN as an inverse problem to relate structural response to the loads applied is presented in this section. A back-propagation ANN was designed and developed and trained within the MATLAB simulation environment to measure transverse load on a flat composite marine panel. The estimated output was then validated by comparing it against both experimental and numerical data.

#### **4.2.1.1 Generation of Training Data**

In order to investigate the general behaviour of a composite panel used in marine structures and the application of the ANN methodology under large displacement, the same composite panel is employed to set up a fully fixed test rig (see section 2.4.1 for equipment set up). Normal loads were randomly applied to the top surface of the panel at thirteen grid intersections (L1-13). Depending on the proximity of the gauge and the applied loads different gauges exhibited different levels of sensitivity which was as expected. To produce efficient training data the strain data should be captured at the sensitive regions (i.e. the strain at those locations must vary significantly due to changes in load level). In addition, the strain data collected must provide a unique response for each load distribution. If strain is collected from non-sensitive regions of the panel and/or the strain data collected is not unique for each load distribution the ANN is less likely to be able to find a function relating the input and output.

For nonlinear structures an alternative approach is needed in order to generate the required training data. There are two ways in which such data can be generated, a) experimentally or b) using a nonlinear Finite Element Analysis (FEA) solver. Generating the required training data experimentally is very time consuming and labour intensive. Therefore, nonlinear FEA analysis using a script that allowed automatic generation of random load on the panel was utilised to generate the training data. ABAQUS 6.10-1 FEA software (SIMULIA) was used. A script

function written in Python language was used to iteratively run the software in a batch using different random loads applied at each of the thirteen loading locations on the panel. The FEA model was initially validated to ensure that it represented the actual panel accurately. The validation was achieved by comparing strains collected experimentally with the FEA strains under the same loading conditions. Loads from 100 N to 800 N applied in 100 N increments were placed on the panel one at a time on locations L1 to L13. The strain readings at locations S1 to S16 on the panel were saved for each test. The same tests were performed with FEA to compare with the experimental results.

Once validated, a large number of training (load/strain response) data was able to be generated from the FEA model. In order to increase the efficiency of generating the training data, it was possible to reduce the number of FEA models required to establish the nonlinear strain response for each gauge location. This was achieved by fitting nonlinear curves to data collected for each strain location and using the curves to interpolate strain data for different load magnitudes.

The structural responses of the panel in terms of strain were saved to be used as the input training data set. The corresponding load for each input data set was also saved and utilised as the output training set. Some of these input and outputs were saved separately for testing the network and error minimisation. In this study, sixteen single strain gauge readings (inputs) and thirteen applied loads (outputs) constitute one training data set. At each loading location (L1-L13), static load ranging between 24.525 N and 784.8 N was applied in steps of 24.525 N. In total 1040 training data sets were generated from the nonlinear FEA model.

#### ***4.2.1.2 ANN Architecture/Topology***

Due to inherent non-linearity in data of this experiment, it is expected that the training would not be as fast as training of linear data. Therefore, in addition to the ANN architecture employed in the small displacement experiment, a novel ANN architecture is introduced and investigated. This means two different ANN architectures are studied for this experiment and their performances are compared. The architectures utilised were:

- 1) One network with sixteen neurons in the input layer and thirteen neurons in the output layer is trained to estimate the load on the panel from the strain responses (Figure 4-1).

- 2) Thirteen networks each with sixteen neurons in the input layer and one neuron in output layer are trained and used to estimate the load on the panel from the strain responses (Figure 4-2).

The number of hidden layers and neurons in each hidden layer of the two network architectures were flexible. These were dependent on the complexity of the training data sets and were optimised according to the network performance. The Sum of Squared Errors (SSE) and Mean of Squared Errors (MSE) are common network performance indicators. Through the testing of various network architectures, the optimum network having the lowest performance indicator can be determined. Once the ANN is trained, it can be employed to estimate new loading cases where the same patterns exist. In other words, whenever the same pattern of strain reading as an input is introduced to the network, it will be able to estimate the loads that caused those structure responses. Depending on how well the network is trained (the performance of the network), there will be error between the output data set and the network estimated output (load).

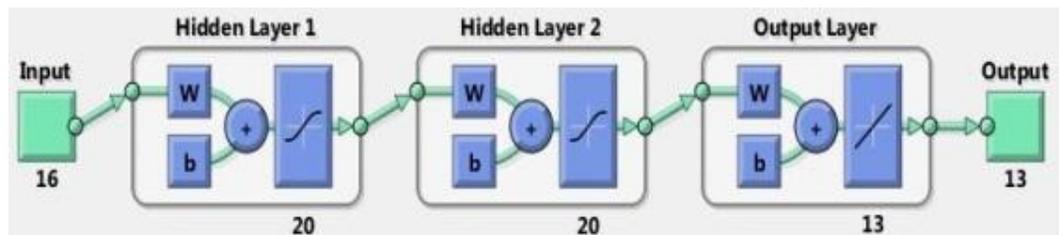


Figure 4-1: MATLAB representation of ANN architecture for method one

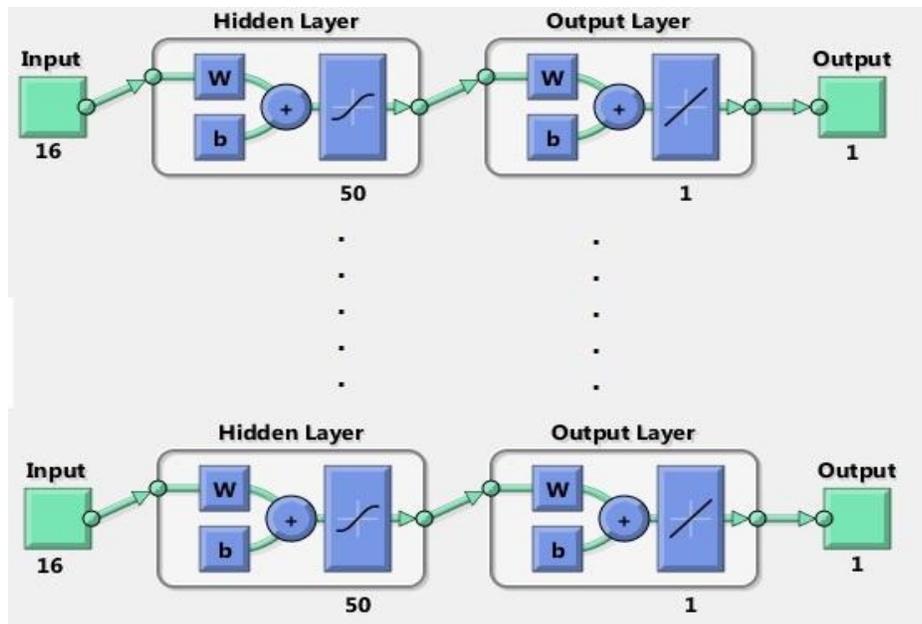


Figure 4-2: MATLAB representation of ANN architecture for method two

#### 4.2.1.3 ANN Validation and Performance

The validity and performance of the ANN method was evaluated by comparing the load estimated by the ANN with known loads applied to the panel which were not seen by the network during the training process. The first validation study utilised load and strain data generated from the FEA model and was compared with estimated loads from the ANN. In the second study, problem strain data was captured directly from the panel and again the estimated loads were compared with the actual applied loads.

#### 4.2.1.4 PYTHON Script of Large Displacement Experiment Description

As it was described earlier, training data for ANN is generated using FEA modelling. This was due to the fact that for nonlinear structures an alternative approach to the superposition theorem could be employed. Furthermore, since experimentally acquiring all the training data is very time consuming and labour intensive, nonlinear FEA analysis using a script that allowed automatic generation of random load on the panel was utilised to generate the training data. In this section, the script utilised to model and simulate various experiments in ABAQUS 6.10-1 FEA software (SIMULIA) is described. Figure 4-3 indicates a flow diagram of this code. For clarification purposes, this script is divided in several parts and can be accessed in Appendix 1.

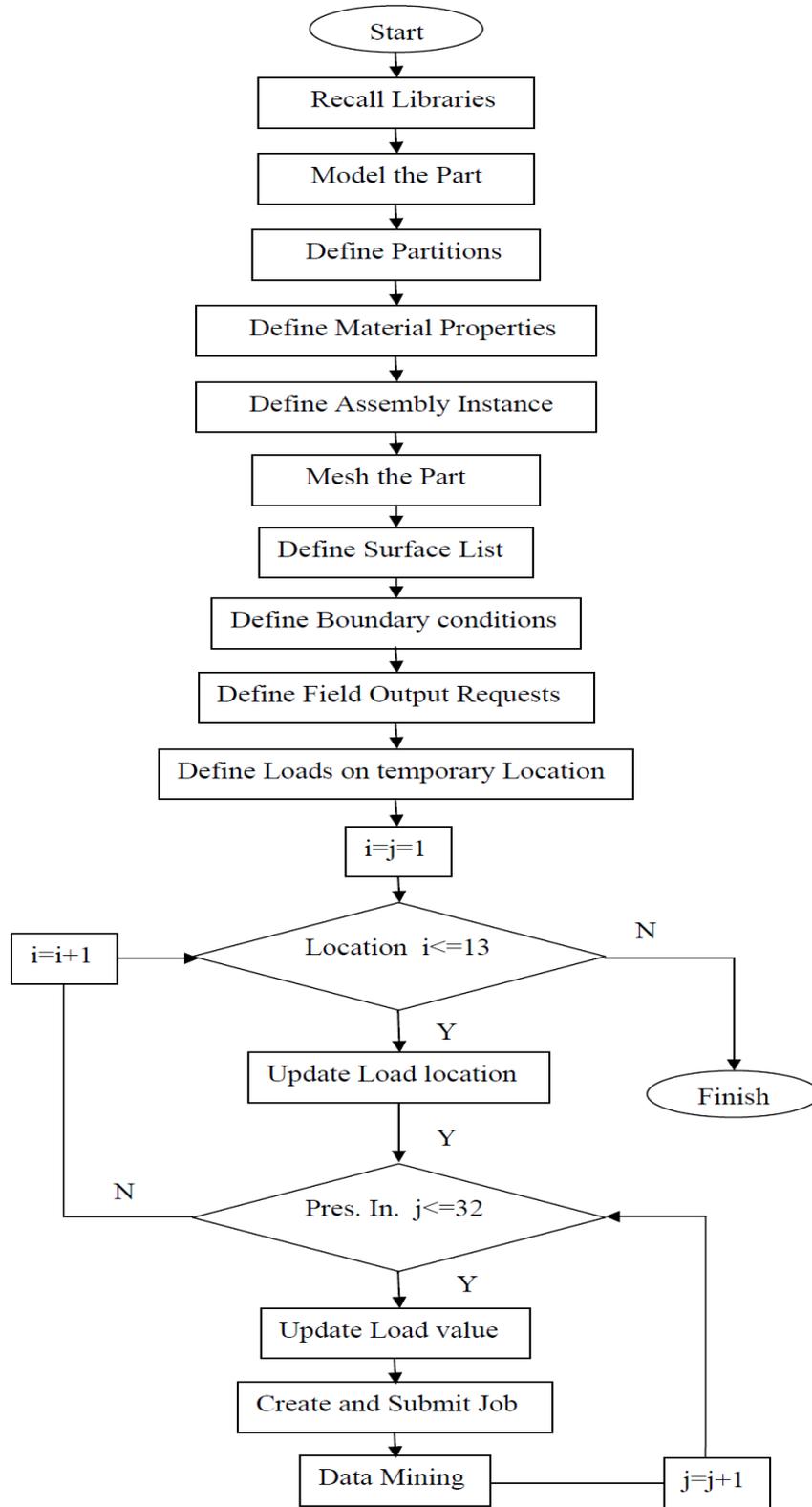
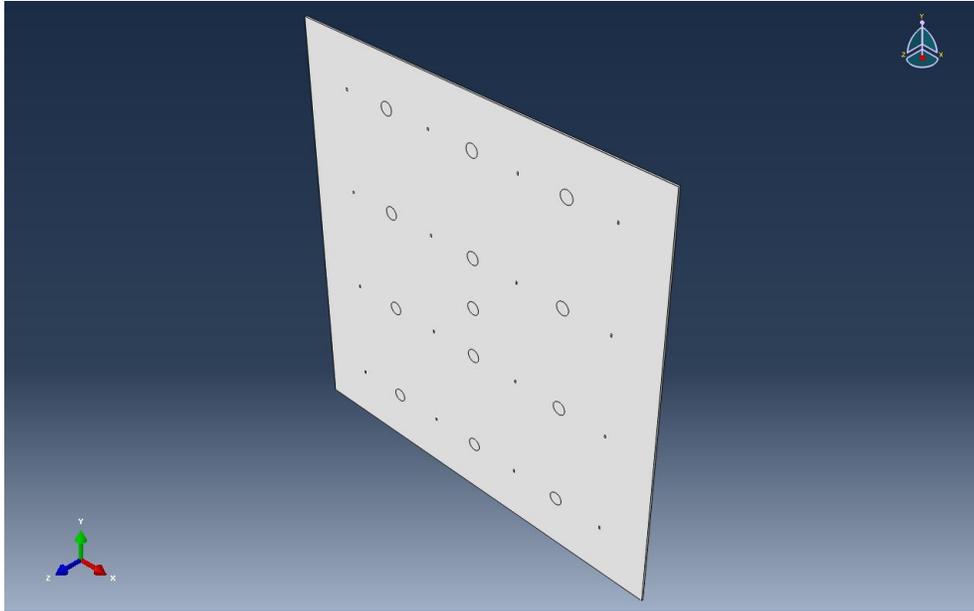


Figure 4-3: Flowchart of PYTHON Script of Large Displacement Experiment Description

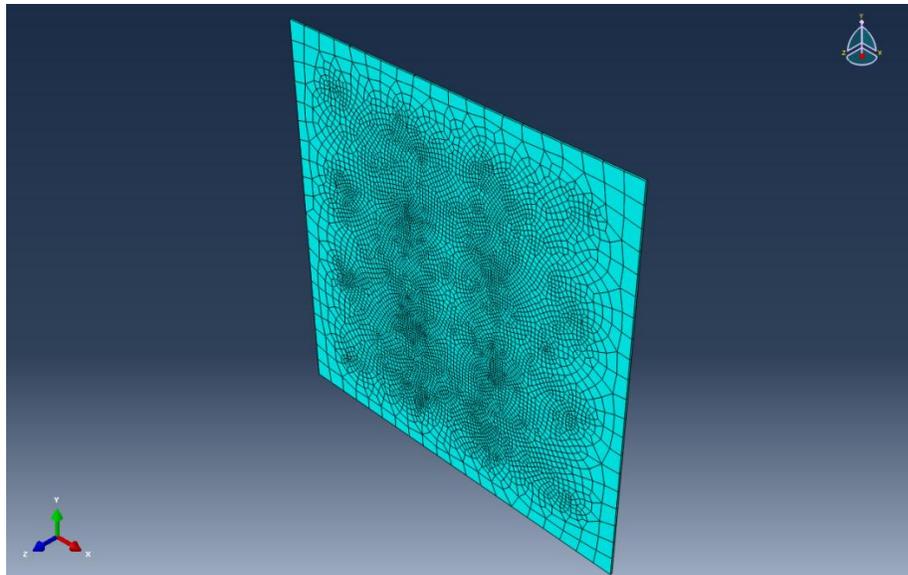
The first part is essential to recall different libraries in programme (script lines 1-15). In Part 2, Modelling of the panel has been defined. If there is no input in each parameter, a default input number is considered. The script lines 16-20 define the test structure under consideration (see section 2.4.1).

The script lines 21-53 define partitions on the panel surface. Within the Part module, Partition toolset can be used to partition a part into additional regions. After a part is partitioned, different properties can be assigned to the resulting regions. For instance, here this helps to define where the loads are applied and where the structural responses are collected. As it is indicated in the Figure 4-4, several partitions are defined on the panel part. This figure indicates that 13 circles are defined to specify loading surface locations as well as 16 rectangles representing the location of strain gauges.

In part 3 (script lines 54-62), the material properties as well as the section are defined and related together. The assembly instance is defined and the part is meshed. The panel part is meshed with seeds having approximate global seed size of 0.05 and the element type used is hexahedral isoparametric. This seed global size is a relative value based on the size of the part instance with no dimension and is selected based on a simple mesh convergence study. After meshing the part, 7031 elements are generated on part. Furthermore, the meshed part as well as all the resulting elements is illustrated in Figure 4-5.



**Figure 4-4: Partitioned panel (16 rectangles and 13 circles)**

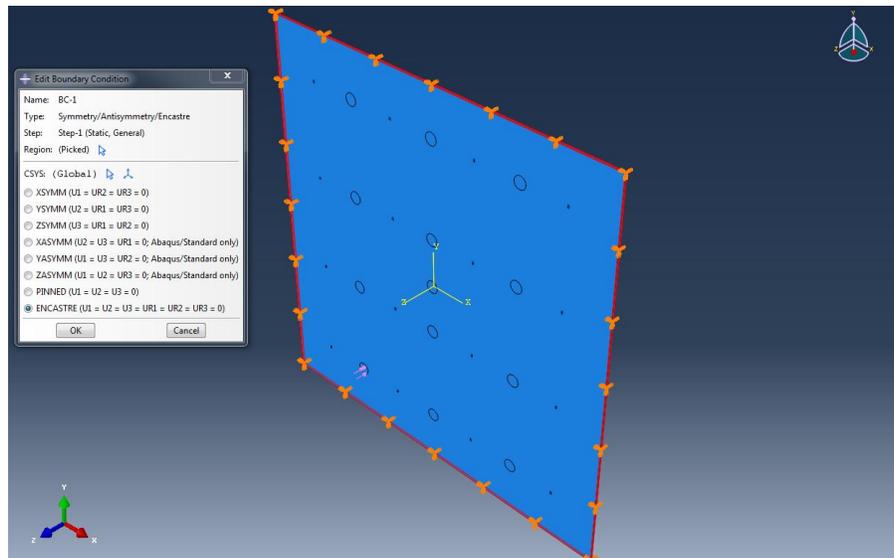


**Figure 4-5: Meshed Part (7031 elements)**

In part 4, the partitions are selected in order and they are assigned to a specific surface feature in the assembly toolset (script lines 63-96). Here, the partitions are used to define a new surface list in the assembly toolset. Each of these surfaces will have particular names which can be easily addressed in the script. This helps the program access each surface easily. The script line 97,

defines a row vector containing the name of each of the created surfaces. This vector can be then accessed easier in the numerical manipulation in a loop.

Part 5 (script line 98) is for definition of the boundary conditions. Due to the fact that the panel is fully fixed, ENCASTRE ( $U1=U2=U3=UR1=UR2=UR3=0$ ) is selected as the boundary condition for all 4 edge surfaces of the panel. The boundary condition of the panel is illustrated in Figure 4-6.



**Figure 4-6: Fully fixed boundary condition**

The field output requests are defined in Part 6 (script line 99). The necessary field output of this model for this experiment is elastic strain components (EE). Furthermore, it is necessary to define the loading of the structure before job creation. The loading is defined in Part 7 (script lines 100\_106). The loading of this experiment is modelled with distributed pressure type loads over the circular area. The modelling simulates the loading from the pressure load applied from the weights distributed over the circular area with radius of 16 mm. Figure 4-7 indicates the loading of Load\_Surf\_1.

The PYTHON script is developed in the way that iteratively runs the software in a batch using different random loads applied at each of the thirteen loading locations on the panel. For this purpose, loops are defined in the script to iteratively simulate various loading scenarios. The flowchart of this iteration in summary is that the model is defined in the first 6 parts and then a

loop is used to iteratively define loads over various loading locations as well as creating a job. Each of the jobs are submitted for analysis individually and the results are post processed and the necessary structural responses from the strain gauge locations are collected and saved in a specific file with a particular order.

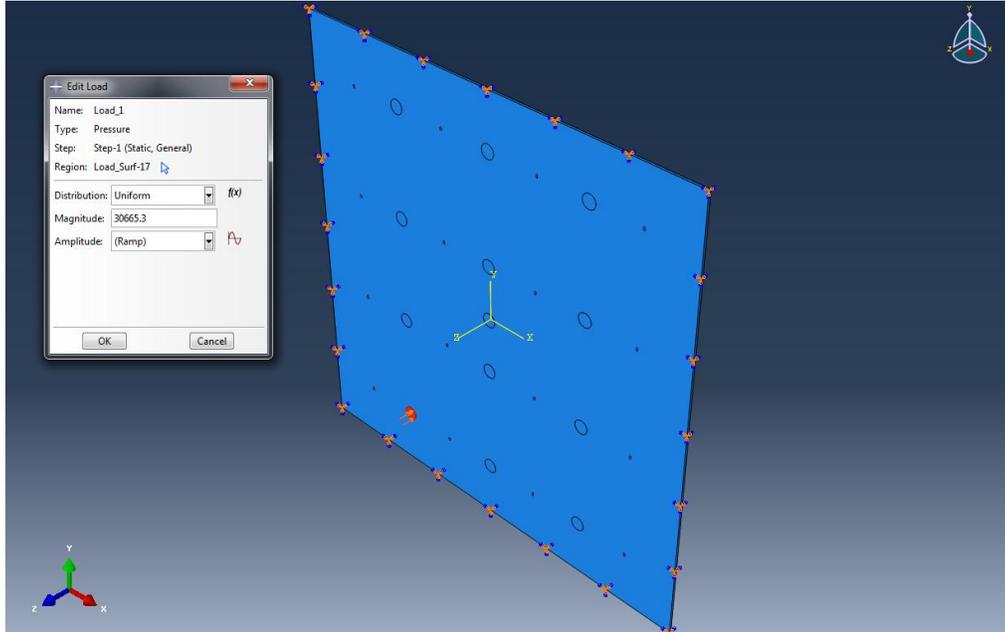


Figure 4-7: distributed Pressure load (Load\_Surf\_1)

The main iteration loop starts from part 7 where the loading location is defined and a pressure load is applied on that location. In part 8 another loop is defined inside the previous one which enables it to have several simulations for each loading scenario. In other words, in part 7 for example, L1 is considered for loading and in part 8, the value of the load is updated in each iteration and a new job is created and then submitted for analysis. It is important to note that by the end of each analysis, ABAQUS saves all the results for the field output requests in an output file (\*.odb) which can be accessed anytime later. On the other hand, when the job is completed by Python or by opening the desired \*.odb file, all different output results can be viewed in ABAQUS without the need for submitting the job again each time. Figure 4-8 indicates the schematic of one simulation where the load is only applied at one loading location L1.

In the last part of this script (script lines 115-132), each of the simulation \*.odb output files are accessed and the desired results are read and saved in specific order in a text file. In this

experiment elastic strain components for particular element of the panel part are saved in a text file.

FEA modelling is used to save time and cost compared to experimental analysis. However, still it is possible to save more using curve fitting for data generation. In order to clarify this, assume that a data set with 100 values between 1 and 100 is needed. One way is to randomly generate loads in each simulation and after 100 simulations, all the data is ready. An alternative approach can do fewer simulations, which covers all the load range, and then fit a curve for each loading condition. An infinite number of data can then be read from the curves to generate the training data set.

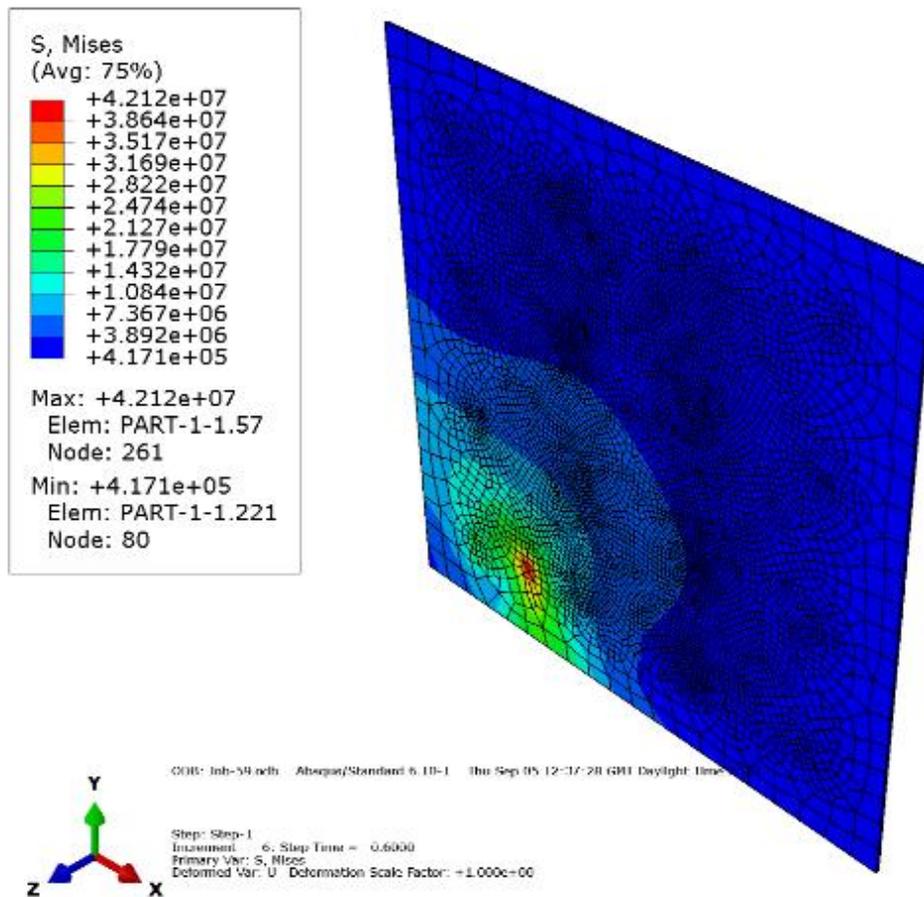


Figure 4-8: Schematic of a Large displacement experiment simulation result

### 4.2.2 Results

As it was mentioned earlier in this chapter, in order to investigate the general behaviour of composite panel with attached strain gauges under large displacement, a specific test rig and equipment setup is employed and modelled in FEA as well. This experiment is performed under static loading conditions. Loading is achieved by applying gravity force of various dead weights.

In this section, to ensure that the data acquisition system reading is acceptable, validity of assumptions is checked. For this purpose, a set of experiments were performed over several days to investigate repeatability of the system as well as the amount of drift it may experience in a normal room condition where it is located. The strain data was collected through bespoke data acquisition/ANN software linked to the NI cDAQ data acquisition system. This software was developed by the author in MATLAB utilising NI Direct Data Exchange (DDE) protocols to acquire the strain data and the MATLAB Artificial Neural Network Toolbox capabilities. It is found that the data is experiencing a small drift over time (usually in just some microstrains in hours). In order to eliminate the effect of drifts after some hours of running the device, it is decided to introduce extra data acquisition when the system is unloaded before each loading. This data set is then employed as a reference for zeroing the strain readings for the unloaded system. The results indicated that there is no visible effect of drifts in strain reading anymore. In reality this can be avoided by employing other gauges like FBG sensors or more accurate DAQ systems with auto zeroing possibilities. Although the system was assumed to have almost no effect of drifts anymore, there have been other difficulties such as very small noise in data. Most of the readings experienced a maximum sudden change of up to 2 microstrains. Although, this noise is very small and should not be an issue in reality, it can be reduced by introducing an averaging algorithm in the data acquisition function in the program. However, in practice the structural responses to the applied loads have much bigger variations (in hundreds) rather than just one or two micro strain noise and it can be handled with a well-trained generalised network.

Another thing to check is the repeatability of the data reading for the same loading condition. This is confirmed by having the same pattern of data for the same loading condition. For this purpose several data sets are acquired over a relatively long period indicating that the data acquisition gives acceptable repeatable readings over time. The validity of utilising FEA for

training data generation and the ANN validity and performance are detailed in the following sections.

#### 4.2.2.1 FEA Model Validation

For this study, the finite element models are developed and simulated and a script function written in Python language was used to iteratively run the software in a batch using different random loads applied at each of the thirteen loading locations on the panel.

The FEA model is initially validated to ensure that it represented the actual panel accurately. The validation was achieved by comparing strains collected experimentally with the FEA strains under the same loading conditions. Loads from 100 N to 800 N applied in 100 N increments were placed on the panel one at a time on locations L1 to L13. The strain readings at locations S1 to S16 on the panel were saved for each test. The same tests were performed with FEA to compare with the experimental results. Figure 4-9 and Figure 4-10 indicate comparison of FEA and experimental data for two example strain gauges (S6 and S10) when only location L13 is loaded. It can be illustrated that there is reasonable agreement between the strain readings of FEA tests and experimental tests. The average percentage error between FEA and Experimental data for the selected strain gauges are less than 7% (Figure 4-11 and Figure 4-12). These results indicate that the FEA model can be confidently used to simulate various loading conditions and to generate the required training input data.

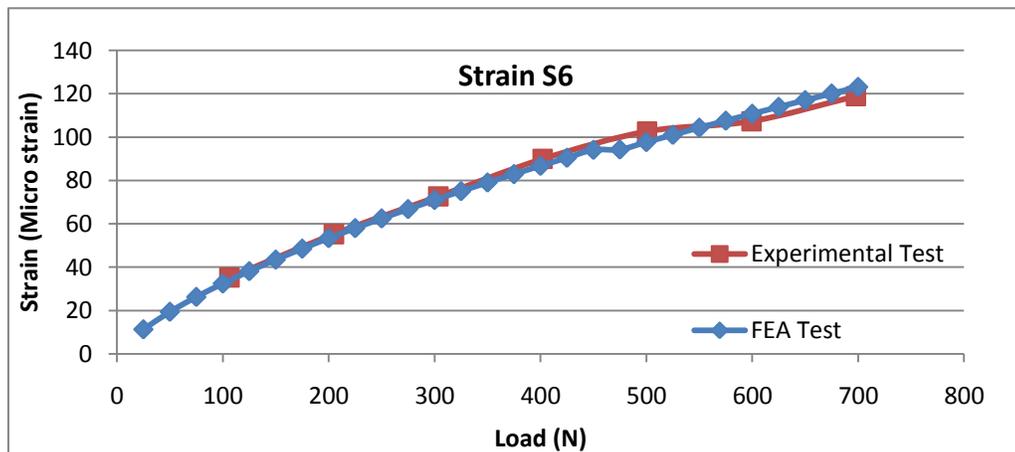


Figure 4-9: Comparison of FEA and Experimental data selected strain gauges (S6)

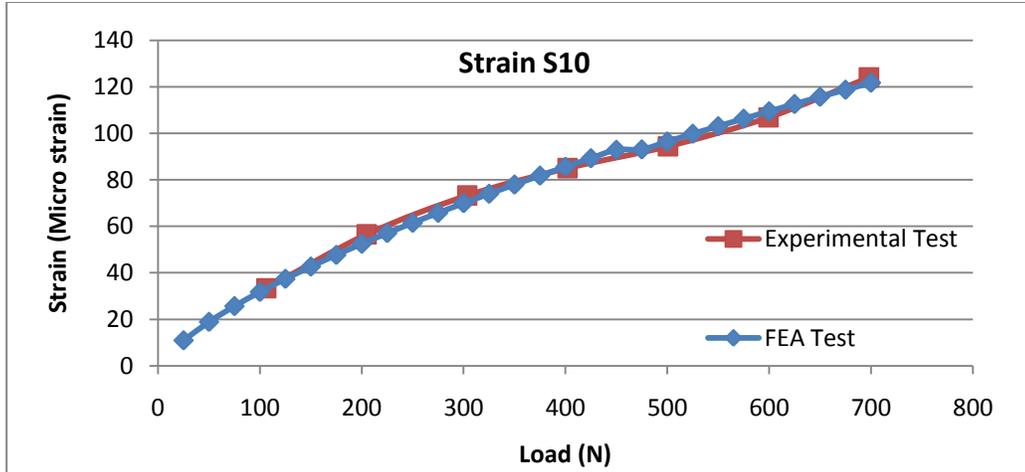


Figure 4-10: Comparison of FEA and Experimental data selected strain gauges (S10)

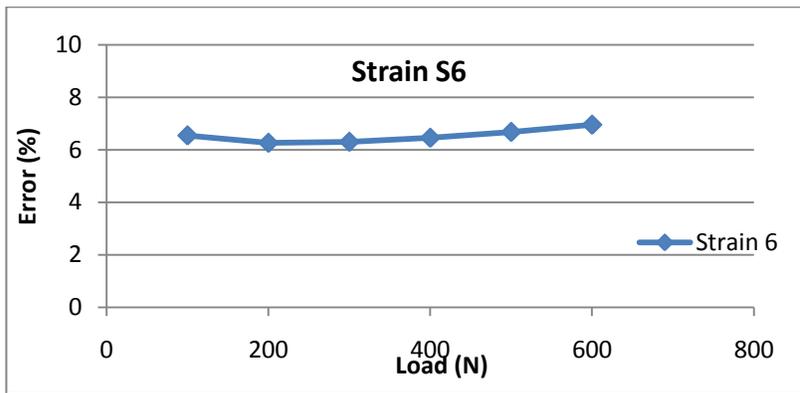


Figure 4-11: Error between FEA and Experimental data selected strain gauges (S6)

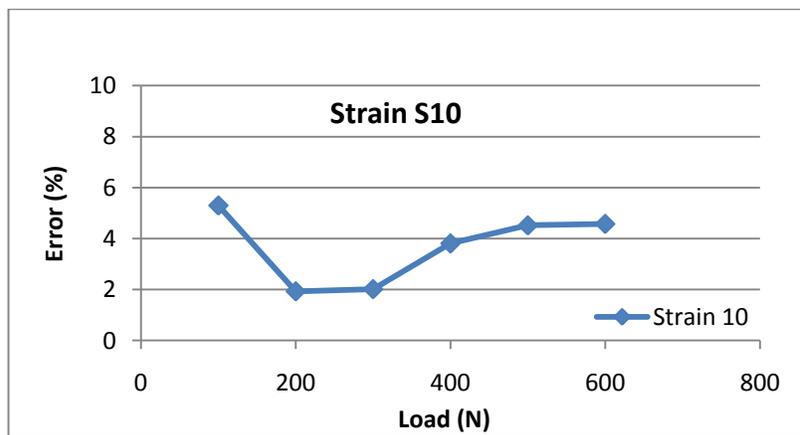


Figure 4-12: Error between FEA and Experimental data selected strain gauges (S10)

#### 4.2.2.2 ANN Validation and Performance

As mentioned in section 4.2.1.2, two different methods are employed to define the networks. Table 4-1 lists the major parameters of the network architecture used in the two methods. It was determined, through the testing of various network architectures that the optimum network (lowest SSE) for method one had two hidden layers with twenty neurons and used a tan-sig transfer function. The output layer had thirteen neurons (representing the thirteen loads to be estimated) and used a pure-lin transfer function. Similarly, it was determined that the thirteen networks for method two had one hidden layer each with fifty neurons and used a tan-sig transfer function. The output layer of each network had one neuron (each network estimates corresponding load of one location) and used a pure-lin transfer function.

**Table 4-1: ANN Architectures**

	1 Network with 16 Strain input and 13 Load outputs	13 Networks each 16 Strain input and 1 Load output
Number of networks	1	13
Architecture	Feed Forward	Back-propagation
Number of layers in each network	2	1
Range of load estimation	24.525 - 784.8 (N)	24.525 - 784.8 (N)
No. of inputs (surface strains)	16	16
No. of output layer neurons (loads)	13	1
No. of each hidden layer neurons	[20 20]	[50]
Number of training patterns	1040	1040
Number of testing patterns	1040	1040

In this study, SSE is used as performance indicator. Once the networks were trained, SSE values between the estimated loads and training load data were calculated. Each network has an individual SSE value. This means that although the first method has only one SSE value, the second method had thirteen SSE values. Figure 4-13 indicates SSE performance of all thirteen networks each having sixteen inputs (all strain readings) and one output (load at one location)

generated from the second network architecture.

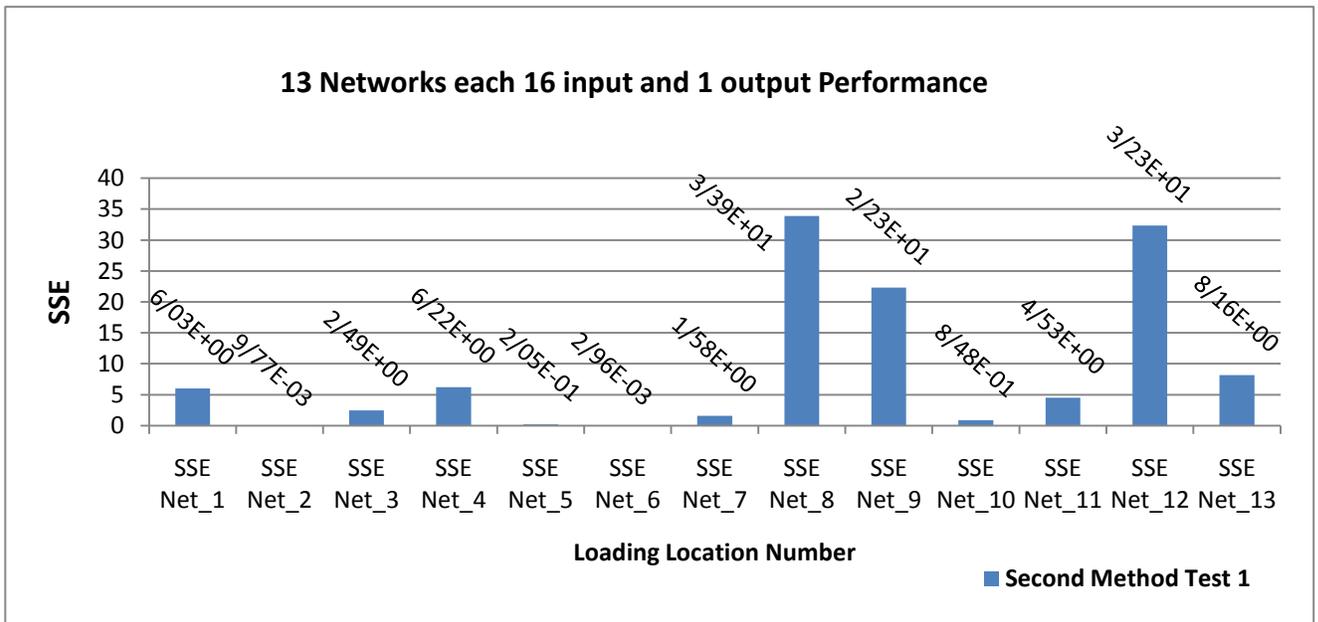


Figure 4-13: SSE Performance of network architecture 2

In order to compare the two methods, the summation of all the networks SSE values in the second method is compared to the SSE value of the first method when only a network with sixteen inputs and thirteen outputs were used to train the system. As it is indicated in Figure 4-14, a better performance for the second method is achieved.

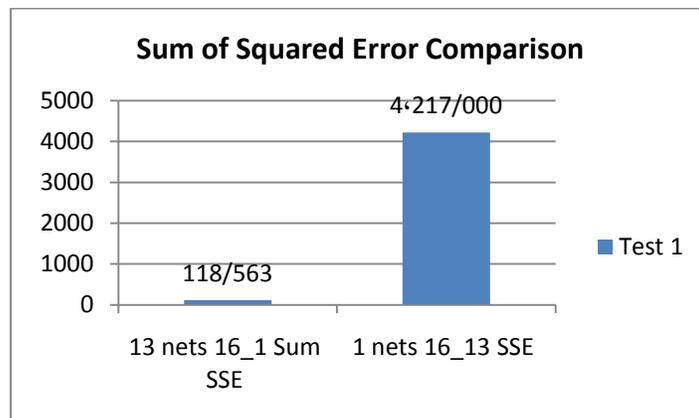
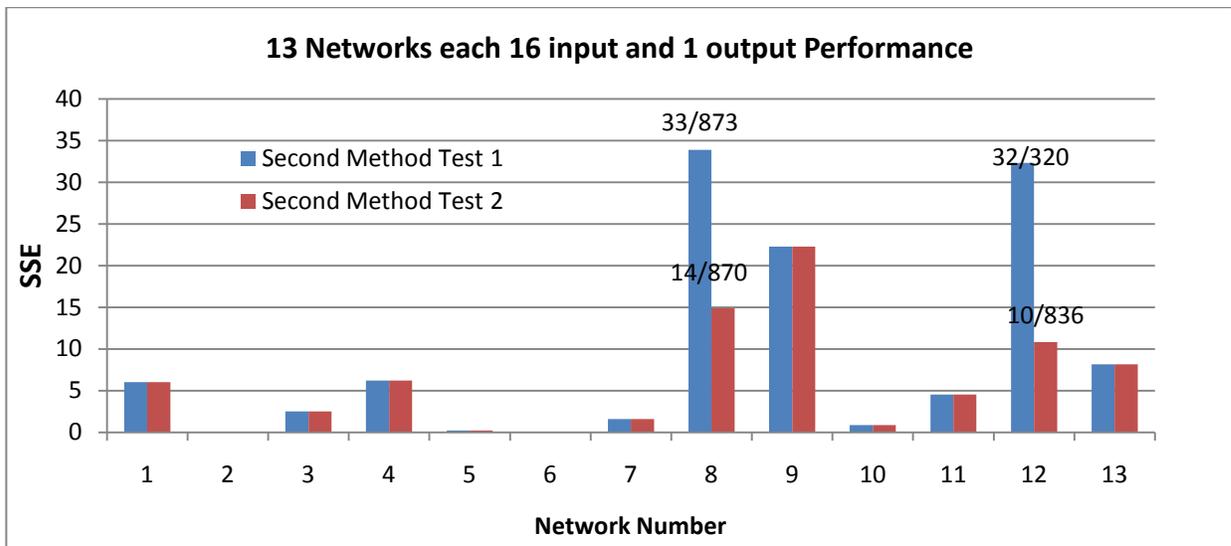


Figure 4-14: Comparison of the SSE values of the two network architectures

In addition to having a better performance, the second method has more flexibility. This means having thirteen independent networks, for each load location a separate new network architecture and parameters can be employed. For instance, the sum of the estimation performances of networks in the second method can be improved by changing the network architecture of those networks (eight and twelve from Figure 4-13) having relatively higher SSE values. As it is illustrated in Table 4-2, for locations eight and twelve, networks with two layers with twenty neurons are used. The improvement in SSE for networks eight and twelve with the new architectures can be seen in Figure 4-15.

**Table 4-2: Optimum ANN Architecture (method 2)**

13 Networks each 16 Strain input and 1 Load output	
Number of networks	13
Architecture	Feed Forward Back-propagation
Number of layers in each network	Most of it has 1 and for location 8 and 12 are 2
Range of loads	0 - 809.3 (N)
Number of inputs (surface strains)	16
Number neurons in output layer (normal loads)	1
Number of neurons in in each hidden layer	[50] or [20 20]
Number of training patterns	1040
Number of testing patterns	1040



**Figure 4-15: Flexibility of Second method in training stage**

In order to investigate the capability of the ANN to estimate loads in real time, once the ANN is trained, new strain data from different loading cases are introduced to it. Having a good performance, the ANN should be able to estimate the external pressure loads that caused those structure responses. For instance, introducing new sets of strain data that have not been used to train the network, the ANN estimates the corresponding load data. Depending on how well the network is trained (the performance of the network), there will be error between the expected output data set and the network estimated output (loads). Figure 4-16 depict a random example of estimated loads with the ANN for both FEA and experimental tests against the desired loads applied in tests when there is only one external load of 300 N at location L13 (data is used to train the network). Figure 4-17 and Figure 4-18 indicate other examples when there is only one external load of 200 N at location L1 and load of 200 N at location L7. For both sets of problem data it can be seen that the ANN can again estimate the load at the loaded locations with a high degree of accuracy. However, the error size of estimated loads with the ANN for experimental tests is slightly bigger. Such small error is normal and it could be from initial error between FEA data and experimental data, errors induced from the repeatability of the data acquisition system with resolution of +/- 0.1 microstrain as well as possible overtraining of the ANN.

The estimated negative load values at the unloaded locations were due to the differences between the strain data collected to generate the training data and the collected problem strain data. Due to these errors, slightly different strain patterns are introduced to the ANN producing the errors in the estimated loads. The introduction of further noisy patterns in the training data set may reduce these small errors, indicating that further work could be carried out to improve the accuracy further.

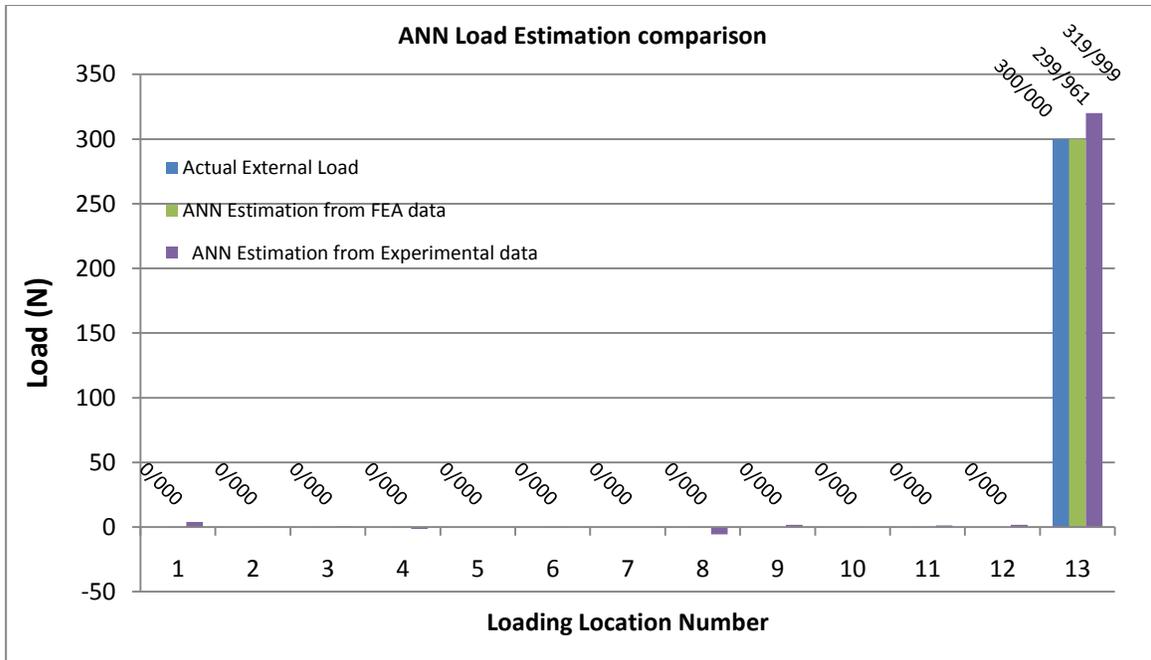


Figure 4-16: ANN estimation for FEA and experimental data Vs. Expected real data for L13

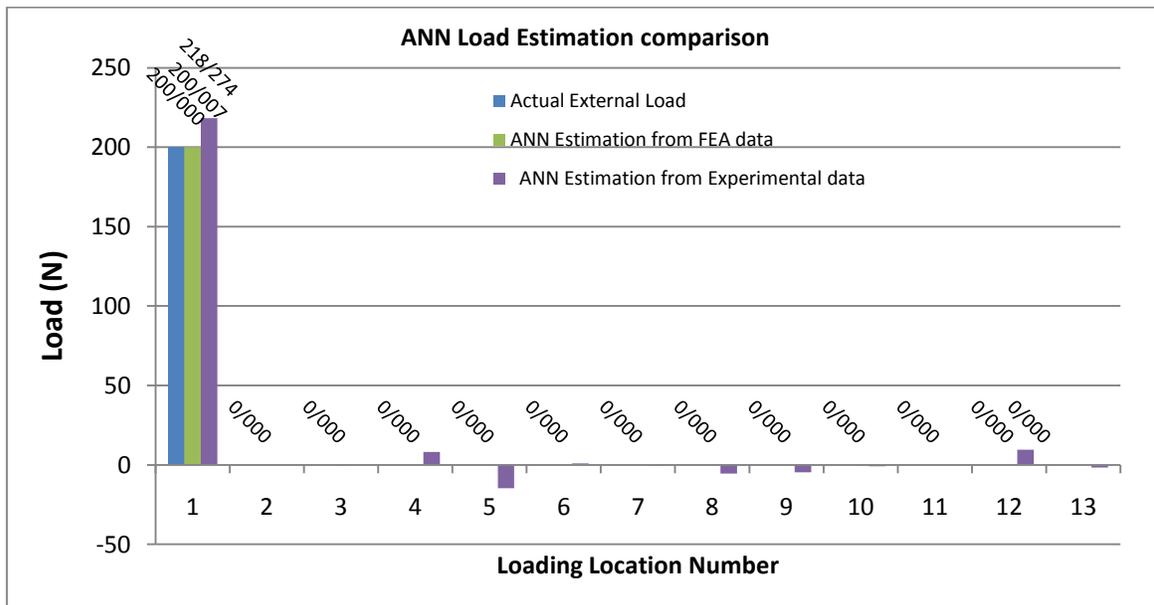


Figure 4-17: ANN estimation for FEA and experimental data Vs. Expected real data for L1

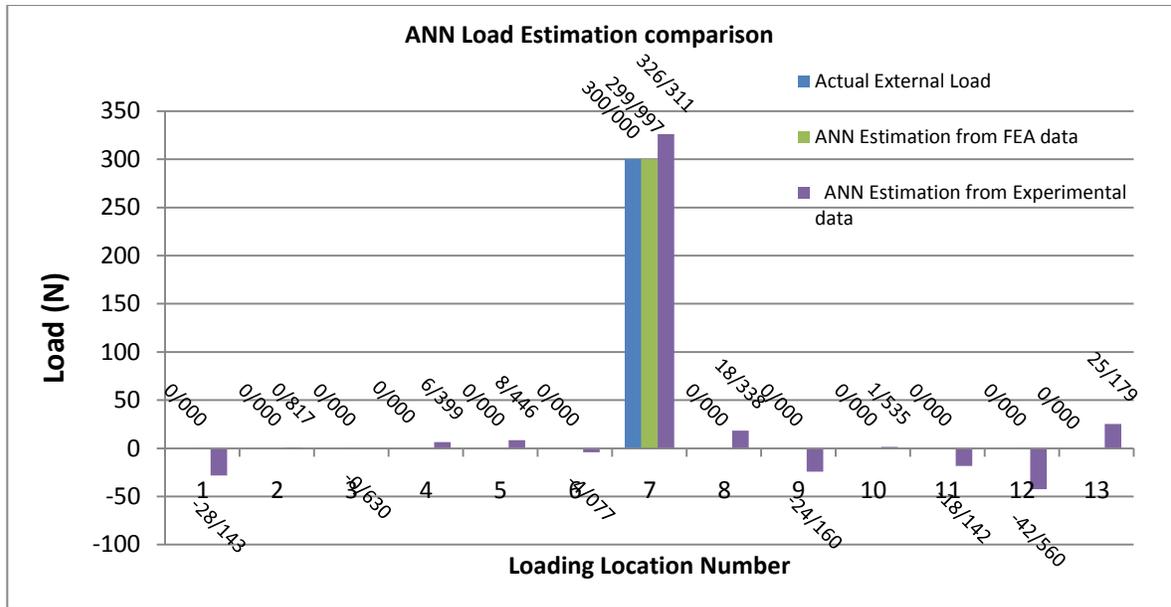


Figure 4-18: ANN estimation for FEA and experimental data Vs. Expected real data for L7

### 4.2.3 Conclusion

In this study, it is shown that the inverse problem method, utilising an ANN, is capable of estimating magnitude and position of the static pressure loads on a marine composite panel under large displacement from nonlinear strain measurements. The results of this study can be summarised as follows:

- FEA data can be used to generate training data for ANN inverse load estimation problems.
- Two different ANN architectures are used and the performances are compared.
- Having nonlinear relationships between the applied load and the surface strains, the system always converges and the SSE is in the range of acceptable error.
- The system is capable of estimating the position and magnitude of static pressure loads on a marine composite panel under large displacement.
- Having a large difference between the training data sets and the problem data set makes the ANN unable to estimate the load accurately.
- The main source of error was found to be in initial error between FEA data and experimental data.

In this study, a comparison of the ANN loads with the actual applied loads indicated a very good performance of the methodology. This was achieved in real-time, providing an accurate load history. This potentially makes the system ideal for solving many classes of complex engineering problem that require load monitoring.

It is proposed that the ANN methodology, with further research and development, could be utilised for the quantification of in-service, transient loads in real-time acting on the craft from the craft's structural response (strain response to load). This would provide valuable information to influence future craft design. In order to fully evaluate the proposed methodology for in-service load monitoring of marine structures the following areas require investigation:

- The behaviour of marine structures under transient load conditions (dynamic load is applied).
- The effect of size of the structure on the ANN estimation accuracy.
- Validation of the methodology on a craft in-service.

Finally, a GUI should be developed allowing control of various parameters of the data acquisition and load monitoring system, as well as graphical display in real-time.

### **4.3 Predicting Impact Loads Using Artificial Neural Networks**

So far, it is shown that the inverse problem approach can be used to estimate the static loads applied on a marine composite panel from the strain measurements when behaving both linearly and nonlinearly. A comparison of the ANN loads with the actual applied loads indicated a very good performance of the methodology. However, it was discussed that more investigation is necessary to further evaluate the suitability of the proposed methodology for in-service load monitoring of marine structures under transient load conditions such as slamming. In brief, transient or impact loading is important due to the fact that they can cause damage and delamination. Furthermore, the impulses from an impact are substantially larger than static loads which cause larger displacement and excitation of the hull structure and leads to shock transfer into the hull and passengers which can also affect the stability of the craft. For reasons like these, it is desirable to link the effect of impact on the hull to the knowledge of the dynamic or transient load intensities and their locations. That is why it is needed to develop a test that detects impact and the impact loads are substantially larger than static loads causing larger displacement. This section reports on the research undertaken to further develop the ANN methodology for quantifying pressure loads on a marine composite panel under transient load conditions such as a drop test from strain measurements. For this purpose, an experiment is designed and performed to further develop the ANN methodology for quantifying pressure loads on a marine composite panel under transient load conditions from strain measurements (see section 2.4.3 for equipment set up). In this study, the impact loads (the cause/output) on a composite panel are quantified by acquiring repeatable peak strain responses (the effect/input) to these loads from the panel.

#### **4.3.1 Methodology**

The methodology employed to evaluate the suitability and performance of utilising an ANN as an inverse problem solver for quantifying the transient load applied to the composite panel is presented in this section. The first stage of the investigation was to design an impact load quantification methodology for the panel utilising an ANN. In the second stage the load quantification methodology was validated by comparing loads estimated by the ANN with the known loading cases of the panel.

#### **4.3.1.1 Simulation Setup and Generation of ANN Training Data**

The structure under consideration is described earlier in section 2.4.3. Loading in this experiment is achieved by simulating a free fall impact of a rigid mild steel cylinder (length 0.103 m, diameter of 0.02 m and mass of 0.254 kg) normal to the panel surface at 13 locations (L1-13) from various heights. For this study, the finite element models are developed and simulated in ABAQUS 6.10-1 (SIMULIA). The panel has 7031 elements and the cylinder has 40 elements. Mesh type used is hexahedral isoparametric.

Generating the required training data sets may be through experimental tests or the use of simulation such as FEA. However, having a validated FEA model would dramatically save in time and costs compared to the experimental tests. In this study, an FEA model is developed and validated against experimental results. Employing a Python script in the FEA model allows automatic generation of various loading conditions by changing the velocity of cylinder just before impact. The structural response of the panel in terms of strain readings and velocity values for specific locations are evaluated.

#### **4.3.1.2 PYTHON Script of Drop Test Experiment**

This section describes the script written in PYTHON language for automatic modelling and simulation of various loading scenarios used in the drop test experiment. The panel model employed for this test is the same one that has already been validated with static loads. Figure 4-19 indicates a flow diagram of this code. For clarification purposes, this script is divided in several parts as well and can be accessed in Appendix 2. The script is developed in the way that iteratively runs the software in a batch using different random heights which is used to calculate the velocity of cylinder just before impact. The cylinder position can be changed in each of the iterations so that the impact can happen at each of the thirteen loading locations on the panel. In this script, loops are defined to iteratively simulate various impact loading scenarios. The flowchart of this iteration in summary is that the model is defined in first 4 parts and then a loop is used to iteratively define loads over various loading locations as well as creating a job. Each of the jobs are submitted for analysis individually and the results are post processed and the necessary structural responses from the strain gauge locations are collected and saved in a specific file with a particular order.

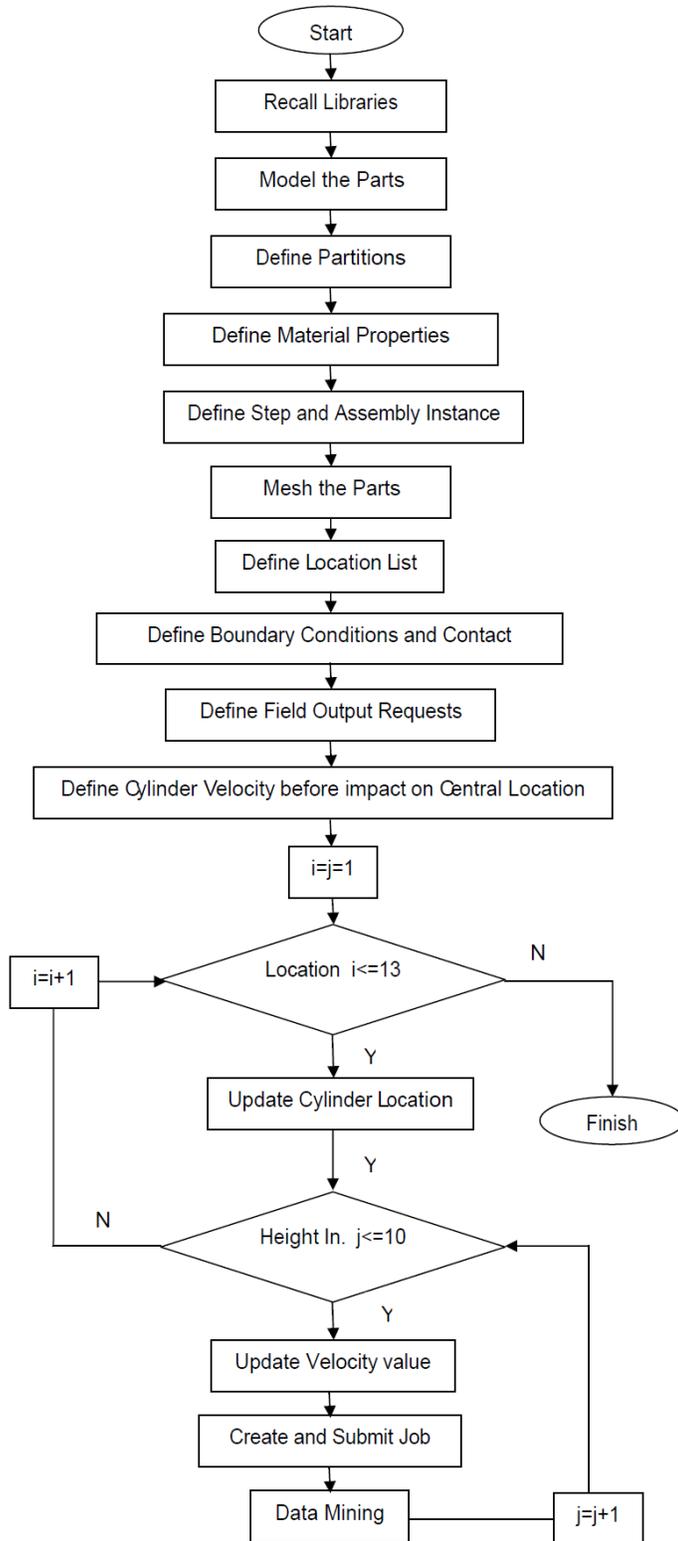
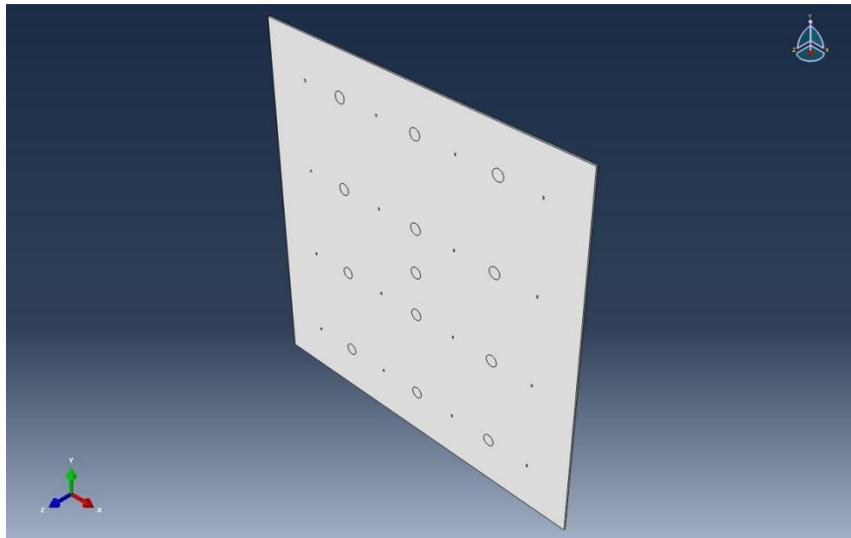
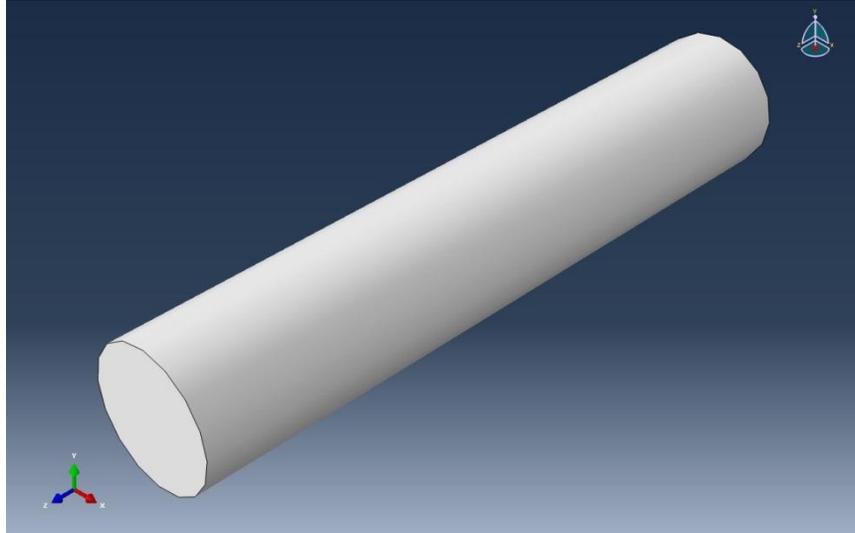


Figure 4-19: Flowchart of PYTHON Script of Drop Test Experiment

As in the previous script, the first part is essential to recall different libraries in the programme (script lines 1-14). In Part 2, modelling of the panel and the cylinder are defined. The script lines 15-57 define the test structure under consideration which was a 1 m<sup>2</sup> GFRP composite panel and a mild steel cylinder. Figure 4-20 and Figure 4-21 indicate the modelled panel and cylinder as a part in ABAQUS. In this part partitions are defined on the panel surface as well. After a part is partitioned, different properties can be assigned to or read from the resulting regions. For instance, here this helps to define different mesh values to different regions of the panel as well as having the same sized element for all the strain locations. As is indicated in the Figure 4-20, several rectangular and circular partitions are defined on the panel part.

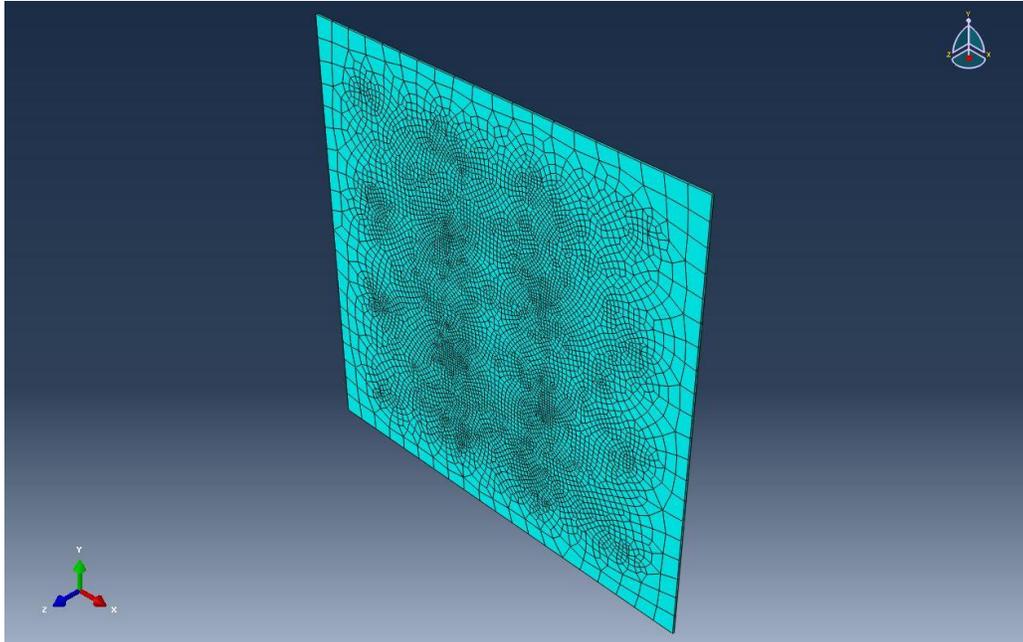


**Figure 4-20: Panel part in modelling of the drop test**

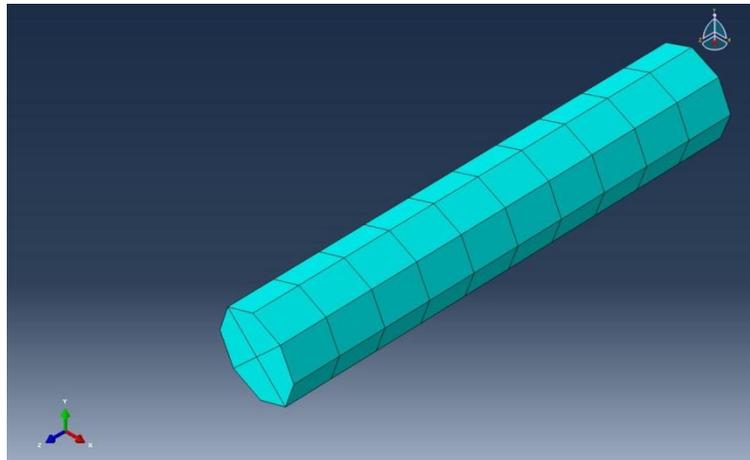


**Figure 4-21: Cylinder part in modelling of the drop test**

In part 3 (script lines 58-72), the material properties as well as the sections are defined, related together and the assembly instance is defined. In this part, the panel part is meshed with seeds having approximate global seed size of 0.05 and the mesh element type used is hexahedral isoparametric. After meshing the part, 7031 elements are generated on part. Furthermore, the cylinder part is meshed with seeds having approximate global seed size of 0.01 and the mesh element type used is hexahedral isoparametric. After meshing the part, 40 elements are generated on the part. It should be noted that these global seed sizes are selected based on simple mesh convergence studies. The meshed parts as well as all the resulting elements are illustrated in Figure 4-22 and Figure 4-23.



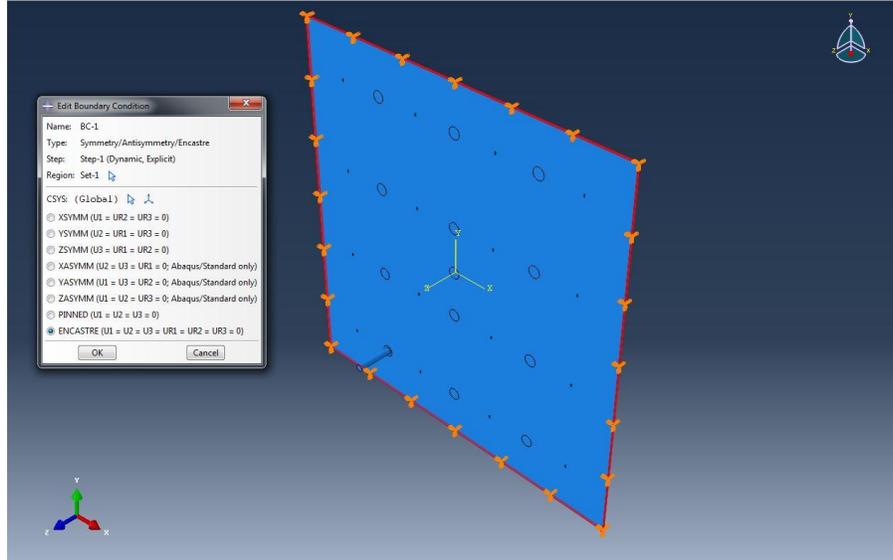
**Figure 4-22: Panel part is meshed and has 7031 elements**



**Figure 4-23: Cylinder part is meshed and has 40 elements**

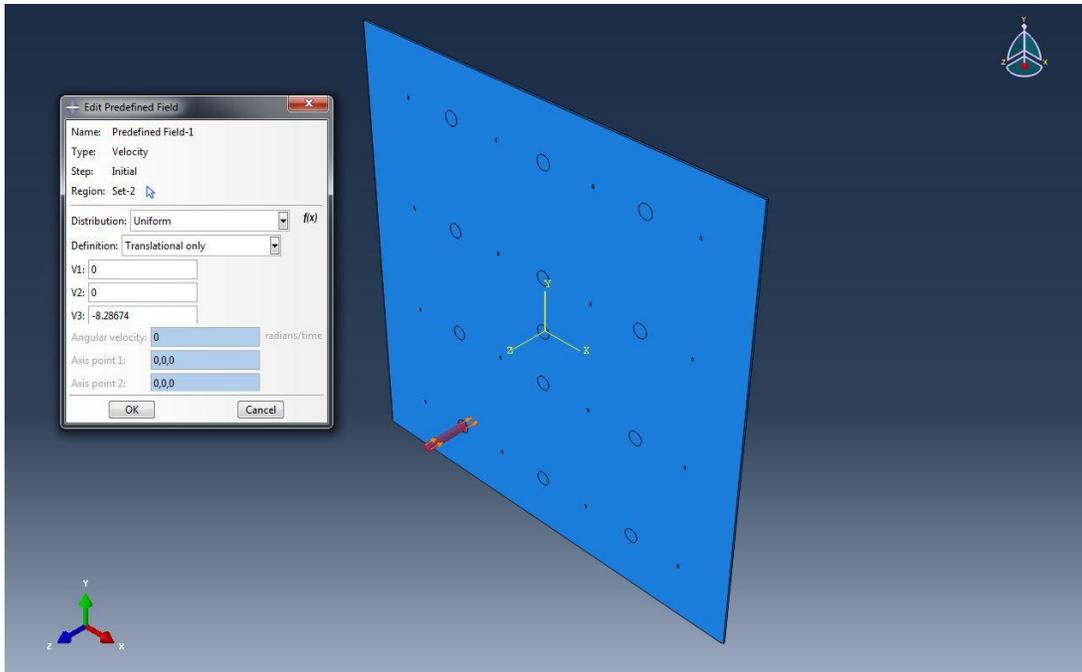
In part 4 (script lines 73-84), the dynamic step and field output requests are defined. The necessary field output of this model for this experiment is calculated in 100 increments in just 0.02 (dynamic step time) seconds after impact. Furthermore the panel boundary conditions are defined in this part. Due to the fact that the panel is fully fixed, ENCASTER ( $U1=U2=U3=UR1=UR2=UR3=0$ ) is selected as the boundary condition for all 4 edge surfaces

of the panel. The boundary condition of the panel is illustrated in Figure 4-24. In this part, the contact properties between the panel and the cylinder parts are defined. In addition an initial velocity is defined for the cylinder. This is the velocity of the cylinder just before impact. However the value of this velocity is updated later in the loops of the script based on the desired free fall height.



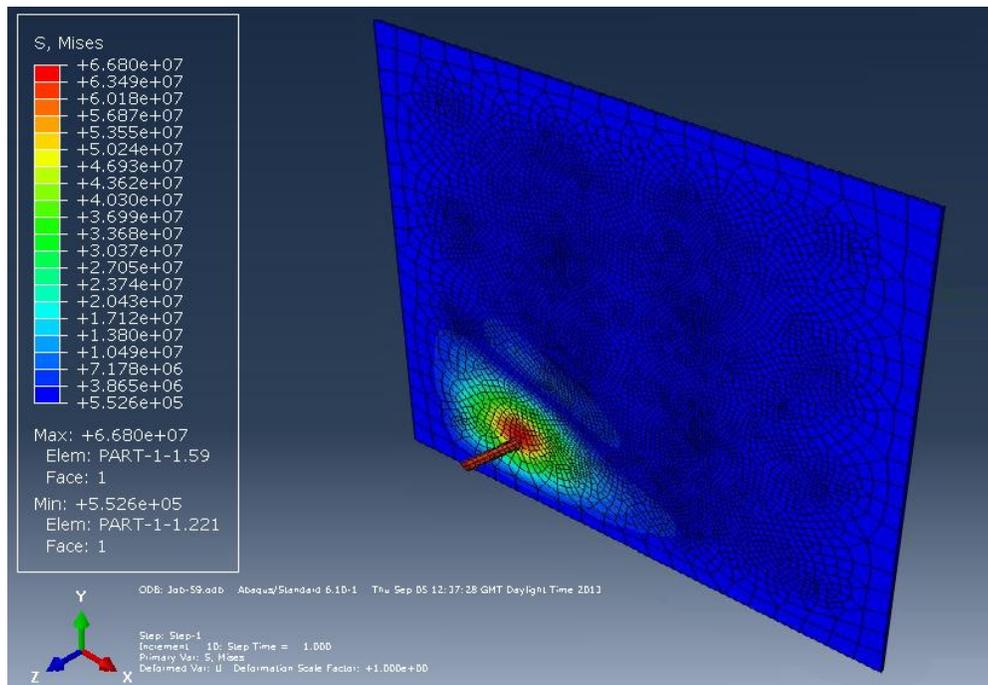
**Figure 4-24: Fully Fixed Boundary condition for the panel edges**

The main iteration loop starts from part 5 (script lines 85-97), where the loading location and the height of cylinder are defined and an impact happens at that location. The impact is defined in each iteration based on the value of the free fall height defined in script line 92-93. The energy from the falling cylinder is transferred to the panel during the impact. The modelling simulates the loading from the impact as well as the system response to the impact. Figure 4-25 indicates the defined velocity of the cylinder just before impact.



**Figure 4-25: Velocity just before impact is calculated for specific free fall height of the cylinder**

Once the value of the height is updated in each iteration and a new job is created and then will be submitted for analysis. **Figure 4-26** indicates the schematic of one simulation where the load is only applied at one loading location.



**Figure 4-26: Schematic of impact simulation results**

In the last part of this script (script lines 98-121), each of the simulation \*.odb output files are accessed and the desired results are read and saved in specific order in a text file. In this experiment normal strain, elastic strain and displacement components for particular elements of the panel part are saved in a text file. Furthermore, the velocity of the cylinder in each interval is saved as well.

#### ***4.3.1.3 ANN Architecture/Topology***

In this study a common Back-propagation, ANN architecture is used and trained employing MATLAB Artificial Neural Network toolbox capabilities. The network has an input layer with 16 neurons (as there are 16 strain readings), output layer with 13 neurons (as there are 13 loading positions) and some hidden layers each having any number of neurons. An iterative process was used to determine the optimum network architecture for the panel based on the performance of each network tested. In this study, three hidden layers each having 20 neurons was found to be the optimum.

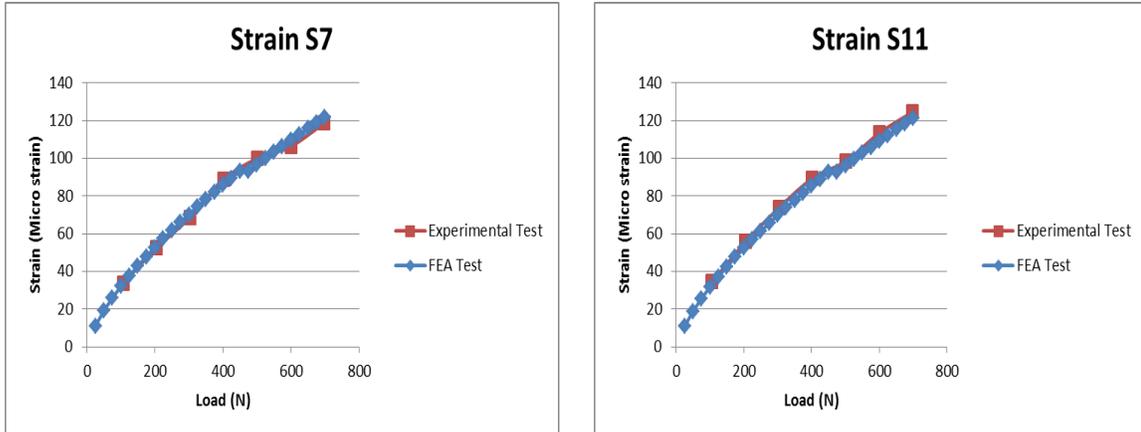
#### ***4.3.1.4 ANN Validation and Performance***

The validity and the performance of the ANN method were evaluated by comparing the load estimated by the ANN with known loads applied to the panel (problem data). Experimental problem data is the strain data from the same 16 nodes on the panel while it is being loaded. The Sum of Squared Errors (SSE) between a known target and ANN estimation is a common network performance indicator. For this validation study, new loading cases simulated by FEA and corresponding load and strain data is employed to evaluate ANN estimation performance when it is introduced with new data sets.

### **4.3.2 Results**

A script written in Python language is used to model the structure and simulate various loading scenarios up to 0.02 seconds after impact. Since this test is under high loads, large displacements analysis is used to simulate the model using a nonlinear solver. Furthermore, the structural response of the panel in terms of strain as well as the cylinder velocity over the simulation time is saved to be used to generate training data sets. In order to validate the FEA

model against the real structure, the panel is loaded from 100 to 800 N in 100 N increments at all 13 load locations (L1 – L13) separately. The strain readings at all 16 locations (S1 - S16) on the panel are saved for each test. The same tests are performed with FEA to compare the results with the experimental results. For instance, Figure 4-27 indicates that for loading only location L13, there is reasonable agreement between the strain curves (S7 and S11) of the FEA model and experimental tests.



**Figure 4-27: FEA Vs. experimental test results**

Once the model is validated, it can be simulated for various loading conditions to generate the required data. Having strain readings from the FEA model at selected nodes S1-16, the corresponding peak strain value of the first impulse is used as input training data. Figure 4-28 shows example FEA strain data collected from the gauge S1 when the impact location was L1.

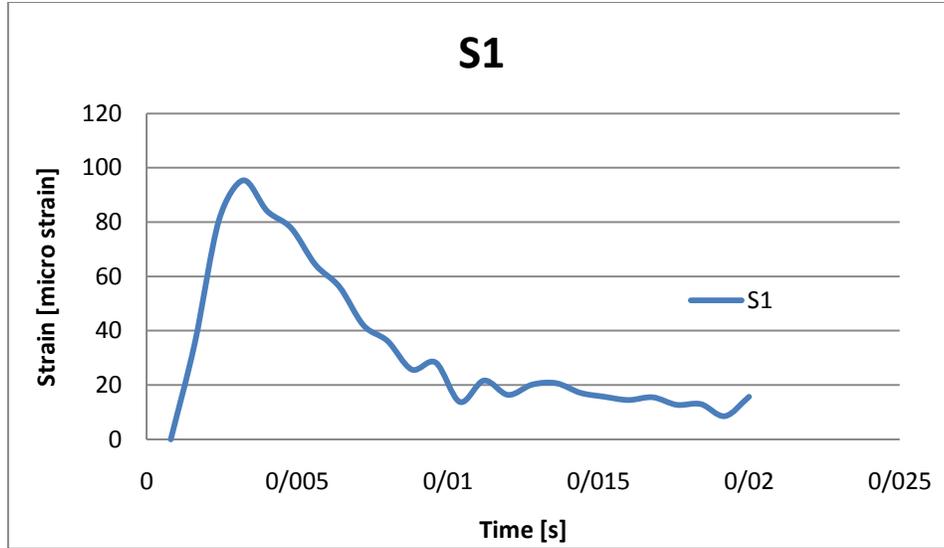


Figure 4-28: Typical FEA strain data during the impact for gauge S1

In order to calculate the impact force of the first impulse, Equation 17 is used where  $V_1$  is the velocity of cylinder just before impact and  $V_2$  is its velocity after impact.  $\Delta t$  is the duration of peak impulse. Impact forces of each loading cases are saved and utilised as training data set targets.

$$F = \frac{m (V_2 - V_1)}{\Delta t} \quad (\text{Eq. 17})$$

In this study 16 strain readings (inputs) and 13 load readings from 13 locations (outputs) are needed to have one set of training data. Enough training sets of 13 various loads at each location on the panel and the resultant 16 strains caused by these loads were required to find the relationship between the input/output data. For each loading location (L1 - L13), 75 training data sets are generated by loadings from 600 N to 6071N making a total of 975 training data sets from FEA. Loadings are changed based on the velocity of the cylinder just before impact. Introducing the training data to the trained network, the ANN output should be as similar as possible to the impact load set that the ANN has been trained with. Figure 4-29 and Figure 4-30 indicate some random examples of estimated impact load data by the ANN compared with the expected values for impact locations L3 and L9. It can be noticed that there is reasonable agreement between the scatters.

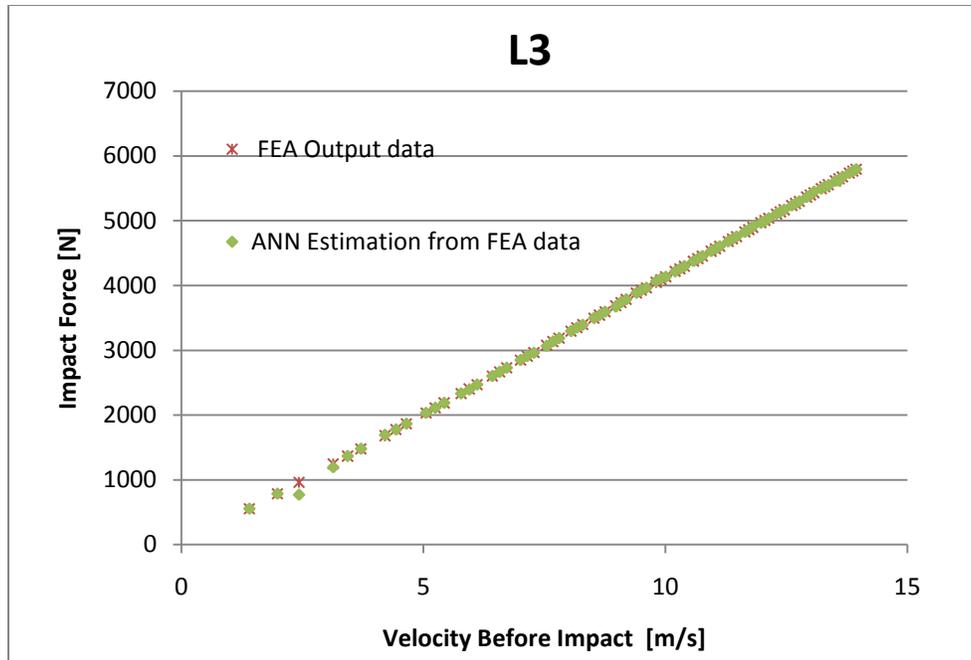


Figure 4-29: Estimated impact load data by ANN Vs. expected values from training data (L3)

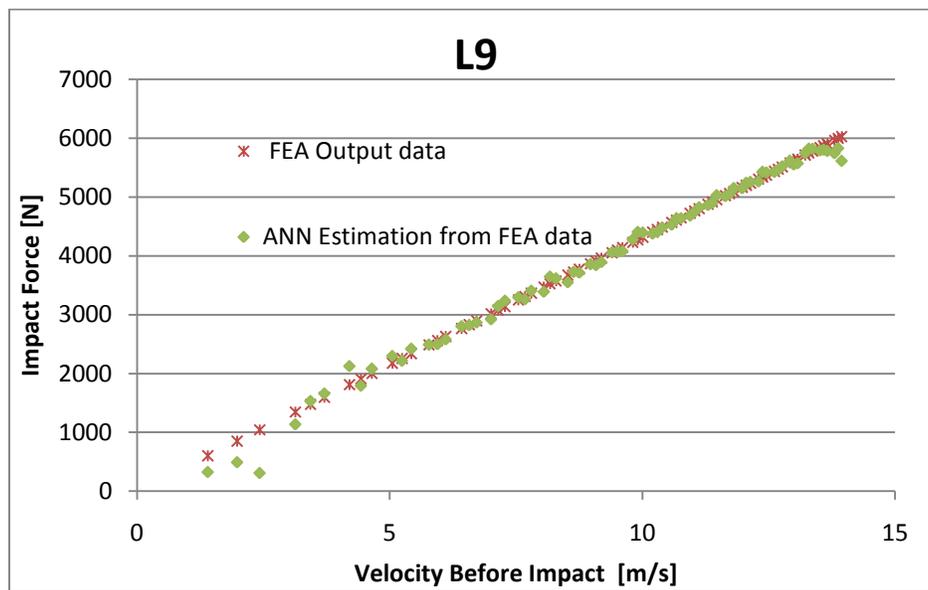


Figure 4-30: Estimated impact load data by ANN Vs. expected values from training data (L9)

#### 4.3.2.1 Validation using FEA Data

For each loading location (L1- L13), 25 training data sets are generated by loading from 600 N to 6071N making a total of 325 training data sets from FEA. This set of problem data was not introduced to the network during its training procedure. The strain set of this problem data set is

introduced to the previously trained ANN and the corresponding estimated load values are calculated. Figure 4-31 and Figure 4-32 show some random examples of estimated impact load data by ANN and is compared with the expected values.

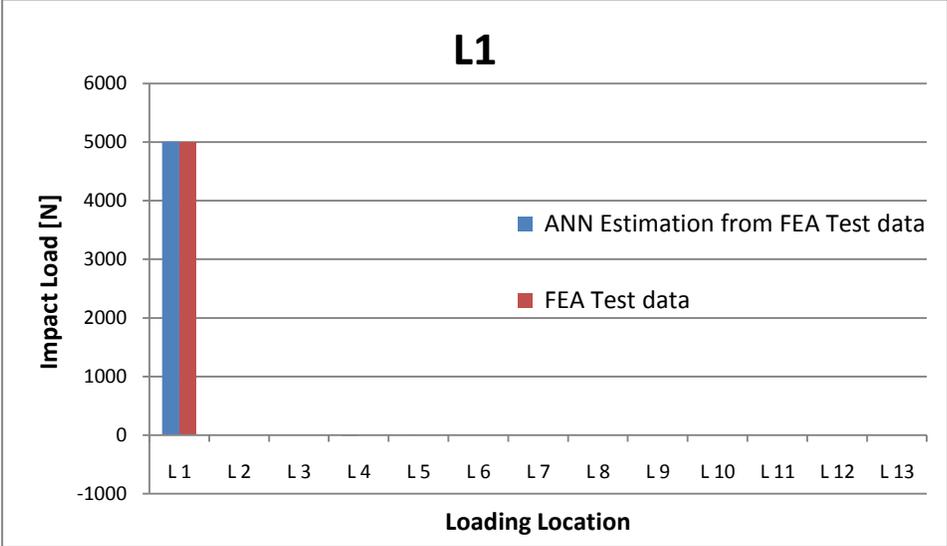


Figure 4-31: Estimated impact load data by ANN Vs. expected values from test data (L1)

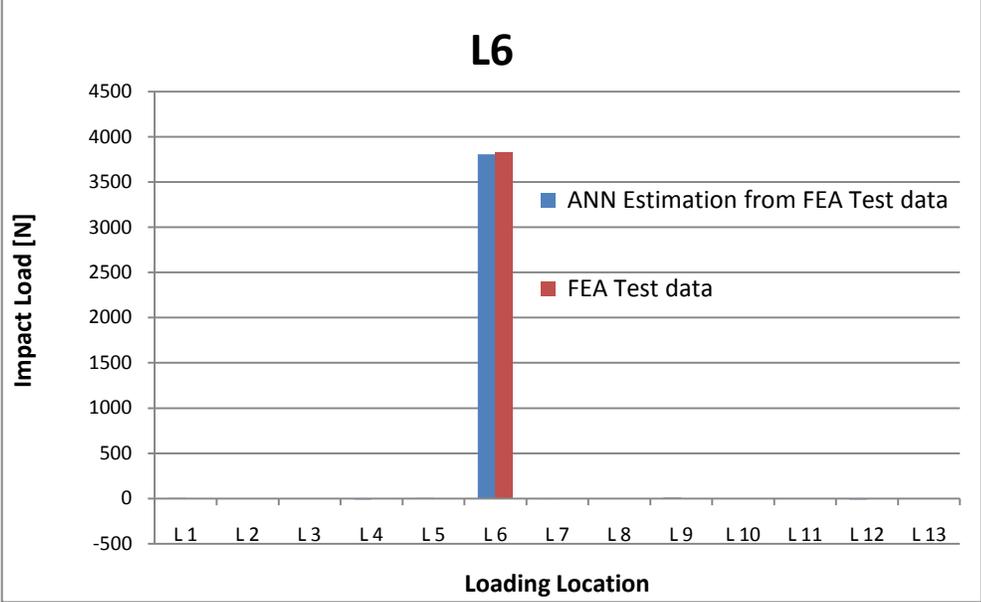


Figure 4-32: Estimated impact load data by ANN Vs. expected values from test data (L6)

### **4.3.3 Conclusion**

The results presented in this section show that the inverse problem method, utilising an ANN, can accurately estimate the position and magnitude of 13 impact loads applied to the composite panel from the captured strain data of 16 nodes spread over the panel surface. The results indicate that the system always converges and the ANN can be trained using FEA data to solve inverse problems and accurately estimate the impact loads. Once the ANN is sufficiently trained it can be utilised to estimate the output in real-time where new inputs (problem data) are presented and processed by the ANN and impact loads are estimated.

### **4.4 Summary**

The findings from both experiments of this chapter indicate that should the structural response have a nonlinear behaviour or transient nature, the ANN can be trained employing FEA data. In the first experiment of this chapter, it is shown that the inverse problem method, utilising an ANN, is capable of estimating magnitude and position of the static pressure loads on a marine composite panel under large displacement from nonlinear strain measurements. The comparison of the ANN loads with the actual applied loads indicated a very good performance of the methodology. Furthermore, the results from the drop test show that the system always converges and the ANN can be trained using FEA data to solve inverse problems and accurately estimate the impact loads.

## **Chapter 5. DISCUSSION, EVALUATION, FINAL CONCLUSION AND FUTURE RESEARCH**

This chapter is a summary and evaluation of the topics and results outlined in this thesis. Furthermore, future research is described

## 5.1 Summary and Discussion

The Author hypothesised that ANN can be employed to find an inverse solution to predict hydrodynamic loads applied on marine structures which may be used to inform designers in the stage of preliminary ship design. In this research, the aim was to employ and investigate the ability of this approach to immediately predict/estimate the load by analysing any similar new response data introduced to the solver. The suitability of the proposed methodology could be indicated by successful prediction of the applied loads.

In order to investigate the applicability of the new system in online load monitoring of marine structures such as Rigid Inflatable Boats (RIB), some critical objectives were recognised that have not been investigated before and are novel. To answer the research question and investigate the research objectives, the performance of the load monitoring system (in terms of MSE or SSE) is evaluated by performing some experimental tests by means of applying this monitoring system on a marine structure representative such as composite panel. The developed system used the surface strains from the structure and computed the weight/forces by ANN trained using FEA/ experimental data. Several tests were performed and experimental tests were carried out to validate the results.

The first objective of this research was to investigate the general behaviour of a marine structure representative with attached strain gauges under small displacement. In order to achieve this, the small displacement experiment is designed and performed in which a composite panel representative of a boat's hull structure is instrumented and calibrated to function as a complex load cell capable of measuring external normal loads. The panel is divided into 16 patches (one strain gauge rosettes placed in middle of each patch) and 12 loading positions are picked to be loaded utilising weights. Structure response is collected with a data acquisition system directly to the written programme in MATLAB. Generated superimposed data from these readings are used to train an Artificial Neural Networks. A satisfactory trained ANN can be used as an alternative analytical tool on other new collected data. New structural response as inputs can be presented to the network to predict load data as outputs. Preliminary results from static loading of a composite panel indicated very promising results with a very low error margin. The results were encouraging and lead to the Hybrid Inverse Problem Engine.

However, for an ideal system, the number of sensors should be minimised to reduce the time to train the system, cost and weight. To achieve another objective of the research, an optimisation experiment is designed and carried out in which the small displacement test is optimised in terms of the required sensors for accurate load estimation. The optimisation results indicated an area of  $1\text{ m}^2$  with 12 loading positions can be efficiently monitored with only 4 rosettes attached close together in the middle of the panel.

Since more research was needed to investigate the effect of higher loads and non-linearity, other experiments were performed having a larger load range. Higher loads lead to large displacement in panels and non-linearity cannot be ignored. Another objective of this research was to investigate the general behaviour of a marine structure representative with attached strain gauges under large displacement. In order to achieve this objective, the large displacement experiment is designed and performed. In large displacement cases, since the system responses to applied loads were not linear anymore, superposition could not be employed to generate training data. To save in costs and time, a FEA model was developed in ABAQUS and validated using experimental data. The validated model was then employed to simulate various nonlinear loading conditions. In this study, two different ANN structures capable of relating the load data set to the corresponding strain data set were described and the performances were compared. The results from this experiment indicate that the inverse problem method, utilising an ANN, is capable of estimating magnitude and position of the static pressure loads on a marine composite panel under large displacement from nonlinear strain measurements. Therefore, this novel load monitoring system can also be utilised for load monitoring of marine structures effectively even when the relation between the applied loads and the structural responses are nonlinear.

In addition to static loading conditions, another objective of the research was to perform more investigations on the suitability of the proposed methodology for in-service load monitoring of marine structures under transient load conditions such as slamming. This objective is achieved by satisfactory results of the drop test experiment designed and performed to further develop the ANN methodology for quantifying pressure loads on a marine composite panel under transient load conditions from strain measurements. In this study, the impact loads (the cause/output) on a composite panel were quantified by acquiring repeatable peak strain responses (the effect/input)

to these loads from the panel. The training data were generated employing a validated FEA model (developed in ABAQUS) and used to train the networks. The results indicate that the system can be trained to relate applied loads and structural responses and accurately estimate the impact loads.

The last objective of this research was to develop a GUI which can be used easily to develop a load monitoring system. This objective is also achieved by developing a GUI in MATLAB allowing control of various parameters of the data acquisition and load monitoring system, as well as graphical display in real-time.

In summary, according to the project time and budget constraints the research objectives were defined and investigated. The results of the experiments indicated that these objectives are achieved satisfactorily and the research question is answered. It is concluded that the ANN methodology could be utilised for the quantification of in-service, transient loads in real-time acting on the marine structure from its structural response (strain response to load). This would provide valuable information to influence its future design. Furthermore, the inherent advantages of the load monitoring system using ANN as an inverse method over other methods make the load estimation possible in real time without the need for information about the material and the geometry of the marine structure. The ability to measure the actual load history of a marine structure in-service would enable the designer to validate the load estimation and structural design tools used during the design stage. This would lead to the development of more optimal structure designs for this type of marine structure. The operational safety of the craft can also be improved by having a real-time load monitoring system that is able to detect any degradation of the structural integrity and defects within the structure. Providing an accurate load history potentially makes the system ideal for solving many classes of complex engineering problem that require load monitoring.

## **5.2 Evaluation of the proposed In-Service Load Monitoring System**

It is very important to evaluate the outcomes of this research, which is the application of hybrid methods for in-service monitoring of marine structures. The main aim of this study was an investigation into computing the loads applied on a composite panel as a marine structure representative using artificial intelligence. Since during the term of this research ANN, FEA,

experimental techniques were employed frequently, the evaluation of the proposed in-service load monitoring system was broken into the following sections: Artificial Neural Network, Finite Element Analysis, and experimental techniques.

### **5.2.1 Experimental Set-up**

During the course of this research, a series of novel experiments were designed and various quantitative tests were performed. In order to investigate the behaviour of marine structures and the application of the methodology, a 1 m<sup>2</sup> Glass Reinforced Fibre Polymer (GRFP) composite panel as a representative of panels in a boat is employed. The panel is chosen as a representative of a marine structure due to the fact that marine crafts are manufactured from a number of flat panels attached to the hull frames.

The experiments of the project employed two different data acquisition systems. Results gained from Microlink data acquisition system used for the small displacement test have some noises which have also affected the accuracy of the results. Although care was taken to reduce the error, still this system was not as accurate as the new standard NI cDAQ data acquisition system used for other tests. Further to this it was not possible to use the Microlink data acquisition system to perform dynamic tests. This was due to the fact that it has very slow data acquisition speed and high level of noise in data.

The attachment process of the strain gauges is very time consuming and demands related experience. Due to the fact the strain gauges are very small and delicate objects; extra caution is needed to avoid possible misplacement and breakage. However, many types of strain gauges are commercially available and can be employed to save time. Loading the panel was achieved manually in this research which was very time consuming and hazardous. Possible alternatives could be design and manufacturing of a specific device to automatically apply loads.

The final load monitoring system from this approach requires wide general knowledge of engineering from different fields including preparing the test set up, operating with the FE software, connecting strain gauges, training an ANN, working with different technical devices such as data acquisition, etc. It is not possible to leave all the technical work to one operator or engineer. Despite valid results gained by the author from different tests, the system is not ready

to be used commercially yet. The system should be designed in such a manner that it can be used by an operator with a short training course.

### **5.2.2 Artificial Neural Network**

In this study due to the lack of a clearly stated mathematical solution or algorithm the ANN was considered as the most suitable technique. ANN can provide suitable solutions for problems with a high degree of non-linearity and high dimensionality. ANN also provides accurate solutions for noisy, complex, imprecise, imperfect and/or error prone sensor data. ANN can provide accurate results for some problems that cannot be analysed using standard techniques. When an ANN is trained the output of the system can be computed in a fraction of second which make it suitable for dynamic and real time analysis.

The ANN was widely used during the course of this project. Despite the encouraging results gained by ANN, it has its own limitations. These limitations can be summarised as follows:

- Training an ANN requires producing sufficient amounts of training and testing patterns, which is a time consuming and expensive procedure.
- In general, when employing ANNs, no unique solution or general design theory exists. Generally it is not guaranteed that an ANN will converge to its global minimum or occasionally even will converge at all. Training an ANN requires consideration about the local and global minimas as well as poor training.
- There is no standard regulation to select the most efficient network's parameters such as learning weight, momentum parameter, transfer function or even the number of training and testing patterns.
- It is difficult to determine which ANN gives the best solution when considering several well-trained ANNs for a batch of training and testing patterns.
- Obtaining a low performance value for training and testing patterns requires a high effort. It is not always possible to consider a low pre-selected performance value and reach that just by continuing the training procedure.
- May be too slow in the case of large-scale problems when common serial processing digital computers are employed.

- The number of inputs to the ANN needs to be equal or greater than the number of outputs.

Since there is no general training rule for an ANN and the training process is a relative process according to the training data set of a particular problem, it is very probable that the network results may not be satisfactory on the first design pass. In this situation, it is necessary to redo one or more of the process steps repeatedly until better results are achieved. When the training process results in the network which is not accurate enough, several approaches are highly recommended. One way is to initialise the network again and perform the training again. Initialising a network leads to new weights and biases. When the network parameters are changed, it might result in different solutions. Another solution is to increase the number of hidden neurons, usually 20 or more. The reason is that the network will have more flexibility having a larger numbers of neurons in the hidden layer. It is recommended to increase the layer size gradually. Training a network with too large a hidden layer is more time consuming and may cause the network to be under-characterised as well. This means that the network has more parameters to optimise than there are data vectors to constrain them. The third popular approach is to try a different training function. Furthermore, different training data can be tried to train the network. Additional data sets can be used in the training data set or in contrast redundant or challenging data may be removed from the training data set. Presenting a network with additional data is expected to produce a more general network that can handle new data well.

### **5.2.3 Finite Element Analysis and Experimental Techniques**

Finite element analysis (FEA) is widely used in this research for predicting the responses of the model to environmental factors such as velocity, force and strain values. The process starts with the creation of a geometric model, which is then divided into smaller shapes connected at specific nodal points. Finally, the material behaviour and boundary conditions are applied to each element, and the analysis is performed.

Despite the advantage of FEA for fast stress/strain/deflection computing it has its own limitations. Usually for simplistic models, FE results are close to those obtained from the experiments. When the subject is complicated not only modelling but also the validation of FEA results is a very difficult task. In this research modelling the composite panel using FEA had

some limitations which make the validation difficult. This is due to the fact that modelling complex geometries and new materials such composites is more difficult as well as time and labour intensive. Sometimes such modelling may have results which show poor repeatability behaviour of the system and a poor correlation between the FEA and experiential strains. Considering these characteristics of the system it is very difficult or sometimes impossible to use the standard techniques, because similar loads do not produce similar strains in FEA models with even slightly different parameters (such as mesh type).

Due to the manufacturing and budget restriction, it was not possible to manufacture a uniform thickness panel. From the other point of view, it was difficult to compare the FEA and experimental results due the varying thickness of the actual panel. As the results of using an even thickness for the panel, some error (less than 7 percent) between the FEA and experimental results were obtained. However, the panel modelling was a simple task compared to the modelling of the whole marine structure like a boat or ship.

### **5.3 Final Conclusion**

This research has explored relevant aspects of the application of hybrid combinations of ANN, FEA and experimental techniques both in direct and inverse studies. The hybrid technique was found to produce solutions to problems that existing measurement/computing techniques could not solve. This technique can provide more information faster than standard existing techniques without the need for knowledge of material/geometrical properties. The findings presented in this study should stimulate future research in the area of in-service load monitoring of marine structures (internal structures, hull, panels, etc.) and provides a powerful tool to determine the position of the high loads or impact areas and information regarding the rectification required for a more efficient design of the marine structure to produce lighter, faster and more durable marine structures.

### **5.4 Future Research**

The final aim of this research could be developing a load monitoring system capable of in service load monitoring of various marine structures. The future of this research should be to

develop a system which is commercially available and easy to use for a variety of marine structures. To achieve this goal several considerations should be made:

- 1) Developing a general and standard routine to define the loading circumstances of the marine structures in real life time.
- 2) Investigate in possibility of employing re-attachable strain gauges. This reduces the technical work, time and cost radically.
- 3) Developing a system which analyses the marine structure using identified load sensitive areas to estimate the best position of strain gauges.
- 4) Designing an automatic and repeatable load applicator. Using this device, it is possible to apply different loads to different positions of the marine structure considering load sensitive and tolerant areas.
- 5) Validation of methodology using a real marine structure such as a boat in sea condition.
- 6) Optimisation of the quantity of sensors quantity and investigate the effects of network training parameters for a specific marine structure.
- 7) Full investigation of the effect of geometry size, DAQ system and quantity and type of gauges in load estimations.
- 8) Further investigations in how to turn this load monitoring system into a commercial system.

## Reference

- Abs. 2011. *Guide for building and classing high speed craft*. American Bureau of Shipping.
- Adamchak, J. C., 1984. *An approximate method for estimating the collapse of a ship's hull in preliminary design*. Paper presented at the Ship Structural Symposium, Arlington, Virginia.
- Al-Assaf, Y., and El Kadi, H. 2001. Fatigue life prediction of unidirectional glass fiber/epoxy composite laminate using neural networks (Vol. 53, pp. 65–71): *Composite Structures*.
- Aleksander, I., and Morton, H., 1990. *An introduction to neural computing*. Chapman and Hall.
- Allen, R., Jones, R., and Taylor, D., 1978. *A simplified method for determining structural design limit pressures on high performance marine vehicles*. Paper presented at the Proceedings of AIAA/SNAME Advanced Marine Vehicle Conference, San Diego.
- Amali, R., Noroozi, S., and Vinney, J., 2000. *The application of combined artificial neural network and finite element method in domain problems*. Paper presented at the Sixth International Conference on Engineering Applications of Neural Networks (EANN 2000), Kingston upon Thames.
- Amali, R., Noroozi, S., Vinney, J., Sewell, P., and Andrews, S., 2006. Predicting interfacial loads between the prosthetic socket and the residual limb for below-knee amputees—a case study.
- Anderson, J. A., 1972. A simple neural network generating on interactive memory. *Mathematical Biosciences*, 14, 197-220
- Anderson, J. A., and Rosenfeld, E. E., 1988. *Neurocomputing, foundations of research*. MIT Press.
- Aymerich, F., and Serra, M. 1998. Prediction of fatigue strength of composite laminates by means of neural networks (Vol. 144, pp. 231-240): *Key Engineering Materials*.
- Bae. 2009. *Bae systems annual report 2008*.
- Bai, Y., 2003. *Marine structural design*. Oxford: ELSEVIER SCIENCE Ltd.
- Baldwin, C., Niemczuk, J., Kiddy, J., Chen, P., Christiansen, M., and Chen, S., 2002. Structural testing of navy vessels using bragg gratings and a prototype digital spatial wavelength domain multiplexing (dswdm) system. *NAVAL ENGINEERS JOURNAL*, 63-70.

- Beale, M. H., Hagan, M. T., and Demuth, H. B. Neural network toolbox 7 user's guide.  
Available from: [Accessed].
- Belik, O., Bishop, R. E. D., and Price, W. G., 1983. A simulation of ship responses due to slamming in irregular head waves. *Trans R.I.N.A.* , 125.
- Beukelman, W., 1980. Bottom impact pressures due to forced oscillation. *International Shipbuilding Progress*, 27 (309).
- Bishop, C. M. 1995. Neural networks for pattern recognition. Oxford: Oxford University Press.
- Boresi, A. P., Schmidt, R. J., and Sidebottom, O. M., 1993. *Advanced mechanics of materials*. John Wiley.
- Bunting, E., and Sheahan, M. 2009. Broken by design?, *Yachting World* (pp. 66-79): IPC INSPIRE Ltd.
- Bv. 2008. *Rules for the classification and certification of yachts*. Bureau Veritas.
- Caianiello, E. R., 1961. Outline of a theory of thought-processes and thinking machines. *Journal of Theoretical Biology*, 2, 204-235.
- Cao, X., Sugiyama, Y., and Mitsui, Y., 1998. Application of artificial neural networks to load identification. *Computers & Structures*, 69, 63-67.
- Capecchi, V., 2010. *Applications of mathematics in models, artificial neural networks and arts : Mathematics and society*. Dordrecht ; New York: Springer.
- Caudill, M., 1989. *Neural networks primer*. San Francisco: Miller Freeman Publications.
- Caudill, M., and Butler, C., 1992. Understanding neural networks:Caudill, m., and c. Butler, understanding neural networks. *Computer Explorations*, 1 and 2.
- Chalmers, D. W., 1993. *Design of ship's structure*, London.
- Coello Coello, C. A., and Lamont, G. B., 2004. *Applications of multi-objective evolutionary algorithms*. World Scientific. Available from:  
<http://search.ebscohost.com/login.aspx?authtype=ip,shib&custid=s7547708&direct=true&db=nlebk&db=nlabk&site=ehost-live&scope=site&AN=148582>.
- Cowan, J. D., and Sharp, D. H., 1988. Neural nets and artificial intelligence. *Daedalus*, 117 (1), 85-121.
- Darpa. 1988. *Neural network study*. Lexington: MA: M.I.T. Lincoln Laboratory.
- Dnv. 2011. *Rules for classification of high speed light craft*. Det Norske Veritas.

- Efstathiades, C., Baniotopoulos, C. C., Nazarko, P., Ziemianski, L., and Stavroulakis, G. E., 2007. Application of neural networks for the structural health monitoring in curtain-wall systems. *Engineering Structures*, 29, 3475–3484.
- El Kadi, H., and Al-Assaf, Y. 2002. Prediction of the fatigue life of unidirectional glass fiber/epoxy composite laminate using different neural network paradigms (Vol. 55, pp. 239–246): *Composite Structures*.
- Faltinsen, O., 2000. Hydroelastic slamming. *J. Marine Science and Technology*, 5 (2), 49-65.
- Faltinsen, O. M., 2005. *Hydrodynamics of high-speed marine vehicles*. New York: Cambridge University Press.
- Faltinsen, O. M., 2007. Challenges in hydrodynamics of ships and ocean structures. *J. Naval Architecture and Shipbuilding Industry*, 5 (3), 268-277.
- Farely, B. G., and Clark, W. A., 1954. *Simulation of self-organising systems by digital computer*. IRE Transactions of Professional Group of Information Theory.
- Fausett, L. V. 1994. *Fundamentals of neural networks* (1st ed.): Prentice Hall.
- Frederick, J., Gravetter, Lori-Ann, B., Forzano, 2011, *Research Methods for the Behavioral Sciences*.
- Fukushima, K., 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybernetics*, 36, 193-202.
- Fukushima, K., 1982. Neocognitron: A new algorithm for pattern recognition of deformations and shifts in position. *Pattern Recognition*, 15, 455-469.
- Fukushima, K., 1986. A neural network model for selective attention in visual pattern recognition. *Biol. Cybernetics*, 55, 5-15.
- Fukushima, K., Miyake, S., and Ito, T., 1983. *Ieee transactions on systems. Man and Cybernetics*.
- Gerritsma, J., and Beukelman, W., 1964. *The distribution of the hydrodynamic forces on a heaving and pitching ship model in still water*. Paper presented at the Fifth Symposium on Naval Hydrodynamics, Washington.
- Greco, M., Colicchio, G., and Faltinsen, O. M., 2007. Shipping of water on a two-dimensional structure, part ii.

- Guo-Dong, X., and Wen-Yang, D., 2009. Review of prediction techniques on hydrodynamic impact of ships. *Marine Science Application*, 8, 204-210.
- Hagan, M. T., Demuth, H. B., and Beale, M. H., 1996. *Neural network design*. Boston: MA: PWS Publishing.
- Hagan, M. T., and Menhaj, M., 1999. Training feed-forward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 5 (6), 989-993.
- Haque, M., and Sudhakar, K. V. 2001. Prediction of corrosion-fatigue behaviour of dp steel through artificial neural network (Vol. 23, pp. 1-4): *International Journal of Fatigue*.
- Haykin, S., 1994. *Neural networks, a comprehensive foundation*. Macmillan College Publishing Co. Inc.
- Hebb, D. O., 1949. The organisation of behaviour. *In*. New York: Wiley, 60-78.
- Heller, S., and Jasper, N., 1961. On the structural design of planing craft. *Transaction of RINA*, 103, 49-65.
- Helliwell, I. S., Turega, M. A., and Cottis, R. A., 1995. *Accountability of neural networks trained with 'real world' data*. Paper presented at the Artificial Neural Networks.
- Hentinen, M., and Holm, G., 1994. Load measurement on the 9.4m sailing yacht sail lab. *13th International Symposium Yacht Design and Yacht Construction*, 131-161.
- Holden, S. B., and Rayner, P. J. W., 1995. *Generalization and pac learning: Some new results for the class of generalized single-layer networks*. Paper presented at the Transactions on Neural Networks.
- Hopfield, J. J., 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79, 2554-2558.
- Hughes, O. F., 1983. *Ship structural design. A rationally-based, computer-aided, optimisation approach*. New York: John Wiley and Sons.
- Hydrolance. 2013. Available from: <http://www.hydrolance.net/Common/DeakWash-ship.jpg>, [Accessed: 05/04/2013].
- Iso. 2008. *Small craft – hull construction and scantlings, part 5: Design pressures for monohulls design stresses, scantlings determination.*: International Organization for Standardization.
- Iwashita, H., Ito, A., Okada, T., Ohkusu, M., Takaki, M., and Mizoguchi, S., 1993. Wave force acting on a blunt ship with forward speed in oblique sea. 173, 195-208.

- J.M. Ko, Y. Q. N., July 2005. Technology developments in structural health monitoring of large-scale bridges. *Engineering Structures*, 1715–1725.
- James, W., 1890. Psychology (briefer course). In. New York: Holt, 253-279.
- Jensen, J. J., and Pedersen, P. T., 1979. Wave-induced bending moments in ships: A quadratic theory. *J. Trans. Roy. Soc. Nav. Archit.*, 121, 151-165.
- Jones, N., 1972. *Slamming damage*. MIT Department of Ocean Engineering.
- Joubert, P., 1982. Strength of bottom plating of yachts. *J. Ship Research*, 26 (1), 45-49.
- Joubert, P., 1996. Tests on yacht hull plating. *J. Marine Technology*, 33, 130-140.
- Kapsenberg, G., Veer, A., Hackett, J., and Levadou, M., 2003. Aftbody slamming and whipping loads. *SNAME Annual Meeting*, 111, 213-231.
- Karman, T. V., 1929. *The impact on seaplane floats during landing - technical report*. Aachen: Aerodynamical Institute of the Technical High School.
- Katsikeros, C. E., and Labeas, G. N., 2009. Development and validation of a strain-based structural health. *Mechanical Systems and Signal Processing*, 23, 372-383.
- Ko, J. M., and Ni, Y. Q., 2005. Technology developments in structural health monitoring of large-scale bridges. *Engineering Structures*, 1715–1725.
- Koelbel, G., and Jr., J., 1995. Comments on the structural design of high speed craft. *Marine Technology*, 32 (2).
- Kohonen, T., 1972. Correlation matrix memories. *IEEE Transaction on Computers*, 21 (4), 353-359.
- Kohonen, T., 1982. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43, 59-69
- Kohonen, T., 1987. *Self-organization and associative memory*. Berlin: Springer-Verlag.
- Kohonen, T., 1988. An introduction to neural computing. *Neural Networks*, 1, 3-16.
- Kohonen, T., 1997. *Self-organizing maps*. Berlin: Springer-Verlag.
- Kohonen, T., 1998. An introduction to neural computing. *Neural Networks*, 1.
- Korvin-Kroukovsky, B. V., 1955. *Investigation of ship motions in regular waves*. Transactions of SNAME.
- Korvin-Kroukovsky, B. V., and Jacobs, W. R., 1997. *"Pitching and heaving motions of a ship in regular waves*. Trans SNAME.

- Kukkanen, T., 2010. Wave load predictions for marine structures. *Rakenteiden Mekaniikka (Journal of Structural Mechanics)*, 43 (3), 150-166.
- Lee, J., and Wilson, P. A., 2009. Experimental study of the hydro-impact of slamming in a modern racing sailboat *J. ships Marine Science Application*, 8, 204-210.
- Lee, J. A., Almond, D. P., and B., H. 1999. The use of neural networks for the prediction of fatigue lives of composite materials (Vol. 30, pp. 1159–1169): *Composites Part A: Applied Science and Manufacturing*.
- Li, H. C. H., Herszberg, I., Davis, C. E., Mouritz, A. P., and Galea, S. C., 2006. Health monitoring of marine composite structural joints using fibre optic sensors. *Composite Structures*, 75, 321–327.
- Lippman, R. P., 1987. An introduction to computing with neural nets. *IEEE ASSP Magazine*, 4–22.
- Liu, P., and Li, H.-X., 2004. *Fuzzy neural network theory and application*. World Scientific. Available from:  
<http://search.ebscohost.com/login.aspx?authtype=ip,shib&custid=s7547708&direct=true&db=nlebk&db=nlabk&site=ehost-live&scope=site&AN=235586>.
- Livingstone, D., 2008. *Artificial neural networks : Methods and applications*. Totowa, NJ: Humana Press.
- Lynch, J. P., Law, K. H., Kiremidjian , A. S., Carryer, E., Farrar, C. R., and Sohn, H., 2004. Design and performance validation of a wireless sensing unit for structural monitoring applications. *Structural Engineering and Mechanics*, 17(13–14):393–408.
- M. Raudenský, J. H., J. Krejsa and L. Sláma. 1996. Usage of artificial intelligence methods in inverse problems for estimation of material parameters. *INT. J. NUM. METH. HEAT FLUID FLOW*, 6, 19-29.
- Mackay, D. J. C., 1992. Bayesian interpolation. *Neural Computation*, 4 (3), 415–447.
- Majumder, M., Gangopadhyay, T. K., and Chakraborty, A. K., 2008. Fibre bragg gratings in structural health monitoring—present status and applications. *Sensors and Actuators A: Physical*.
- Mamdani, E. H., and Assilian, S., 1975. An experiment in linguistic synthesis with a fuzzy logic controller. *Journal of Man-Machine Studies*, 7, 1-13.

- Manganelli, P., 2006. *Experimental investigation of dynamic loads on offshore racing yachts*. University of Southampton, UK: Ph.D thesis.
- Mcculloch, W. S., and Pitts, W. A., 1943. A logical calculus of the ideas immanent in nervous activity. *Buttetin of Mathematics and Biophysics*, 5, 115-133.
- Moller, M. F., 1993. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6, 525–533.
- Mousumi Majumder, T. K. G., Ashim Kumar Chakraborty,. 2008. Fibre bragg gratings in structural health monitoring—present status and applications. *Sensors and Actuators A: Physical*.
- Mufti Aa, T. G., Jones Pr. 1997. Field assessment of fibre-optic bragg grating strain sensors in the confederation bridge. *Canadian Journal of Civil Engineering*, 24(26):963–966.
- Murray, R., Neumerkel, D., and Sbarbaro, D., 1992. *Neural networks for modeling and control of a non-linear dynamic system*.
- Nakos, D., and Sclavounos, P., 1990. *Ship motions by a three-dimensional rankine panel method*. Paper presented at the Proceedings of 18th symposium on naval hydrodynamics, Ann Arbor.
- Ni. 2013. Ni compactdaq - national instruments.
- Nilson, N. J., 1965. *Learning machines*. McGraw-Hill.
- Ochi, M. K., and Motter, L. E., 1969. Prediction of extreme values of impact pressures associated with ship slamming. *J. Ship Research*, 13 (2).
- Ochi, M. K., and Motter, L. E., 1973. Prediction of slamming characteristics and hull responses for ship design. *SNAME Trans*, 8, 144 - 176.
- Pan, S. A., 2012. *On demand dbs for parkinson's disease : Tremor prediction using artificial neural networks*. Thesis. University of Reading.
- Panigrahi, B. K., Abraham, A., and Das, S., 2010. *Computational intelligence in power engineering*. Berlin: Springer.
- Petrov, S. 2012. Berge stahl, Available from: [http://maritimebg.com/wp-content/uploads/Berge Stahl Bulk carrier.jpg](http://maritimebg.com/wp-content/uploads/Berge_Stahl_Bulk_carrier.jpg) [Accessed: 06/04/2013].
- Phelps, B. P., 1997, November. Determination of wave loads for ship structural analysis. DSTO Aeronautical and Maritime Research Laboratory: Maritime Platforms Division (MPD),

- DSTO-RR-0116, 29 pages. Available from:  
<http://dspace.dsto.defence.gov.au/dspace/bitstream/1947/3313/1/DSTO-RR-0116%20PR.pdf> [Accessed: 02/04/2013].
- Pleune, T. T., and Chopra, O. K. 2000. Using artificial neural networks to predict the fatigue life of carbon and low-alloy steels (Vol. 197, pp. 1–12): Nuclear Engineering and Design.
- Rao, V. B., and Rao, H. V., 1993. *C++ neural networks and fuzzy logic*. Management Information Source.
- Rawson, K. J., and Tupper, E. C., 1976. *Basic ship theory vol. 2*. 2nd ed. ed. [S.l.]: Longman.
- Rawson, K. J., and Tupper, E. C., 2001. *Basic ship theory*. 5th ed. ed. Oxford: Butterworth-Heinemann, 2002.
- Rawson, K. J. A. T., and Tupper, E. C., 1968. *Basic ship theory*. Harlow: Longmans.
- Reichard, R., 1984. The structural response of small craft to dynamic loading. *Proceedings of the 14th AIAA Symposium on the Aero/Hydronautics of Sailing*, 30, 105-110.
- Rosenblatt, F., 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65 (6), 386–408.
- Rudas, I. J., Fodor, J., and Kacprzyk, J., 2010. *Computational intelligence in engineering*. Berlin: Springer.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J., 1986. Learning internal representations by error propagation. *In: Vol. 1*. Cambridge: The M.I.T.Press, 318–362.
- Salvesen, N., Tuck, E. O., and Faltinsen, O., 1970. Ship motions and sea loads, 78, 250 - 287.
- Savitsky, D., and Brown, P. W., 1976. Procedures for hydrodynamic evaluation of planing hulls in smooth and rough water. 13 (4).
- Schalkoff, R., 1997. *Artificial neural network*. Singapore: The McGRAW-HILL Companies, Inc.
- Schmidt, A., 1996. *A modular neural network architecture with additional generalization abilities for high dimensional input vectors*.
- Schumann, J. M., and Liu, Y., 2010. *Applications of neural networks in high assurance systems*. Berlin: Springer.
- Sewell, P., Noroozi, S., Vinney, J., Amali, R., and Andrews, S., 2010. Improvements in the accuracy of an inverse problem engine's output for the prediction of below-knee

- prosthetic socket interfacial loads. *Engineering Applications of Artificial Intelligence*, 23, 1000–1011.
- Shar, S., and Palmieri, F., 1990. *Meka-a fast, local algorithm for training feed forward neural networks*. Paper presented at the International Joint Conference on Neural Networks.
- Shockmitigation. Bae systems halmatic fast interceptor craft. Available from: <http://www.shockmitigationdirectory.com/inner-category/3/6/313/> [Accessed.
- Sikora, J. P., Dinsbacher, A., and Beach, J. E., 1983. A method for estimating lifetime loads and fatigue lives for swath and conventional monohull ships. *Naval Engineers Journal*.
- Sivakumar, B., Berndtsson, R., and Ebrary Inc., 2010. *Advances in data-based approaches for hydrologic modeling and forecasting*. World Scientific. Available from: <http://site.ebrary.com/lib/bournemouth/Doc?id=10479909>.
- Soo-Young, K., Moon, B.-Y., Kim, D.-E., and Shin, S. C., 2006. Automation of hull plates classification in ship design system using neural network method. *Mechanical Systems and Signal Processing*, 20 (2), 493–504.
- Spencer, J. B. F., Ruiz-Sandoval, M. E., and N., K., 2004. Smart sensory technology: Opportunities and challenges. *Structural Control and Health Monitoring [CDROM]*, 11(14):349–368.
- Ssc. 1995. *Hydrodynamic impact on displacement ship hulls*. an assessment of the state of the art Ship Structure Committee 385.
- St. Denis, M., and Pierson, W. J., 1953. *On the motions of ships in confused sea*. Transactions of SNAME.
- Stavovy, A., and Chuang, S., 1976. Analytical determination of slamming pressures for high-speed vehicles in waves. *J. Ship Research*, 20 (4), 190-198.
- Sumitro, S., and M.L., W., 2003. *Sustainable structural health monitoring system*. Paper presented at the Proceedings of the international workshop on advanced sensors, structural health monitoring and smart structures.
- Sun, H., and Faltinsen, O. M., 2012. Hydrodynamic forces on a semi-displacement ship at high speed. *J. Applied Ocean Research*, 34, 68-77.
- Swingler, K. 1996. *Applying neural networks: A practical guide*: Academic Press.

- Takagi, K., 1993. Takagi k. Calculation of unsteady pressure by rankine source method. *J Kansai Soc Nav Archit*, 219, 47–56.
- Tang, J., Cao, X., and Mitsui, Y., 1995. *A study on determining the vertical tail loads of an aircraft by strain method*. Paper presented at the PICAST'95 (Proceedings of Second Pacific International Conference on Aerospace Science & Technology), Australia.
- Tupper, E. C., 2004. *Introduction to naval architecture*. 4th ed. ed. Amsterdam ; London: Elsevier, Butterworth Heinemann.
- Velten, K., Reinicke, R., and Friedrich, K. 2000. Wear volume prediction with artificial neural networks. (Vol. 33, pp. 731–736): Tribology International
- Venkatesh, V., and Rack, H. J. 1999. A neural network approach to elevated temperature creep-fatigue life prediction (Vol. 21, pp. 225–234): International Journal of Fatigue.
- Vishay\_Group. 2010. Strain gage selection: Criteria, procedures, recommendations. Available from: <http://www.vishaypg.com/docs/11055/tn505.pdf> [Accessed.
- Vogl, T. P., Mangis, J. K., Rigler, A. K., Zink, W. T., and Alkon, D. L., 1988. Accelerating the convergence of the back-propagation method. *Biological Cybernetics*, 59, 256–264.
- Watanabe, I., 1994. Practical method for diffraction pressure on a ship running in oblique wave. *J. Kansai Soc. Nav. Archit.*, 221, 83-89 [in Japanese].
- Westergaard, H. M., and Salter, W. A., 1921. Moments and stresses in slabs. *Proceeding of Amer. Concrete Inst.*, 17, 415-538.
- Widrow, B., and Hoff, M. E., 1960. *Adaptive switching circuits*. 96-104: WESTCON Convention.
- Windmill. 2011. Windmill data acquisition catalogue, <http://www.windmillsoft.com>: Windmill Software Ltd.
- Xu, R., and Wunsch, D. C., 2009. *Clustering*. Wiley. Available from: <http://search.ebscohost.com/login.aspx?authtype=ip,shib&custid=s7547708&direct=true&db=nlebk&db=nlabk&site=ehost-live&scope=site&AN=254099>.
- Xu, S., Deng, X., Tiwar, V., Sutton, M. A., and Fourney, W. L., 2010. An inverse approach for pressure load identification. *International Journal of Impact Engineering*, 37 (7), 865-877.

- Yang, J. Intelligent data mining using artificial neural networks and genetic algorithms [electronic resource] : Techniques and applications. Available from: <http://wrap.warwick.ac.uk/3831/> [Accessed.
- Yasukawa, H., 1990. A rankine panel method to calculate unsteady ship hydrodynamic forces. *J Soc Nav Archit*, 168, 131–140.
- Yu, T., 2013. *Computational intelligent data analysis for sustainable development*. Boca Raton: Taylor & Francis.
- Zainun, N. Y. B. A., 2011. *Computerized model to forecast low-cost housing demand in urban area in malaysia using artificial neural networks (ann)*. Thesis. Loughborough University.
- Zaknich, A. 2003. *Neural networks for intelligent signal processing*. Singapore: World Scientific publishing co.
- Zeng, Z., and Wang, J., 2010. *Advances in neural network research and applications*. Berlin: Springer Verlag.
- Zhang, Z., and Friedrich, K. 2003. Artificial neural networks applied to polymer composites: A review (Vol. 63, pp. 2029–2044): *Composites Science and Technology*.
- Zhang, Z., Friedrich, K., and Velten, K. 2002. Prediction of tribological properties of short fiber composites using artificial neural networks (Vol. 252, pp. 668–675): *Wear*.
- Ziemianski, L., and Harpula, G., 1999. The use of neural networks for damage detection in eight storey frames. *Proceedings of the 5th international conference: Engineering applications of neural networks*, 292–297.
- Zingoni, A., 2005. Structural health monitoring, damage detection and long-term performance. *Engineering Structures*, 1713-1714.
- Zurada, J. M., 1992. *Introduction to artificial neural systems*. Boston: PWS Publishing Company.

## Appendices

### Appendix 1: PYTHON Script of Large Displacement Experiment

#### Part 1

1. # -\*- coding: mbc -\*-
2. import sys,math
3. import visualization
4. from part import \*
5. from material import \*
6. from section import \*
7. from assembly import \*
8. from step import \*
9. from interaction import \*
10. from load import \*
11. from mesh import \*
12. from job import \*
13. from sketch import \*
14. from visualization import \*
15. from connectorBehavior import \*

#### Part 2

16. mdb.models['Model-1'].ConstrainedSketch(name='\_\_profile\_\_', sheetSize=.200)
17. mdb.models['Model-1'].sketches['\_\_profile\_\_'].rectangle(point1=(.500, .500),  
point2=(-.500, -.500))
18. mdb.models['Model-1'].Part(dimensionality=THREE\_D, name='Part-1', type=  
DEFORMABLE\_BODY)
19. mdb.models['Model-1'].parts['Part-1'].BaseSolidExtrude(depth=.005, sketch=

```

mdb.models['Model-1'].sketches['__profile__'])

20. del mdb.models['Model-1'].sketches['__profile__']
21. mdb.models['Model-1'].ConstrainedSketch(gridSpacing=.07071, name='__profile__',
sheetSize=2.82844, transform=mdb.models['Model-1'].parts['Part-
1'].MakeSketchTransform(
sketchPlane=mdb.models['Model-1'].parts['Part-1'].faces[4],
sketchPlaneSide=SIDE1,
sketchUpEdge=mdb.models['Model-1'].parts['Part-1'].edges[4],
sketchOrientation=RIGHT, origin=(0.0, 0.0, .005)))
22. mdb.models['Model-1'].parts['Part-1'].projectReferencesOntoSketch(filter=
COPLANAR_EDGES, sketch=mdb.models['Model-1'].sketches['__profile__'])
23. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(0.0,
0.0), point1=(0.0, .016))
24. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(-0.375,
0.250), point1=(-0.375, 0.266))
25. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(-0.375,
-0.250), point1=(-0.375, -0.266))
26. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(-0.375,
0.0), point1=(-0.375, 0.016))
27. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(-0.125,
0.250), point1=(-0.125, 0.266))
28. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(-0.125,
0.0), point1=(-0.125, 0.016))
29. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(0.375,
0.250), point1=(0.375, 0.266))

```

30. `mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(0.375, 0.0), point1=(0.375, 0.016))`
31. `mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(0.125, 0.250), point1=(0.125, 0.266))`
32. `mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(-0.125, -0.250), point1=(-0.125, -0.266))`
33. `mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(0.125, 0.0), point1=(0.125, 0.016))`
34. `mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(0.125, -0.250), point1=(0.125, -0.266))`
35. `mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(0.375, -0.250), point1=(0.375, -0.266))`
36. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-0.1285, -0.3765), point2=(-0.1215, -0.3735))`
37. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-0.1285, 0.3765), point2=(-0.1215, 0.3735))`
38. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0.1285, -0.3765), point2=(0.1215, -0.3735))`
39. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0.1285, 0.3765), point2=(0.1215, 0.3735))`
40. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-0.3785, -0.3765), point2=(-0.3715, -0.3735))`
41. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-0.3785, 0.3765), point2=(-0.3715, 0.3735))`
42. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0.3785, -0.3765), point2=(0.3715, -0.3735))`
43. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0.3785, 0.3765), point2=(0.3715, 0.3735))`

```

44. mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-0.1285, -0.1265),
    point2=(-0.1215, -0.1235))
45. mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-0.1285, 0.1265),
    point2=(-0.1215, 0.1235))
46. mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0.1285, -0.1265),
    point2=(0.1215, -0.1235))
47. mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0.1285, 0.1265),
    point2=(0.1215, 0.1235))
48. mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-0.3785, -0.1265),
    point2=(-0.3715, -0.1235))
49. mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-0.3785, 0.1265),
    point2=(-0.3715, 0.1235))
50. mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0.3785, -0.1265),
    point2=(0.3715, -0.1235))
51. mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0.3785, 0.1265),
    point2=(0.3715, 0.1235))
52. mdb.models['Model-1'].parts['Part-1'].PartitionFaceBySketch(faces=
    mdb.models['Model-1'].parts['Part-1'].faces.getSequenceFromMask(('[#10 ]',
    ), ), sketch=mdb.models['Model-1'].sketches['__profile__'], sketchUpEdge=
    mdb.models['Model-1'].parts['Part-1'].edges[4])
53. del mdb.models['Model-1'].sketches['__profile__']

```

### Part 3

```

54. mdb.models['Model-1'].Material(name='Material-1')
55. mdb.models['Model-1'].materials['Material-1'].Density(table=((1786.0, ), ))
56. mdb.models['Model-1'].materials['Material-1'].Elastic(table=((21220000000, 0.128), ))
57. mdb.models['Model-1'].HomogeneousSolidSection(material='Material-1', name=
58. 'Section-1', thickness=None)

```

```

59. mdb.models['Model-1'].rootAssembly.DatumCsysByDefault(CARTESIAN)
60. mdb.models['Model-1'].rootAssembly.Instance(dependent=ON, name='Part-1-1',
    part=mdb.models['Model-1'].parts['Part-1'])
61. mdb.models['Model-1'].parts['Part-1'].seedPart(deviationFactor=0.1, size=0.05)
62. mdb.models['Model-1'].parts['Part-1'].generateMesh()

```

#### Part 4

```

63. mdb.models['Model-1'].rootAssembly.regenerate()
64. mdb.models['Model-1'].parts['Part-1'].SectionAssignment(offset=0.0,
    offsetField="", offsetType=MIDDLE_SURFACE, region=Region(
    cells=mdb.models['Model-1'].parts['Part-1'].cells.getSequenceFromMask(
    mask=('#1 ]', ), )), sectionName='Section-1', thicknessAssignment=
    FROM_SECTION)
65. mdb.models['Model-1'].rootAssembly.regenerate()
66. mdb.models['Model-1'].StaticStep(name='Step-1', nlgeom=ON,
    previous='Initial', initialInc=0.1, minInc=5e-5, maxInc=0.1)
67. mdb.models['Model-1'].rootAssembly.Instance(dependent=ON, name='Part-1-1',
    part=mdb.models['Model-1'].parts['Part-1'])
68. mdb.models['Model-1'].rootAssembly.Surface(name='Surf-1', side1Faces=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(
    ('#4000000 ]', ), ))
69. mdb.models['Model-1'].rootAssembly.Surface(name='Surf-2', side1Faces=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(
    ('#1000000 ]', ), ))
70. mdb.models['Model-1'].rootAssembly.Surface(name='Surf-3', side1Faces=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(
    ('#200000 ]', ), ))

```



81. `mdb.models['Model-1'].rootAssembly.Surface(name='Surf-14', side1Faces=  
mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(  
(['#40 '], ), ))`
82. `mdb.models['Model-1'].rootAssembly.Surface(name='Surf-15', side1Faces=  
mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(  
(['#400000 '], ), ))`
83. `mdb.models['Model-1'].rootAssembly.Surface(name='Surf-16', side1Faces=  
mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(  
(['#80 '], ), ))`
84. `mdb.models['Model-1'].rootAssembly.Surface(name='Load_Surf-17', side1Faces=  
mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(  
(['#100 '], ), ))`
85. `mdb.models['Model-1'].rootAssembly.Surface(name='Load_Surf-18', side1Faces=  
mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(  
(['#100000 '], ), ))`
86. `mdb.models['Model-1'].rootAssembly.Surface(name='Load_Surf-19', side1Faces=  
mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(  
(['#80000 '], ), ))`
87. `mdb.models['Model-1'].rootAssembly.Surface(name='Load_Surf-20', side1Faces=  
mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(  
(['#2000 '], ), ))`
88. `mdb.models['Model-1'].rootAssembly.Surface(name='Load_Surf-21', side1Faces=  
mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(  
(['#40000 '], ), ))`
89. `mdb.models['Model-1'].rootAssembly.Surface(name='Load_Surf-22', side1Faces=  
mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(  
(['#200 '], ), ))`
90. `mdb.models['Model-1'].rootAssembly.Surface(name='Load_Surf-23', side1Faces=  
mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(  
(['#20000 '], ), ))`

91. `mdb.models['Model-1'].rootAssembly.Surface(name='Load_Surf-24', side1Faces=  
mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(  
('#10000 '], ), ))`
92. `mdb.models['Model-1'].rootAssembly.Surface(name='Load_Surf-25', side1Faces=  
mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(  
('#800 '], ), ))`
93. `mdb.models['Model-1'].rootAssembly.Surface(name='Load_Surf-26', side1Faces=  
mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(  
('#400 '], ), ))`
94. `mdb.models['Model-1'].rootAssembly.Surface(name='Load_Surf-27', side1Faces=  
mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(  
('#1000 '], ), ))`
95. `mdb.models['Model-1'].rootAssembly.Surface(name='Load_Surf-28', side1Faces=  
mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(  
('#8000 '], ), ))`
96. `mdb.models['Model-1'].rootAssembly.Surface(name='Load_Surf-29', side1Faces=  
mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(  
('#4000 '], ), ))`
  
97. `Surfacelist=['Surf-0','Surf-1','Surf-2','Surf-3','Surf-4','Surf-5','Surf-6','Surf-7','Surf-8','Surf-  
9','Surf-10','Surf-11','Surf-12','Surf-13','Surf-14','Surf-15','Surf-16','Load_Surf-  
17','Load_Surf-18','Load_Surf-19','Load_Surf-20','Load_Surf-21','Load_Surf-  
22','Load_Surf-23','Load_Surf-24','Load_Surf-25','Load_Surf-26','Load_Surf-  
27','Load_Surf-28','Load_Surf-29']`

#### Part 5

98. `mdb.models['Model-1'].EncastreBC(createStepName='Step-1', name='BC-1', region=  
Region(`

```
faces=mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask(  
mask=(['#e0000000 #1 ], , )))
```

#### Part 6

```
99. mdb.models['Model-1'].fieldOutputRequests['F-Output-1'].setValues(variables=(  
'S', 'E', 'VE', 'PE', 'VEEQ', 'PEEQ', 'PEEQT', 'PEEQMAX', 'PEMAG', 'PEQC',  
'EE', 'IE', 'THE', 'NE', 'LE', 'ER', 'SE', 'SPE', 'SEPE', 'SEE', 'SEP',  
'SALPHA', 'U', 'RF', 'CF', 'CSTRESS', 'CDISP'))
```

#### Part 7

# Note that for pressure applied to circular area with radius of 16 mm equals 1kg weight(9.81 N) = 12203.92118 N/m<sup>2</sup>

```
100.     unit_load_pressure=12266.123  
101.     Initial_load_step=2.5  
102.     desired_load_step=2.5  
103.     for j in range(1,14):  
104.         loadName = 'Load_%s' % (j)  
105.         surfName='Load_Surf-%s'%(j+16)  
106.         mdb.models['Model-1'].Pressure(amplitude=UNSET, createStepName='Step-1',  
distributionType=UNIFORM, field="", magnitude=12440.2866,  
name=loadName,  
region=mdb.models['Model1'].rootAssembly-surfaces[Surfacelist[16+j]])
```

#### Part 8

```
107.     for i in range(1,33):
```

```

108.         pressure_value=( unit_load_pressure*Initial_load_step)
           +((unit_load_pressure*desired_load_step)*(i-1))
109.         mdb.models['Model-1'].Pressure(amplitude=UNSET,
           createStepName='Step-1', distributionType=UNIFORM, field=" ,
           magnitude=pressure_value, name=loadName, region= mdb.models['Model-1']
           .rootAssembly-surfaces[ Surfacelist[16+j]])
110.         jobName = 'Test_surf_%s_JOB_%s' % (j,i)
111.         print jobName
112.         mdb.Job(atTime=None, contactPrint=OFF, description=" , echoPrint=OFF,
           explicitPrecision=SINGLE, getMemoryFromAnalysis=True, historyPrint=OFF,
           memory=98, memoryUnits=PERCENTAGE, model='Model-1', modelPrint=OFF,
           multiprocessingMode=DEFAULT,
name=jobName,nodalOutputPrecision=SINGLE,
           numCpus=1, queue=None, scratch=" , type=ANALYSIS, userSubroutine=" ,
           waitHours=0, waitMinutes=0)

113.         mdb.jobs[jobName].submit()
114.         mdb.jobs[jobName].waitForCompletion()

Part 9

115.         print jobName
116.         openOdb(path=jobName + '.odb')
117.         myViewport = session.Viewport(name='myviewport', origin=(0, 0),
           width=250, height=135)
118.         myOdb = visualization.openOdb(path=jobName + '.odb')
119.         myViewport.setValues(displayedObject=myOdb)

```

```

120. myViewport.odbDisplay.display.setValues(plotState=CONTOURS_ON_DEF)
121. myViewport.odbDisplay.commonOptions.setValues(renderStyle=FILLED)
122. myOdb.steps['Step-1'].frames[1].fieldOutputs['EE'].values[5].data*1000000
123. mystring=[5,7, 8, 6 , 1, 3, 4 , 2, 68, 66, 65, 67, 64, 62, 61, 63 ]
124. outputFile = open('Strain_EE_'+jobName + '.txt','w')
125. outputFile.write('
    Element\tmaxPrincipal\tmidPrincipal\tminPrincipal\tEE[11]\t\tEE[22]\t\tEE[33]\t\tpressu
    re_value\n')
126. for m in range(0,16):
127.     k=mystring[m]
128.     v=myOdb.steps['Step-1'].frames[10].fieldOutputs['EE'].values[k-1]
129.     outputFile.write('%d\t%.6e\t%.6e\t%.6e\t%.6e\t%.6e\t%.6e\n' %
        (v.elementLabel,v.maxPrincipal,v.midPrincipal,v.minPrincipal ,v.data[0], v.data[1],
        v.data[3],pressure_value ))
130.     outputFile.close()
131.     myOdb.close()
132. del mdb.models['Model-1'].loads[loadName]

```

## Appendix 2: PYTHON Script of Drop Test Experiment

### Part 1

```
# -*- coding: mbcs -*-
```

1. import sys,math
2. import visualization
3. from part import \*
4. from material import \*
5. from section import \*
6. from assembly import \*
7. from step import \*
8. from interaction import \*
9. from load import \*
10. from mesh import \*
11. from job import \*
12. from sketch import \*
13. from visualization import \*
14. from connectorBehavior import \*

### Part 2

15. mdb.models['Model-1'].ConstrainedSketch(name='\_\_profile\_\_', sheetSize=.200)
16. mdb.models['Model-1'].sketches['\_\_profile\_\_'].rectangle(point1=(.500, .500), point2=(-.500, -.500))
17. mdb.models['Model-1'].Part(dimensionality=THREE\_D, name='Part-1', type=DEFORMABLE\_BODY)
18. mdb.models['Model-1'].parts['Part-1'].BaseSolidExtrude(depth=.005, sketch=mdb.models['Model-1'].sketches['\_\_profile\_\_'])
19. del mdb.models['Model-1'].sketches['\_\_profile\_\_']

```

20. mdb.models['Model-1'].ConstrainedSketch(gridSpacing=.07071, name='__profile__',
    sheetSize=2.82844, transform=mdb.models['Model-1'].parts['Part-
    1'].MakeSketchTransform(sketchPlane=mdb.models['Model-1'].parts['Part-1'].faces[4],
    sketchPlaneSide=SIDE1, sketchUpEdge=mdb.models['Model-1'].parts['Part-1'].edges[4],
    sketchOrientation=RIGHT, origin=(0.0, 0.0, .005)))
21. mdb.models['Model-1'].parts['Part-
    1'].projectReferencesOntoSketch(filter=COPLANAR_EDGES,
    sketch=mdb.models['Model-1'].sketches['__profile__'])
22. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(0.0,
    0.0), point1=(0.0, .016))
23. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(-0.375,
    0.250), point1=(-0.375, 0.266))
24. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(-0.375,
    -0.250), point1=(-0.375, -0.266))
25. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(-0.375,
    0.0), point1=(-0.375, 0.016))
26. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(-0.125,
    0.250), point1=(-0.125, 0.266))
27. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(-0.125,
    0.0), point1=(-0.125, 0.016))
28. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(0.375,
    0.250), point1=(0.375, 0.266))
29. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(0.375,
    0.0), point1=(0.375, 0.016))
30. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(0.125,
    0.250), point1=(0.125, 0.266))
31. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(-0.125,
    -0.250), point1=(-0.125, -0.266))
32. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(0.125,
    0.0), point1=(0.125, 0.016))

```

33. `mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(0.125, -0.250), point1=(0.125, -0.266))`
34. `mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=(0.375, -0.250), point1=(0.375, -0.266))`
35. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-0.1285, -0.3765), point2=(-0.1215, -0.3735))`
36. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-0.1285, 0.3765), point2=(-0.1215, 0.3735))`
37. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0.1285, -0.3765), point2=(0.1215, -0.3735))`
38. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0.1285, 0.3765), point2=(0.1215, 0.3735))`
39. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-0.3785, -0.3765), point2=(-0.3715, -0.3735))`
40. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-0.3785, 0.3765), point2=(-0.3715, 0.3735))`
41. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0.3785, -0.3765), point2=(0.3715, -0.3735))`
42. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0.3785, 0.3765), point2=(0.3715, 0.3735))`
43. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-0.1285, -0.1265), point2=(-0.1215, -0.1235))`
44. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-0.1285, 0.1265), point2=(-0.1215, 0.1235))`
45. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0.1285, -0.1265), point2=(0.1215, -0.1235))`
46. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0.1285, 0.1265), point2=(0.1215, 0.1235))`
47. `mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-0.3785, -0.1265), point2=(-0.3715, -0.1235))`

```

48. mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-0.3785, 0.1265),
    point2=(-0.3715, 0.1235))
49. mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0.3785, -0.1265),
    point2=(0.3715, -0.1235))
50. mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0.3785, 0.1265),
    point2=(0.3715, 0.1235))
51. mdb.models['Model-1'].parts['Part-1'].PartitionFaceBySketch(faces=mdb.models['Model-
    1'].parts['Part-1'].faces.getSequenceFromMask(['[#10 ]',  ), ),
    sketch=mdb.models['Model-1'].sketches['__profile__'], sketchUpEdge=
    mdb.models['Model-1'].parts['Part-1'].edges[4])
52. del mdb.models['Model-1'].sketches['__profile__']
53. mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=200.0)
54. mdb.models['Model-1'].sketches['__profile__'].CircleByCenterPerimeter(center=( 0.0,
    0.0), point1=(0.01, 0.0))
55. mdb.models['Model-1'].Part(dimensionality=THREE_D, name='cylinder', type=
    DEFORMABLE_BODY)
56. mdb.models['Model-1'].parts['cylinder'].BaseSolidExtrude(depth=0.103,
    sketch=mdb.models['Model-1'].sketches['__profile__'])
57. del mdb.models['Model-1'].sketches['__profile__']

```

### Part 3

```

58. mdb.models['Model-1'].HomogeneousSolidSection(material='Material-1', name=
    'Section-1', thickness=None)
59. mdb.models['Model-1'].HomogeneousSolidSection(material='Material-2', name=
    'Section-2', thickness=None)
60. mdb.models['Model-1'].parts['Part-1'].SectionAssignment(offset=0.0, offsetField="",
    offsetType=MIDDLE_SURFACE, region=Region( cells=mdb.models['Model-
    1'].parts['Part-1'].cells.getSequenceFromMask( mask=['[#1 ]', ), ), ),
    sectionName='Section-1', thicknessAssignment= FROM_SECTION)
61. mdb.models['Model-1'].rootAssembly.regenerate()

```

```

62. mdb.models['Model-1'].parts['cylinder'].Set(cells= mdb.models['Model-
    1'].parts['cylinder'].cells.getSequenceFromMask(['#1'], ), name='Set-1')
63. mdb.models['Model-1'].parts['cylinder'].SectionAssignment(offset=0.0, offsetField=",
    offsetType=MIDDLE_SURFACE, region=mdb.models['Model-
    1'].parts['cylinder'].sets['Set-1'], sectionName='Section-2',
    thicknessAssignment=FROM_SECTION)
64. mdb.models['Model-1'].rootAssembly.DatumCsysByDefault(CARTESIAN)
65. mdb.models['Model-1'].rootAssembly.Instance(dependent=ON, name='Part-1-1',
    part=mdb.models['Model-1'].parts['Part-1'])
66. mdb.models['Model-1'].parts['Part-1'].seedPart(deviationFactor=0.1, size=0.05)
67. mdb.models['Model-1'].parts['Part-1'].generateMesh()
68. mdb.models['Model-1'].rootAssembly.regenerate()
69. mdb.models['Model-1'].rootAssembly.Instance(dependent=OFF, name='cylinder-1',
    part=mdb.models['Model-1'].parts['cylinder'])
70. mdb.models['Model-1'].rootAssembly.seedPartInstance(deviationFactor=0.1,
    minSizeFactor=0.1, regions=(mdb.models['Model-1'].rootAssembly.instances['cylinder-
    1'], ), size=0.01)
71. mdb.models['Model-1'].rootAssembly.generateMesh(regions=(mdb.models['Model-
    1'].rootAssembly.instances['cylinder-1'], ))
72. mdb.models['Model-1'].rootAssembly.translate(instanceList=('cylinder-1', ),
    vector=(0.0, 0.0, 0.005))

```

#### Part 4

```

73. mdb.models['Model-1'].ExplicitDynamicsStep(name='Step-1', previous='Initial',
    timePeriod=0.03)
74. mdb.models['Model-1'].FieldOutputRequest(createStepName='Step-1', name= 'F-
    Output-1', variables=('S', 'MISES', 'MISESMAX', 'E', 'PE', 'NE', 'LE', 'ER', 'ERV', 'U',
    'UT', 'UR', 'V', 'VT', 'VR', 'A'))
75. mdb.models['Model-1'].fieldOutputRequests['F-Output-1'].setValues(numIntervals= 100)

```

```

76. mdb.models['Model-1'].rootAssembly.Set(faces=mdb.models['Model-
    1'].rootAssembly.instances['Part-1-1'].faces.getSequenceFromMask( ('[#e0000000 #1 ]',
    ), ), name='Set-1')
77. mdb.models['Model-1'].EncastreBC(createStepName='Step-1', localCsys=None, name=
    'BC-1', region=mdb.models['Model-1'].rootAssembly.sets['Set-1'])
78. mdb.models['Model-1'].rootAssembly.Set(faces= mdb.models['Model-
    1'].rootAssembly.instances['cylinder-1'].faces.getSequenceFromMask( ('[#3 ]', ), ),
    name='Set-2')
79. mdb.models['Model-1'].ContactProperty('fric')
80. mdb.models['Model-1'].interactionProperties['fric'].TangentialBehavior( dependencies=0,
    directionality=ISOTROPIC, elasticSlipStiffness=None, formulation=PENALTY,
    fraction=0.005, maximumElasticSlip=FRACTION, pressureDependency=OFF,
    shearStressLimit=None, slipRateDependency=OFF,
    table=((0.3, ), ), temperatureDependency=OFF)
81. mdb.models['Model-1'].ContactExp(createStepName='Step-1', name='Int-1')
82. mdb.models['Model-1'].interactions['Int-1'].includedPairs.setValuesInStep(
    stepName='Step-1', useAllstar=ON)
83. mdb.models['Model1'].interactions['Int1'].contactPropertyAssignments.appendInStep(
    assignments=((GLOBAL, SELF, 'fric'), ), stepName='Step-1')
84. mdb.models['Model-1'].Velocity(distributionType=MAGNITUDE, field="", name=
    'Predefined Field-1', omega=0.0, region= mdb.models['Model-1'].rootAssembly.sets['Set-
    2'], velocity1=0.0, velocity2= 0.0, velocity3=-3.0)

```

#### Part 5

```

85. Xstring=[-0.25,-0.25,-0.25,-0.25, 0.0 ,0.0 ,0.0 ,0.0, 0.25, 0.25, 0.25, 0.25, 0,0]
86. Ystring=[-0.375, -0.125, 0.125, 0.375, -0.375, -0.125, 0.125, 0.375, -0.375, -0.125, 0.125,
    0.375, 0,0 ]
87. g=9.81
88. for L in range (1,14):

```

```

89.     mdb.models['Model-1'].rootAssembly.translate(instanceList=('cylinder-1', ),
        vector=(Xstring[L-1], Ystring[L-1], 0.0))
90.     for j in range(1,11):
91.         Jobname ='Job_%s_Location_%s' % (j,L)
92.         h=(.5*(j-1))+.5
93.         velocity=-sqrt(2*g*h)
94.         mdb.models['Model-1'].predefinedFields['Predefined Field-1'].setValues(omega=
        0.0, velocity1=0.0, velocity2=0.0, velocity3=velocity)
95.         mdb.Job(activateLoadBalancing=False, atTime=None,
        contactPrint=OFF, description="", echoPrint=OFF, explicitPrecision=SINGLE,
        historyPrint=OFF, model='Model-1', modelPrint=OFF,
        multiprocessingMode=DEFAULT, name=Jobname, nodalOutputPrecision=SINGLE,
        numCpus=1, numDomains=1, parallelizationMethodExplicit=DOMAIN, queue=None,
        scratch="", type=ANALYSIS, userSubroutine="", waitHours=0, waitMinutes=0)
96.         mdb.jobs[Jobname].submit(consistencyChecking=OFF)
97.         mdb.jobs[Jobname].waitForCompletion()

```

## Part 6

```

        # Open the output database and display a
        # default contour plot.
98. openOdb(path=Jobname + '.odb')
99.     myViewport = session.Viewport(name='myviewport', origin=(0, 0), width=250,
        height=135)
100.         myOdb = visualization.openOdb(path=Jobname + '.odb')
101.         myViewport.setValues(displayedObject=myOdb)
102.         myViewport.odbDisplay.display.setValues(plotState=
        CONTOURS_ON_DEF)

```

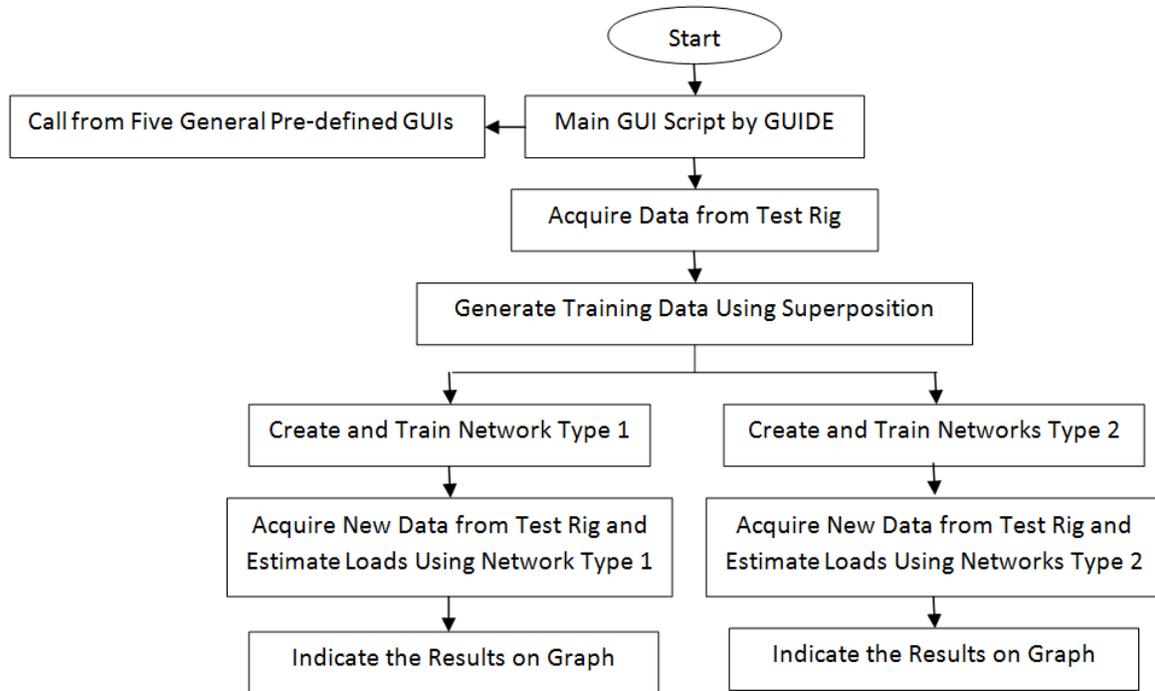
```

103.         myViewport.odbDisplay.commonOptions.setValues(
           renderStyle=FILLED)
104.         mystring=[5,7, 8, 6 , 1, 3, 4 , 2, 68, 66, 65, 67, 64, 62, 61, 63 ]
105.         outputFile = open('disp_U_'+Jobname + '.txt','w')
106.         outputFile.write('Node\tmagnitude\tU[1]\t\tU[2]\t\tU[3]\n')
107.         for i in range(0,16):
108.             m=mystring[i]
109.             for k in range(0,100):
110.                 v=myOdb.steps['Step-1'].frames[k].fieldOutputs['U'].values[m]
111.                 outputFile.write ('%d\t%.6e\t%.6e\t%.6e\t%.6e\n' % (v.nodeLabel,
           v.magnitude,v.data[0], v.data[1], v.data[2] ))
112.             outputFile.close()
113.             outputFile = open('Strain_NE_'+Jobname + '.txt','w')
114.             outputFile.write
           ('Element\tmaxPrincipal\tmidPrincipal\tminPrincipal\tEE[11]\t\tEE[22]\t\tEE[33]\t\tEE[1
           2]\t\tEE[13]\t\tEE[23]\n')
115.             for i in range(0,16):
116.                 m=mystring[i]
117.                 for k in range(0,100):
118.                     v=myOdb.steps['Step-1'].frames[j].fieldOutputs['NE'].values[i]
119.
           outputFile.write('%d\t%.6e\t%.6e\t%.6e\t%.6e\t%.6e\t%.6e\t%.6e\t%.6e\t%.6e\n' %
           (v.elementLabel,v.maxPrincipal,v.midPrincipal,v.minPrincipal ,v.data[0], v.data[1],
           v.data[2], v.data[3], v.data[4], v.data[5] ))
120.             outputFile.close()
121.             mdb.models['Model-1'].rootAssembly.translate(instanceList=('cylinder-1', ),
           vector=(-Xstring[L-1], -Ystring[L-1], 0.0))

```

### **Appendix 3: GUI Development Description**

As was mentioned previously in Chapter 2, MATLAB NNT can be used in different ways such as directly from MATLAB basic command-line operations as well as predefined GUIs. Alternatively, in order to access the full functionality of various MATLAB toolboxes as well as MATLAB NNT all at the same time, it is possible to employ the capabilities of M-file scripts using MATLAB programming language to write one customised programme for a particular project. Throughout this project, MATLAB has been widely used in order to acquire data directly from data acquisition systems, for pre-processing and analysing the data as well as ANN training and estimation of various data sets. Furthermore, a GUI is developed in MATLAB allowing control of various parameters of the data acquisition and load monitoring system, as well as graphical display in real time. All the necessary steps in developing a load monitoring system are gathered in one user friendly MATLAB program through a GUI. This allows the operator to modify and set the necessary information in the program according to the specific test experiment setup. In this section, the MATLAB program is described in brief. Figure 1-1 indicates a flow diagram of this code. For clarification purposes, this script is divided in several parts and can be accessed in Appendix 4. In addition, in order to introduce MATLAB NNT, a training example is described in Appendix 5.



**Figure 5-1: Flowchart of the MATLAB GUI**

The first part of this script has functions that appear when the MATLAB GUIDE generates the code after designing the GUI. One function is the first initialisation code generated by MATLAB for the GUI and should not be edited. The next function executes just before Prediction is made visible in the GUI. The Outputs from the last function of this part are returned to the command line. Once the user runs the script, the main GUI window will be illustrated and the user can press any button to run the desired function. The main GUI window is indicated in Figure 1-2. The main GUI has several push buttons which pressing any of them calls its call back function. In addition an axis type element is also added to the GUI. This enables the graphical display of the results (loads) in this window (no. 6 in ). Push buttons are sorted in order at both sides of this diagram. The six buttons at the right side are all related to the functions specifically developed for this project experiment enabling a load monitoring system. The other five buttons at the left side are all for using the pre-defined GUIs of the MATLAB NNT directly from this main GUI. Although the GUI is designed particularly for linear cases (small displacement experiment), it is possible to use it for training networks with nonlinear data (large displacement and drop test experiments).

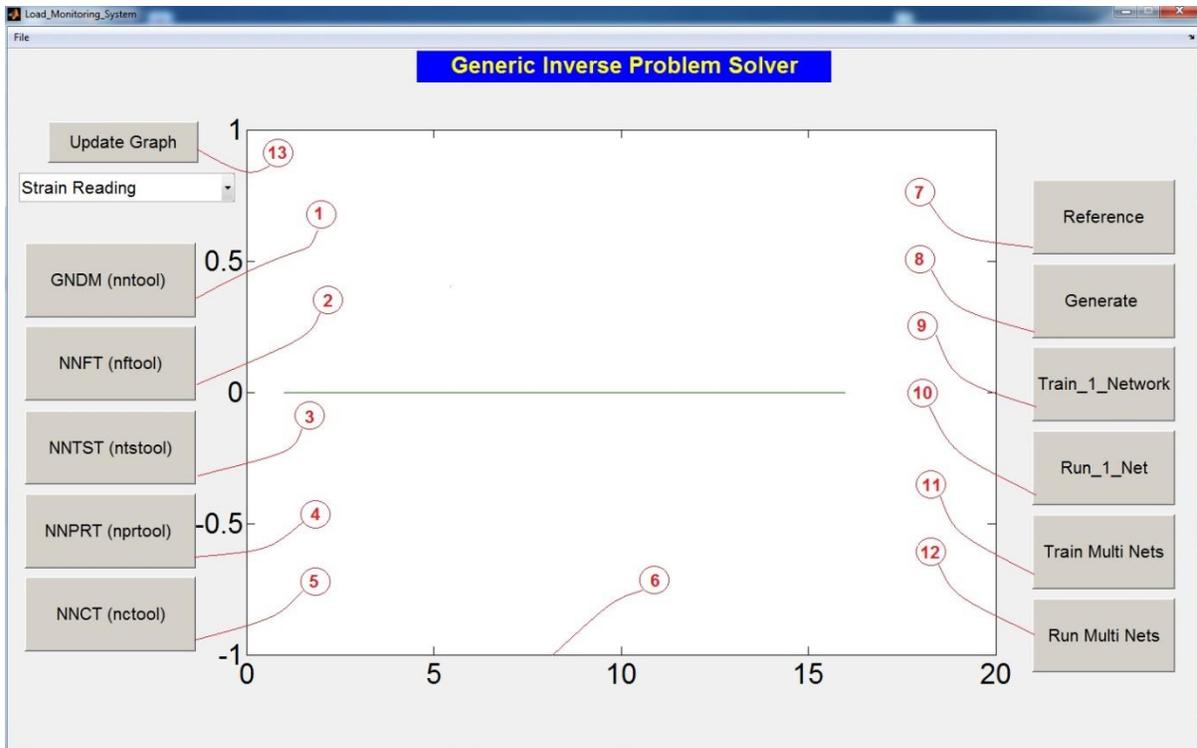


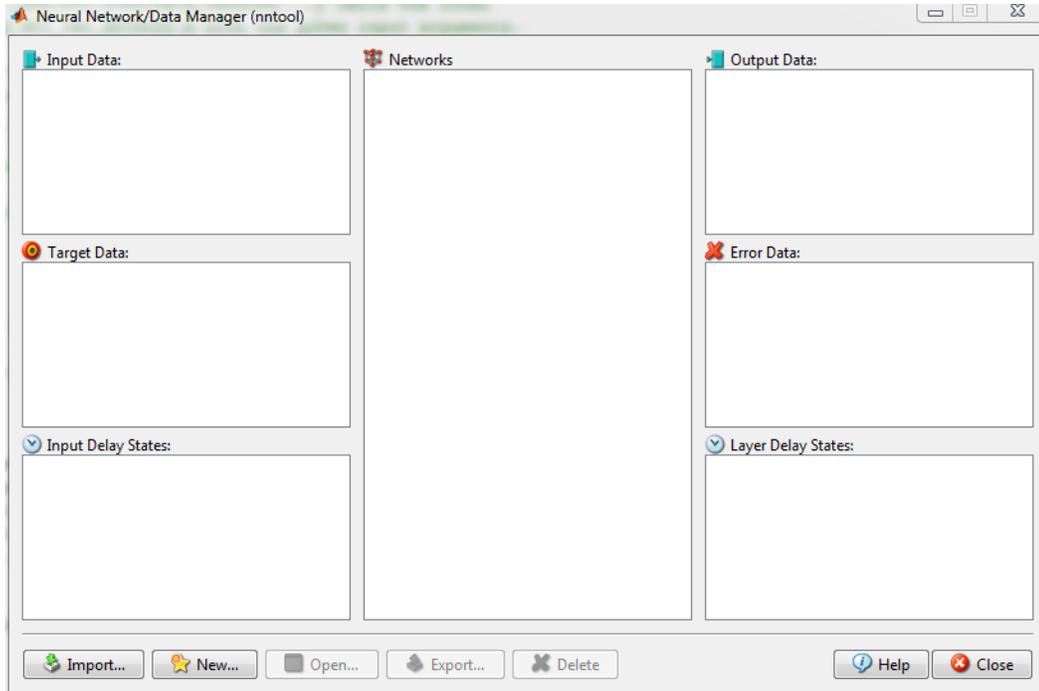
Figure 1-2: Developed MATLAB Programme main GUI

The second part of the script includes several functions which appear when MATLAB GUIDE generates the code after designing the GUI as well. They are for file, open, print and close Menu options in the designed GUI.

Part 3 of the script allows the user to easily call any of the main five general pre-defined GUIs of the MATLAB NNT and employ their full functionality straight from this GUI. All the five push buttons at the left side of the main GUI indicated in are defined in part 3 in the following order: The main GUIs are:

1. General Network Data Manager (*nntool* button no. 1 in Figure 1-2)

*nntool* command opens the Network/Data Manager window, which allows to import, create, use, and export all sorts of neural networks and data available in MATLAB (Figure 1-3).



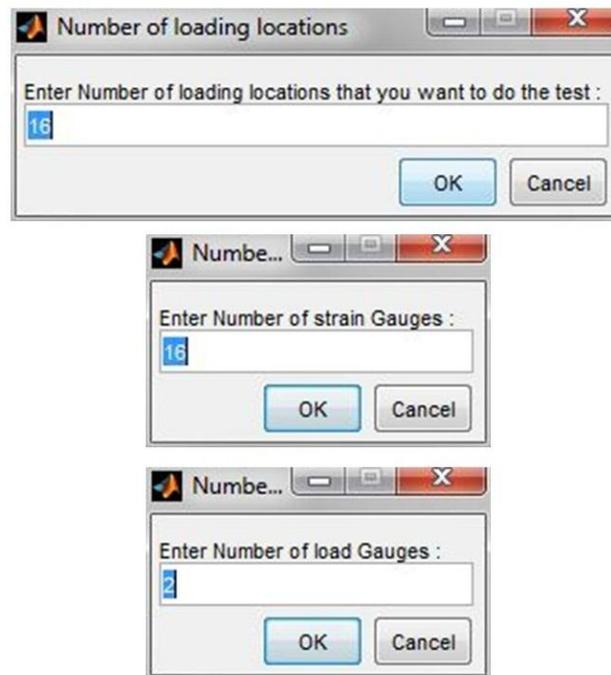
**Figure 1-3: General Network Data Manager (nntool) GUI**

2. Neural Network Fitting Tool (*nftool* button no. 2 in Figure 1-2)
3. Neural network time series tool (*ntstool* button no. 3 in Figure 1-2)
4. Neural network pattern recognition tool (*nprtool* button no. 4 in Figure 1-2)
5. Neural network classification or clustering tool (*nctool* button no. 5 in Figure 1-2)

In the part 4 of the M-file script, a function is defined in order to acquire data using MATLAB data acquisition toolbox directly into MATLAB from NI DAQ systems. Whenever this function is called back, structural responses are acquired from NI driver software through windows dynamic data exchange protocols. Depending on the DAQ system used to acquire data for each experiment either this function is used or the function defined in part 5. In part of the M-file script, a function is defined in order to acquire data using windows dynamic data exchange requests from WINDMILL Logger software (the commercial package for MICROLINK 751 devices). It should be noted that to have successful data acquisition from this function, Windmill Logger should be set up already and be running at the same time as MATLAB file execution.

These functions are called in the program whenever an experimental data acquisition is needed.

In the part 6 of the M-file script, a call back button is defined which executes training data acquisition (button no. 7 in Figure 1-1). The first step in data acquisition is to configure the number of loading locations, strain and load gauges used in the experiment (Figure 1-4). These GUIs are defined in part 7 of M-file script. Once the system gets the inputs from the user, a loop is used to get data from all the strain gauges for particular number of loading locations. Several functions are called back in this part and at the end the acquired data is saved and exported to MATLAB workspace.



**Figure 1-4: number of loading locations, strain and load gauges configuration GUIs**

The load monitoring system is designed in the way that for each loading location, two separate data set readings from all strain gauges are collected from the structure. One is when the structure is unloaded and the other is when it is only loaded at that specific loading location. This is defined in a function in part 8 of M-file script. Once both readings are acquired the system asks the user to verify and check that these data acquisitions are performed accurately. Should the user be happy about the procedure, the yes button is selected and the differences in two data set readings are used to indicate the deflections at the certain strain gauge locations due to that specific applied load. However, should the user be not sure about the consistency of the data

acquisition process, no button is selected and the same procedure would be repeated until the user is pleased with the procedure. The checking function is defined in part 9 of M-file script. Figure 1-5 indicates sample GUIs designed for this stage when strain readings are collected for loading location number one. Once deflections of the systems due to the one applied load at the time to the first loading location are achieved, the same procedure applies to the next loading location.

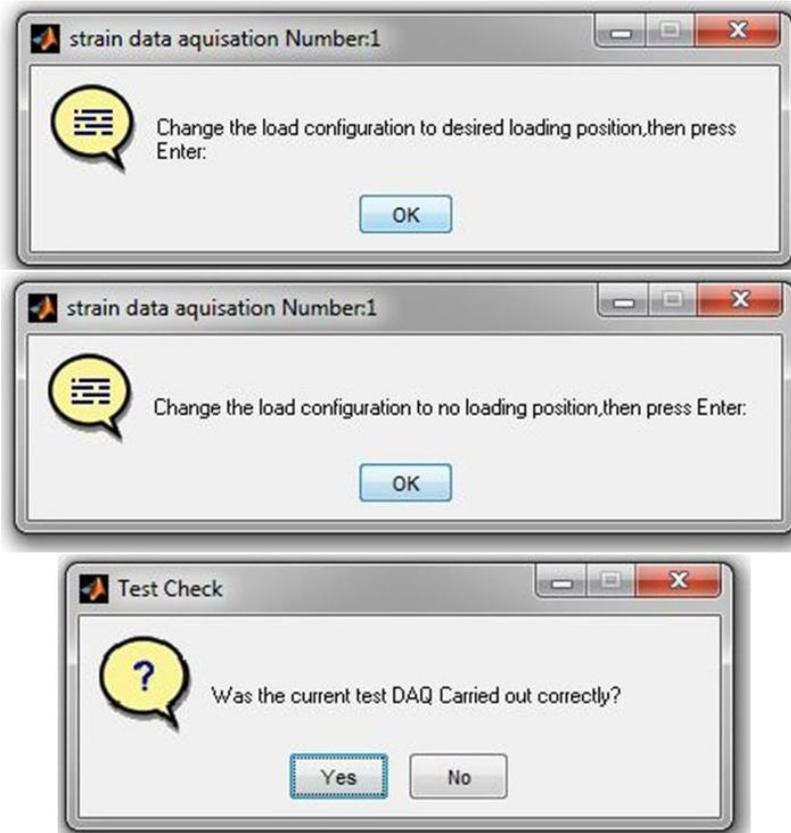


Figure 1-5: Strain data acquisition No.1 GUIs

The loop in part 7 continues till all the loading locations are covered. At the end of part 7 when all the locations are loaded at once and the strain readings for each loading location is collected, data sets are put in order in matrices, saved and should the user be willing, data sets can be exported to the MATLAB workspace (Figure 1-6). Strain data sets and load data sets are put together in one matrix (original reference data matrix) which is used later to generate more data.

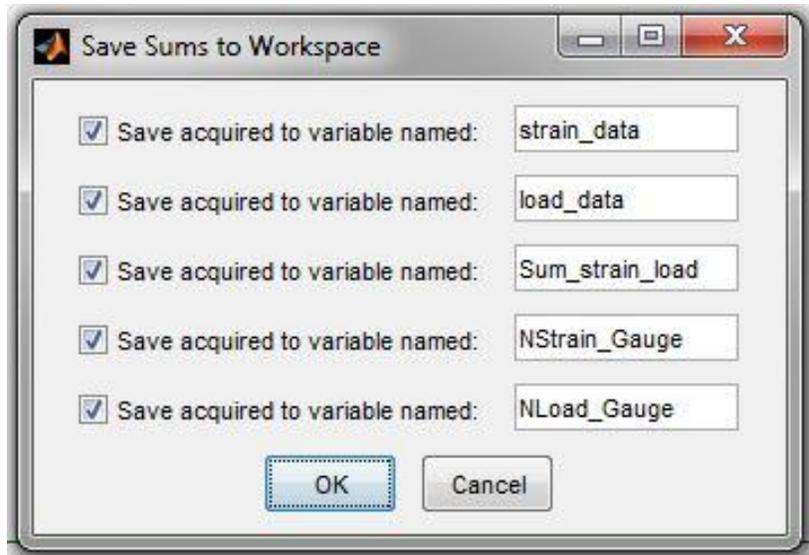
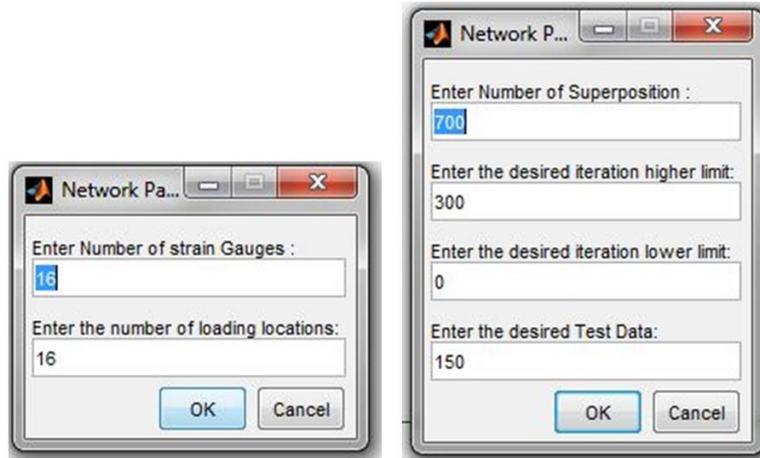
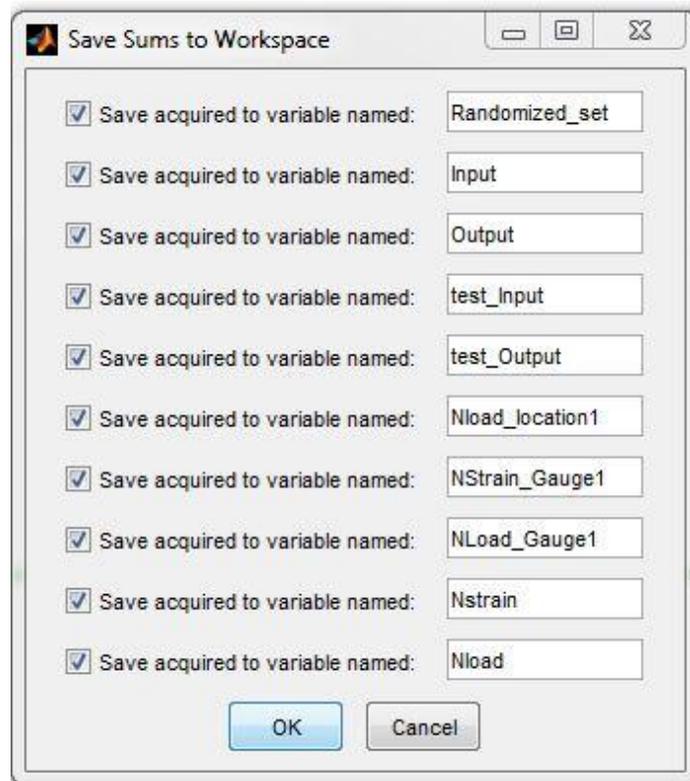


Figure 1-6: Export the acquired data to the MATLAB workspace

Once the original reference data matrix is built employing the experimental data acquired from the structure, for a linear system it can be used to generate enough training data using the superposition theorem. In the part 10 of M-file script, a call back button is defined which executes training data generation algorithm (button no. 8 in Figure 1-2). The algorithm is based on superposition theorem and uses a random generation function defined in the part 11 to generate random loading case scenarios having random numbers in the range lower limit to higher limit of the linear loading response defined by the user. Training data generation parameters are configured first by the user employing GUIs indicated in Figure 1-7. Training data generated with this script consist of two parts. One part has loading scenarios where all the locations are loaded with random loads and the other part has loading scenarios where only one loading location is loaded with random load values. All the generated data sets are put in specific order in various matrices, saved and should the user be willing, data sets can be exported to the MATLAB workspace (Figure 1-8).



**Figure 1-7: Training data generation parameter configuration GUIs**



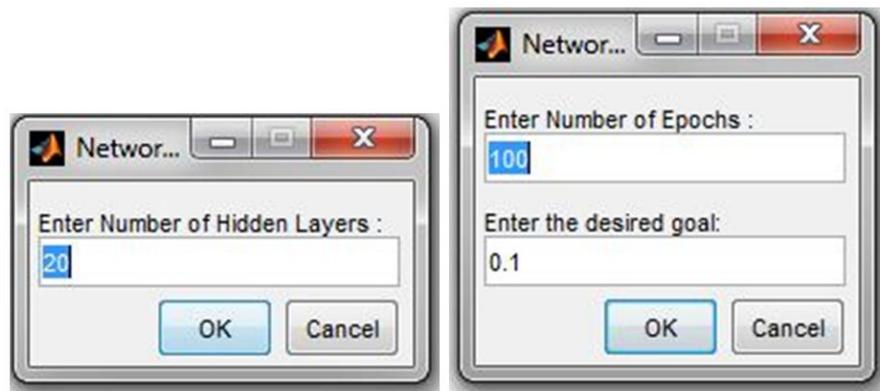
**Figure 1-8: Export the generated data to the MATLAB workspace**

When training data is ready, it is time to create an ANN and train it. In this research two different back-propagation ANN architectures are used to relate structural response to the applied loads. The architectures utilised are:

- 1) One network with Nstrain (number of strain gauges) neurons in the input layer and Nload (number of loading locations) neurons in the output layer is trained to estimate the load on the panel from the strain responses.
- 2) Nload networks each with Nstrain neurons in the input layer and one neuron in output layer are trained and used to estimate the load on the panel from the strain responses.

Other network parameters such as the number of hidden layers and neurons in each hidden layer are flexible and can be configured by the user according to the data sets.

In part 12 of this M-file script, first ANN architecture is used to create and train the network. The user first selects a call back button designed to execute training with one network (button no. 9 in Figure 1-2) then configures the network parameters using GUIs indicated in Figure 1-9.



**Figure 1-9: Network Parameter Configuration GUIs for ANN architecture 1**

This part of the script will then create a network according to these network parameters. The network will be trained with the input and target data sets of the training data. During the training process, the neural network training tool GUI will be displayed. The user will be able to monitor the training process. This window indicates the state of the training progress in terms of training time and epochs, the network performance, the magnitude of the gradient of performance and the number of validation checks. These values are of most interest to monitor during the training progress. This is due to the fact that the user may use the early stopping method to achieve a more general network. The training process could be stopped at any time of the progress by clicking the Stop Training button in the training window. This is usually done when the performance function fails to decrease considerably over many training iterations. From the

Neural Network Training GUI, three main plots can be accessed as well for post-processing: performance, training state and regression plots. Figure 1-10 indicates the neural network training tool GUI.

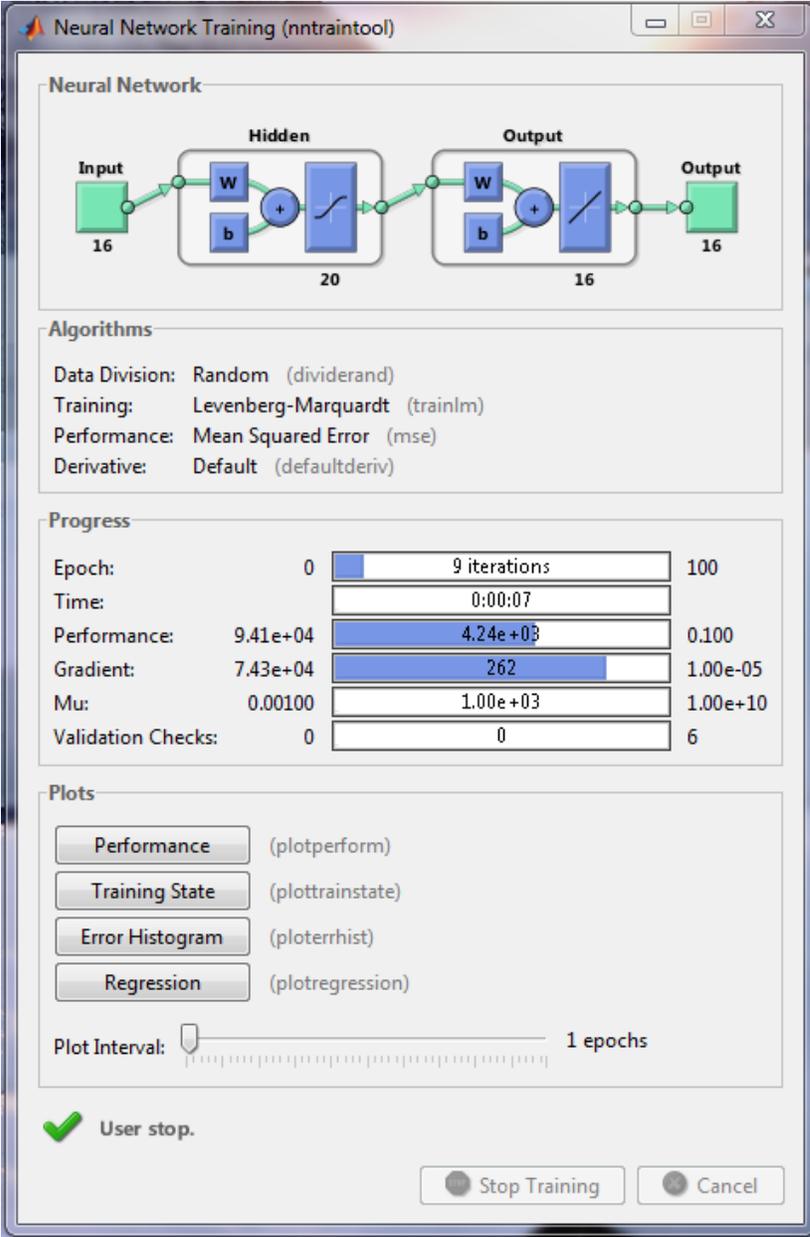


Figure 1-10: Neural network Training tool

Once the training is done, the network and the main parameters of the training state will be saved and should the user be willing, data sets can be exported to the MATLAB workspace (Figure 1-11).

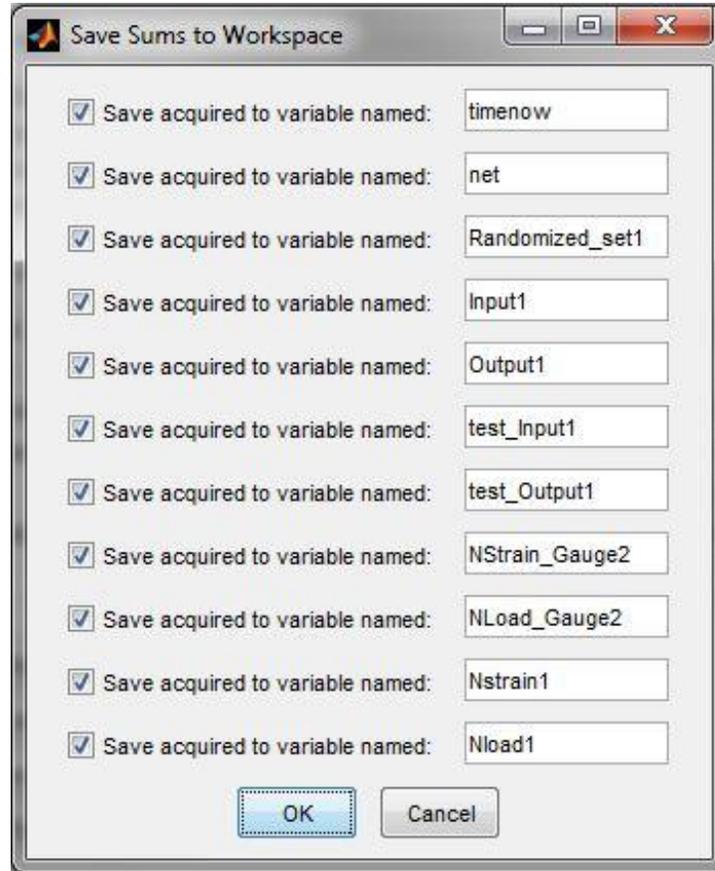


Figure 1-11: Export the network and its main parameters to the MATLAB workspace

Having a well-trained network, the ANN should be able to estimate the external loads from new introduced structure response data sets. In the part 13 of the M-file script, a call back button is defined which executes new load estimation using the trained network having the first ANN architecture (button no. 10 in Figure 1-2). By pressing this button a new set of data is acquired and introduced to a trained network. The data acquisition would be performed once only in two parts again. One is when the structure is unloaded and the other is when it is only loaded at desired specific loading locations. The GUIs used to guide through the data acquisition process are indicated in Figure 1-5. The network will simulate and predict load from this new set of data. The loads are then indicated as a bar graph in the main GUI (Figure 1-12). Once the load

estimation is done, the network output will be saved and should the user be willing, data sets can be exported to the MATLAB workspace (Figure 1-13).

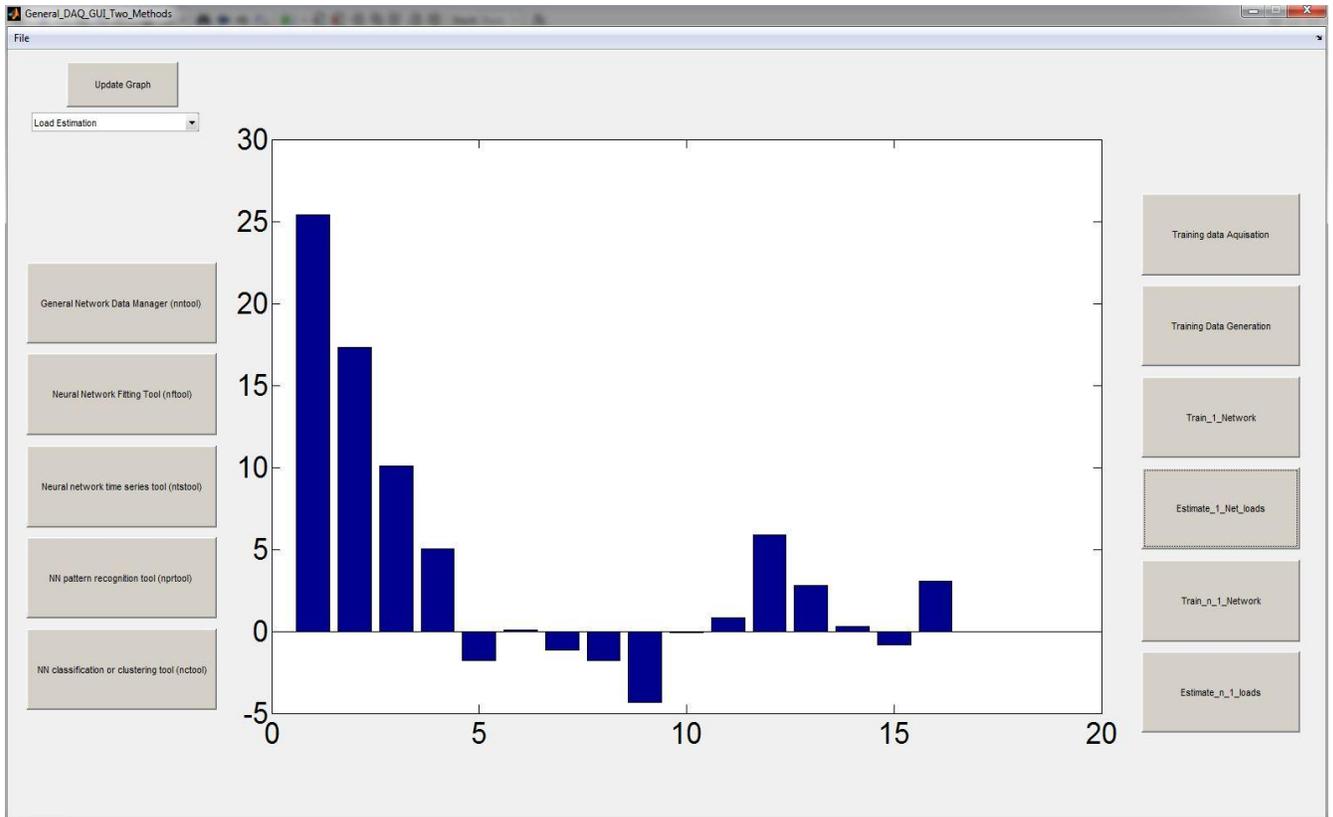


Figure 1-12: Load estimation results indicated as a bar chart

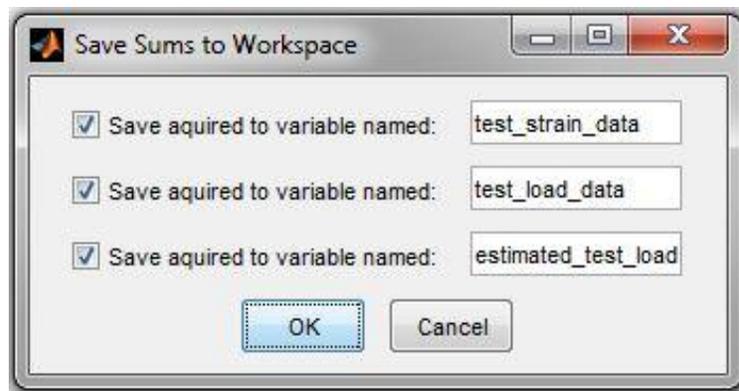


Figure 1-13: Export the network output to the MATLAB workspace

In part 14 of this M-file script, the second ANN architecture is used to create and train the network. A call back button should be selected to execute training with Nload networks (button

no. 11 in Figure 1-2) which then configures the number of network hidden layers and neurons in each hidden layer using the GUI indicated in Figure 1-14.



**Figure 1-14: Nload networks hidden layer and neuron number configuration**

This part of the script will then create Nload target data sets from the training target data set using a loop and Nload networks according to the network parameters are created. Each of the networks will be trained with the input and one of the target data sets created earlier from the training data. During the training process, the neural network training tool GUI will be displayed for each of the networks and the user will be able to monitor the training processes. Once the training is done, the networks and the main parameters of the training states will be saved.

Having well-trained networks, the ANN should be able to estimate the external loads from new introduced structure response data sets. In the part 15 of the M-file script, a call back button is defined which executes new load estimation using the trained network having the second ANN architecture (button no. 12 in Figure 1-2). By pressing this button a new set of data is acquired and introduced to a trained network. The data acquisition would be performed once only in two parts again. One is when the structure is unloaded and the other is when it is only loaded at desired specific loading locations. The GUIs used to guide through the data acquisition process are indicated in . The networks will simulate and estimate loads from this new set of data. Each network will estimate a load which is related to one location only. However, the results of all the networks together provide the estimation of all loading locations from the same strain data set input. The loads are then indicated as a bar graph in the main GUI. Once the load estimation is done, the network output will be saved and should the user be willing, data sets can be exported to the MATLAB workspace (Figure 1-3).

Finally, in part 16 of this the M-file script, a call back button is defined which updates the main GUI graph based on its popup menu (button no. 13 in Figure 1-2). The user should first select from the menu between new strain reading and load estimation. Once the update button is selected, there will be a new data acquisition and the strain readings are used to update the graph.

Throughout this project, MATLAB has been widely used in order to acquire data directly from data acquisition systems, for pre-processing and analysing the data as well as ANN training and estimation of various data sets, the main developed MATLAB program was described in this chapter. In fact, the capabilities of M-file script using MATLAB programming language is employed to write one customised programme for this particular project. As described in detail in this chapter, all the necessary steps in developing a load monitoring system are gathered in one user friendly MATLAB program through a GUI which allows the operator to modify and set the necessary information in the program according to the specific test experiment setup. Furthermore, the user can control various parameters of the data acquisition and load monitoring system and visualise data in real time. In the next chapter, a summary and evaluation of the topics and results of this thesis as well as possible future research and recommendations are covered.

## Appendix 4: MATLAB Script

### Part 1

```
function varargout = General_DAQ_GUI_Two_Methods(varargin)
% GENERAL_DAQ_GUI_TWO_METHODS MATLAB code for
General_DAQ_GUI_Two_Methods.fig
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',  gui_Singleton, ...
                  'gui_OpeningFcn', @General_DAQ_GUI_Two_Methods_OpeningFcn, ...
                  'gui_OutputFcn',  @General_DAQ_GUI_Two_Methods_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

%{
This function appears when MATLAB GUIDE generates the code after designing the GUI.
It is the first initialization code generated by MATLAB for GUI.
%}

% End initialization code - DO NOT EDIT
% --- Executes just before General_DAQ_GUI_Two_Methods is made visible.
function General_DAQ_GUI_Two_Methods_OpeningFcn(hObject, eventdata,
handles, varargin)
% Choose default command line output for General_DAQ_GUI_Two_Methods
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% This sets up the initial plot - only do when we are invisible
% so window can get raised using General_DAQ_GUI_Two_Methods.
%load('C:\Users\Mohammad\Desktop\Hosptest1\Train_leg.mat')
if strcmp(get(hObject,'Visible'),'off')
    plot(zeros(16));
    end
%{ This function appears when MATLAB GUIDE generates the code after designing the GUI.
It executes just before Prediction is made visible in GUI.%}

% --- Outputs from this function are returned to the command line.
function varargout = General_DAQ_GUI_Two_Methods_OutputFcn(hObject,
eventdata, handles)
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
%{ This function appears when MATLAB GUIDE generates the code after designing the GUI.
Outputs from this function are returned to the command line.% }
```

## Part 2

```
% -----
function FileMenu_Callback(hObject, eventdata, handles)
```

```
    %{ This function appears when MATLAB GUIDE generates the code after designing the GUI.
It is for the File Menu option in the designed GUI.
% }
```

```
% -----
function OpenMenuItem_Callback(hObject, eventdata, handles)
% hObject handle to OpenMenuItem (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
file = uigetfile('*.fig');
if ~isequal(file, 0)
    open(file);
end
uiopen('LOAD');
```

```
    %{ This function appears when MATLAB GUIDE generates the code after designing the GUI.
It is for the Open Menu option in the designed GUI.% }
```

```
% -----
function PrintMenuItem_Callback(hObject, eventdata, handles)
printdlg(handles.figure1)
```

```
%{ This function appears when MATLAB GUIDE generates the code after designing the GUI.
It is for the Print option in the designed GUI.% }
```

```
% -----
function CloseMenuItem_Callback(hObject, eventdata, handles)
selection = questdlg(['Close ' get(handles.figure1, 'Name') '?'], ...
                    ['Close ' get(handles.figure1, 'Name') '...'], ...
                    'Yes', 'No', 'Yes');
```

```
if strcmp(selection, 'No')
    return;
end
delete(handles.figure1)
%{ This function appears when MATLAB GUIDE generates the code after designing the GUI.
It is for the close option in the designed GUI.% }
```

### Part 3

```
% --- Executes on button press in nntool.
function nntool_Callback(hObject, eventdata, handles)
nntool
% --- Executes on button press in nftool.
function nftool_Callback(hObject, eventdata, handles)
nftool
% --- Executes on button press in ntstool.
function ntstool_Callback(hObject, eventdata, handles)
ntstool
% --- Executes on button press in nprtool.
function nprtool_Callback(hObject, eventdata, handles)
nprtool
% --- Executes on button press in nctool.
function nctool_Callback(hObject, eventdata, handles)
nctool
%{ This function appears when MATLAB GUIDE generates the code after designing the GUI.
```

It is for predefined MATLAB NNT GUI options access in the designed GUI.

The first function will be executed on button press in of nntool in GUI and will open predefined GUIs for nntool function. The second function will be executed on button press in of nftool in GUI and will open predefined GUIs for nftool function. The third function will be executed on button press in of ntstool in GUI and will open predefined GUIs for ntstool function. The third function will be executed on button press in of ntstool in GUI and will open predefined GUIs for ntstool function. The fourth function will be executed on button press in of nprtool in GUI and will open predefined GUIs for nprtool function. The fifth function will be executed on button press in of nctool in GUI and will open predefined GUIs for nctool function.% }

### Part 4

```
function [data]=data_session
DAQ_Logger_Set_No=2;
DAQ_Logger_Channel_No=8;
s = daq.createSession('ni');
s.Rate=10000;
for j=1:DAQ_Logger_Set_No
for i=0:DAQ_Logger_Channel_No-1
S=['ch',num2str(i),'=s.addAnalogInputChannel('cDAQ1Mod',num2str(j),'',
'ai',num2str(i),'', 'Bridge'));'];
eval(S);
S=['ch',num2str(i),'.ADCTimingMode = 'HighSpeed';'];
eval(S);
S=['ch',num2str(i),'.BridgeMode = 'Quarter';'];
eval(S);
S=['ch',num2str(i),'.NominalBridgeResistance=350;'];
eval(S);
end
end
data = 1000000*s.startForeground;
data=mean(data,1);
```

```

checkLabels = {'Save acquired to variable named:'};
varNames = {'Signal_1'};
items = {data};
export2wsdlg(checkLabels,varNames,items,...
            'Save Sums to Workspace');
s.release()

```

%{ This function is defined in order to acquire data using MATLAB data acquisition toolbox directly into MATLAB from NI DAQ systems. Whenever this function is called back, structural responses are acquired from NI driver software through windows dynamic data exchange .

Note: In order to have successful data acquisition NI drivers should be installed set up already %}

## Part 5

```

function [DAQ]=DAQ_Logger
    DAQ_Logger_Set_No=2;
    DAQ_Logger_Channel_No=15;
    channel = ddeinit('Logger','Data');
    for j=0:DAQ_Logger_Set_No-1
        for i=0:DAQ_Logger_Channel_No-1
            S=['DAQ(1,j*(DAQ_Logger_Channel_No)+i+1) =
ddereq(channel,','',num2str(j),'000',num2str(i),'')'];
            eval(S);
        end
    end
end

```

%{ This function is written by the author. Whenever this function is called back through windows dynamic data exchange request s, structural responses are acquired from WINDMILL Logger software (the commercial package for MICROLINK 751 devices).

Note: In order to have successful data acquisition Windmill Logger should be set up already and be running at the same time as MATLAB file execution.%}

## Part 6

```

% --- Executes on button press in Training Data Acquisition.
function Training_data_Aquisition_Callback(hObject, eventdata, handles)
strain_data_Null=data_session;
data=Training_data_acquisition;
    %{ This function appears when MATLAB GUIDE generates the code after designing the GUI.
Whatever written in this function will be executed on button press in Training Data Acquisition in GUI.%}

```

## Part 7

```

function [strain_data,load_data,NStrain_Gauge,Nload_location]=
Training_data_acquisition
input_answer{1,1}='1';

```

```

prompt = {'Enter Number of loading locations that you want to do the test
: '};
dlg_title = 'Number of loading locations ';
num_lines = 1;
def = {'16'};
input_answer = inputdlg(prompt,dlg_title,num_lines,def);
Nload_location=str2num(input_answer{1,1});
input_answer{1,1}='1';
prompt = {'Enter Number of strain Gauges : '};
dlg_title = 'Number of strain Gauges ';
num_lines = 1;
def = {'16'};
input_answer = inputdlg(prompt,dlg_title,num_lines,def);
NStrain_Gauge=str2num(input_answer{1,1})
input_answer{1,1}='1';
prompt = {'Enter Number of load Gauges : '};
dlg_title = 'Number of Load Gauge ';
num_lines = 1;
def = {'2'};
input_answer = inputdlg(prompt,dlg_title,num_lines,def);
NLoad_Gauge=str2num(input_answer{1,1})
for i=1:Nload_location
strain_data_all(i,:)=get_one_set_data(i,NStrain_Gauge,NLoad_Gauge);
end
size(strain_data_all)
strain_data=strain_data_all(:,1:NStrain_Gauge);
size(strain_data)
load_data=strain_data_all(:,NStrain_Gauge+1:NStrain_Gauge+NLoad_Gauge);
size(load_data)
Sum_strain_load=strain_data;
for i=1:NLoad_Gauge
Sum_strain_load(:,NStrain_Gauge+i)=load_data(:,i);
end
Sum_strain_load
checkLabels = {'Save acquired to variable named:', 'Save acquired to
variable named:', 'Save acquired to variable named:', 'Save acquired to
variable named:', 'Save acquired to variable named:'};
varNames =
{'strain_data', 'load_data', 'Sum_strain_load', 'NStrain_Gauge', 'NLoad_Gauge'};
items = {strain_data, load_data, Sum_strain_load, NStrain_Gauge, NLoad_Gauge};
export2wsdlg(checkLabels, varNames, items, ...
'Save Sums to Workspace');
name=[num2str(NStrain_Gauge) '_Strain_Gauge_'
num2str(Nload_location) '_loading_location_' num2str(NLoad_Gauge)
'_Load_Gauge' ];
save name 'name'
save(name, 'strain_data', 'load_data', 'Sum_strain_load',
'NStrain_Gauge', 'NLoad_Gauge', 'Nload_location', 'name');

```

## Part 8

```

function
[strain_data]=get_one_set_data(Ndata_location,NStrain_Gauge,NLoad_Gauge)

```

```

    S=strcat('strain data aquisition Number: ', num2str(Ndata_location));
    helpdlg1=helpdlg('Change the load configuration to no loading
position,then press Enter: ',S);
    waitfor(helpdlg1);
    strain_data_Null=data_session;
    helpdlg2=helpdlg('Change the load configuration to desired loading
position,then press Enter:',S);
    waitfor(helpdlg2);
    strain_data=data_session-strain_data_Null;
    checking(Ndata_location)

```

## Part 9

```

function checking(Ndata_location)
choice = questdlg('Was the current test DAQ Carried out correctly?', ...
    'Test Check', 'Yes','No','Yes');
switch choice
    case 'Yes'
        case 'No'
            msgbox1= msgbox('Do the same loading test again');
            waitfor(msgbox1);
            get_one_set_data(Ndata_location)
    end
end

```

## Part 10

```

% --- Executes on button press in Training_Data_Generation.
function Training_Data_Generation_Callback(hObject, eventdata, handles)
    prompt = {'Enter Number of strain Gauges :','Enter the number of loading
locations:'};
    dlg_title = 'Training Data generation Parameters';
    num_lines = 1;
    def = {'16','16'};
    input_answer = inputdlg(prompt,dlg_title,num_lines,def);

    Nstrain=str2num(input_answer{1,1})
    Nload=str2num(input_answer{2,1});
    Data_Generation(Nstrain,Nload)
    %{ This function appears when MATLAB GUIDE generates the code after designing the GUI.

    Whatever written in this function will be executed on button press in Training Data Generation in GUI.%}

function Data_Generation(Nstrain,Nload)
    prompt = {'Enter Number of Superposition :','Enter the desired iteration
higher limit:','Enter the desired iteration lower limit:','Enter the desired
Test Data:'};
    dlg_title = 'Training Data generation Parameters';
    num_lines = 1;
    def = {'700','300','0', '150'};
    input_answer = inputdlg(prompt,dlg_title,num_lines,def);
    NSuperposition=str2num(input_answer{1,1})
    Niteration_higher_limit=str2num(input_answer{2,1});

```

```

Niteration_lower_limit=str2num(input_answer{3,1});
NTest_data=str2num(input_answer{4,1})
load('C:\Users\Mohammad\Desktop\Matlab Final\name.mat');
S=['C:\Users\Mohammad\Desktop\Matlab Final\' name '.mat']
load(S)
%the random number in the range lower limit to higher limit for the first
part
Sum=strain_data;
    for i=1:Nload
        Sum(Nstrain+i,i)=load_data(i,1);
    end

Sum
for i=1:Nload
Sum(:,i)=Sum(:,i)/Sum(Nstrain+i,i);
end
% the random number in the range Niteration_lower_limit to
Niteration_higher_limit for the first part
% First part consist of NSuperposition data (set initially 700) which is
% summation of all the Nload data sets (Nload +Nstrain in Nload coloumns
% where each coloumn is randomized and then all the coumn are added
% together to demonstrate the randome cases for when all the locations are
loaded at the same time with random loads
Temp_Column_NEW_SUM=zeros(Nstrain+Nload,NSuperposition);
for k=1:(NSuperposition)
NEW_SUM=randomized_data(Nload,Niteration_lower_limit,Niteration_higher_limit,
Sum);
    for j=1:Nload
        Temp_Column_NEW_SUM(:,k)=Temp_Column_NEW_SUM(:,k)+NEW_SUM(:,j);
    end
end
Randomized_set=Temp_Column_NEW_SUM;
%the random number in the range Niteration_lower_limit to
Niteration_higher_limit for the Second part
%Second part consists of round(NSuperposition/Nload)* Nload coloumns which
%is one random load at a time on each location
for k=1:round(NSuperposition/Nload)

NEW_SUM=randomized_data(Nload,Niteration_lower_limit,Niteration_higher_limit,
Sum);
tempsum(:, :, k)=NEW_SUM(:, :);
end
for k=1:round(NSuperposition/Nload)
    for i=1:Nload
        Randomized_set(:,NSuperposition+i-Nload+(k*Nload))=tempsum(:, i, k);
    end
end

for i=1:Nstrain
    Input(i,:)= Randomized_set(i,:);
end
Input;
for i=Nstrain+1:Nstrain+Nload
    for j=1:(round(NSuperposition/Nload)*Nload)+NSuperposition
        Output(i-Nstrain,j)= Randomized_set(i,j);
    end
end

```

```

    end
end
Output;
% This part is the test data where NTest_data * Nload data sets with
% order is generated this means that each loading location is loaded from 1
% to NTest_data individually and data sets are generated
for k=1:NTest_data
    for i=1:Nstrain+Nload
Temp_test_data(i,:,k)=k*Sum(i,:);
    end
end
for k=1:150
    for i=1:Nstrain+Nload
        for j=1:Nload
            test_data(i,((k-1)*Nload)+j)=Temp_test_data(i,j,k);
        end
    end
end

for i=1:Nstrain
test_Input(i,:)=test_data(i,:);
end

for i=Nstrain+1:Nstrain+Nload
test_Output(i-Nstrain,:)=test_data(i,:);
end
name2=['General_data_' num2str(NStrain_Gauge) '_Strain_Gauge_'
num2str(Nload_location) '_loading_location_' num2str(NLoad_Gauge)
'_Load_Gauge' ];
    save name2 'name2'
    save(name2, 'Randomized_set' , 'Input', 'Output', 'test_Input',
'test_Output', 'NStrain_Gauge', 'NLoad_Gauge', 'Nstrain',
'Nload','name','Nload_location');

    checkLabels = {'Save acquired to variable named:', 'Save acquired to
variable named:', 'Save acquired to variable named:', 'Save acquired to
variable named:', 'Save acquired to variable named:', 'Save acquired to
variable named:', 'Save acquired to variable named:', 'Save acquired to
variable named:'};
    varNames = {'Randomized_set', 'Input', 'Output', 'test_Input',
'test_Output', 'Nload_location', 'NStrain_Gauge', 'NLoad_Gauge', 'Nstrain', 'Nload
'};
    items = {Randomized_set, Input, Output, test_Input,
test_Output, Nload_location, NStrain_Gauge, NLoad_Gauge, Nstrain, Nload};
    export2wsdlg(checkLabels, varNames, items, ...
        'Save Sums to Workspace');

```

## Part 11

```

function[NEW_SUM]
=randomized_data(Nload, Niteration_lower_limit, Niteration_higher_limit, Sum)

```

```

Rand= Niteration_lower_limit + (Niteration_higher_limit-
Niteration_lower_limit).*rand(1,Nload);
for j=1:Nload
NEW_SUM(:,j)=Rand(1,j)*Sum(:,j);
end

```

## Part 12

```

% --- Executes on button press in Train Network.
function Train_1_Network_Callback(hObject, eventdata, handles)
net=Train_1_Network;
%{

```

This function appears when MATLAB GUIDE generates the code after designing the GUI.

Whatever written in this function will be executed on button press in Train 1 Network in GUI.

```

%}

function [net]=Train_1_Network
load('C:\Users\Mohammad\Desktop\Matlab Final\name2.mat');
S=['C:\Users\Mohammad\Desktop\Matlab Final\' name2 '.mat'];
load(S)

prompt = {'Enter Number of Hidden Layers :'};
dlg_title = 'Network Parameters ';
num_lines = 1;
def = {'20'};
input_answer = inputdlg(prompt,dlg_title,num_lines,def);
Nhidden_lay=str2num(input_answer{1,1});

net = feedforwardnet(Nhidden_lay);

prompt = {'Enter Number of Epochs :','Enter the desired goal:'};
dlg_title = 'Network Parameters ';
num_lines = 1;
def = {'100','0.1'};
input_answer = inputdlg(prompt,dlg_title,num_lines,def);
net.trainParam.epochs=str2num(input_answer{1,1})
net.trainParam.goal=str2num(input_answer{2,1})
%net.trainParam.goal=0.1;
%net.trainParam.epochs=100;
net = train(net,Input,Output);
timenow=datestr(clock);
filename=strcat('Randomized_numbers_',date,'_',timenow(13:14),'_',timenow(1
6:17),'_',timenow(19:20));
name3=['General_net_data_' num2str(NStrain_Gauge) '_Strain_Gauge_'
num2str(Nload_location) '_loading_location_' num2str(NLoad_Gauge)
'_Load_Gauge' ];
save name3 'name3'
save(name3, 'timenow', 'net' ,'Randomized_set' , 'Input',
'Output', 'test_Input', 'test_Output', 'NStrain_Gauge', 'NLoad_Gauge',
'Nstrain', 'Nload','name');

```

```

    checkLabels = {'Save acquired to variable named:', 'Save acquired to
variable named:', 'Save acquired to variable named:', 'Save acquired to
variable named:', 'Save acquired to variable named:', 'Save acquired to
variable named:', 'Save acquired to variable named:', 'Save acquired to
variable named:', 'Save acquired to variable named:'};
    varNames = {'timenow', 'net', 'Randomized_set', 'Input', 'Output', 'test_Input',
'test_Output', 'NStrain_Gauge', 'NLoad_Gauge', 'Nstrain', 'Nload'};

    items = {timenow, net, Randomized_set, Input, Output, test_Input,
test_Output, NStrain_Gauge, NLoad_Gauge, Nstrain, Nload};
    export2wsdlg(checkLabels, varNames, items, ...
                'Save Sums to Workspace');

```

### Part 13

%{ This function appears when MATLAB GUIDE generates the code after designing the GUI.

Whatever written in this function will be executed on button press in Estimate 1 Net loads in GUI.

By pressing this button a new set of data is acquired and introduced to a trained network. The network will simulate and predict load from this new set of data. The loads are then indicated as a bar graph in GUI. At the end there will be the option to save key data such as: 'timenow', 'Strain\_reading\_average', 'test\_output' and 'first\_reading\_average'.%

```

% --- Executes on button press in Estimate Load.
function Estimate_1_Net_loads_Callback(hObject, eventdata, handles)
load('C:\Users\Mohammad\Desktop\Matlab Final\name3.mat');
S=['C:\Users\Mohammad\Desktop\Matlab Final\' name3 '.mat']
load(S)
location=5;
strain_data_all(1,:)=get_one_set_data(1,NStrain_Gauge,NLoad_Gauge);
strain_data=strain_data_all(:,1:NStrain_Gauge)';
load_data=zeros(Nstrain,1);
load_data(location,1)=strain_data_all(:,NStrain_Gauge+1)';

    checkLabels = {'Save aquired to variable named:', 'Save aquired to variable
named:', 'Save aquired to variable named:'};
    varNames = {'test_strain_data', 'test_load_data', 'estimated_test_load'};
    estimated_test_load=sim(net, strain_data);
    bar(estimated_test_load);
    items = {strain_data, load_data, estimated_test_load};
    export2wsdlg(checkLabels, varNames, items, ...
                'Save Sums to Workspace');

```

### Part 14

```

% --- Executes on button press in Train_n_1_Network.
function Train_n_1_Network_Callback(hObject, eventdata, handles)
mean_nets_mse=Train_n_1_Network;
function [mean_nets_mse]=Train_n_1_Network
load('C:\Users\Mohammad\Desktop\Matlab Final\name2.mat');
S=['C:\Users\Mohammad\Desktop\Matlab Final\' name2 '.mat'];
load(S)
prompt = {'Enter Number of Hidden Layers :'};

```

```

dlg_title = 'Network Parameters ';
num_lines = 1;
def = {'50'};
input_answer = inputdlg(prompt,dlg_title,num_lines,def);
Nhidden_lay=str2num(input_answer{1,1});
N_temp=size(Output);
Nload=N_temp(1,1);
N_temp=size(Input);
Nstrain=N_temp(1,1);
for i=1:Nstrain
s=strcat('Output_',num2str(i),'=Output(i,:);')
eval(s)
s=strcat('net_',num2str(i),' = feedforwardnet([' ,num2str(Nhidden_lay),
']);')
eval(s)
s=strcat('[net_',num2str(i),' ,tr_',num2str(i),' ] =
train(net_',num2str(i),' ,Input,Output_',num2str(i),' );')
eval(s)
end
for i=1:Nstrain
s=strcat('net_esstimation(i,:)=sim(net_',num2str(i),' ,Input);')
eval(s)
s=strcat('nets_mse(1,i)=mse(net_',num2str(i),' ,Output_',num2str(i),' ,net_es
stimation(i,:) );')
eval(s)
end
error_net_esstimation= net_esstimation - Output;
mean_nets_mse=mean(nets_mse)
name4=['General_n_1_net_data_' num2str(NStrain_Gauge) '_Strain_Gauge_'
num2str(Nload_location) '_loading_location_' num2str(NLoad_Gauge)
'_Load_Gauge' ];
save name4 'name4'
for i=1:Nstrain

S= ['netdata.net_',num2str(i),'=net_',num2str(i),';'];
eval(S)
S= ['netdata.tr_',num2str(i),'=tr_',num2str(i),';'];
eval(S)
save(name4, 'netdata'
,'error_net_esstimation','nets_mse','net_esstimation','Randomized_set' ,
'Input', 'Output', 'test_Input', 'test_Output', 'NStrain_Gauge',
'NLoad_Gauge', 'Nstrain', 'Nload','name4');
end

```

## Part 15

```

% --- Executes on button press in Estimate_n_1_Net_loads.
function Estimate_n_1_Net_loads_Callback(hObject, eventdata, handles)
load('C:\Users\Mohammad\Desktop\Matlab Final\name4.mat');
S=['C:\Users\Mohammad\Desktop\Matlab Final\' name4 '.mat'];
load(S)
location=5;
strain_data_all(1,:)=get_one_set_data(1,NStrain_Gauge,NLoad_Gauge)

```

```

size(strain_data_all)
Input=strain_data_all(:,1:NStrain_Gauge)';
Output=zeros(Nstrain,1);
Output(location,1)=strain_data_all(:,NStrain_Gauge+1)';
for i=1:Nstrain
s=strcat('net_esstimation(i,:)=sim(netdata.net_',num2str(i),'',Input);')
eval(s)
end
checkLabels = {'Save aquired to variable named:', 'Save aquired to variable
named:', 'Save aquired to variable named:'};
varNames = {'test_strain_data', 'test_load_data', 'estimated_test_load'};
bar(net_esstimation);
items = {Input,Output,net_esstimation};
export2wsdlg(checkLabels,varNames,items,...
'Save Sums to Workspace');

```

## Part 16

```

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end
set(hObject, 'String', { 'Strain Reading', 'Load Estimation (Bar)', });
% --- Executes on button press in Update_Graph.
function Update_Graph_Callback(hObject, eventdata, handles)
axes(handles.axes1);
cla;
load('C:\Users\Mohammad\Desktop\Matlab Final\name3.mat');
S=['C:\Users\Mohammad\Desktop\Matlab Final\' name3 '.mat']
load(S)
strain_data_all(1,:)=get_one_set_data(1,NStrain_Gauge,NLoad_Gauge);
strain_data=strain_data_all(:,1:NStrain_Gauge)';
load_data=strain_data_all(:,NStrain_Gauge+1:NLoad_Gauge)';
estimatedload=sim(net,strain_data);
for i=1:NStrain_Gauge
if estimatedload(i,1)<=0
estimatedload(i,1)=0;
end
end
popup_sel_index = get(handles.popupmenu1, 'Value');
switch popup_sel_index
case 1
plot(strain_data);
case 2
bar(estimatedload);
end
end

```

## Appendix 5: **Introduction to MATLAB NNT through a Training Example**

MATLAB NNT can be used in different ways. First is to employ predefined GUIs designed for many problems of pattern recognition, clustering, function fitting and time series analysis.

The second is to use basic command-line operations which use simple argument with default settings for function parameters (all of the default settings can be accessed and overridden)

The third approach is customization of the toolbox and create own customised neural networks, while still having access to the full functionality of the toolbox. The fourth approach is to employ M-file scripts of MATLAB and access all the above ways of using toolbox at one customised programme. In this thesis, the first two methods will be described.

### **Using MATLAB NNT Predefined GUIs**

In this section, how to use MATLAB NNT predefined GUIs is briefly described through a simple example. In order to open any of the GUIs, their specific command should be written in MATLAB command line. The main GUIs are:

1. General Network Data Manager (*nntool*)

*nntool* command opens the Network/Data Manager window, which allows to import, create, use, and export all sorts of neural networks and data available in MATLAB.

2. Neural Network Fitting Tool (*nftool*)

*nftool* opens a window which leads the user step by step through solving a data fitting problem employing a two-layer feed-forward network.

3. Neural network time series tool (*ntstool*)

*ntstool* opens the neural network time series wizard and leads the user through solving a nonlinear time series problem with a dynamic neural network.

4. Neural network pattern recognition tool (*nprtool*)

*nprtool* opens the neural network pattern-recognition GUI which guides the user through solving a pattern recognition or classification problem using a two-layer feed-forward network with sigmoid output neurons

5. Neural network classification or clustering tool (*nctool*)

nctool opens the neural network clustering GUI which guides the user through solving a clustering problem using a self-organizing map.

The MATLAB NNT predefined GUIs are very straight forward and clear and lead the user through solving the problem employing various neural networks. In this thesis, the most general GUI, General Network Data Manager (*nntool*), is described through an example. For this purpose, training data set was already available and it is only needed to load it into MATLAB workspace. Data set includes strain values from 8 sensors (strain gauges) when various weights in the range of 0 to 20 Kg are applied on 2 locations. The aim is to design a network which relates these strain data to the weight values. In other words, strain values are inputs to network and weight values are targets. Training data set consists of 1396 random loading scenarios. Imported input/target pairs (examples) are in the form of two matrices: Input and Output. In this example, Input is a matrix of  $8 \times 1396$  elements (1396 columns each has 8 strain readings) and Output is a matrix of  $2 \times 1396$  elements (1396 columns each has 2 weight values).

As it was stated earlier, in order to call a particular GUI, its command (*nntool*) should be written in MATLAB command line. The first window of General Network Data Manager GUI will be opened (Figure 1-15). In this window, the user can import, export MATLAB workspace data (Figure 1-16 and Figure 1-17) or alternatively use open option to load data from hardware. Once necessary data are loaded to the GUI, the user should create the network using. Selecting create new option leads the user to a new window (Figure 1-18) where a new network or data set can be created. In the Network index of this window, a new network can be selected and configured. Main features of a network can be customized for the new network including: type, input and output data, training and learning functions, number of layers (hidden layers), number of neurons and their type of transfer functions in each hidden layer of the network.

In this example, data should be imported first to General Network Data Manager GUI. Input data set is imported as Input and Output data set as target Data. This will categorize data for easier use later in GUI. Once the data set are imported, the user should define the specific network for the problem. In this example, a back-propagation feedforward network is selected with one hidden layer each having 10 neurons. The training and performance functions are set to be the default of the MATLAB NNT: the Levenberg-Marquardt (*trainlm*) training method and

the mean square error performance function ( $MSE$ ). In fact this GUI enables many different configurations of ANN architecture to be employed. A list of possible network types, training and performance functions are listed in Table 1-2-Table 1-3. In general, the default training function for feedforwardnet, *trainlm*, is the fastest training function in MATLAB NNT. The quasi-Newton method, *trainbfg*, is pretty fast as well. However, for large networks with thousands of weights, due to the fact that more memory and more computation time are required, both of these training functions are less efficient. Furthermore, *trainlm* indicates better performances on nonlinear regression problems than on pattern recognition problems (Hagan & Menhaj, 1999). For training very large networks, and pattern recognition problems, *trainscg* and *trainrp* are good selections. This is mostly due to their relatively small memory requirements. In addition, their algorithms are much faster than standard gradient descent algorithms (Beale, et al., 2012).

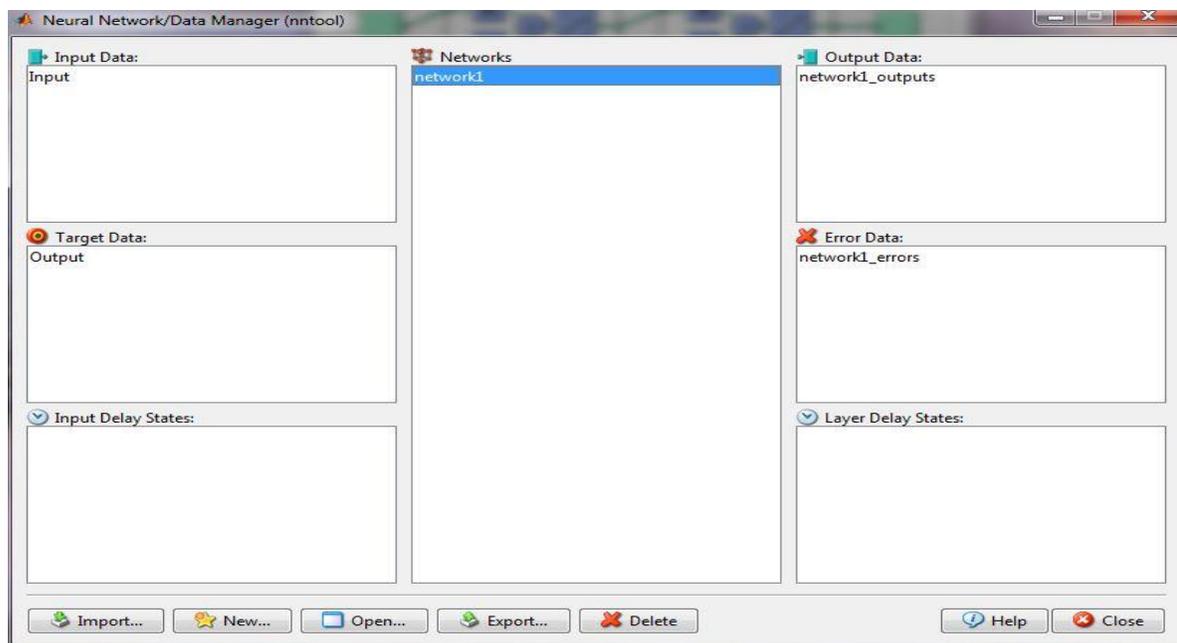


Figure 1-15: General Network Data Manager GUI

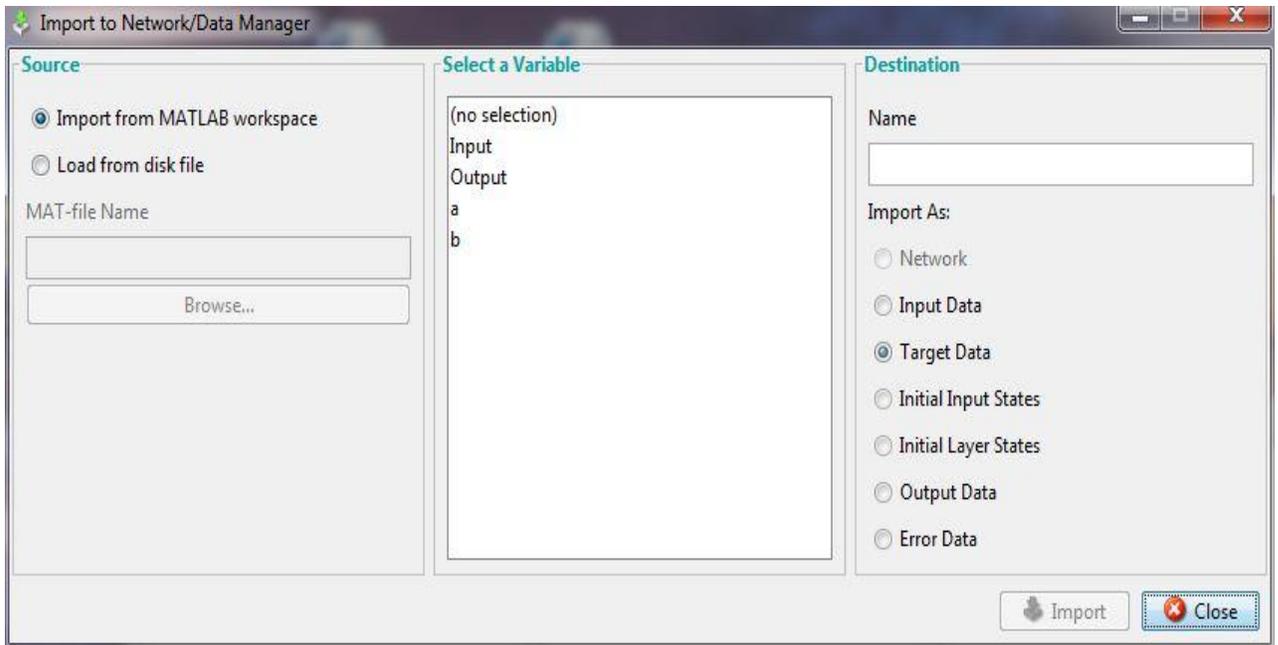


Figure 1-16: Import to Network/Data Manager GUI

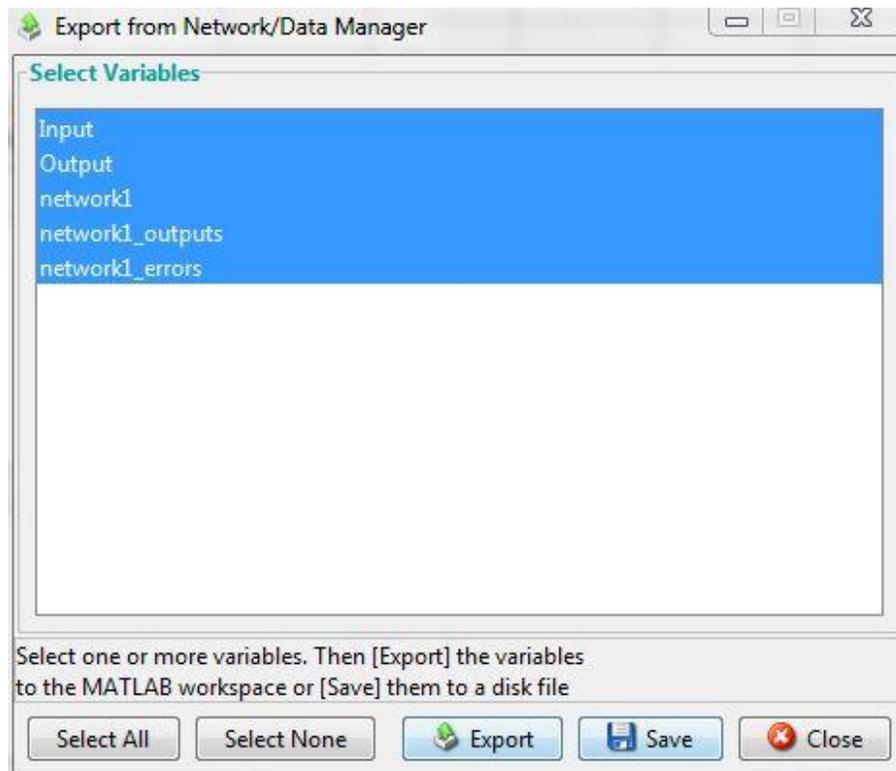


Figure 1-17: Export from Network/Data Manager GUI

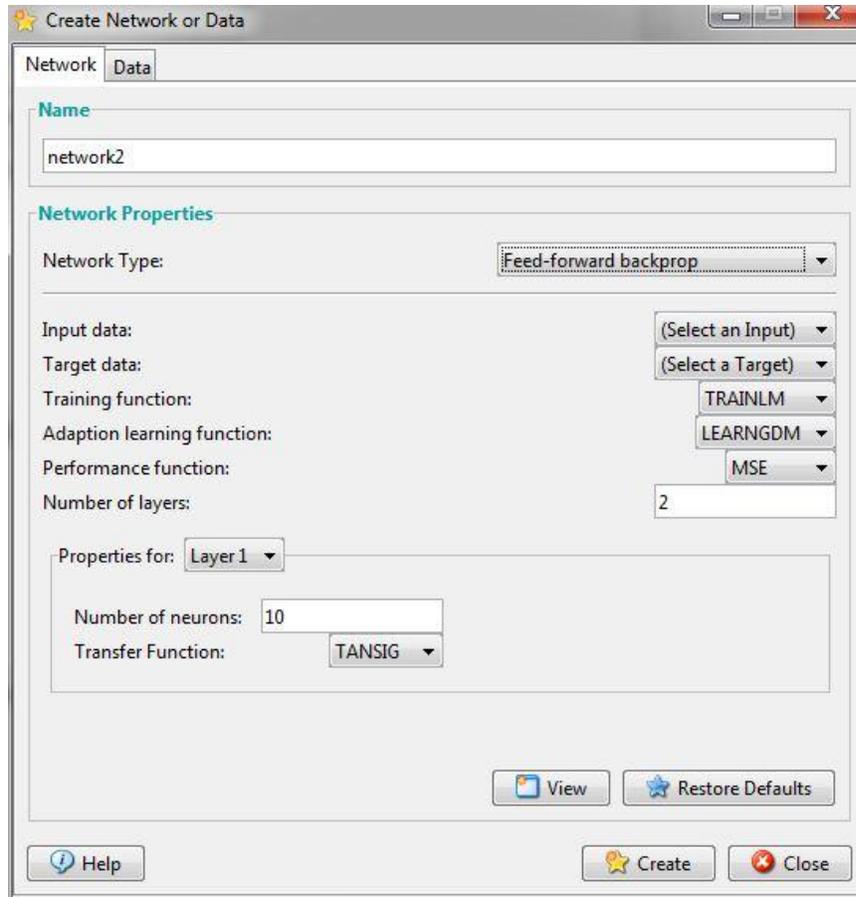


Figure 1-18: Create Network or Data GUI

Table 1-1: List of MATLAB NNT Training Functions (Beale, et al., 2012)

Training Functions	
Function	Algorithm
<i>trainlm</i>	Levenberg-Marquardt
<i>trainbr</i>	Bayesian Regularization
<i>traingd</i>	Gradient Descent
<i>trainrp</i>	Resilient Backpropagation
<i>trainscg</i>	Scaled Conjugate Gradient
<i>traincgb</i>	Conjugate Gradient with Powell/Beale Restarts
<i>traincgf</i>	Fletcher-Powell Conjugate Gradient
<i>traincgp</i>	Polak-Ribière Conjugate Gradient
<i>trainoss</i>	One Step Secant
<i>traingdx</i>	Variable Learning Rate Gradient Descent
<i>traingdm</i>	Gradient Descent with Momentum
<i>trainbfg</i>	BFGS Quasi-Newton

Table 1-2: List of MATLAB NNT Network Types (Beale, et al., 2012)

Network Types	
New Networks Functions	Description
<i>network</i>	Create custom neural network
<i>cascadeforwardnet</i>	Cascade-forward neural network
<i>competlayer</i>	Competitive neural layer
<i>distdelaynet</i>	Distributed delay neural network
<i>elmannet</i>	Elman neural network
<i>feedforwardnet</i>	Feed-forward neural network
<i>fitnet</i>	fitnet
<i>layrecnet</i>	Layer recurrent neural network
<i>linearlayer</i>	Linear neural layer
<i>lvqnet</i>	Learning vector quantization (LVQ) neural network
<i>narnet</i>	Nonlinear auto-associative time series network
<i>narxnet</i>	Nonlinear auto-associative time series network with external input
<i>newgrnn</i>	Generalized regression neural network
<i>newlind</i>	Designed linear layer
<i>newpnn</i>	Probabilistic neural network
<i>newrb</i>	Radial basis network
<i>newrbe</i>	Exact radial basis network
<i>patternnet</i>	Pattern recognition network
<i>perceptron</i>	Perceptron
<i>selforgmap</i>	Self-organizing map
<i>timedelaynet</i>	Time-delay neural network

Table 1-3: List of MATLAB NNT Performance Functions (Beale, et al., 2012)

Performance Functions	
Function	Description
<i>mae</i>	Mean absolute error performance function
<i>mse</i>	Mean squared error performance function
<i>sse</i>	Sum squared error performance function

Once network is created, MATLAB NNT automatically initiated the required number of weights and biases according to the network and its input and target data sets. Double clicking on the network in the General Network Data Manager GUI opens a new window that enables the user to employ the network. The first index of this window (Figure 1-19), View, indicates the

neural network as a graphical diagram that clearly describes network type and its number of layers. Other indexes of network GUI enable its principle functions such as training, simulation as well as weight and biases set up. As mentioned earlier there are two training styles: incremental and batch training. MATLAB NNT enables the user to train the network in either way employing specific window or functions. In the network GUI, Train index employs batch training style. The user can use the predefined parameter or override them as required (Figure 1-20 and Figure 1-21). In contrast to the Train index, the Adopt index uses incremental training style (Figure 1-22).

For this example, batch training is employed. This means that the user should use Train index options to train the network. Training Inputs/Targets are selected from imported data sets and training parameters are set to be the default values of the GUI (Figure 1-20 and Figure 1-21). Selecting the train button begins the training process. This opens the Neural Network Training GUI indicated in Figure 1-23 automatically. During training, the progress is constantly updated in the training GUI. It can be illustrated from this window that the data is divided using the *dividerand* function, and the Levenberg-Marquardt (*trainlm*) training method is employed with the mean square error performance function. With the *dividerand* setting, the input vectors and target vectors will be randomly divided, by default 70% is used for training, 15% for validation and 15% for testing.

This window also indicates the state of training progress in terms of training time and epochs, the network performance, the magnitude of the gradient of performance and the number of validation checks. These values are of most interest to monitor during the training progress. This is due to the fact that the user may use early stopping method to achieve a more general network. The training process could be stopped at any time of the progress by clicking the Stop Training button in the training window. This is usually done when the performance function fails to decrease considerably over many training iterations.

The magnitude of the training time and epochs as well as the gradient and the number of validation checks are utilised to terminate the training. The gradient will become very small as the training reaches a minimum of the performance. In this example, should the magnitude of the gradient become less than  $1e^{-5}$ , the training process will stop. The number of validation checks

denotes the number of consecutive iterations that the validation performance has not decreased. In this example the magnitude is 6 and reaching to this number stops the training process. Reaching to the magnitude of any of these limit values during the training progress stops the progress. . In this run, the training did stop because of the number of validation checks after just 40 iterations.

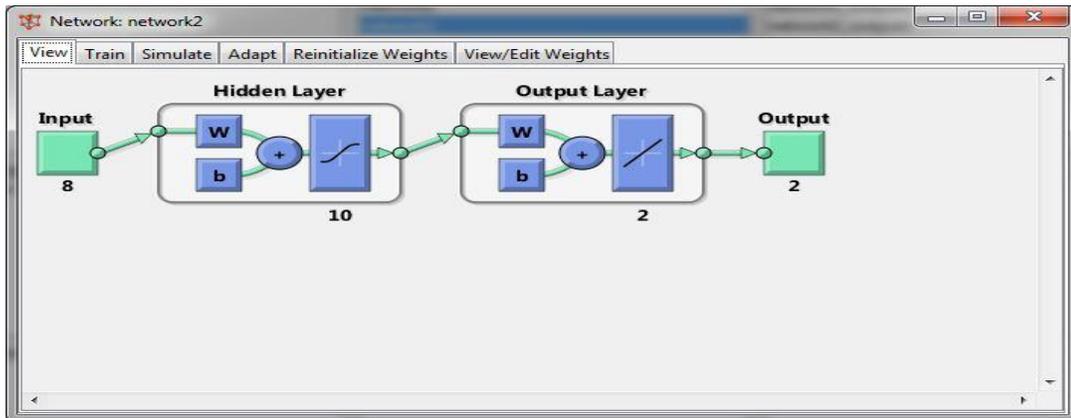


Figure 1-19: Network Architecture View

The screenshot shows the 'Train' window for 'Network: network1'. It features a menu bar with 'View', 'Train', 'Simulate', 'Adapt', 'Reinitialize Weights', and 'View/Edit Weights'. Below the menu are two tabs: 'Training Info' (selected) and 'Training Parameters'. The 'Training Info' section is divided into 'Training Data' and 'Training Results'.

Training Data		Training Results	
Inputs	Input	Outputs	network1_outputs
Targets	Output	Errors	network1_errors
Init Input Delay States	(zeros)	Final Input Delay States	network1_inputStates
Init Layer Delay States	(zeros)	Final Layer Delay States	network1_layerStates

At the bottom right of the window is a 'Train Network' button with a play icon.

Figure 1-20: Network Train window (Training Info index)

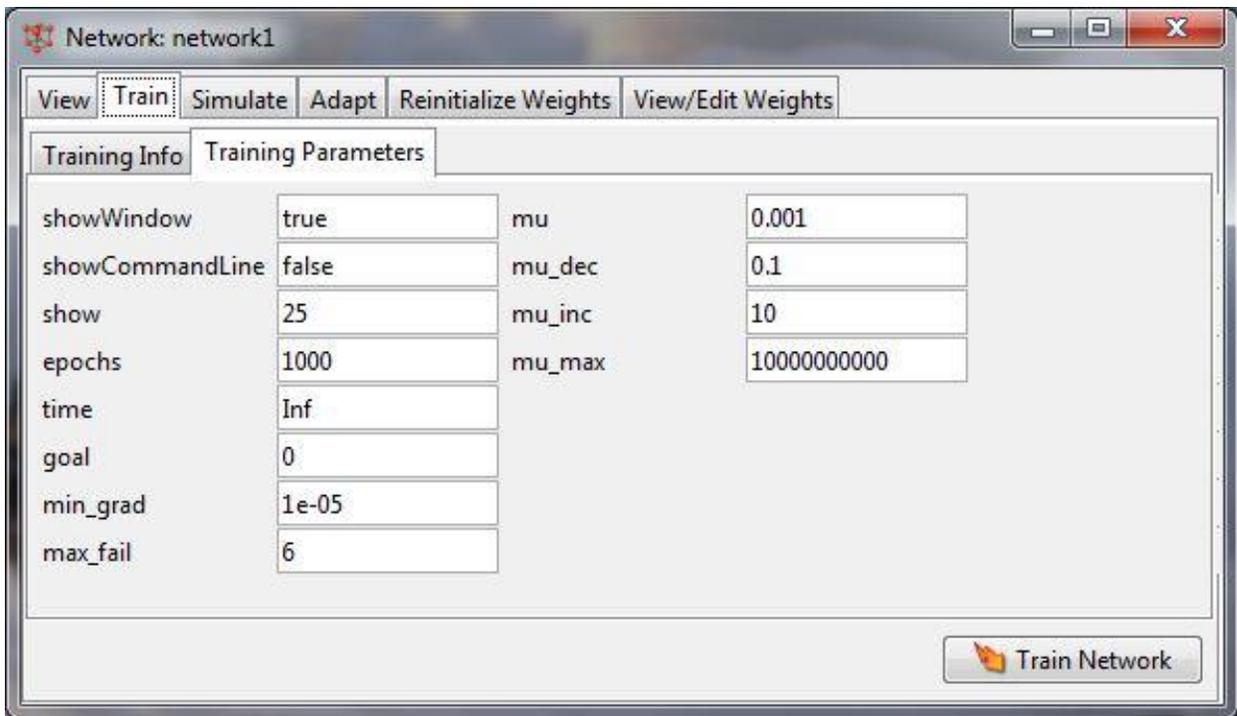


Figure 1-21: Network Train window (Training Parameters index)

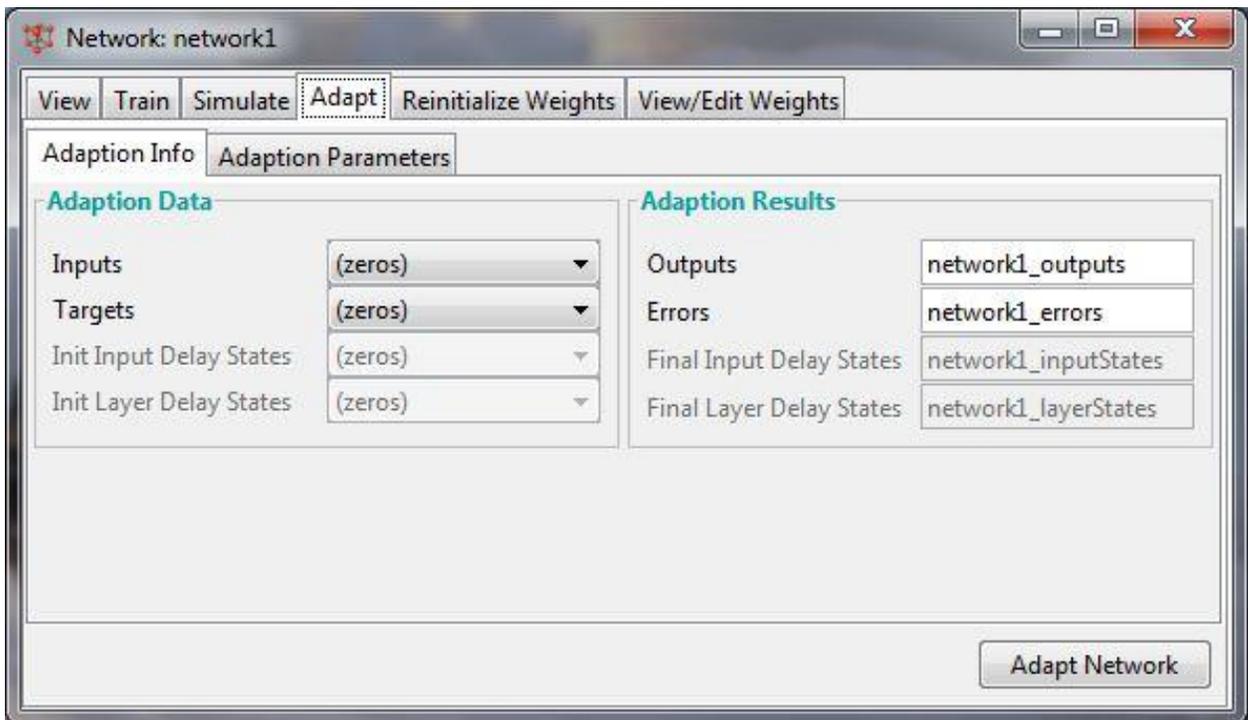


Figure 1-22: Network Adapt window (Adaption Info index)

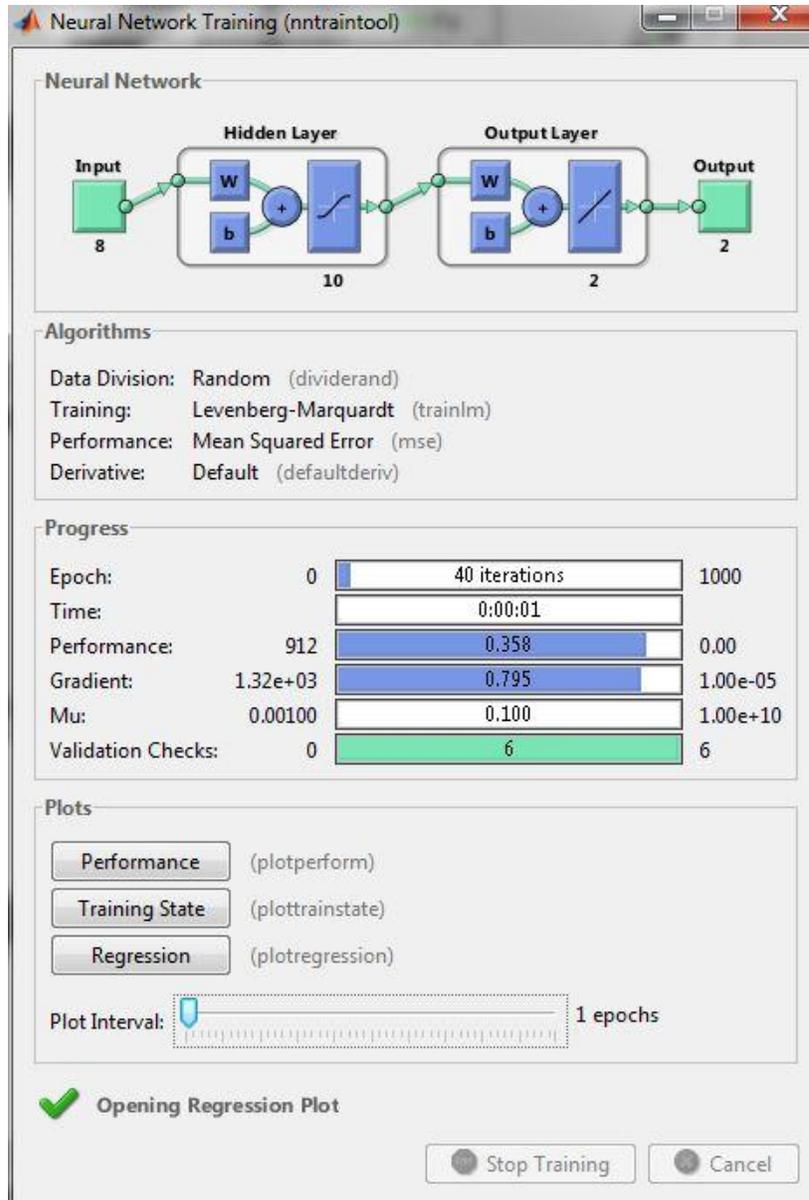
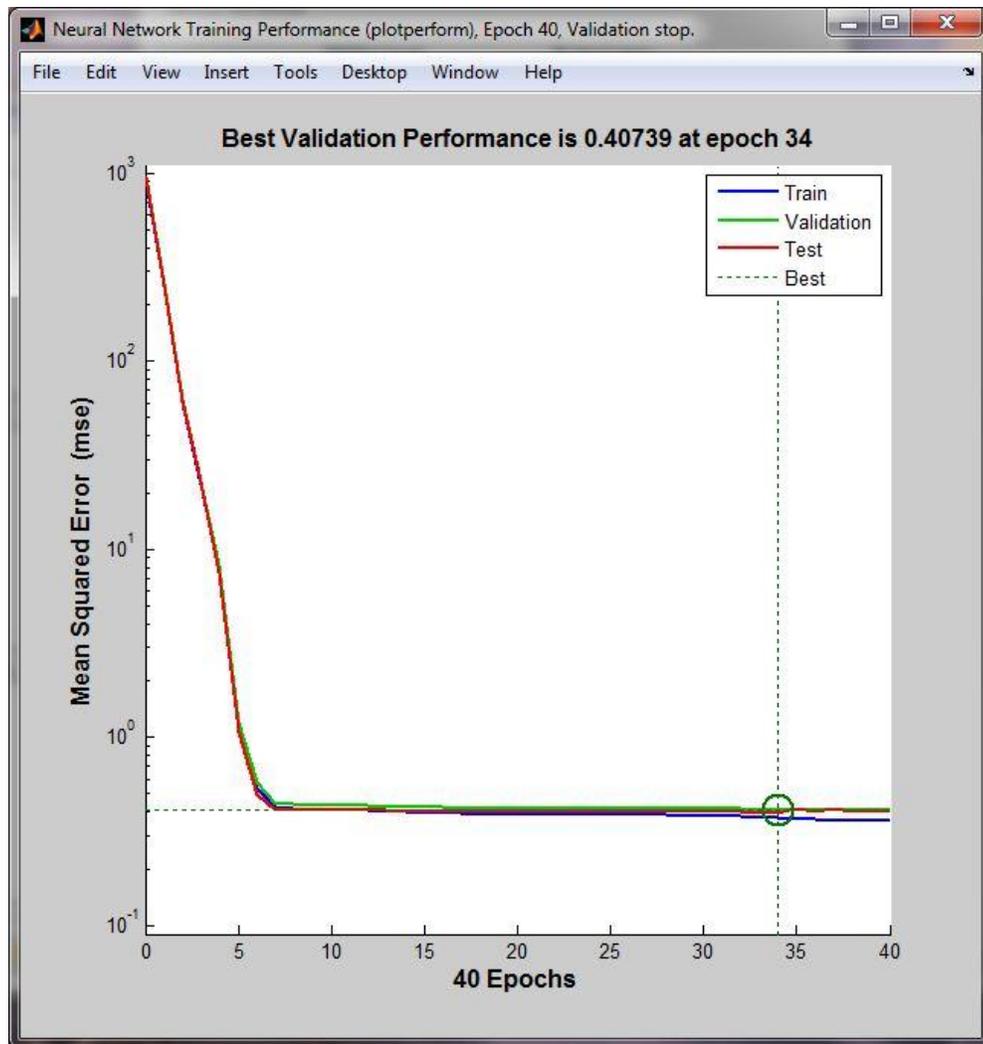


Figure 1-23: Network Network Training GUI

From the Neural Network Training GUI, three plots can be accessed: performance, training state and regression plots. The performance plot indicates the value of the performance function versus the iteration number (Figure 1-24). Training, validation and test performances diagrams are plotted simultaneously and their trends can be compared. In this example, no major problems are illustrated with the training in the performance plot. The validation and test curves are very similar.

The training state plot shows the progress of other training variables, such as the gradient magnitude, the number of validation checks, etc. The regression plot shows a regression between network outputs and network targets and is used mainly for post-processing purposes to validate network performance. The regression plot shows the relationship between the outputs of the network and the targets. The regression plot for this example is shown in the **Figure 1-25**. The three axes represent the training, validation and testing data. The solid line represents the best fit linear regression line between outputs and targets. The R value is an indication of the relationship between the outputs and targets. For this example, the training data indicates a good fit. The validation and test results also show R values that greater than 0.99.



**Figure 1-24: Network Performance Plot**

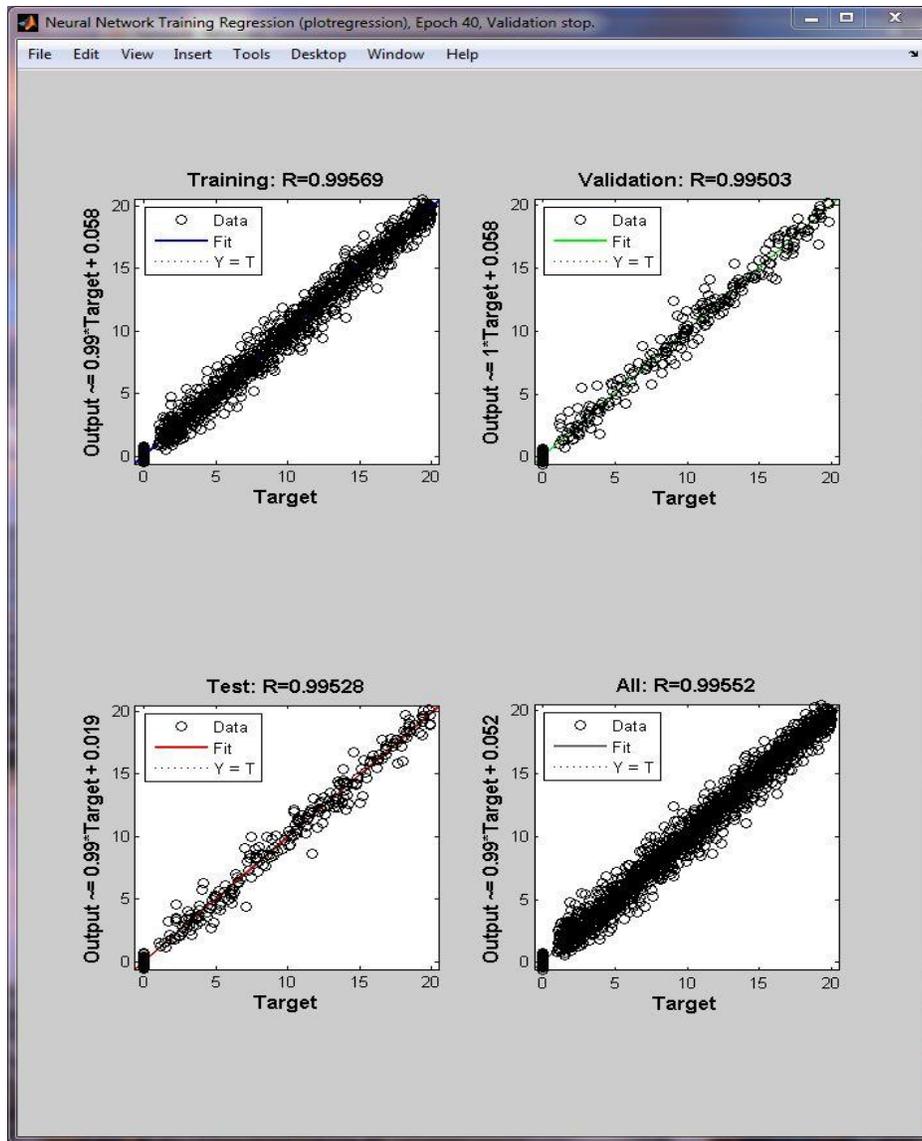


Figure 1-25: Network Regression Plot

If the network is not sufficiently accurate, the network initialization and the training can be performed again. Each time a network is initialized, the network parameters may be different and different solutions might be produced. In order to reinitialize the network weights and biases automatically based on the training data sets, the Reinitialize Weights index of the network window can be used (Figure 1-26). Furthermore, View/Edit Weights index enables to view and edit individual or all the weight and biases of a layer (Figure 1-27). As a second approach, due to the fact the larger numbers of neurons in the hidden layer give the network more flexibility, the number of hidden neurons can be increased to above 20. As another option a different training

function can be used. For instance, sometimes Bayesian regularization training (*trainbr*) can produce better generalization capability than using early stopping (Beale, et al., 2012). At last additional training data can be employed as well. Having a trained and validated network, the network object can be used to calculate the network response to any input. Properly trained multilayer networks tend to give reasonable answers when presented with inputs that they have never seen. Typically, a new input leads to an accurate output, if the new input is similar to inputs used in the training set. The Simulate index of the network GUI enables the user to present new inputs to the trained network and save the output of the network as a new data set (Figure 1-28).

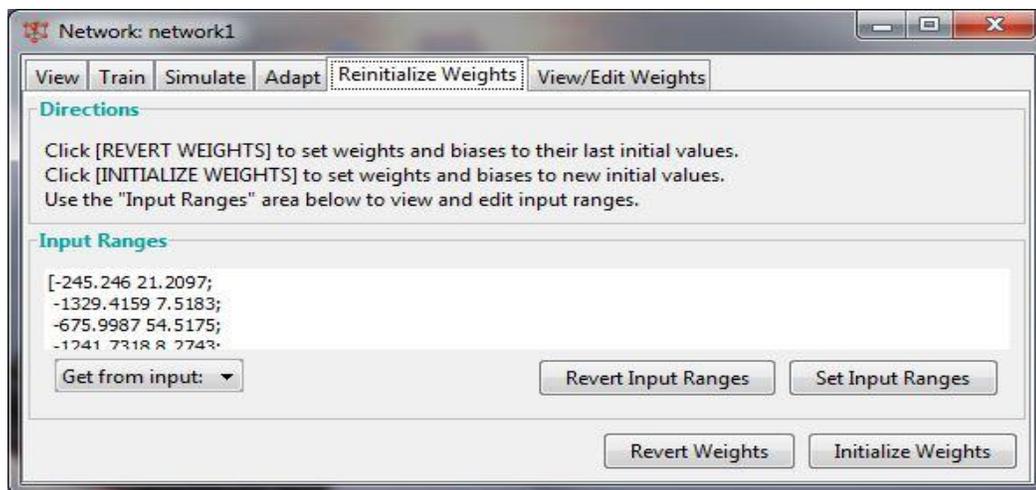


Figure 1-26: Network Reinitialize Weights Window

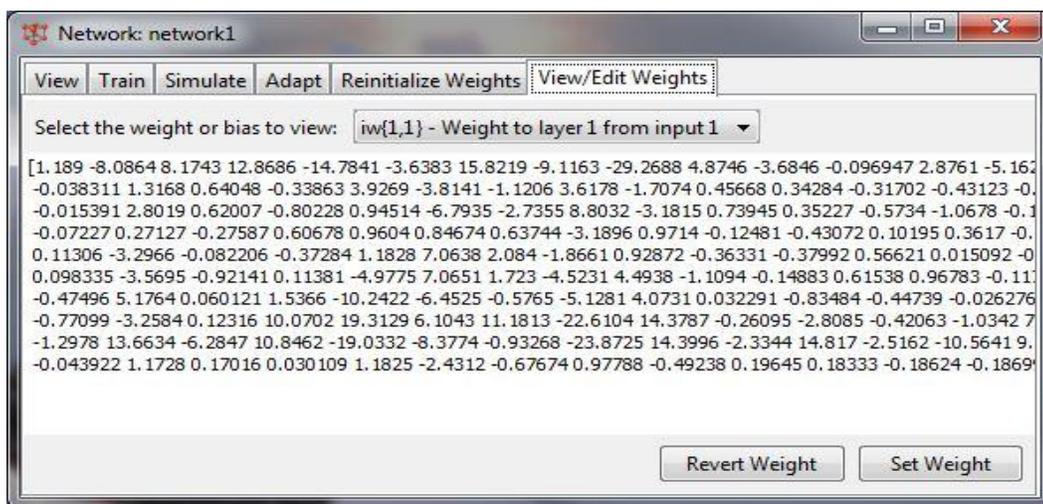


Figure 1-27: Network View/Edit Weights Window

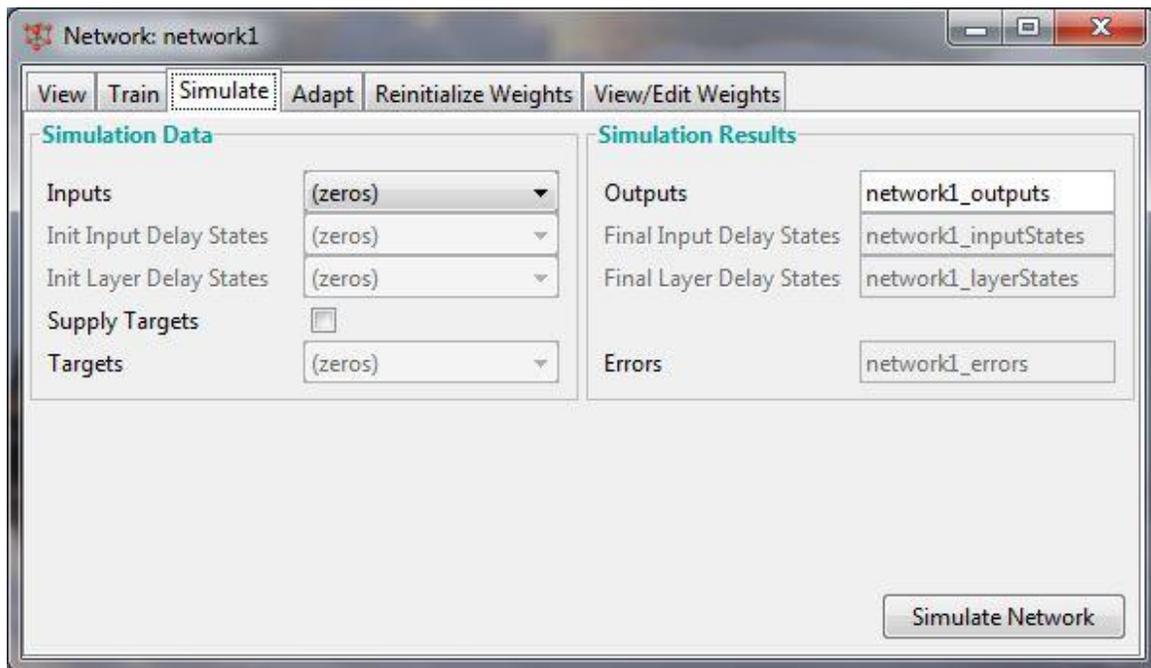


Figure 1-28: Network Simulate Window

## Using MATLAB NNT Command-Line Functions

In this section, how to solve the same simple example by using command-line functions is described. Following scripts are the functions that perform major tasks of the GUIs directly from MATLAB command-line. Again it is assumed that training variables are available and should be loaded:

*Load training\_data*

In order to create a feedforward network with the default tan-sigmoid transfer functions in the hidden layer and pure\_linear in output output layer, following script is written in the command-line:

```
network1 = feedforwardnet([10], 'trainlm');
```

Note that ten neurons (somewhat arbitrary) are assigned to the one hidden layer and Levenberg-Marquardt (*trainlm*) training method is selected. In order to view the network architecture following command can be employed:

```
view(network1);
```

The network has some default values for dividing training data set to three data sets. By default 70% is used for training, 15% for validation and 15% for testing. However, these ratios can be set as the user wish using following scripts:

```
network1.divideParam.trainRatio = 70/100;  
network1.divideParam.valRatio = 15/100;  
network1.divideParam.testRatio = 15/100;
```

In order to train and save the training results with the default values for the feedforward network following script is used:

```
[network1,tr] = train(network1,Input,Output);
```

Notice that no configuration or initialization commands are used. This is due to the fact that weight and bias initialization and network configuration is done automatically by the *train* function. Since there are two target values associated with each input vector having 8 elements, the network will have 2 output neurons. By default, the training window will appear during training (Figure 1-23). Should the user wish to have this window not displayed during training, the network parameter *network1.trainParam.showWindow* to 0 otherwise it should be 1.

Once the training is finished by early stopping or reaching one of the limits, the performance of the network should be tested. Below is the script that computes the network outputs for the same input data set presented at training process, errors and overall performance:

```
net_outputs = network1(Input);  
errors = gsubtract(Output, net_outputs);  
performance = perform(network1, Output, net_outputs);
```

In general, some post-processing is needed to decide on network performance and check if any changes need to be made to the training process, the network architecture or the data sets. The first thing to do is to check the training record, *tr*, which was the second argument returned from the training function. This structure contains all of the necessary information regarding the

network training process. For instance, *tr.trainInd*, *tr.valInd* and *tr.testInd* contain the indices of the data points that were used in the training, validation and test sets, respectively. If the user decides to retrain the network using the same division of data, *net.divideFcn* can be set to *'divideInd'*, *net.divideParam.trainInd* to *tr.trainInd*, *net.divideParam.valInd* to *tr.valInd*, *net.divideParam.testInd* to *tr.testInd*. These codes can be written in command line as below:

```
network1.divideFcn = 'divideind';  
network1.divideParam.trainInd = tr.trainInd;  
network1.divideParam.valInd = tr.valInd;  
network1.divideParam.testInd = tr.testInd;
```

The *tr* structure also keeps track of several variables during the course of training, such as the value of the performance function, the magnitude of the gradient, best iteration, etc. For instance, employing *tr.best\_epoch*, the iteration at which the validation performance reached a minimum can be identified. In order to plot the training, validation and test performance, following code can be used:

```
plotperform(tr);
```

In order to plot training state graph following code can be used:

```
plottrainstate(tr);
```

In order to plot training state graph following code can be used:

```
plotregression(Output,net_outputs,'Train');
```

The regression plot can be created with the following commands: The first command calculates the trained network response to all of the inputs in the data set. The following six commands extract the outputs and targets that belong to the training, validation and test subsets. The final command creates three regression plots for training, testing and validation.

```
net_outputs = network1(Input);  
trOut = net_outputs (tr.trainInd);  
vOut = net_outputs (tr.valInd);  
tsOut = net_outputs (tr.testInd);  
trTarg = Output (tr.trainInd);
```

```
vTarg = Output (tr.valInd);  
tsTarg = Output (tr.testInd);  
plotregression(trTarg,trOut,'Train',vTarg,vOut,'Validation',tsTarg,tsOut,'Testing');
```

In case it is desired to initialize the network, following command can be employed:

```
network1= init(network1);  
[network1,tr] = train(network1,Input,Output);
```

If it is required to increase the number of hidden neurons, say to 20, and train the network again the following codes can be used:

```
network1 = feedforwardnet([20],'trainlm');  
[network1,tr] = train(network1,Input,Output);
```

And finally, to try a different training function like Bayesian regularization training, *trainbr*, for a possible better generalization capability, the following codes can be used:

```
network1 = feedforwardnet([20],'trainbr');  
[network1,tr] = train(network1,Input,Output);
```

Once the user is satisfied with the training process and the network performance is validated, the network can be used to calculate the network response to any input. For instance, the network response to the 5th input vector in the building data set, the following code can be used:

```
a = network1 (Input (:,5));
```

## **Appendix 6: Publications**

**Paper 1: Determination of the static pressure loads on a marine composite panel from strain measurements utilising artificial neural networks**

# Determination of the static pressure loads on a marine composite panel from strain measurements utilising artificial neural networks

Proc IMechE Part M:  
*J Engineering for the Maritime Environment*  
227(1) 12–21  
© IMechE 2013  
Reprints and permissions:  
sagepub.co.uk/journalsPermissions.nav  
DOI: 10.1177/1475090212449231  
pim.sagepub.com  


Mohammad Reza Ramazani<sup>1</sup>, Siamak Noroozi<sup>1</sup>,  
Mehran Koohgilani<sup>1</sup>, Bob Cripps<sup>2</sup> and Philip Sewell<sup>1</sup>

## Abstract

One of the first steps in marine structural design is to calculate the wave-induced loads and load combinations. In contrast with both the hydrostatic loads and the self-weight loads which can be evaluated with a high degree of confidence, it is more difficult to measure the in-service hydrodynamic loads generated by sea waves. Direct pressure load measurement techniques can currently provide only data at finite locations while classical analytical techniques require knowledge of all the parameters that influence the load and that each parameter is studied independently. Therefore, a novel technique is required to overcome these limitations by providing a method of measuring the pressure load over large areas with relatively few sensors and minimal data collection. This paper reports research undertaken to develop an inverse problem approach utilising an artificial neural network for measurement of the pressure loads experienced by marine structures. The suitability and performance of utilising an artificial neural network for quantifying the pressure load applied to a marine structure is presented. It was found that the artificial neural network was able to estimate accurately the pressure loads applied to up to 12 locations on the structure. It is concluded that the inverse problem approach can be used to estimate the applied loads on the marine structure in real time from strain measurements.

## Keywords

Composite, marine, artificial intelligence, artificial neural network, structural analysis, load

Date received: 18 August 2011; accepted: 30 April 2012

## Introduction

A review of methods for evaluating the hydrodynamic loads on ships by Phelps<sup>1</sup> outlined that, for given ship conditions, both the hydrostatic loads and the self-weight loads can be evaluated with a high degree of confidence. In contrast, the determination of the hydrodynamic loads generated by sea waves is more difficult to quantify. Most designers of high-speed craft use the rules, guidelines and procedures of the various classification societies to calculate the bottom loads and pressures. Unfortunately, there are some uncertainties in these calculations that can lead to overdesign.<sup>2</sup> Recent studies demonstrated that measurement of the in-service hydrodynamic load on marine structural components is also particularly difficult owing to the transient nature of the load intensities.<sup>3</sup>

Xu and Duan<sup>3</sup> have reviewed the techniques used to estimate the impact loads (slamming, wave slap, flare slamming and green water on the deck). In summary,

these techniques can be categorised into four groups: theoretical approaches, experiments, empirical formulae and numerical simulations. Theoretical approaches and numerical simulations are the most flexible approaches. Experimental methods are thought to be most reliable but can be costly and suffer from scale effects. The measured results, such as a two-dimensional wedge, a two-dimensional cross section of a ship and a three-dimensional drop cone or sphere, are usually used for validation of a theoretical method

<sup>1</sup>School of Design, Engineering and Computing, Bournemouth University, Poole, Dorset, UK

<sup>2</sup>Longitude Consulting Engineers Ltd, Southampton, UK

### Corresponding author:

Mohammad Reza Ramazani, School of Design, Engineering and Computing, Bournemouth University, Fern Barrow, Poole, Dorset BH12 5BB, UK.

Email: mramazani@bournemouth.ac.uk

or numerical simulation. There are also sea-keeping tests conducted in the towing tank in either regular waves or irregular waves. The objective of these experimental studies is to obtain the relationship between the pressure magnitude and the velocity at the instant of impact on a given body.

Some modern composite laminate hulls have sensors integrated within them during manufacture. Stresses and strains generated because of the loads acting on a craft are used to monitor the structural integrity, and also to monitor stresses at key locations where there are geometrical or material discontinuities. These systems provide real-time information concerning the state of the ship structure, can be employed to validate computer models, increase operational attainability and ship service life and can lead to reductions in the ongoing maintenance costs.<sup>4-6</sup>

Computational fluid dynamics (CFD) techniques provide a method to solve these problems; however, it is generally accepted that CFD codes have difficulties in estimating the impact loads.<sup>7</sup> CFD methods are time consuming, making statistical estimations of the response variables in the sea difficult.<sup>3</sup>

Finite element analysis (FEA) can be employed for assessment of a ship's structural strength. This requires a suitable method for applying loads to the FEA model. This approach is dependent upon the accuracy of the methods used to define and calculate the loads in the first place and also the specific FEA software being used. It is more difficult to apply hydrodynamic loads to finite element models than to apply static loads. This is due to the dynamic nature of the problem and less clearly defined loads. In addition, the material properties are usually not available, difficult or too costly to obtain and are very variable for typical marine-type laminates. According to Phelps,<sup>1</sup> the work carried out so far has been limited and correlation between numerical predictions, test data and further evaluations are necessary.

As vessels and craft, in most cases, are extremely complicated structures, the mechanical properties, or relations between externally induced excitation and structural responses, are difficult to formulate. An appropriate load-monitoring system and technique have to be developed for naval assets and large structures.<sup>8</sup>

A novel approach for determination of the pressure loads experienced by marine structures is the utilisation of artificial intelligence (AI). Different approaches employing AI methods may be utilised to solve complex problems. Fuzzy logic and artificial neural networks (ANNs) are examples of various AI techniques available.<sup>9</sup> These methods have been utilised in research areas where problems are solved by pattern recognition, generalisation and pattern classification.<sup>10</sup> ANNs have attracted considerable attention and show promise for modelling complex non-linear relationships.

This approach has the potential to overcome issues related to both direct measurement of pressure and

other analytical estimation techniques. Transducers allow measurement of the pressure in only a finite number of regions, provide data at only the location at which they are attached and are relatively heavy. Therefore, to provide an adequate profile of the pressure on the boat a large number of transducers would be required, adding significant weight to it. A common analytical estimation technique involves developing a complex relationship relating the load to all the key design parameters. The sensitivity constants and influence factors for the relationship can then systematically be calculated by a parametric study using linear regression analysis. Once all the constants have been determined, the relationship can be transformed into an empirical formula, relating the pressure load to the design parameters. This technique requires knowledge of all the parameters that influence the load and that each parameter is studied independently of all others to find the relationship between each parameter and the applied load. This methodology therefore requires a large amount of testing to collect the required data.

ANNs have been used extensively in other fields. For instance, ANNs have been widely used for damage identification.<sup>11</sup> In a study by Cao et al.,<sup>8</sup> an approach was developed to identify the loads acting on aircraft wings where an ANN was utilised to model the load-strain relationship for structural analysis. The research demonstrated that using an ANN to identify loads is feasible and a well-trained ANN reveals an extremely fast convergence and a high degree of accuracy in the process of load identification for a cantilevered beam model. In a study by Amali et al.,<sup>12</sup> it was demonstrated that an ANN can be combined with experimental methods to create a hybrid inverse problem analysis tool or inverse problem engine. The hybrid approach can be applied to both direct problems (calculation of the structural response from known loads applied to the structure) and inverse problems (calculation of the applied load from a known structural response). Additionally, the approach avoids the need for having information on the component geometry and the material properties.<sup>13</sup>

Applying AI to a ship or a boat has been limited to the use of an ANN for preliminary ship design.<sup>14,15</sup> Research by Kim et al.<sup>16</sup> presented a new method to classify surface hull plates effectively in preliminary ship design using neural networks.

This paper reports the research undertaken to develop an ANN methodology for quantifying the static pressure loads on a marine composite panel from strain measurements collected from the panel.

## Methodology

The methodology employed to evaluate the suitability and performance of utilising an ANN as an inverse problem solver for quantifying the load applied to the composite panel is presented in this section. The first

stage of the investigation was to design a load quantification methodology for the panel utilising an ANN. In the second stage the load quantification methodology was validated by comparing loads estimated by the ANN with the actual loads applied to the panel.

### Inverse problem analysis methodology

Inverse problem analysis is based on accurately calculating the external loads or boundary conditions that generate a known amount of strain, stress or displacement at predetermined locations on a structure. An ANN, as an inverse problem solver, can be utilised to solve structural problems where the structure's response to the load is known but the load which causes this response is not. In a study by Xu et al.,<sup>17</sup> it was indicated that many fields of science and industry employ inverse analysis such as material property estimation, radar tracking, medical tomography, residual stress determination, oil reservoir identification and non-destructive testing.

According to Kohonen,<sup>18</sup> WS McCulloch and WA Pitts introduced the fundamentals of neural computing for the first time in 1943. An ANN, consisting of neurons (units) and connections between them, is based on simulating the way that the brain processes different data. Therefore, ANNs are able to perform behaviours similar to brain activities such as learning, generalisation, categorisation, association, optimisation and feature extraction.

The inverse method is used to determine a relationship between the cause and its effect. In this study, the static loads (the cause (output)) on a composite panel are quantified by acquiring repeatable strain responses (the effect (input)) to these loads from the panel. By introducing examples to an ANN, it can learn the relationships between the input and the output through a training process. The ANN requires a number of known input and output data for training (i.e. relating the ANN inputs to outputs using a transfer function and series of weighting values). New input strain data can then be introduced to the trained ANN (problem data) to quantify or estimate the load (Figure 1).

The ANN is adjusted or trained such that a particular input leads to a unique and specific desired response (testing inputs). Weight adjustment (described in the section on the ANN architecture or topology) is based

on a comparison of the network response and the desired response. Looping continues until the network response matches the desired response or at least the error function becomes an acceptably small value (i.e. minimisation of the error function or convergence). A back-propagation ANN uses a mean square error (MSE) function which is defined as a sum of the squared errors between the desired response and the network response over all the network responses for all the input–desired response pairs.

Further input–desired response pairs, which have not been seen previously by the network (testing patterns), are introduced after each loop and fed through the network to find the MSE between the desired response and the network response. This is a further check to ensure that the error function is being reduced and that the network is moving towards the global minima or convergence. The testing MSE value also indicates the level of accuracy of the network (i.e. if the testing MSE is 10%, then the network should estimate accurately to within 90%).

ANN analysis often requires a high number of individual loops to determine the best solution. However, the training time can be reduced (i.e. the number of loops to minimise the error equation can be reduced) by pre-processing the data that are given to the network to train. Introducing noisy data to the ANN is one technique to improve the ANN's training efficiency. These data are generated from the original input and output data and have a noise level either added to or subtracted from a predefined proportion of the training data. The noise level depends on the nature of the input and output data collected (i.e. if data are collected from strain gauges the noise value will depend on the accuracy of the data acquisition equipment). Introducing noisy data during training allows the ANN still to recognise input data that may have some random fluctuation in its magnitude owing to the accuracy of the data acquisition device.

Once the ANN is sufficiently trained, it can be utilised to estimate the output in real time. New inputs (problem data) are presented and processed by the ANN as though training were taking place. However, at this point the output is retained and no back propagation occurs. Establishing an inverse problem analysis approach can result in important advantages over simulation, numerical or theoretical methods. Through utilising such a method, knowledge of the component material constitutive laws and component geometry are not required. In contrast, they are necessary for valid and accurate simulation, numerical or theoretical analysis.

### Experimental set-up

The structure under consideration was a 1 m<sup>2</sup> glass-reinforced fibre polymer marine composite panel (Figure 2(a)). The properties of the composite material were unknown; however, as discussed in the section on

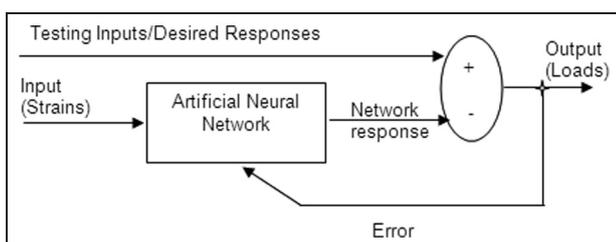
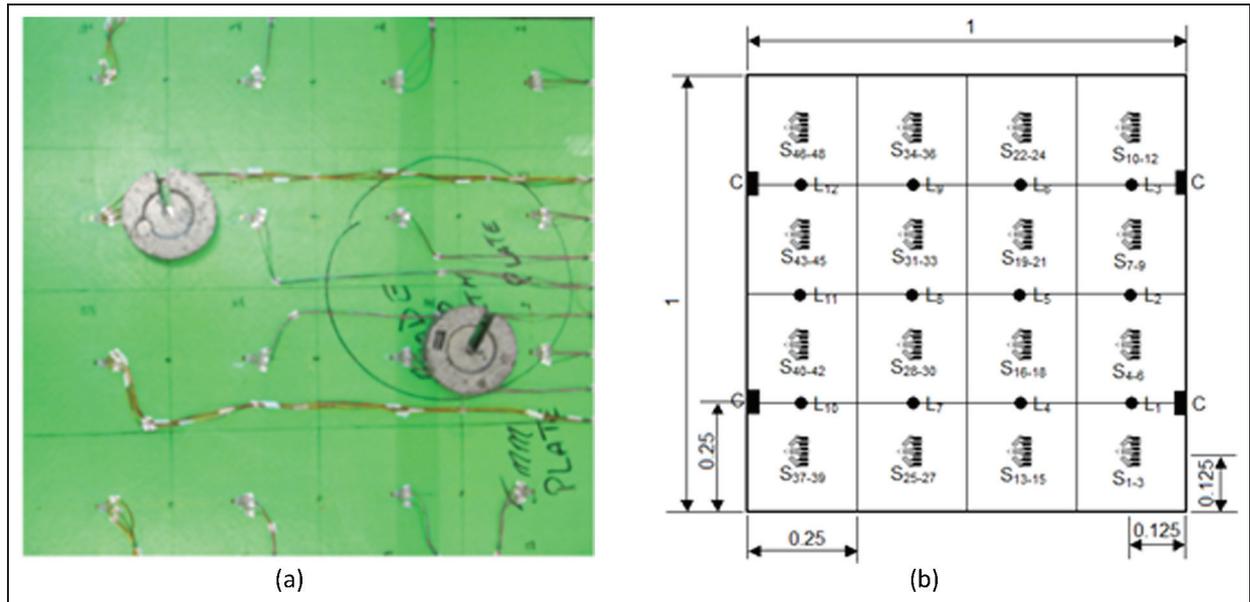


Figure 1. ANN training loop to minimise the error.



**Figure 2.** Top view of the marine composite panel showing (a) the weights used for loading and (b) the locations L of the loads, the locations S of the strain gauges and the locations C of the constraints (all dimensions are in metres).

inverse problem analysis methodology, a major advantage of the inverse approach is that the material properties are not required to quantify the load applied to the structure.

The panel was divided into a four-by-four grid, producing 16 equal regions of area  $0.25\text{ m} \times 0.25\text{ m}$  (Figure 2(b)). The positions of the strain readings collected from the panel are also important. To produce efficient training data the strain data should be captured in the sensitive regions (i.e. the strain at those locations must vary significantly owing to changes in the load level). In addition, the strain data collected must provide a unique response for each load distribution. If strain data are collected from non-sensitive regions of the panel and/or the strain data collected are not unique for each load distribution, the ANN is less likely to be able to find a function relating the input and the output. A rectangular strain gauge rosette was attached to the top surface of the panel at the centre of each region, giving 48 strain readings ( $S_1$  to  $S_{48}$ ). The specifications of the strain gauges used are detailed in Table 1.

The bottom surface of the panel was supported at four points using blocks, each  $0.035\text{ m}$  wide,  $0.009\text{ m}$  deep and  $0.025\text{ m}$  tall (Figure 2(b)). Loading was achieved by applying weights normal to the panel

surface at the 12 locations indicated in Figure 2(b) ( $L_1$  to  $L_{12}$ ).

A Windmill 751-SG strain-monitoring and control data acquisition system (Windmill Software Ltd, Manchester, UK) with a resolution of  $\pm 1$  microstrain was used to capture the strain data. The 751-SG package consists of Windmill 6 software, a universal serial bus (USB) unit which provides differential inputs to monitor 16 strain gauges at up to 80 samples/s. Eight USB units can be connected to one computer to monitor up to 128 strain gauges.

### Generation of the ANN training data

The efficiency of the process for collection of the training data can be increased by reducing the amount of data collected. This is achieved by using the theory of superposition to generate training and testing patterns from the independent parent patterns, as discussed in detail elsewhere.<sup>19</sup> The theory of superposition states that the strain at a point on a structure due to a series of loads is equal to the sum of the strains from each individual load case. Using this theory an infinite number of training patterns can be generated by applying one known load to each location on the structure individually. However, the superposition method is only valid if the structure under investigation has a linear stress–strain or load–displacement response within the expected load range applied to it. The results of the investigation to study the linearity of the panel can be found in the section on the linearity of the composite panel.

In this study, 48 strain readings from 16 rosettes (inputs) and 12 load readings from 12 locations (outputs) were used with the superposition method to

**Table 1.** Strain gauge rosette specifications.

Type	Rectangular rosette
Resistance	$120\ \Omega \pm 0.6\%$
Gauge factor	$2.100 \pm 0.5\%$
Gauge length	3.18 mm
Gauge width	1.78 mm

generate 1116 sets of training data. This meant that 1116 sets of 12 random loads at each location on the panel and the resultant 48 strains caused by these random loads were required to find the relationship between the input data and the output data. Some 280 testing patterns were also generated using the superposition method and introduced to the network during training to ensure convergence. This represented 20% of the original 1396 total patterns generated. Some 200 noisy patterns were also added to the training data based on the level of noise in the data acquisition system ( $\pm 1$  microstrain). Once training data and testing data had been generated, they were normalised by converting all data to a range between 0 and 1 to improve the accuracy of the solution.

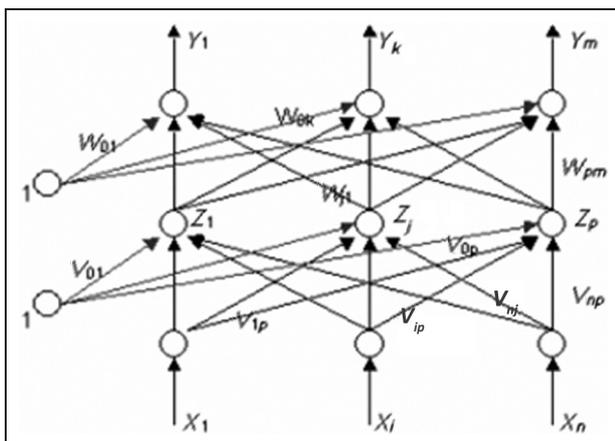
### ANN architecture or topology

Network architecture or topology is the way that neurons are connected to each other in the ANN. Key architectural issues are as follows:

- the number of layers in the ANN;
- the number of neurons per layer;
- the type and parameters of the neuron, which are usually the same throughout;
- the number of calculations per iteration during learning and recall.

An example of a back-propagation neural network with three layers, namely an input layer, a hidden layer and an output layer, is shown in Figure 3.

A common network has one or more hidden layer with a log-sig transfer function connected to an output layer with a pure-lin function: pure-lin is a linear transfer function used for linear approximation, and log-sig



**Figure 3.** Three-layer back-propagation ANN, where the inputs are denoted  $X$ , the outputs are denoted  $Y$  and the output from every neuron in the hidden layer is denoted  $Z$ . The  $V$  values are the weights between the neurons in the input layer and the neurons in the hidden layer. Similarly the  $W$  values are the weights between the neurons in the hidden layer and neurons in the output layer.

is a non-linear sigmoid transfer function that accepts any value as an input between  $+\infty$  and  $-\infty$ . If the last layer of a network has a sigmoid function, then the outputs of the network are limited to a small range. If a linear function is used as the final layer, the network outputs can take on any value as is required to estimate the actual load on the panel. Since the log-sig function is differentiable, it is usually utilised in back-propagation networks.

A transfer function calculates the layer's output from its input. The input data are fed through the ANN and they are adjusted using the transfer functions. The input to the transfer function is the sum of the weighted inputs  $W$  and the bias  $b$  (Figure 4). The values of these parameters are randomly generated when training is initiated. The weight controls the slope of the transfer function while the bias allows the transfer function to shift from left to right. Having multiple hidden layers of neurons with non-linear transfer functions enables the network to understand both non-linear and linear relationships between the input data and the output data.

Unsatisfactory performance of the ANN can be due to a wide range of reasons, such as the following:

- an unsuitable ANN architecture or learning method;
- insufficient representative data (not a sufficient number of example strain-load data);
- inadequate pre-processing (noisy data from data acquisition system have been ignored);
- unsuitable ANN training parameters.

However, most of the time this is not the case, and the ANN will be well trained and performs satisfactorily even on a new untrained data set. An iterative process was used to determine the optimum network architecture for the panel based on the value of the final MSE of each network tested. The number of layers chosen can have an effect on the efficiency of the network. However, because of the relatively low number of inputs and outputs involved in this example this was not deemed to be an important factor to consider when selecting the architecture.

It was determined, through the testing of various network architectures, that the optimum network (lowest MSE) had one hidden layer with 22 neurons and used a tan-sig transfer function which is similar to a log-sig function but compresses the output to a value between  $-1$  and  $+1$  (Figure 4). The output layer had 12 neurons (representing the 12 loads to be estimated) and used a pure-lin transfer function. The final specification of the ANN is shown in Table 2.

It is possible to over-train the ANN, which means that the ANN has been trained to respond to only one type of input. If this should happen, then learning can no longer occur. In real-world applications this situation is not very useful as a separate over-trained ANN for each new kind of input would be required.

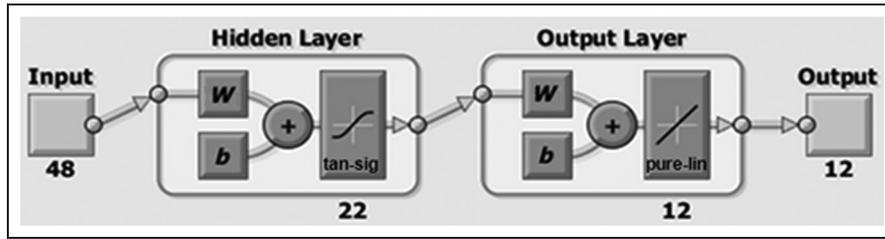


Figure 4. MATLAB representation of the optimum ANN architecture.

Table 2. Architecture of the ANN.

Architecture	Feedforward back propagation
Number of layers	2
Data process	Normalisation
Noise generator	$\pm 1$ microstrain on 200 of the training patterns
Range of loads	0–196.2 N
Number of inputs (surface strains)	48
Number neurons in the output layer (normal loads)	12
Number of neurons in the hidden layer	22
Number of training patterns	1116
Number of testing patterns	280
Number of problem patterns	Depends on the number of patterns collected

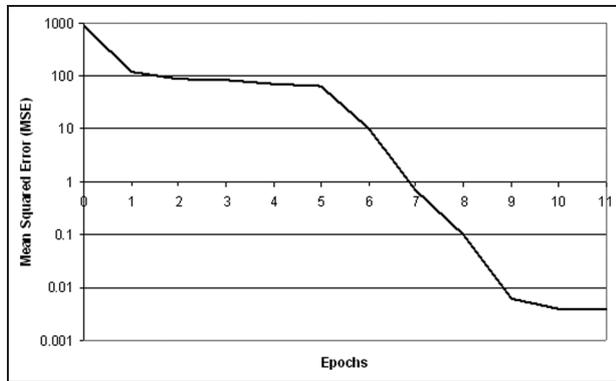


Figure 5. Training and testing errors graph.

To confirm that the ANN was not over-trained the training was supervised to ensure that the MSE of the testing data did not increase, which is an indication of over-training. The ANN was trained to minimise the MSE (Figure 5). The final testing MSE was approximately  $4 \times 10^{-3}\%$ , which took under 3 min to reach. This meant that the ANN should estimate accurately to within 99.9% of the actual load value.

**ANN validation and performance**

The validity and performance of the ANN method were evaluated by comparing the load estimated by the

ANN with known loads applied to the panel (problem data). Experimental problem data are the captured strain data from the 48 strain gauges attached to the panel while it is being loaded. It is essential that the strain data are captured at identical locations for both the training data and the problem data.

The first validation study utilised load and strain data generated from the original superposition data collected to produce the training data. This meant that any issues with the repeatability of the strains collected for a given load were removed. In the second study, problem strain data were captured directly from the panel under different loading conditions (i.e. one or two random loads were placed at random locations on the panel) and again the estimated loads were compared with the actual applied loads.

The strain data were collected through bespoke data acquisition and ANN software linked to the Windmill data acquisition system. This software was developed by the present authors in MATLAB (MathWorks, Natick, Massachusetts, USA) utilising Windmill Direct Data Exchange protocols to acquire the strain data and the MATLAB Artificial Neural Network Toolbox capabilities.

**Results**

The validity of utilising the theory of superposition for the panel and the ANN validity and performance are detailed in the following sections.

**Linearity of the composite panel**

The application of the theory of superposition was shown to be valid through studying the linearity of the material within the expected load range used in this study (discussed in the section on the generation of ANN training data). The load was applied in increments between 19.62 N and 117.72 N at each of the 12 locations and the 48 strains were collected. Figure 6 shows the typical strain response to the applied load. The strains decreased as the load increased as the strain gauges were attached to the compressive surface of the panel. Plotting a linear trend line through the data showed that the strain changed linearly with load for all gauge locations as the coefficient of determination ( $R^2$  value) was very close to one in each case.

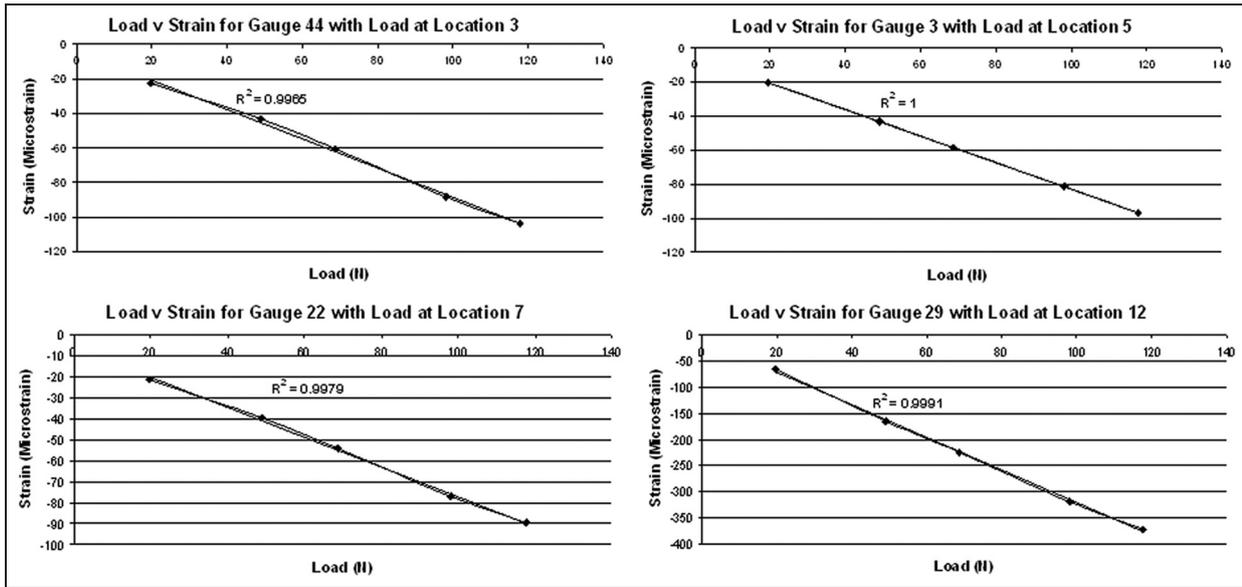


Figure 6. Typical linear response of the composite panel to the applied load.

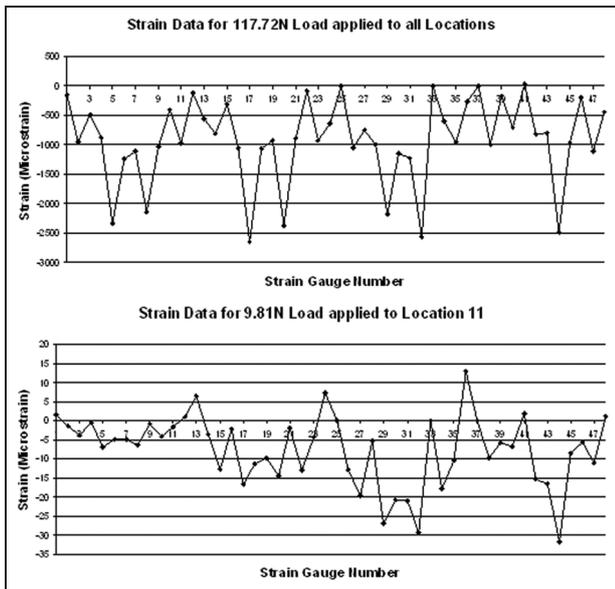


Figure 7. Typical problem strain data.

**ANN validation and performance**

In order to validate the trained network, a set of problem strain data was generated from the panel. Since this set of data had not been seen by the network during the training stage, it can be used to evaluate the accuracy of the ANN’s load estimation.

*Validation using superposition data.* The theory of superposition was used to generate problem data (12 loads and 48 strains generated on the panel surface from these loads) for loads between 0 N and 196.2 N. Figure 7 shows example strain data collected from the 48 gauges used as the problem data in the trained ANN.

Table 3 shows the ANN estimated loads for all 12 load locations for actual loads of 9.81 N and 196.2 N applied to all 12 locations at once. The average of the estimated loads together with the average estimated loads for all load increments were used to generate Figure 8.

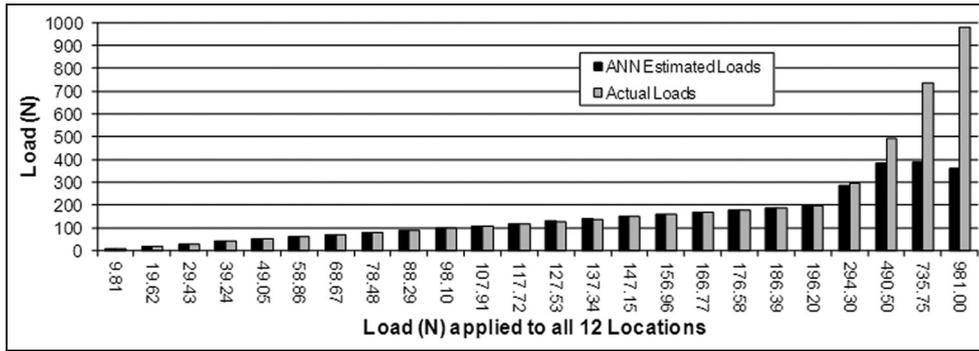
Figure 8 shows the comparison of the actual loads applied to the panel with the ANN estimated loads generated from the introduced problem data. If the ANN estimates the loads accurately, the actual load data and the ANN load data should be identical. In this example, loads were applied to all 12 locations on the panel at once for loads ranging from 0 N to 981 N.

The accuracy of the estimated load is high within the range of loads (0–196.2 N) that the ANN was trained to estimate, thus indicating the suitability of

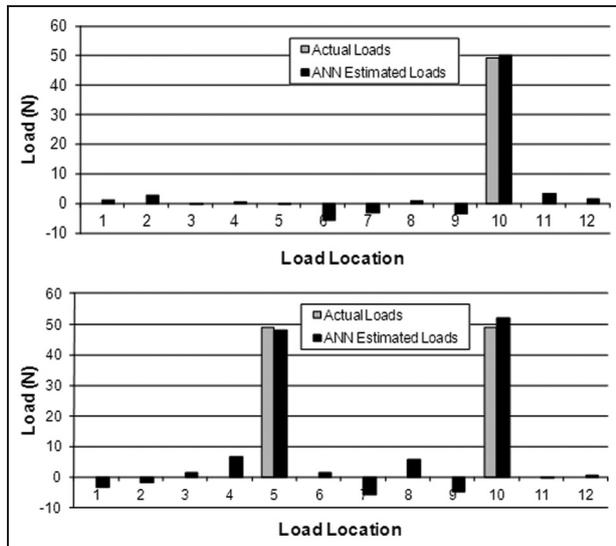
**Table 3.** ANN estimated loads for 9.81 N and 196.2 N loads applied to all 12 load locations.

Load location	ANN estimated load (N)	
	9.81 N applied	196.2 N applied
1	9.72	200.91
2	9.76	195.04
3	9.65	196.45
4	9.59	187.23
5	10.09	205.00
6	9.84	191.51
7	9.82	195.39
8	9.69	198.70
9	9.83	198.09
10	9.90	198.86
11	9.76	196.07
12	9.80	195.26
Average	9.79	196.54

ANN: artificial neural network.



**Figure 8.** Comparison of the actual loads and the ANN loads. ANN: artificial neural network.



**Figure 9.** Example comparisons of the ANN estimated loads and the actual applied loads. ANN: artificial neural network.

this method for load estimation. The results, however, indicate that the network is capable of accurate estimation up to 294.3 N. As the applied loads increase further outside the trained load range, the accuracy of the estimated load decreases. In summary, the network is most accurate in the load range that it has been trained to estimate, as would be expected. However, loads just outside the training envelope can still be estimated accurately.

*Validation using direct data.* The MATLAB program was developed to be able to acquire strain data directly from panel and to introduce them to the ANN as new problem data. This enabled the system to estimate the applied load on the panel in real time. The robustness of the system was investigated by introducing strain data gathered by applying loads at the extremes of the range that it was trained to estimate (0–196.2 N). Figure 9 shows typical examples of the comparison between the actual loads applied to the panel with the ANN estimated loads generated from the introduced problem data. In these examples, either one or two

loads are applied to the panel. For both sets of problem data it can be seen that the ANN can again estimate the load at the loaded locations with a high degree of accuracy.

The estimated negative load values at the unloaded locations were due to the differences between the strain data collected to generate the training data and the collected problem strain data. Because of these errors, slightly different strain patterns are introduced to the ANN, producing the errors in the estimated loads. The introduction of further noisy patterns in the training data set may reduce these small errors, indicating that further work could be carried out to improve the accuracy further.

### Discussion

The results presented in this paper show that the inverse problem method, utilising an ANN, can accurately estimate the position and magnitude of a number of static loads applied to the composite panel from the captured strain data. The results of this study can be summarised as follows.

1. An ANN can be trained using experimental data to solve inverse problems.
2. As there is a mathematical relationship between the applied load and the surface strains, the system always converges and the MSE is in the range of acceptable error.
3. The system is capable of accurately estimating the static loads.
4. Using a combination of pure and noisy data (as used in this study) improves the accuracy of the results.
5. The main source of error was found to be in the reliability of the data acquisition system utilised owing to the large variance in the strain data collected at different time intervals. The ANN cannot estimate the load accurately if there is a large variance in the strain data used to generate the training data set and the problem data set collected at a later date.

The ability to measure the actual load history of a craft in service would enable the designer to validate the load estimation and structural design tools used during the design stage of a craft. This would lead to the development of more optimal structure designs for this type of craft. The operational safety of the craft can also be improved by having a real-time load-monitoring system that is able to detect any degradation of the structural integrity and defects within the structure.

It is proposed that the ANN methodology, with further research and development, could be utilised for the quantification of in-service transient loads in real time acting on the craft from the craft's structural response (strain response to the load). This would provide valuable information to influence future craft design. In order to evaluate fully the proposed methodology for in-service load monitoring of marine structures the following areas require investigation:

- (a) the behaviour of marine structures under large displacement (a static load is applied) where the structure may behave non-linearly;
- (b) the behaviour of marine structures under transient load conditions (a dynamic load is applied);
- (c) the effect of size of the structure on the ANN estimation accuracy;
- (d) the number of sensors required for accurate load estimation by optimising the method (while some vessels do have integrated sensors, most do not; the number of sensors should be minimised to reduce the time to train the system, the cost and the weight);
- (e) the effect of modifying the ANN training parameters, including the number and type of training patterns introduced to the ANN;
- (f) validation of the methodology on an in-service craft.

Finally, a graphical user interface should be developed, allowing control of various parameters of the data acquisition and load-monitoring system as well as a graphical display in real time.

## Conclusion

It was shown that the inverse problem approach can be used to estimate the loads applied on a marine composite panel from the strain measurements. A comparison of the ANN loads with the actual applied loads indicated a very good performance of the methodology. This was achieved in real time, providing an accurate load history for a component without requiring knowledge of the material properties or the component geometry. This potentially makes the system ideal for solving many classes of complex engineering problem that require load monitoring. Areas for future investigation including evaluation of the suitability of the approach for estimating transient loads were discussed.

## Funding

This work was supported by BAE Systems Surface Ships Ltd, Portsmouth and Bournemouth University (grant no. DAKG22X).

## Acknowledgement

The authors would like to thank Steve Austen, Head of Engineering Support of the Royal National Lifeboat Institution, for providing the marine composite panel used in this investigation.

## References

1. Phelps BP. Determination of wave loads for ship structural analysis. Report DSTO-RR-0116, Aeronautical and Maritime Research Laboratory, Defence Science and Technology Organisation, Melbourne, Victoria, Australia, 1997.
2. Koelbel Jr JG. Comments on the structural design of high speed craft. *Mar Technol* 1995; 32(2): 77–100.
3. Xu G-D and Duan W-Y. Review of prediction techniques on hydrodynamic impact of ships. *Mar Sci Appl* 2009; 8: 204–210.
4. Adamchak JC. An approximate method for estimating the collapse of a ship's hull in preliminary design. In: *SSC-SNAME structural symposium*, Arlington, VA, USA, 15–16 October 1984, pp. 37–61.
5. Baldwin C, Niemczuk J, Kiddy J, et al. Structural testing of navy vessels using bragg gratings and a prototype digital spatial wavelength domain multiplexing (DSWDM) system. *Nav Engrs J* 2002; 114(1): 63–70.
6. Sikora JP, Dinsenbacher A and Beach JE. A method for estimating lifetime loads and fatigue lives for swath and conventional monohull ships. *Nav Engrs J* 1983; 95(3): 63–85.
7. Faltinsen OM. Challenges in hydrodynamics of ships and ocean structures. *J Nav Archit Shipbuilding Ind* 2007; 58(3): 268–277.
8. Cao X, Sugiyama Y and Mitsui Y. Application of artificial neural networks to load identification. *Comput Struct* 1998; 69: 63–67.
9. Mamdani EH and Assilian S. An experiment in linguistic synthesis with a fuzzy logic controller. *J Man-Mach Stud* 1975; 7(1): 1–13.
10. Rao VB and Rao HV. *C++ neural networks and fuzzy logic*. New York: M & T Books, 1995.
11. Ziemianski L and Harpula G. The use of neural networks for damage detection in eight storey frames. In: *5th international conference on engineering applications of neural networks*, Warsaw, Poland, 13–15 September 1999, pp. 292–297. Toruń: Wydawnictwa Adam Marszałek.
12. Amali R, Noroozi S and Vinney J. The application of combined artificial neural network and finite element method in domain problems. In: *6th international conference on engineering applications of neural networks (EANN 2000)*, Kingston upon Thames, Surrey, UK, 17–19 July 2000, pp. 1–7. London: Kingston University.
13. Amali R, Noroozi S, Vinney J et al. Predicting interfacial loads between the prosthetic socket and the residual limb for below-knee amputees – a case study. *Strain* 2006; 42(1): 3–10.

14. Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev* 1958; 65(6): 386–408.
15. Shar S and Palmieri F. MEKA – a fast, local algorithm for training feed forward neural networks. In: *International joint conference on neural networks*, Como, Italy, 24–27 July 1990, vol 3, pp. 41–46. New York: IEEE.
16. Kim S-Y, Moon B-Y, Kim D-E and Shin SC. Automation of hull plates classification in ship design system using neural network method. *Mech Systems Signal Processing* 2006; 20(2): 493–504.
17. Xu S, Deng X, Tiwar V, et al. An inverse approach for pressure load identification. *Int J Impact Engng* 2010; 37(7): 865–877.
18. Kohonen T. An introduction to neural computing. *Neural Networks* 1998; 1(1): 3–16.
19. Sewell P, Noroozi S, Vinney J, et al. Improvements in the accuracy of an inverse problem engine's output for the prediction of below-knee prosthetic socket interfacial loads. *Engng Applic Artif Intell* 2010; 23(6): 1000–1011.

S	locations of strain gauges
V	weights between the neurons in the input layer and the neurons in the hidden layer
W	weights between the neurons in the hidden layer and the neurons in the output layer
X	artificial neural network inputs
Y	artificial neural network outputs
Z	output from the hidden layer of the artificial neural network

### Abbreviations

AI	artificial intelligence
ANN	artificial neural network
CFD	computational fluid dynamics
FEA	finite element analysis
MSE	mean square error
USB	universal serial bus

## Appendix

### Notation

<i>b</i>	bias
C	locations of constraints
L	locations of loads

**Paper 2: Sensor Optimisation for In-service Load Measurement of a Large Composite Panel under Small Displacement**

## **Sensor Optimisation for In-service Load Measurement of a Large Composite Panel Under Small Displacement**

Mohammad Reza Ramazani<sup>1,a</sup>, Philip Sewell<sup>1,b</sup>, Siamak Noroozi<sup>1,c</sup>,  
Mehran Koohgilani<sup>1,d</sup> and Bob Cripps<sup>2,e</sup>

<sup>1</sup>School of Design, Engineering & Computing, Bournemouth University, Poole, UK.

<sup>2</sup>Longitude Consulting Engineers Ltd, Southampton, UK.

<sup>a</sup>mramazani@bournemouth.ac.uk, <sup>b</sup>psewell@bournemouth.ac.uk, <sup>c</sup>snoroozi@bournemouth.ac.uk,  
<sup>d</sup>mkoohgil@bournemouth.ac.uk, <sup>e</sup>b.cripps@longitude.uk.com

**Keywords:** Composite, Marine, Artificial Intelligence, Artificial Neural Network, Structural Analysis, Load, Sensor Optimisation

**Abstract.** Current methods to estimate the behaviour of composite structures are based on trial and error or oversimplification. Normally the nonlinearities in these structures are neglected and in order to cover this inadequacy in design of composite structures, an overestimate safety factor is used. These methods are often conservative and leading to the heavier structures. A novel technique employs Artificial Neural Network (ANN) as an inverse problem approach to estimate the pressure loads experienced by marine structures applied on a composite marine panel from the strain measurements. This can be used in real-time to provide an accurate load history for a marine structure without requiring knowledge of the material properties or component geometry. It is proposed that the ANN methodology, with further research and development, could be utilised for the quantification of in-service, transient loads in real-time acting on the craft from the craft's structural response (strain response to load). However, to fully evaluate this methodology for load monitoring of marine structures further research and development is required such as sensor optimisation. The number of sensors should be minimised to reduce the time to train the system, cost and weight. This study investigates the number of sensors required for accurate load estimation by optimising the method.

### **Introduction**

Measurement of hydrodynamic loads generated by sea waves is difficult due to the fact that the sea is highly irregular. The common practice to determine wave loads is based on applying rules and standards which often relies on conservative methods due to large uncertainties in the theoretical calculations used for wave load predictions for ships. In addition, for unconventional ships with new structural designs, it can be sometimes difficult to apply general standards and rules. Direct calculation procedures are needed specifically for complex structures and designs. However, the direct calculation procedures, especially the calculation of the wave loads, are less applied in the shipbuilding industry [1]. One reason is the rather large uncertainties in the wave load predictions for ships as well as lack of experience. In addition, the theoretical basis of the calculation methods is not necessarily sufficient to achieve reliable predictions. Furthermore, uncertainties exist also in all assumptions involved in stochastic methods and prediction procedures including environmental and operational conditions. Sometimes these are difficult to determine accurately in advance and hence assumptions need to be made to estimate them [1].

The current techniques to measure hydrodynamic loads indicate that many techniques developed are either simplified or very expensive and time-consuming [2]. Many studies have been performed to date for evaluating hydrodynamic loads on ships. Such methods are summarised in the reviews by Phelps [3] and Guo-Dong and Wen-Yang [4]. Various strip methods [5,6] have been developed and used to estimate wave induced ship motions and wave loads. The conventional strip theory [7] is successfully used in the seakeeping analysis of normal displacement ships. However, its validity can be questioned when it is used for ships with higher maximum operating speeds [8]. Nonlinearities

become more significant for higher speed ships, however, the conventional strip theory is a linear theory. Strip methods are extensively used as a standard tool to predict the nonlinear loads and motions [9]. Though, there are some concerns on the accuracy of the strip methods for estimation in short waves as they do not precisely consider hydrodynamic interference effects of reflected waves among strips lengthwise or three-dimensional effects.

In order to enhance the accuracy of estimation especially in short waves, many numerical methods considering the three-dimensional effects have been proposed. Among them are the three-dimensional Green function method [10] and Rankine source method [11–13] based on three-dimensional potential theory. However, nonlinear theories and three-dimensional load prediction methods require greater computational effort and have not yet proven to be significantly more accurate than the two dimensional methods [3,4].

Computational Fluid Dynamics (CFD) techniques provide a method to solve these problems. So far, it is generally accepted that CFD codes have difficulties in estimating impact loads [14]. CFD methods are time consuming, making statistical estimations of response variables in sea difficult [4]. Although CFD solutions have been greatly improved in both speed and accuracy in recent years, it is still very challenging to perform complete nonlinear three-dimensional computations for the present problem [15]. As vessels and craft, in most cases, are extremely complicated structures, the mechanical properties, or relations between externally induced excitation and structural responses, are difficult to formulate. An appropriate load monitoring system and technique has to be developed for naval assets and large structures [16].

A novel approach for the determination of pressure loads experienced by marine structures is the utilisation of Artificial Neural Networks (ANN) [17]. ANN methods are utilised in research areas where problems are solved by pattern recognition, generalisation and pattern classification [18]. ANN has attracted considerable attention and shown promise for modelling complex nonlinear relationships. ANNs have been used extensively in many fields [16, 19-24].

Ramazani et al. [17] have recently shown that the inverse problem approach can be used to estimate the loads applied on a composite marine panel from the strain measurements when responding linearly under load. A comparison of the ANN loads with the actual applied loads indicated a very good performance of the methodology. However, it was discussed that more investigation is necessary to further evaluate the suitability of the proposed methodology for in-service load monitoring of marine structures. For instance, the number of sensors required for accurate load estimation needs to be optimised. This is due to the fact that while some vessels do have integrated sensors most do not. Hence, the minimum number of sensors should be employed to reduce the time to train the system, cost and weight.

This paper reports on the research undertaken to further develop the ANN methodology for quantifying static pressure loads on a marine composite panel under large displacement from nonlinear strain measurements collected from the panel.

## Methodology

The methodology employed to generate training data sets and optimise the quantity of sensors based on the performance of various trained ANN as an inverse problem solver for quantifying the load applied to the composite panel is presented in this section. The first stage of the investigation was to design a load quantification methodology for the panel utilising an ANN. In the second stage the load quantification methodology was validated by comparing loads estimated by the ANN with the known loading cases of the panel.

**Inverse Problem Analysis Methodology.** Inverse problem analysis is based on accurately calculating the external loads or boundary conditions that generate a known amount of strain at pre-determined locations on a structure. An ANN, as an inverse problem solver, can be utilised to determine a relationship between the cause and its effect [17]. In this study, the static loads (the cause/output) on a composite panel are quantified by acquiring repeatable strain responses (the effect/input) to these loads from the panel. By introducing examples to an ANN, it can learn the relationships between the input and output through a training process. Once the ANN is sufficiently

trained it can be utilised to estimate the output in real-time. New inputs (problem data) are presented and processed by the ANN to quantify/estimate the load.

**Experimental Setup.** The structure under consideration was a 1 Glass Reinforced Fibre Polymer (GRFP) marine composite panel made of 7 layers of stitched biaxial E-glass cloth and with Ampreg 22 epoxy resin system (all provided by SP Gurit Systems), hand laid up with a total thickness of m (Fig. 1). The panel was divided into a four-by-four grid producing sixteen equal regions of area m<sup>2</sup>. A 120 ohm rectangular strain gauge rosette was attached to the top surface of the panel at the centre of each region giving 16 gauges (S<sub>1-16</sub>) with 48 strain readings (S<sub>1-48</sub>). Four rectangular strain gauge rosettes were also placed in the middle of the panel with 12 strain readings (S<sub>49-60</sub>) attached at the corners of a m<sup>2</sup> square. The specifications of the strain gauges used are detailed in Table 1.

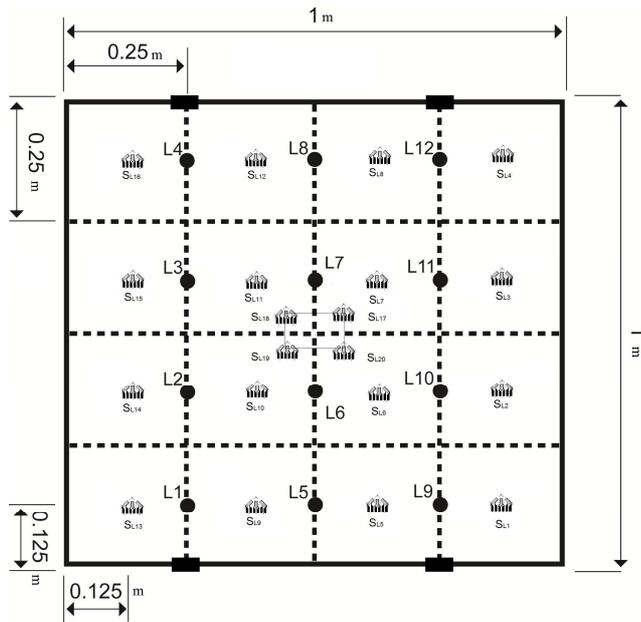


Table 1. Strain gauge specification.

Type	Rectangular Rosette
Resistance	120 [ohms] ±0.6%
Gauge factor	2.100±0.5%
Gauge length	[m]
Gauge width	[m]

Fig 1. Schematic of composite panel indicating strain gauge, loading and support locations.

The bottom surface of the panel was supported at four points using blocks, each 0.035 m wide, 0.009 m deep and 0.025 m high (Fig. 1). Loading was achieved by applying weights normal to the panel surface at the 12 locations indicated in Fig. 1 (L<sub>1-12</sub>). A Windmill 751-SG strain monitoring and control data acquisition system (Windmill Software Ltd, Manchester, UK) with a resolution of +/- 1 microstrain was used to capture the strain data. The 751-SG package comprises of Windmill 6 software, a Universal Serial Bus (USB) unit which provides differential inputs to monitor 16 strain gauges at up to 80 samples per second. Eight USB units can be connected to one computer to monitor up to 128 strain gauges. The strain data was collected through bespoke data acquisition/ANN software linked to the Windmill data acquisition system. This software was developed by the authors in MATLAB (MathWorks, Natick, Massachusetts, USA) utilising Windmill Direct Data Exchange (DDE) protocols to acquire the strain data and the MATLAB Artificial Neural Network Toolbox capabilities.

**Generation of ANN Training Data.** The efficiency of the training data collection process can be increased by reducing the amount of data collected. This is achieved by using the theory of superposition to generate training and testing patterns from the independent parent patterns, as discussed in detail elsewhere [19]. The theory of superposition states that the strain at a point on a structure due to a series of loads is equal to the sum of the strains from each individual load case. Using this theory, an infinite number of training patterns can be generated by applying one known load to each location on the structure individually. However, the superposition method is only valid if the structure under investigation has a linear stress/strain or load/displacement response within the expected load range applied to it. The results of the investigation to study the linearity of the panel indicate a linear relation between load and strain readings. For instance Fig. 2 indicates that the strain changed linearly with load for strain readings S<sub>49</sub> when location L<sub>9</sub> was loaded.

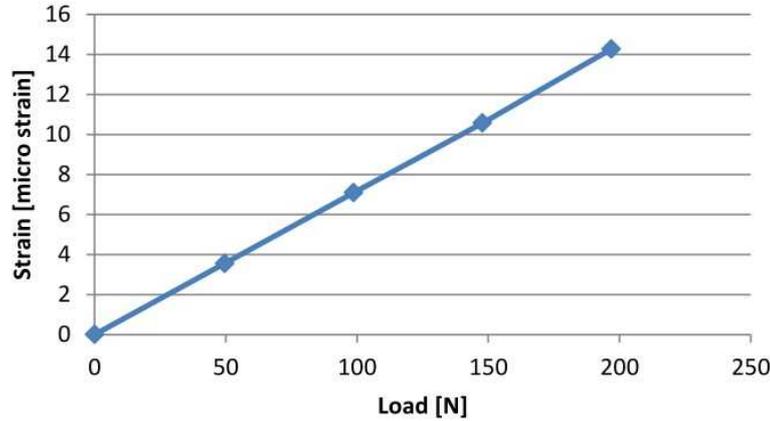


Fig. 2. Linearity of strain data for  $S_{49}$  when loaded on location  $L_9$  is increased from 0 to 196.2 N.

In this study, 48 strain readings ( $S_{1-48}$ ) from 16 rosettes (inputs) and 12 load readings from 12 locations (outputs) were used with the superposition method to generate 1116 sets of training data. This meant that 1116 sets of 12 random loads at each location on the panel and the resultant 48 strains caused by these random loads were acquired to find the relationship between the input/output data. 280 testing patterns were also generated using the superposition method and introduced to the network during training to ensure convergence. 200 noisy patterns were also added to the training data based on the level of noise in the data acquisition system ( $\pm 1$  microstrain). Once training and testing data had been generated it was normalised by converting all data to a range of between zero and one to improve the accuracy of the solution.

**ANN Architecture/Topology.** In this study a common Backpropagation ANN architecture is used and trained employing MATLAB Artificial Neural Network toolbox capabilities. An iterative process was used to determine the optimum network architecture for the panel based on the value of the final Sum of the Squares Errors (SSE) of each network tested. SSE is a measure of the discrepancy between the data and an estimation model. A small SSE indicates a tight fit of the model to the data. Table 2 lists the major parameters of the network architecture used in this study.

Table 2. ANN architecture parameters.

Architecture	Feed Forward Backpropagation
Number of layers in each network	2
Range of load estimation	0-196.2 [N]
No. of inputs (surface strains)	48,18,15,12
No. of output layer neurons (loads)	12
No. of each hidden layer neurons	[20 20]
Number of training patterns	1116
Number of testing patterns	280

**Optimisation.** In order to optimise the number of gauges, various sensor configurations are employed to acquire and generate training data sets and the corresponding ANN is trained. The performance of each ANN in terms of SSE is used for comparison. The aim of this optimisation is to minimise the number of strain gauges needed to reasonably estimate the magnitude of load at 12 loading locations on the composite panel.

In this study, the number rosettes are reduced/optimised in several steps and the effect on the performance of the load estimation methodology is investigated. Due to the inherent characteristic of an ANN, reduction of the number of inputs is limited by the number of outputs. Hence, a minimum of 12 strain inputs are required to be able to successfully estimate 12 load outputs. Therefore, the minimum number of rosettes that could be used to successfully predict 12 loads is 4. In order to optimise the number of the strain gauges, the number of gauges are reduced step by step from 16 (48 readings) to 6 (18 readings), 5 (15 readings) and 4 rosettes (12 readings) respectively. There are multiple permutations for selecting 6, 5 or 4 rosettes from the 16 rosettes attached to the panel. Since

the aim of this study was first to find the optimum number of gauges, random permutations of gauges were selected and the ANN performance using the SSE values for each ANN were compared. It was then possible to determine the optimum number of gauges required to achieve a high quality estimation of the 12 loads from this study. A further strategy for also optimising the sensor locations was then also investigated.

**ANN Validation and Performance.** Finally, the validity of the ANN using the optimised strain gauges was evaluated by comparing the load estimated by the ANN with known loads applied to the panel (problem data). Experimental problem data is the captured strain data from the optimised strain gauges attached to the panel while it is being loaded. It is essential that the strain data is captured at identical locations for both the training and problem data. The first validation study utilised load and strain data generated from the original superposition data collected to produce the training data of the optimised sensor configuration. This meant that any issues with the repeatability of the strains collected for a given load were removed. In the second study, problem strain data for the same sensor configuration was captured directly from the panel under different loading conditions (i.e. one or two random loads were placed at random locations on the panel) and again the estimated loads compared with the actual applied loads.

## Results

**Optimisation.** A selection of the optimisation tests and the ANN performance results are presented in Table 3. The results show that as the number of the rosettes is reduced, the SSE values as performance indicators of various random tests are reasonably small values. This implies that even for cases with only 6, 5 or 4 rosettes, the ANN is trained well and it is capable of estimating the magnitude and position of the applied loads. The results indicated that utilising only 4 rosettes (12 strain readings) it is possible to accurately estimate applied loads on all 12 locations ( $L_{1-12}$ ).

Table 3. Comparison of ANN performance SSE for various sensor configurations.

	16 Rosette Locations ()						Performance SSE
Random Test 1	1-16						1.520
	6 Rosette Locations ()						Performance SSE
Random Test 1	2	3	6	11	14	15	1.670
Random Test 2	3	6	8	9	11	14	0.842
Random Test 3	5	6	7	10	11	12	2.376
Random Test 4	6	7	8	9	10	11	2.038
	5 Rosette Locations ()					Performance SSE	
Random Test 1	2	5	7	10	15	1.442	
Random Test 2	3	6	7	10	114	2.963	
	4 Rosette Locations ()				Performance SSE		
Random Test 1	3	6	11	14	2.643		
Random Test 2	4	7	10	13	2.185		
Random Test 3	5	8	9	12	2.642		
Random Test 4	5	8	9	12	1.208		

Furthermore, in practice, it is more desirable to have the minimum number of strain gauges attached only in one small area on the structure. Hence, 4 new rosettes were attached much closer together in the middle of the panel (Fig. 1,  $S_{L17-20}$ ) and the performance of ANN was investigated. Achieving reasonable results enables the gauges to be placed in just a small portion of the panel and yet be able to accurately estimate the position and the value of 12 externally applied loads. The SSE performance value of this test was 8.259 which is still small value and indicates a tight fit of the model to the data.

**Validation using Superposition Data.** In order to validate the trained network, a set of problem strain data was generated from the panel using the 12 strain readings from the optimised patch (S<sub>49-60</sub>). Since this set of data had not been seen by the network during the training stage, it can be used to evaluate the accuracy of the ANN's load estimation.

The theory of superposition was used to generate problem data (12 loads and 12 strains generated on the panel surface from these loads) for loads between 0 N and 196.2 N. These were introduced to the ANN and the estimated loads calculated by the ANN compared to the expected load profile (Fig. 3).

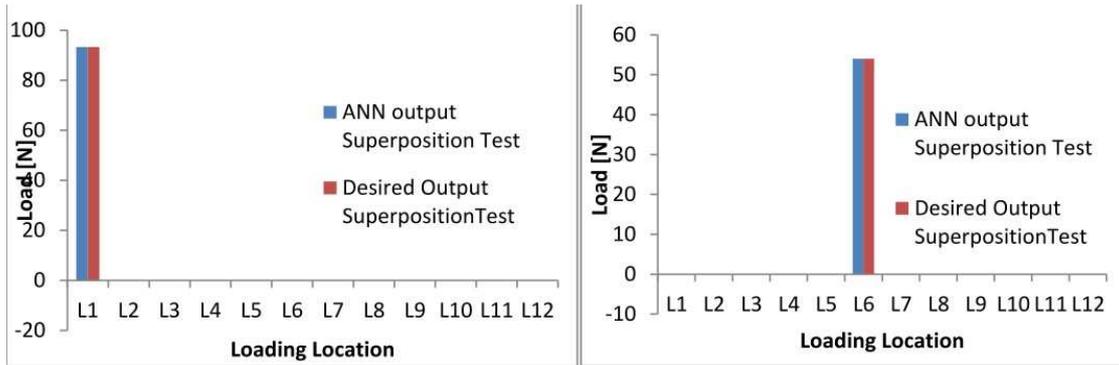


Fig. 3. Estimated load data by ANN vs. expected load values from superposition.

**Validation using Direct Data.** The MATLAB program was developed to be able to acquire strain data directly from the panel and introduce it to the ANN as new problem data. This enabled the system to estimate the applied load on the panel in real-time. The robustness of the system was investigated by introducing strain data gathered by applying loads at the extremes of the range it was trained to estimate (0-196.2 N). Fig. 4 shows typical examples of the comparison between the actual loads applied to the panel with the ANN estimated loads generated from the introduced problem data. In these examples, one load is applied to the panel. For both sets of problem data it can be seen that the ANN can again estimate the load at the loaded locations with a high degree of accuracy.

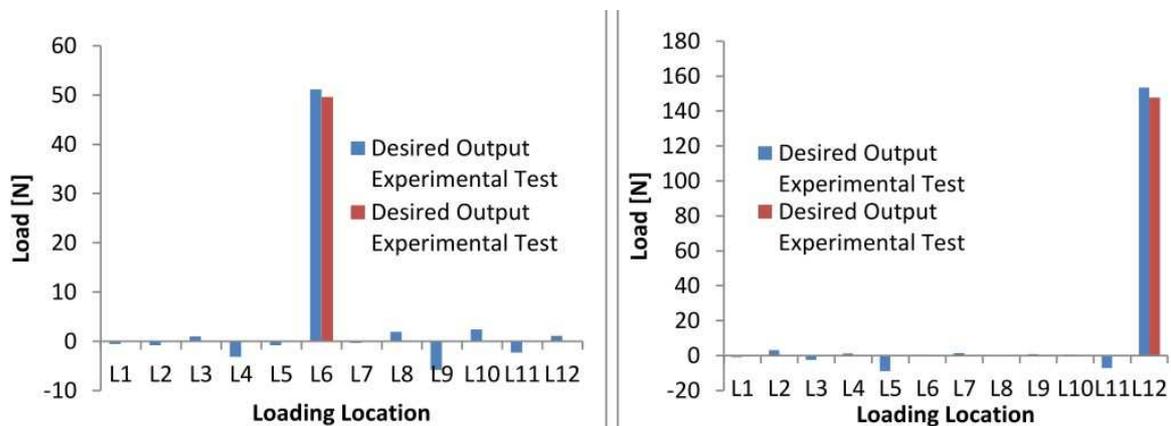


Fig. 4. Estimated load data by ANN Vs. expected load values from experiment.

## Discussion

Establishing an inverse problem analysis approach for structural analysis can result in important advantages over simulation, numerical or theoretical methods. Through utilising such a method, knowledge of the component material constitutive laws and component geometry are not required. In contrast, they are necessary for valid and accurate simulation, numerical or theoretical analysis. The results presented in this paper show that the inverse problem method, utilising an ANN, can accurately estimate the position and magnitude of 12 static loads applied to the composite panel from 4 strain gauge rosettes placed close to each other in centre of the panel. The results indicate that the system always converges, the SSE is small and in the range of acceptable error. This means that an ANN can be trained using experimental data to solve inverse problems and accurately estimate the static loads. Although, the first validation study indicated that the estimated load data by the ANN almost perfectly fits the expected load values from superposition, small error values were seen in second validation study. The main source of error was found to be in the reliability of the data acquisition system utilised due to the large variance in the strain data collected at different time intervals. However, the noise to strain ratio decreases as the load increase which results in less variances in strain data patterns. Having more similar strain data patterns to those employed to train the network leads to better load estimation output.

The ability to measure the actual load history of a craft in-service would enable the designer to validate the load estimation and structural design tools used during the design stage of a craft. This would lead to the development of more optimal structure designs for this type of craft. The operational safety of the craft can also be improved by having a real-time load monitoring system that is able to detect any degradation of the structural integrity and defects within the structure.

## Conclusion

The aim of this research was to establish an inverse load monitoring approach based on directly acquired structural response from an optimised set of sensors. It has been shown that the inverse problem approach can be used to estimate 12 loads applied on a composite marine panel from the strain measurements from 4 strain gauge rosettes. A comparison of the optimised ANN loads with the actual applied loads indicated a very good performance of the methodology. This was achieved in real-time, providing an accurate load history for a component without requiring knowledge of the material properties or component geometry. This potentially makes the system ideal for solving many classes of complex engineering problem that require load monitoring.

## Acknowledgement and Funding

This work was supported by BAE Systems Surface Ships Ltd, Portsmouth and Bournemouth University. The Authors would like to thank Steve Austen, Head of Engineering Support of the Royal National Lifeboat Institution for providing the marine composite panel used in this investigation.

## References

- [1] T. Kukkanen, Wave load predictions for marine structures, *Rakenteiden Mekaniikka (J. Structural Mechanics)*, 43(3) (2010) 150-166.
- [2] Y. Bai, *Marine structural design*, ELSEVIER SCIENCE Ltd, Oxford, 2003.
- [3] B.P. Phelps, *Determination of Wave Loads for Ship Structural Analysis*, DSTO Aeronautical and Maritime Research Laboratory, Australia, 1997.
- [4] X. Guo-dong, D. Wen-yang, Review of prediction techniques on hydrodynamic impact of ships, *J. Marine Science Application*, 8 (2009) 204-210.
- [5] A. Ito, S. Mizoguchi, Hydrodynamic pressure acting on a full ship in oblique short waves, *J. Soc. Nav. Archit. Jpn.*, 222 (1994) 125–32 [in Japanese].

- 
- [6] I. Watanabe, Practical method for diffraction pressure on a ship running in oblique wave, *J. Kansai Soc. Nav. Archit.*, 221 (1994) 83–9 [in Japanese].
- [7] N. Salvesen, E.O. Tuck, O.M. Faltinsen, Ship motions and sea loads, *J. Trans. SNAME*, 78 (1970) 250–87.
- [8] O.M. Faltinsen, *Hydrodynamics of high-speed marine vehicles*, Cambridge University Press, New York, 2005.
- [9] J.J. Jensen, P.T. Pedersen, Wave-induced bending moments in ships: a quadratic theory, *J. Trans. Roy. Soc. Nav. Archit.* 121(1979) 151–65.
- [10] H. Iwashita, A. Ito, T. Okada, M. Ohkusu, M. Takaki, S. Mizoguchi. Wave force acting on a blunt ship with forward speed in oblique sea, *J. Soc. Nav. Archit. Jpn.*, 173 (1993) 195–208 [in Japanese].
- [11] H. Yasukawa, A Rankine panel method to calculate unsteady ship hydrodynamic forces, *J. Soc. Nav. Archit. Jpn.* 168 (1990) 131–40 [in Japanese].
- [12] D. Nakos, P. Sclavounos, Ship motions by a three-dimensional Rankine panel method, *Proc. of 18th symp. on nav. hydrodynamics*, Ann Arbor, (1990) 21–40.
- [13] K. Takagi, Calculation of unsteady pressure by Rankine source method, *J. Kansai Soc. Nav. Archit.*, 219 (1993) 47–56 [in Japanese].
- [14] O.M. Faltinsen, Challenges in Hydrodynamics of Ships and Ocean Structures, *J. Nav. Archit. and Shipbuilding Industry*, 58(3) (2007) 268-277.
- [15] H. Sun, O.M. Faltinsen, Hydrodynamic forces on a semi-displacement ship at high speed, *J. Applied Ocean Research*, 34 (2012) 68– 77.
- [16] X. Cao, Y. Sugiyama, Y. Mitsui, Application of artificial neural networks to load identification, *J. Computers and Structures*, 69 (1998) 63-67.
- [17] M.R. Ramazani, S. Noroozi, M. Koohgilani, B. Cripps, P. Sewell, Determination of static pressure loads on a marine composite panel from strain measurements utilising artificial neural networks, submitted to *Proceedings of the Institution of Mechanical Engineers, Part M: J. Eng. for the Maritime Environment*, (2012).
- [18] V.B. Rao, H.V. Rao, *C++ Neural Networks and Fuzzy Logic*, M & T Books, New York, 1995.
- [19] L. Ziemianski, G. Harpula, The use of neural networks for damage detection in eight storey frames, *Proceedings of the 5th international conference: Engineering applications of neural networks*, (1999) 292–297.
- [20] R. Amali, S. Noroozi, J. Vinney, The application of combined artificial neural network and finite element method in domain problems, *Proceedings of the Sixth International Conference on Engineering Applications of Neural Networks (EANN 2000)*, (2000) 1-7.
- [21] R. Amali, S. Noroozi, J. Vinney, P. Sewell, S. Andrews, Predicting interfacial loads between the prosthetic socket and the residual limb for below-knee amputees: a case study, *J. Strain*, 42(1) (2006) 3-10.
- [22] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain, *J. Psychological Review*, 65(6) (1958) 386–408.
- [23] S. Shar, F. Palmieri, MEKA: a fast, local algorithm for training feed forward neural networks, *Proceedings of the International Joint Conference on Neural Networks*, (1990) 41–46.

- 
- [22] K. Soo-Young, B.Y. Moon, D.E. Kim, S.C. Shin, Automation of hull plates classification in ship design system using neural network method, *J. Mechanical Systems and Signal Processing* 20(2) (2006) 493–504.
- [23] S. Xu, X. Deng, V. Tiwar, M.A. Sutton, W.L. Fournery, An inverse approach for pressure load identification, *Int. J. Impact Engineering*, 37(7) (2010) 865–877.
- [24] R. Amali, S. Noroozi, J. Vinney, P. Sewell, S. Andrews, Improvements in the accuracy of an Inverse Problem Engine's output for the prediction of below-knee prosthetic socket interfacial loads, *J. Engineering Applications of Artificial Intelligence*, 23(6) (2010) 1000–1.

**Paper 3: Using Artificial Neural Networks and Strain Gauges For The Determination of Static Loads On A Thin Square Fully-Constrained Composite Marine Panel Subjected to a Large Central Displacement**

# Using artificial neural networks and strain gauges for the determination of static loads on a thin square fully-constrained composite marine panel subjected to a large central displacement

M R Ramazani, S Noroozi, P Sewell, R Khandan and B Cripps

Submitted 19.07.12

Accepted 06.09.12

*Current methods of estimating the behaviour of marine composite structures under pressure due to slamming as a result of high waves are based on trial and error or oversimplification. Normally under these conditions the non-linearities of these structures are often neglected and, in order to compensate, an overestimated safety factor is employed. These conservative approaches can result in heavier and overdesigned structures. In this paper, a new semi-empirical method is proposed that overcomes some of*

*these problems. This work involved the use of an artificial neural network (ANN) combined with strain gauge data to enable real-time in-service load monitoring of large marine structural panels. Such a tool has other important applications, such as monitoring slamming or other transient hydrostatic loads that can ultimately affect fatigue life. To develop this system, a glass fibre-reinforced polymer (GFRP) composite panel was used due to its potential for providing a non-linear response to pressure or slamming loads. It was found that the ANN was able to predict normal loads applied at different locations on the panel accurately. This method is also capable of predicting loads on the marine structure in real time.*

*Mohammad Reza Ramazani received his MEng degree in mechanical engineering from the University of Birmingham in 2009. He is currently studying for a PhD jointly funded by Bournemouth University and BAE Surface Ships Ltd to investigate in-service load monitoring of marine structures. His interests include artificial intelligence and computer-aided engineering.*

**Keywords:** Composite, marine, artificial intelligence, artificial neural network, structural analysis, load, non-linear structures, large displacement analysis.

*Professor Siamak Noroozi received his PhD from Sheffield University in 1986 in the area of finite element analysis coupled with boundary element analysis. He currently holds the Chair in Advanced Technology at Bournemouth University. His research interests are finite element analysis, boundary element analysis, biomechanics, condition monitoring, general stress analysis, photoelasticity, alternative numerical analysis, composite technology and aeroelasticity.*

## 1. Introduction

In addition to hydrostatic and mass-related loads that can be evaluated with a high degree of accuracy and confidence, it is also desirable to measure transient loads due to slamming as a result of high waves. The current practice to determine wave loads is based on applying standard rules, which often relies on conservative methods due to large uncertainties in the theoretical treatment used for wave load predictions. This leads to a craft that is heavier and slower than it could otherwise be.

*Dr Philip Sewell received his BEng degree in mechanical engineering from the University of the West of England in 1999 and a PhD in the field of prosthetic design in 2003. He is currently employed as a Senior Academic in Design Simulation at Bournemouth University. His research interests include the design of novel tools for prosthetic fitting, the development of techniques to determine prosthetic interfacial pressure distributions and experimental and numerical stress analysis.*

Although sea has an irregular and arbitrary condition, the overall condition can be predicted statistically by superimposing a series of different regular waves of varying heights, lengths, directions and phase<sup>[1-3]</sup>. In order to define the sea-state that the craft are expected to encounter during their lifetime, an enormous amount of data regarding ocean waves has been collected. Hogben *et al*<sup>[4]</sup> collected comprehensive data regarding ocean waves from 104 ocean areas covering all major shipping routes. Having more information about sea states, the wave-induced loads on the craft structure and the response to such loads may be estimated.

*Rasoul Khandan received his BSc degree in mechanical engineering from the Isfahan University of Technology in 2004 and an MSc in mechanical engineering (applied design) in 2008 from Shiraz University. He is currently working on modelling, simulation and optimisation of composite materials as a PhD researcher in the Design Simulation Research Centre at Bournemouth University. His main research interest is modelling and the application of composite and smart materials.*

Techniques used to measure hydrodynamic loads use non-linear equations due to the random and irregular nature of the sea, resulting in a very expensive and time-consuming analysis. Methods have been developed in order to simplify such an analysis<sup>[5]</sup>. Strip theory is one of the most well-known techniques used to determine the wave-induced loads<sup>[6,7]</sup>. The principle is that the craft's hull is divided into a number of segments or strips. The forces acting on the hull are then calculated separately on each segment using a two-dimensional flow theory. This method ignores the longitudinal component of relative velocity and any type of interaction between the different segments. Other shortcomings of this theory include ignoring three-dimensional or viscous effects

*Professor Bob Cripps received his degree in ship science from the University of Southampton in 1976. He was awarded an Honorary Doctor of Engineering from Bournemouth University in 2005. He is currently a Director of Longitude Consulting Engineering, part of London Offshore Consultants (LOC). He is a Visiting Professor at Bournemouth University and a Royal Academy of Engineering Visiting Professor in the Principles of Engineering Design at the University of Southampton.*

*Mohammad Reza Ramazani\*, Siamak Noroozi, Philip Sewell and Rasoul Khandan are with the School of Design, Engineering & Computing, Bournemouth University, Bournemouth, UK.*

*Bob Cripps is with Longitude Consulting Engineers Ltd, Southampton, UK.*

*\*Corresponding author. Tel: 01202 961528; Email: mramazani@bournemouth.ac.uk*

as well as the inability to account for the above-water hull form. In order to resolve the problem with compatibility between strips, flexible beam strip theories were developed that account for the bending and shear stiffness of the hull<sup>[8]</sup>. Although this kind of theory can estimate the distortional higher frequency responses of a hull to slamming and lashing excitation, it is still linear analysis and extreme response is not well modelled.

The accuracy of the strip theory and other codes has been investigated by several researchers and the error associated with predicting a mid-ship bending moment using strip theory is of the order of 10% to 20%. This accuracy is reduced further towards the ends of the vessel and as seas become progressively more beam-on<sup>[9]</sup>. Clarke<sup>[10]</sup> conducted many on-board measurements employing several Royal Navy (RN) ships. The results indicated that strip theory over-estimates wave bending moments, particularly at larger wave heights. Furthermore, the hogging bending moment was over-predicted more so than the sagging moment. It is concluded that these techniques are only accurate for moderate sea conditions and ship speeds meaning an extreme load causing a large displacement in panels is impossible to measure. Moreover, doubts also exist in many of the assumptions that involve stochastic/random data or procedures involving environmental and operational conditions. This is due to the fact that sometimes environmental and operational conditions are difficult to define accurately in advance and therefore assumptions are needed<sup>[11]</sup>.

In order to improve the accuracy of estimation, especially in short waves, many numerical methods considering the three-dimensional effects have been introduced. Among them are the three-dimensional Green function method<sup>[12]</sup> and Rankine source method<sup>[13-15]</sup> based on three-dimensional potential theory. The benefits of these methods include taking the three-dimensional effects into account, having good stability of computations and a moderate computing time. Hence, they are considered as suitable design tools replacing the strip methods.

A review by Phelps<sup>[9]</sup> indicates that non-linear theories and three-dimensional load prediction methods have been introduced, but these require greater computational effort and have not yet proven to be significantly more accurate than the two-dimensional methods. It is concluded that a novel technique is required to overcome current limitations in the practices used to measure and estimate loads experienced by the hull of a small, high-speed boat operating in a seaway. Furthermore, as vessels and craft are, in most cases, extremely complicated structures, the mechanical properties, or relationship between externally-induced excitation and structural responses, are difficult to formulate. An appropriate load monitoring system and technique has to be developed for naval assets and large structures<sup>[16]</sup>.

A novel approach for the determination of pressure loads experienced by marine structures is the utilisation of artificial neural networks (ANN) as an inverse method. In a study by Cao *et al*<sup>[16]</sup>, an approach was developed to identify the loads acting on aircraft wings, where an ANN was utilised to model the load-strain relationship for structural analysis. The research demonstrated that using an ANN to identify loads is feasible and a well-trained ANN reveals an extremely fast convergence and a high degree of accuracy in the process of load identification for a cantilevered beam model. In a study by Amali *et al*<sup>[17]</sup>, it is illustrated that ANN can be combined with experimental methods to create a hybrid inverse problem analysis tool or inverse problem engine. The hybrid approach can be applied to both direct problems (calculation of the structural response from known loads applied to the structure) and inverse problems (calculation of the applied load from a known structural response). Additionally, the approach avoids the need to have information on the component geometry and material properties<sup>[18,19]</sup>.

Ramazani *et al*<sup>[20]</sup> have recently shown that the inverse problem approach can be used to estimate low loads applied on a

composite marine panel from a small deflection and its associated strain measurements. A comparison of the ANN loads with the actual applied loads indicated a very good performance of the methodology. This was achieved in real time, providing an accurate load history for a component without requiring knowledge of the material properties or component geometry. However, a large load results in a large displacement in the panel, where the displacement is no longer predictable. This implies that the superposition method of generating training data for a small displacement can no longer be applied here. However, marine structures do experience large displacement and for that reason load prediction is essential. This paper reports on the research undertaken to further develop the ANN methodology to quantify static pressure/central load on a composite marine panel from its non-linear structural response.

## 2. Methodology

The methodology employed to evaluate the suitability of an ANN as an inverse problem is presented in this section. A backpropagation ANN was designed, developed and trained within the Matlab simulation environment (Mathworks, Natick, Massachusetts, USA) to measure transverse load on a flat composite marine panel. The estimated output was then validated by comparing it against both experimental and numerical data.

### 2.1 Inverse problem analysis methodology

Inverse problem analysis is based on accurately calculating the external loads or boundary conditions that generate a known strain at pre-determined locations on a structure. An ANN, as an inverse problem solver, can be utilised to determine a relationship between the cause and its effect<sup>[20]</sup>. In this study, the static loads (the cause/output) on a composite panel are quantified by acquiring repeatable strain responses (the effect/input) to these loads from the panel. Introducing these examples to an ANN, the system can learn and form the relationships between the input (strains) and output (load) through the transfer function. The ANN requires a number of known input and output data for training (*ie* relating the ANN inputs to outputs using a transfer function and series of weighting values). Once the ANN is sufficiently trained it can be utilised to estimate the output in real time. New inputs (problem data) can then be presented and the load can be estimated in real time.

### 2.2 Experimental set-up

The structure under consideration was a 1 m<sup>2</sup> glass fibre-reinforced fibre polymer/plastic (GFRP) marine composite panel (Figure 1). The sample GFRP composite panel used was made of seven layers of stitched biaxial  $\pm 45$  E-glass cloth with Ampreg 22 epoxy resin system, hand laid-up with a total thickness of  $5 \times 10^{-3}$  m. The fibres were aligned parallel to the edges of the panel. Table 1 shows the experimental mechanical properties of the glass fibre as provided by the manufacturer.

The panel was divided into a four-by-four grid producing sixteen equal regions, each with an area of  $0.25 \times 0.25$  m<sup>2</sup> (Figure

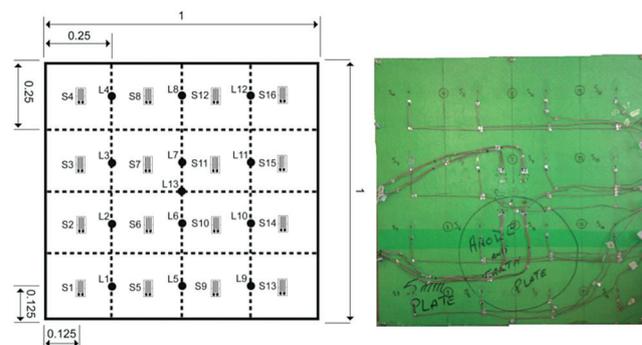


Figure 1. Schematic of composite panel indicating strain gauge and loading locations

**Table 1. Panel material specification provided by SP Gurit Systems (Newport, Isle of Wight, UK)**

Material name			XE905		
Material type			Stitched biaxial		
Fibre volume fraction			0.46		
Longitudinal property		Units	Units		
Longitudinal tensile modulus	N/mm <sup>2</sup>	21220	Poisson's ratio (longitudinal strain)		0.120
Longitudinal tensile strength	N/mm <sup>2</sup>	318.3	Poisson's ratio (transverse strain)		0.120
Longitudinal compressive modulus	N/mm <sup>2</sup>	21220	Longitudinal coeff. of thermal expansion	10-6/°K	14.62
Longitudinal compressive strength	N/mm <sup>2</sup>	254.6	Transverse coeff. of thermal expansion	10-6/°K	14.62
Transverse property			Density	kg/m <sup>3</sup>	1786
Transverse tensile modulus	N/mm <sup>2</sup>	21220	Structural ply thickness	mm	0.75
Transverse tensile strength	N/mm <sup>2</sup>	318.3	Actual ply weight	g/m <sup>2</sup>	1364
Transverse compressive modulus	N/mm <sup>2</sup>	21220	Shear thickness	mm	0.75
Transverse compressive strength	N/mm <sup>2</sup>	254.6			
Shear properties			Derived shear properties @ ±45°		
Interlaminar shear modulus	N/mm <sup>2</sup>	3050	Shear material name	1 x XE905 @ ±45°	
Interlaminar shear strength	N/mm <sup>2</sup>	36.6	Axial modulus with fibres @±45°	N/mm <sup>2</sup>	9737
In-plane shear modulus	N/mm <sup>2</sup>	3050	Shear modulus with fibres @45°	N/mm <sup>2</sup>	9471
In-plane shear strength	N/mm <sup>2</sup>	46.1	Poisson's ratio with fibres @±45°		0.596

1). The bottom surface of the panel was supported on all four edges using aluminium bars, each 0.0381 m high, 0.01905 m wide and 1 m long. Sixteen linear electrical resistance strain gauges (ERSG) (S1-16) were bonded to the centre of each region (specification in Table 2). Two eight-channel NI cDAQ 9236 modules mounted on NI cDAQ 9174 chassis (National Instruments Corporation, Austin, Texas, USA) were used as the strain monitoring and control data acquisition system with a resolution of +/- 0.1 microstrain. The system provides differential inputs to monitor sixteen strain gauges at up to 10,000 samples per second. The strain data was collected using Matlab, utilising Matlab Data Acquisition Toolbox capabilities.

**Table 2. Strain gauge specification**

Type	General purpose linear gauge
Resistance	350 ohms ± 0.6%
Gauge factor	2.100 ± 0.5%
Gauge length	6.35 mm
Gauge width	2.54 mm

Normal loads were randomly applied to the top surface of the panel at thirteen grid intersections (L1-13). Depending on the proximity of the gauge to the applied loads, different gauges exhibited different levels of sensitivity, which was as expected. To produce efficient training data the strain data should be captured at the sensitive regions (*ie* the strain at those locations must vary significantly due to changes in load level). In addition, the strain data collected must provide a unique response for each load distribution. If strain is collected from non-sensitive regions of the panel and/or the strain data collected is not unique for each load distribution the ANN is less likely to be able to find a function relating the input and output.

**2.3 Generation of training data**

Many small marine craft hulls are manufactured from fibreglass strengthened by wood or foam. Their characteristics are such that their thickness is small compared to their other dimensions. In this study, a GFRP panel has been utilised to represent a section of the hull. Panels can be classified according to their thickness and their lateral deflection compared to their thickness<sup>[21]</sup>. They can be

classified as: (1) thick plate, small deflection; (2) thin plate and small deflection; (3) thin plate, large deflection; or (4) very thin plate (membranes) with either small deflection or large deflection. In all cases the solutions are approximate, not exact or closed form. The deflection at the centre of a plate subject to pressure is offered by Westergaard and Slater<sup>[22]</sup> and is based on the modified flexure theory of plates where, depending on the plate aspect ratio, edge boundary conditions and load, different approximate empirical solutions are found. In such cases, a small displacement is defined as displacement less than or equal to half the thickness of the plate. If the displacement exceeds this limit then the problem is treated as a non-linear problem where the displacement can no longer be accurately predicted using the above theory. This is due to highly non-linear double curvature deformation, unlike the displacement function stated above. In large displacement analysis, the transverse shear can also no longer be ignored and if the panel is composite then the transverse shear requires further special treatment. In such cases, the classical inverse approach used previously, based on utilising data generated from superposition, can no longer be employed due to the complexity of the displacement function.

For non-linear structures an alternative approach is needed in order to generate the required training data. There are two ways in which such data can be generated: (a) experimentally; or (b) using a non-linear finite element analysis (FEA) solver. Generating the required training data experimentally is very time consuming and labour intensive. Therefore, non-linear FEA analysis using a script that allowed automatic generation of a random load on the panel was utilised to generate the training data. Abaqus 6.10-1 FEA software (Dassault Systèmes Simulia Corp, Rhode Island, USA) was used. A script function written in Python language was used to iteratively run the software in a batch using different random loads applied at each of the thirteen loading locations on the panel. The FEA model was initially validated to ensure that it represented the actual panel accurately. The validation was achieved by comparing strains collected experimentally with the FEA strains under the same loading conditions. Loads from 100 N to 800 N applied in 100 N increments were placed on the panel one at a time at locations L1 to L13. The strain readings at locations S1 to S16 on the panel were saved for each test. The same tests were performed with FEA to compare with the experimental results.

Once validated, a large number of training (load/strain

response) data was able to be generated from the FEA model. In order to increase the efficiency of generating the training data, it was possible to reduce the number of FEA models required to establish the non-linear strain response for each gauge location. This was achieved by fitting non-linear curves to data collected for each strain location and using the curves to interpolate strain data for different load magnitudes.

The structural responses of the panel in terms of strain were saved to be used as the input training dataset. The corresponding load for each input dataset was also saved and utilised as the output training set. Some of these input and outputs were saved separately for testing the network and error minimisation. In this study, sixteen single strain gauge readings (inputs) and thirteen applied loads (outputs) constitute one training dataset. At each loading location (L1-L13), a static load ranging between 24.525 N and 784.8 N was applied in steps of 24.525 N. In total, 1040 training datasets were generated from the non-linear FEA model.

### 2.4 ANN architecture/topology

ANN analysis often requires a high number of individual loops to determine the best solution. However, the training time can be reduced (*ie* reduce the number of loops to minimise the error equation) by pre-processing the data that is given to the network to train. Having multiple hidden layers of neurons with non-linear transfer functions (such as tan-sig and log-sig) enables the network to understand both non-linear and linear relationships between input and output data. Unsatisfactory performance of the ANN can be due to a wide range of reasons, such as:

- an unsuitable ANN architecture or learning method;
- insufficient representative data (not enough example strain/load data);
- inadequate pre-processing (noisy data from data acquisition system ignored);
- unsuitable ANN training parameters.

Most of the time this is not the case and the ANN will be well trained and perform satisfactorily, even on a new untrained dataset. Key architectural issues that can be optimised include: (i) the number of layers in the ANN; (ii) the number of neurons per layer; (iii) the type and parameters of the neuron, which are usually the same throughout; and (iv) the number of calculations per iteration during learning and recall.

The Matlab Artificial Neural Network toolbox was used in this study to generate two different backpropagation ANN architectures in order to compare their performance. The architectures utilised were:

- One network with sixteen neurons in the input layer and thirteen neurons in the output layer is trained to estimate the load on the panel from the strain responses (Figure 2).
- Thirteen networks each with sixteen neurons in the input layer and one neuron in the output layer are trained and used to estimate the load on the panel from the strain responses (Figure 3).

The number of hidden layers and neurons in each hidden layer of the two network architectures were flexible. These were dependent on the complexity of the training datasets and were optimised according to the network performance. The sum of squared errors (SSE) and mean of squared errors (MSE) are common network performance indicators. Through the testing of various network architectures, the optimum network having the lowest performance indicator can be determined. Once the ANN is trained, it can be employed to estimate new loading cases where the same patterns exist. In other words, whenever the same pattern of

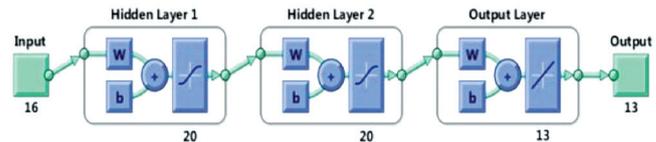


Figure 2. Matlab representation of ANN architecture 1

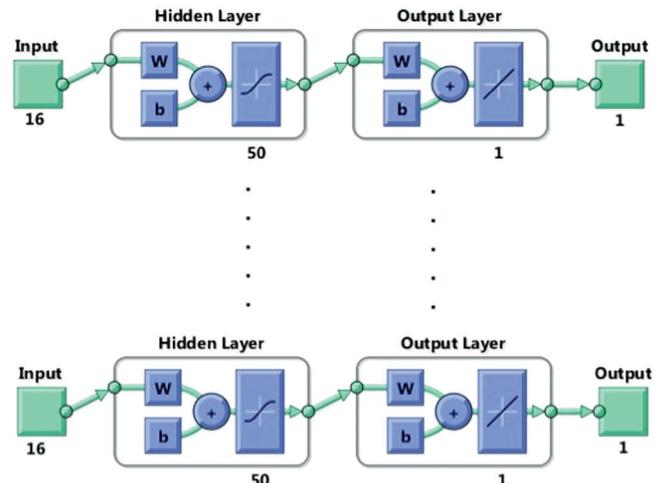


Figure 3. Matlab representation of ANN architecture 2

strain reading as an input is introduced to the network, it will be able to estimate the loads that caused those structure responses. Depending on how well the network is trained (the performance of the network), there will be an error between the output dataset and the network estimated output (load).

### 2.5 ANN validation and performance

The validity and performance of the ANN method was evaluated by comparing the load estimated by the ANN with known loads applied to the panel, which were not seen by the network during the training process. The first validation study utilised load and strain data generated from the FEA model and was compared with estimated loads from the ANN. In the second study, problem strain data was captured directly from the panel and again the estimated loads were compared with the actual applied loads.

## 3. Results

The validity of utilising FEA for training data generation and the ANN validity and performance are detailed in the following sections.

### 3.1 FEA model validation

Figure 4 indicates that for loading only location L13, there is reasonable agreement between the strain readings (S6 and S10) of FEA tests and experimental tests. The average percentage error is

Table 3. ANN architectures

	1 network with 16 strain inputs and 13 load outputs	13 networks each with 16 strain inputs and 1 load output
Number of networks	1	13
Architecture	Feed forward backpropagation	
Number of layers in each network	2	1
Range of load estimation	24.525-784.8 (N)	24.525-784.8 (N)
No of inputs (surface strains)	16	16
No of output layer neurons (loads)	13	1
No of each hidden layer of neurons	[20 20]	[50]
Number of training patterns	1040	1040
Number of testing patterns	1040	1040

less than 7%. These results indicate that the FEA model can be used confidently to simulate various loading conditions and to generate the required training input data.

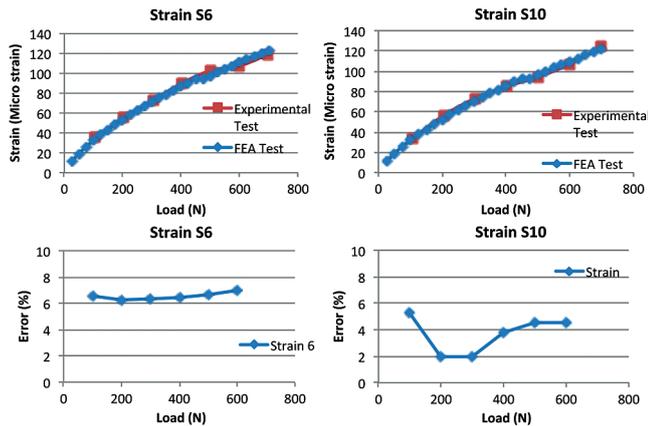


Figure 4. Comparison of FEA and experimental data of selected strain gauges

### 3.2 ANN validation and performance

As mentioned in Section 2.4, two different methods are employed to define the networks. Table 3 lists the major parameters of the network architecture used in the two methods. It was determined, through the testing of various network architectures, that the optimum network (lowest SSE) for method one had two hidden layers with twenty neurons and used a tan-sig transfer (Figure 2). The output layer had thirteen neurons (representing the thirteen loads to be estimated) and used a pure-lin transfer function. Similarly, it was determined that the thirteen networks for method two had one hidden layer each with fifty neurons and used a tan-sig transfer (Figure 3). The output layer of each network had one neuron (each network estimates the corresponding load of one location) and used a pure-lin transfer function.

In this study, SSE is used as a performance indicator. Once the networks were trained, SSE values between the estimated loads and training load data were calculated. Each network has an individual SSE value. This means that although the first method has only one SSE value, the second method had thirteen SSE values. Figure 5 indicates the SSE performance of all thirteen networks, each having sixteen inputs (all strain readings) and one output (load at one location) generated from the second network architecture.

In order to compare the two methods, the summation of all the networks' SSE values in the second method is compared to the SSE value of the first method, when only a network with sixteen inputs and thirteen outputs were used to train the system. As it is indicated in Figure 6, a better performance for the second method is achieved.

In addition to having a better performance, the second method has more flexibility. This means having thirteen independent networks; for each load location a separate new network architecture and parameters can be employed. For instance, the sum of the estimation performances of networks in the second method can be improved by changing the network architecture of those networks (eight and twelve from Figure 5) having relatively higher SSE values. As it is illustrated in Table 4, for locations eight and twelve, networks with two layers with twenty neurons are used. The improvement in SSE for networks eight and twelve with the new architectures can be seen in Figure 7.

In order to investigate the capability of the ANN to estimate loads in real time, once the ANN is trained new strain data from different

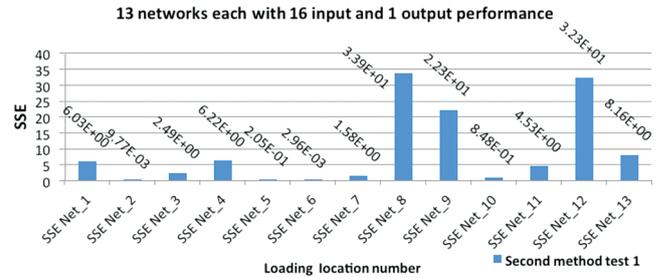


Figure 5. SSE performance of network architecture 2

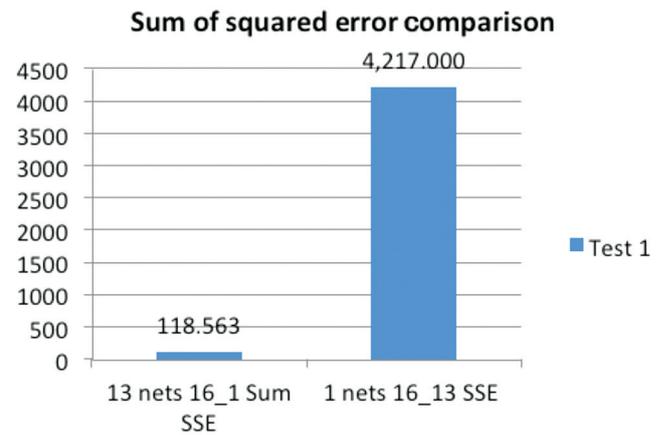


Figure 6. Comparison of the SSE values of the two network architectures

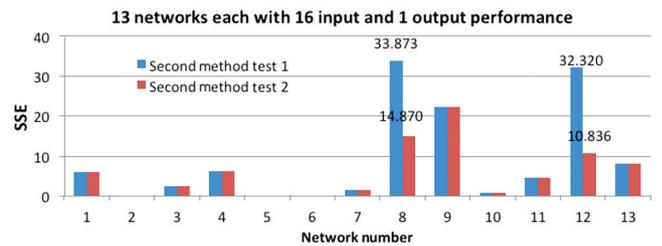


Figure 7. Flexibility of ANN architecture 2 in training stage

loading cases were introduced to it. Having a good performance, the ANN should be able to estimate the external pressure loads that caused those structure responses. For instance, introducing new sets of strain data that have not been used to train the network, the ANN estimates the corresponding load data. Depending on how well the network is trained (the performance of the network), there will be errors between the expected output dataset and the network estimated output (loads). Figure 8 depicts three random examples of the comparison of the ANN results (trained with both FEA and experimental data) with actual externally applied loads of 300 N, 200 N and 300 N applied individually at locations L13, L1 and L7, respectively. For both sets of problem data it can be seen that the ANN can again estimate the load at the loaded locations with a high

Table 4. Optimum ANN architecture 2

	13 networks each with 16 strain inputs and 1 load output
Number of networks	13
Architecture	Feed forward backpropagation
Number of layers in each network	Most of it has 1 and for location 8 and 12 are 2
Range of loads	0-809.3 (N)
Number of inputs (surface strains)	16
Number of neurons in output layer (normal loads)	1
Number of neurons in each hidden layer	[50] or [20 20]
Number of training patterns	1040
Number of testing patterns	1040

degree of accuracy. However, the error size of estimated loads with the ANN for experimental tests is slightly bigger. Such a small error is normal and it could be from an initial error between the FEA data and experimental data, errors induced from the repeatability of the data acquisition system with a resolution of  $\pm 0.1$  microstrain as well as possible overtraining of the ANN.

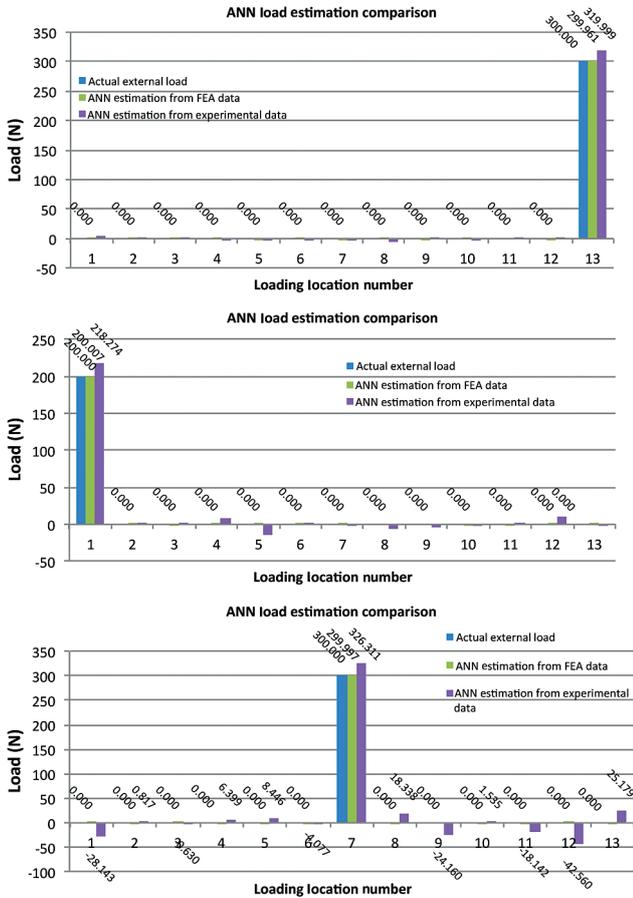


Figure 8. ANN estimation for FEA and experimental data versus expected real data

The estimated negative load values at the unloaded locations were due to the differences between the strain data collected to generate the training data and the collected problem strain data. Due to these errors, slightly different strain patterns are introduced to the ANN producing the errors in the estimated loads. The introduction of further noisy patterns in the training dataset may reduce these small errors, indicating that further work could be carried out to improve the accuracy further.

#### 4. Discussion

In this study, it is shown that the inverse problem method, utilising an ANN, is capable of estimating magnitude and position of the static pressure loads on a marine composite panel under large displacement from non-linear strain measurements. The results of this study can be summarised as follows:

- FEA data can be used to generate training data for ANN inverse load estimation problems.
- Two different ANN architectures are used and the performances are compared.
- Having non-linear relationships between the applied load and the surface strains, the system always converges and the SSE is in the range of acceptable error.
- The system is capable of estimating the position and magnitude of static pressure loads on a marine composite panel under large displacement.

- Having a large difference between the training datasets and the problem dataset makes the ANN unable to estimate the load accurately.
- The main source of error was found to be an initial error between the FEA data and experimental data.

The ability to measure the actual load history of a craft in-service would enable the designer to validate the load estimation and structural design tools used during the design stage of a craft. This would lead to the development of more optimal structure designs for this type of craft. The operational safety of the craft can also be improved by having a real-time load monitoring system that is able to detect any degradation of the structural integrity and defects within the structure.

It is proposed that the ANN methodology, with further research and development, could be utilised for the quantification of in-service, transient loads in real-time acting on the craft from the craft's structural response (strain response to load). This would provide valuable information to influence future craft design. In order to fully evaluate the proposed methodology for in-service load monitoring of marine structures, the following areas require investigation:

- The behaviour of marine structures under transient load conditions (dynamic load is applied).
- The effect of the size of the structure on the ANN estimation accuracy.
- The number of sensors required for accurate load estimation by optimising the method. While some vessels do have integrated sensors most do not. The number of sensors should be minimised to reduce the time to train the system, cost and weight.
- The effect of modifying ANN training parameters, including the number and type of training patterns introduced to the ANN.
- Validation of the methodology on a craft in-service.

Finally, a graphical user interface (GUI) should be developed allowing control of various parameters of the data acquisition and load monitoring system, as well as graphical display in real time.

#### 5. Conclusions

It has been shown that the inverse problem approach can be used to estimate the magnitude and position of static pressure loads on a marine composite panel under large displacement from non-linear strain measurements. A comparison of the ANN loads with the actual applied loads indicated a very good performance of the methodology. This was achieved in real time, providing an accurate load history. This potentially makes the system ideal for solving many classes of complex engineering problem that require load monitoring.

#### References

1. W J Pierson, 'A unified mathematical theory for the analysis of propagation and refraction of storm-generated ocean surface waves', part i and ii, New York University, New York, 1952.
2. M St Denis and W J Pierson, 'On the motions of ships in confused seas', SNAME Transactions, Vol 61, 1953.
3. W J Pierson, G Neumann and R W James, 'Practical methods for observing and forecasting ocean waves by means of wave spectra and statistics', Hydrographic Office Publication, 1955.
4. N Hogben, N M C Dacunha and G F Olliver, Global Wave Statistics, Unwin Brothers Ltd, UK, 1986.
5. Y Bai, Marine Structural Design, Elsevier Science Ltd, Oxford, 2003.
6. B V Korvin-Kroukovsky, 'Investigation of ship motions in regular waves', SNAME Transactions, Vol 63, pp 386-435, 1955.
7. J Gerritsma and W Beukelman, 'The distribution of the

- hydrodynamic forces on a heaving and pitching ship model in still water', paper presented at the Fifth Symposium on Naval Hydrodynamics, Washington, 1964.
8. R E D Bishop and W G Price, *Hydroelasticity of Ships*, Cambridge University Press, Cambridge, 1979.
  9. B P Phelps, 'Determination of wave loads for ship structural analysis', DSTO Aeronautical and Maritime Research Laboratory, Melbourne, 1997.
  10. J D Clarke, 'Wave loading of warships', *Journal of Naval Science*, Vol 12, No 4, 1986.
  11. T Kukkanen, 'Wave load predictions for marine structures', *Rakenteiden Mekaniikka (Journal of Structural Mechanics)*, Vol 43, No 3, pp 150-166, 2010.
  12. H Iwashita, A Ito, T Okada, M Ohkusu, M Takaki and S Mizoguchi, 'Wave force acting on a blunt ship with forward speed in oblique sea', *Journal of Society Naval Architecture Japan*, Vol 176, 1994.
  13. D Nakos and P Sclavounos, 'Ship motions by a three-dimensional rankine panel method', paper presented at the Proceedings of 18th Symposium on Naval Hydrodynamics, Ann Arbor, 1990.
  14. K Takagi, 'Calculation of unsteady pressure by rankine source method', *Journal of Kansai Society of Naval Architecture*, Vol 219, pp 47-56, 1993.
  15. H Yasukawa, 'A rankine panel method to calculate unsteady ship hydrodynamic forces', *Journal of Society Naval Architecture*, Vol 168, pp 131-140, 1990.
  16. X Cao, Y Sugiyama and Y Mitsui, 'Application of artificial neural networks to load identification', *Computers & Structures*, Vol 69, pp 63-67, 1998.
  17. R Amali, S Noroozi and J Vinney, 'The application of combined artificial neural network and finite element method in domain problems', paper presented at the Sixth International Conference on Engineering Applications of Neural Networks (EANN 2000), Kingston upon Thames, 2000.
  18. R Amali, S Noroozi, J Vinney, P Sewell and S Andrews, 'Predicting interfacial loads between the prosthetic socket and the residual limb for below-knee amputees – a case study', *Strain*, Vol 42, pp 3-10, 2006.
  19. P Sewell, S Noroozi, J Vinney, R Amali and S Andrews, 'Improvements in the accuracy of an inverse problem engine's output for the prediction of below-knee prosthetic socket interfacial loads', *Engineering Applications of Artificial Intelligence*, Vol 23, pp 1000-1011, 2010.
  20. M R Ramazani, S Noroozi, M Koohgilani, B Cripps and P Sewell, 'Determination of static pressure loads on a marine composite panel from strain measurements utilising artificial neural networks', submitted to Proceedings of the Institution of Mechanical Engineers, Part M: J Eng for the Maritime Environment, 1475090212449231, first published on 25 June 2012, doi:10.1177/1475090212449231
  21. A P Boresi, R J Schmidt and O M Sidebottom, *Advanced Mechanics of Materials*, Fifth Edition, John Wiley, 1993.
  22. H M Westergaard and W A Salter, 'Moments and stresses in slabs', *Proceedings of Amer Concrete Inst*, Vol 17, pp 415-538, 1921.

Published by The British Institute of Non-Destructive Testing  
on behalf of its Condition Monitoring Group (COMADIT)

# Infrared Thermography Handbooks



## Volume 1: Principles and Practice

by Norman Walker

This book will provide guidance on the subject of Infrared Thermography (IRT), namely heat transfer theory, equipment selection, applications and operational factors; and as such can be used as an information source to support IRT operations.

## Volume 2: Applications

by A N Nowicki

This book is aimed at thermographers who have attained at least Level 1 accreditation in IRT. It is recommended that the reader also has some operational experience in order to have the required level of knowledge to understand the terminology and diagnostic descriptions included.

Volume One: Principles & Practice – BINDT Members: £60; Non-Members: £65

Volume Two: Applications – BINDT Members: £60; Non-Members: £65

Both Volume One & Two – BINDT Members: £100; Non-Members: £110

Available from: The British Institute of Non-Destructive Testing, Newton Building, St George's Avenue, Northampton NN2 6JB, UK. Tel: +44 (0)1604 89 3811; Fax: +44 (0)1604 89 3861; Email: info@bindt.org

Order online via the BINDT Bookstore at [www.bindt.org/Books](http://www.bindt.org/Books)

Copyright of Insight: Non-Destructive Testing & Condition Monitoring is the property of British Institute of Non-Destructive Testing and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.

## **Paper 4: In-service Transient Load Measurement on a Large Composite Panel**

## In-service Transient Load Measurement on a Large Composite Panel

Mohammad Reza Ramazani<sup>1,a</sup>, Philip Sewell<sup>1,b</sup>, Siamak Noroozi<sup>1,c</sup>,  
Mehran Koohgilani<sup>1,d</sup> and Bob Cripps<sup>2,e</sup>

<sup>1</sup>School of Design, Engineering & Computing, Bournemouth University, Poole, UK.

<sup>2</sup>Longitude Consulting Engineers Ltd, Southampton, UK.

<sup>a</sup>mramazani@bournemouth.ac.uk, <sup>b</sup>psewell@bournemouth.ac.uk, <sup>c</sup>snoroozi@bournemouth.ac.uk,  
<sup>d</sup>mkoohgil@bournemouth.ac.uk, <sup>e</sup>b.cripps@longitude.uk.com

**Keywords:** Composite, Marine, Artificial Intelligence, Artificial Neural Network, Structural Analysis, Impact Load, Slamming

**Abstract.** Current practices to estimate the pressure loads on the hull of small high-speed craft in a seaway are based on determination of the wave loads by applying rules and standards which itself relies either on often conservative methods, leading to a craft that is heavier and slower than it could be otherwise. There are rather large uncertainties in the wave load predictions for ships mainly caused by not necessarily sufficient theoretical basis of the calculation methods. Direct pressure measurement techniques can only provide data at each transducer location and classical analytical techniques require a large amount of experimental data to be collected to relate pressure to the structures response. The evaluation of wave generated hydrodynamic loads is less reliable as the dynamic nature of the loading as well as transient effects such as slamming and green water on deck still demands more investigations. Therefore, a novel technique is required to overcome these limitations by providing a method of measuring the pressure load with relatively few sensors and minimal data collection. This paper reports on research undertaken to develop an inverse problem approach utilising an Artificial Neural Network (ANN) for quantification of in-service, transient loads in real-time acting on the craft from the craft's structural response (strain response to load). This study investigates suitability and performance of utilising ANN as an inverse problem approach to estimate impact loads applied to up to 13 locations on the structure in real-time from 16 strain measurements.

### Introduction

The measurement of hydrodynamic impact loads has an important role in the design of reliable cost and weight effective marine structures. Weight is a major factor to optimise the speed of a marine structure. Although employing stiff and light materials such as composites leads to increase in speed, structural damage is still significant [1]. It is expected that the structural damage may be influenced by global hydroelastic behaviour from waves and/or local hydrodynamic impact loads from slamming [2]. Wave impact is a random nonlinear phenomenon which is very sensitive to relative motion and attack angle between the body and free surface of the water. Since the duration of wave impact loads is very short, hydroelastic effects are large. In addition, due to air trapping, the wave impact phenomenon is difficult to describe. As the new generation of high speed naval craft gets larger and faster, slamming impact loads on these vessels becomes a critical design concern. However, the hydrodynamic impact load is one of the least understood areas of marine structure design [3]. Wave impact has challenged many researchers for more than half century and yet more research for an accurate practical estimation method of wave impact loads is required.

Many studies have been performed so far for evaluating hydrodynamic loads on ships. Such methods are summarised in the reviews by Phelps [4] and Guo-Dong and Wen-Yang [3]. It is indicated that the evaluation of wave generated hydrodynamic loads is less reliable than the static loads and there is less guidance as to how to handle the dynamic nature of the loading as well as transient effects such as slamming. In the past four to five decades, research has provided increased knowledge of the nature of hydrodynamic loads, which together with the improvements in computing

power have greatly enhanced the capability to determine the effects of these loads on ship structures. Starting with the work of von Karman [5], various research works have been carried out to describe hydrodynamic loads in Naval Architecture, on motorboats and sailboats in parallel with rules and regulations [6-19]. Furthermore, the introduction of composite materials in marine architecture, such as fast marine crafts, has brought new types of operational failures in panels. In contrast to metallic materials, when the design makes use of composite materials, different types of cracks can appear such as delaminations. This usually happens due to the localised impacts from service loads (such as slamming loads) often observed in composite high-speed crafts. Although, many studies have been carried out to describe the slamming phenomenon and its consequences on the design of ships made of metallic materials [10, 20-26], there are limited studies in this area for composite marine structures. In addition, as it is pointed out in various studies [1, 27], there is a need for more accurate knowledge of the hydrodynamic impact on marine structure problems as the knowledge on wave impact is still far from sufficient.

A novel approach for the determination of pressure loads experienced by marine structures is the utilisation of Artificial Neural Networks (ANN) [28]. ANN methods are utilised in research areas where problems are solved by pattern recognition, generalisation and pattern classification [29]. ANN has attracted considerable attention and shown promise for modelling complex nonlinear relationships. ANNs have been used extensively in many fields [29-38].

In a recent study by the authors [28], it is shown that the inverse problem approach can be used to estimate the loads applied on a marine composite panel from the strain measurements when behaving linearly. A comparison of the ANN loads with the actual applied loads indicated a very good performance of the methodology. However, it was discussed that more investigation is necessary to further evaluate the suitability of the proposed methodology for in-service load monitoring of marine structures under transient load conditions such as slamming. This paper reports on the research undertaken to further develop the ANN methodology for quantifying pressure loads on a marine composite panel under transient load conditions from strain measurements.

## Methodology

The methodology employed to evaluate the suitability and performance of utilising an ANN as an inverse problem solver for quantifying the transient load applied to the composite panel is presented in this section. The first stage of the investigation was to design an impact load quantification methodology for the panel utilising an ANN. In the second stage the load quantification methodology was validated by comparing loads estimated by the ANN with the known loading cases of the panel.

**Inverse Problem Analysis Methodology.** Inverse problem analysis is based on accurately calculating the external loads or boundary conditions that generate a known amount of strain at predetermined locations on a structure. An ANN, as an inverse problem solver, can be utilised to determine a relationship between the cause and its effect [28]. In this study, the impact loads (the cause/output) on a composite panel are quantified by acquiring repeatable peak strain responses (the effect/input) to these loads from the panel. By introducing examples to an ANN, it can learn the relationships between the input and output through a training process. Once the ANN is sufficiently trained it can be utilised to estimate the output in real-time. New inputs (problem data) are presented and processed by the ANN to quantify/estimate the load.

**Simulation Setup.** The structure under consideration was a 1 m<sup>2</sup> glass Reinforced Fibre Polymer (GRFP) marine composite panel made of 7 layers of stitched biaxial  $\pm 45$  E-glass cloth and with Ampreg 22 epoxy resin system (all provided by SP Gurit Systems), hand laid up with a total thickness of  $5 \times 10^{-3}$  m. Fig. 1 shows a schematic of composite panel indicating strain node, loading and support locations. The panel was divided into a four-by-four grid producing sixteen equal regions of area  $0.25 \times 0.25$  m<sup>2</sup>. The bottom surface of the panel was supported on all four edges using aluminium bars, each 0.0381 m high, 0.01905 m wide and 1m long (Fig. 1). Strain readings (S1-16) were captured from the centre of each of the 16 regions on the top surface of the panel. Loading was achieved by simulating a free fall impact of a rigid mild steel cylinder (length 0.103 m, diameter of

0.02 m and mass of 0.254 kg) normal to the panel surface at 13 locations (L1-13) from various heights. For this study, the finite element models are developed and simulated in ABAQUS 6.10-1 (SIMULIA). The panel has 7096 elements and the cylinder has 40 elements. Mesh type used is hexagonal. Fig. 2 indicates a typical Finite Element Analysis (FEA) model output employed in this study.

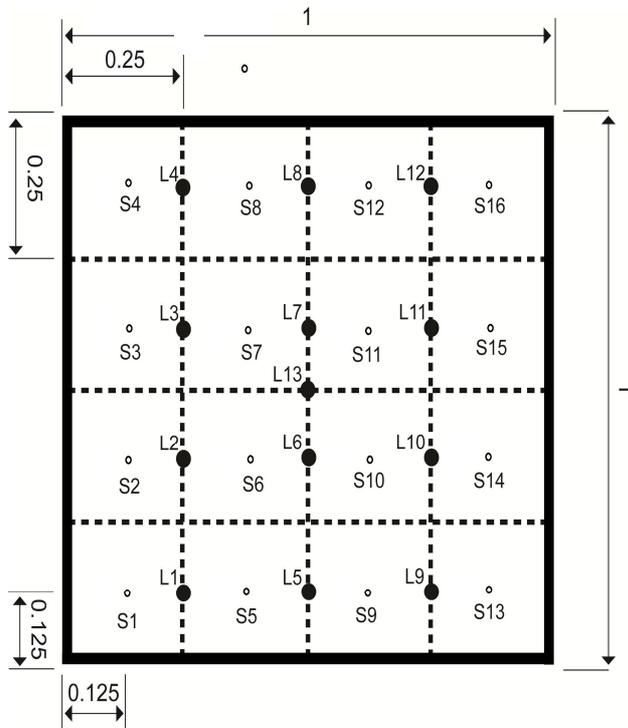


Fig 1. Schematic of composite panel.

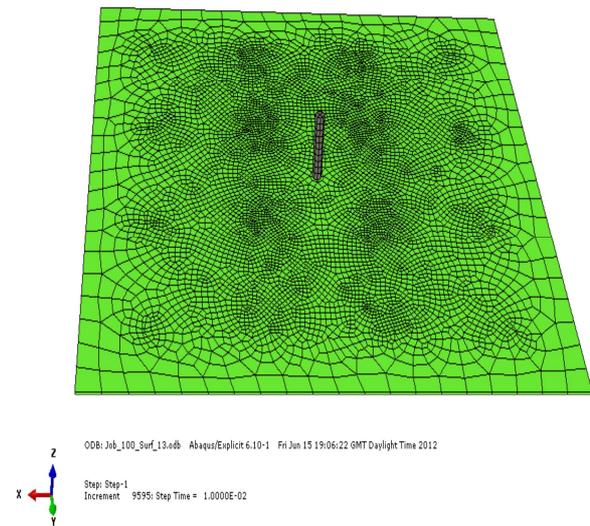


Fig 2. Meshed FEA Panel.

**Generation of ANN Training Data.** Generating the required training data sets may be through experimental tests or the use of simulation such as FEA. However, having a validated FEA model would dramatically save in time and costs compared to the experimental tests. In this study, an FEA model is developed and validated against experimental results. Employing a Python script in the FEA model allows automatic generation of various loading conditions by changing the velocity of cylinder just before impact. The structural response of the panel in terms of strain readings and velocity values for specific locations are evaluated.

**ANN Architecture/Topology.** In this study a common Backpropagation, ANN architecture is used and trained employing MATLAB Artificial Neural Network toolbox capabilities. The network has an input layer with 16 neurons (as there are 16 strain readings), output layer with 13 neurons (as there are 13 loading positions) and some hidden layers each having any number of neurons. An iterative process was used to determine the optimum network architecture for the panel based on the performance of each network tested. In this study, three hidden layers each having 20 neurons was found to be the optimum.

**ANN Validation and Performance.** The validity and the performance of the ANN method were evaluated by comparing the load estimated by the ANN with known loads applied to the panel (problem data). Experimental problem data is the strain data from the same 16 nodes on the panel while it is being loaded. The Sum of Squared Errors (SSE) between a known target and ANN estimation is a common network performance indicator. For this validation study, new loading cases simulated by FEA and corresponding load and strain data is employed to evaluate ANN estimation performance when it is introduced with new data sets.

## Results

A script written in Python language is used to model the structure and simulate various loading scenarios up to 0.02 seconds after impact. Since this test is under high loads, large displacements analysis is used to simulate the model using a nonlinear solver. Furthermore, the structural response of the panel in terms of strain as well as the cylinder velocity over the simulation time is saved to be used to generate training data sets. In order to validate the FEA model against the real structure, the panel is loaded from 100 to 800 N in 100 N increments at all 13 load locations (L1 – L13) separately. The strain readings at all 16 locations (S1 - S16) on the panel are saved for each test. The same tests are performed with FEA to compare the results with the experimental results. For instance, Fig. 3 indicates that for loading only location L13, there is reasonable agreement between the strain readings (S7 and S11) of the FEA model and experimental tests.

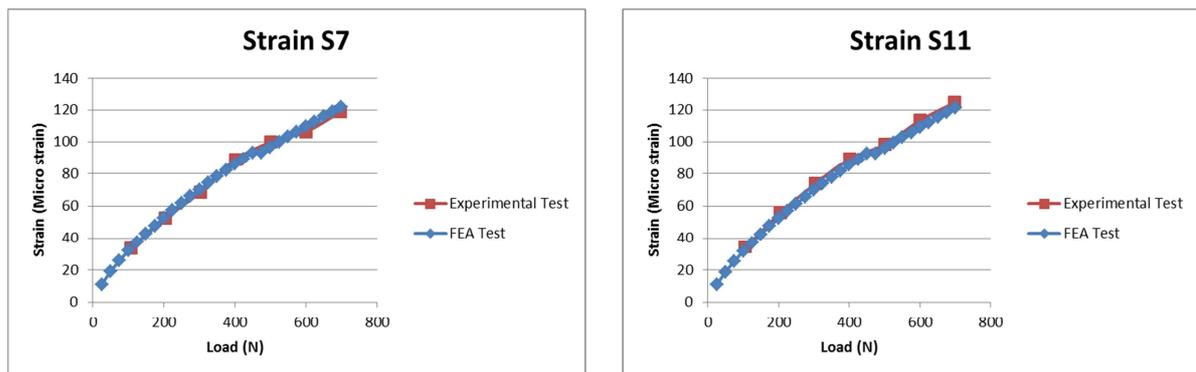


Fig 3. FEA Vs. experimental test results

Once the model is validated, it can be simulated for various loading conditions to generate the required data. The corresponding peak strain values of the first impulse from locations S1-16 (Fig. 1) of the FEA model is used as input training data. Fig. 4 shows example FEA strain data collected from the gauge location S1 when impact location was L1.

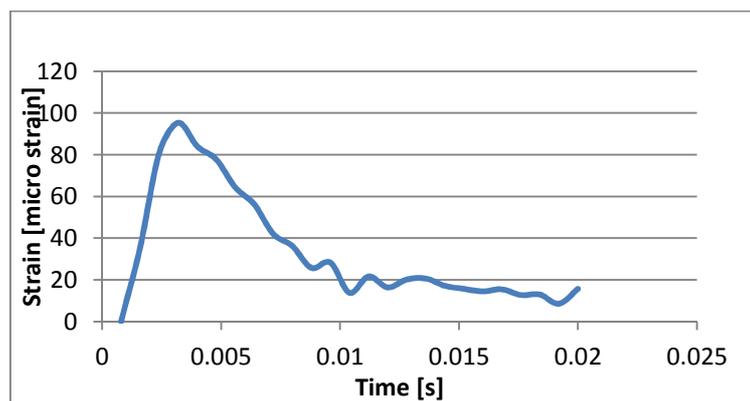


Fig 4. Typical strain data during the impact for gauge S1.

In order to calculate, the impact force of the first impulse ( $F$ ), Eq. 1 is used where  $V_1$  is the velocity of cylinder just before impact and  $V_2$  is its velocity after impact.  $\Delta t$  is the duration of peak impulse. Impact forces of each loading cases are saved and utilised as training data set targets.

$$F = \frac{m (V_2 - V_1)}{\Delta t} \quad (1)$$

In this study 16 strain readings (inputs) and 13 load readings from 13 locations (outputs) are needed to have one set of training data. Enough training sets of 13 various loads at each location on the panel and the resultant 16 strains caused by these loads were required to find the relationship between the input/output data. For each loading location (L1 - L13), 75 training data sets are generated by loadings from 600 N to 6071N making a total of 975 training data sets from FEA. Loadings are changed based on the velocity of the cylinder just before impact. Introducing the training data to the trained network, ANN output should be as similar as possible to the impact load set that the ANN has been trained with. Fig. 5 indicates some random examples of estimated impact load data by the ANN compared with the expected values for impact locations L3 and L9.

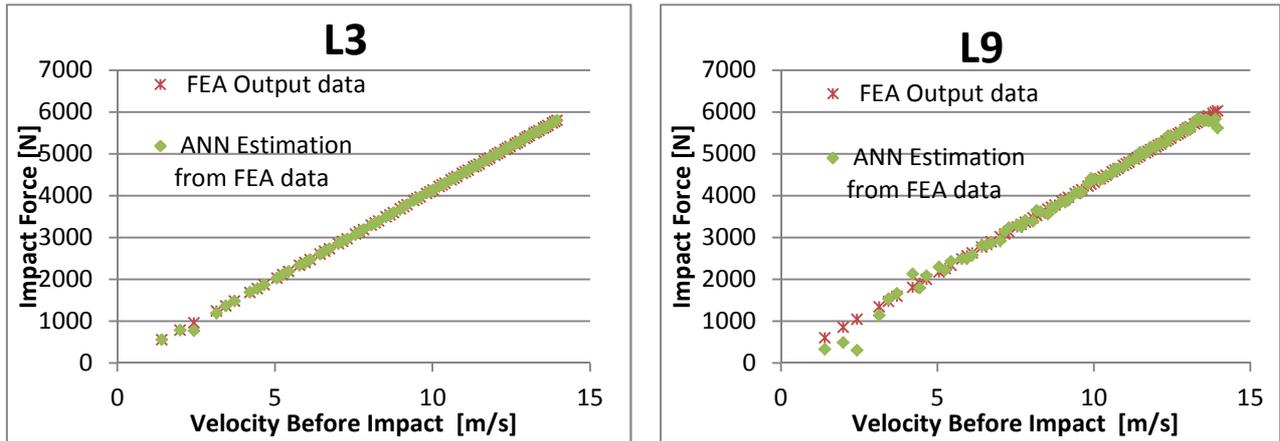


Fig 5. Estimated impact load data by ANN Vs. expected values from training data.

**Validation using FEA Data.** For each loading location (L1- L13), 25 training data sets are generated by loadings from 600 N to 6071N making a total of 325 training data sets from FEA. This set of problem data was not introduced to the network during its training procedure. The strain set of this problem data set is introduced to the previously trained ANN and the corresponding estimated load values are calculated. Fig. 6 shows some random examples of estimated impact load data by ANN and is compared with the expected values.

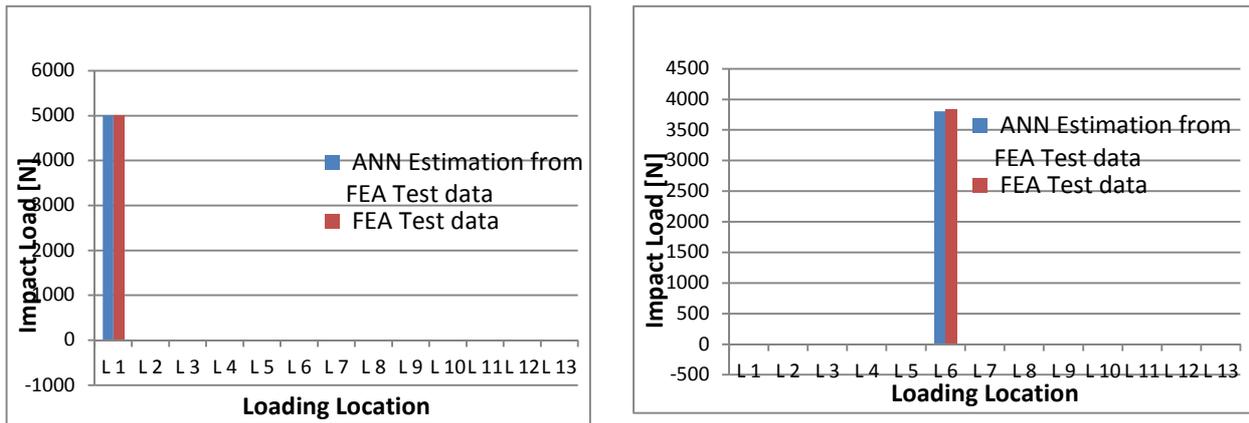


Fig 6. Estimated impact load data by ANN Vs. expected values from test data.

The results presented in this paper show that the inverse problem method, utilising an ANN, can accurately estimate the position and magnitude of 13 impact loads applied to the composite panel from the captured strain data of 16 nodes spread over the panel surface. The results indicate that the system always converges and ANN can be trained using FEA data to solve inverse problems and accurately estimate the impact loads. Once the ANN is sufficiently trained it can be utilised to estimate the output in real-time where new inputs (problem data) are presented and processed by the ANN and impact loads are estimated.

The ability to measure the actual load history of a craft in service would enable the designer to validate the load estimation and structural design tools used during the design stage of a craft. This would lead to the development of more optimal structure designs for this type of craft. The operational safety of the craft can also be improved by having a real-time load monitoring system that is able to detect any degradation of the structural integrity and defects within the structure.

## Conclusion

The aim of this research was to establish an inverse impact load monitoring approach based on structural response from a set of nodes simulated in FEA. It has been shown that the inverse problem approach can be used to estimate 13 impact loads applied on a composite marine panel from the strain measurements of 16 strain gauges. A comparison of the ANN estimated impact loads with the actual applied impact loads indicated a very good performance of the methodology. Once the ANN is sufficiently trained it can be utilised to estimate the output in real-time. This potentially makes the system ideal for solving many classes of complex engineering problem that require impact load monitoring.

## Acknowledgement and Funding

This work was supported by BAE Systems Surface Ships Ltd, Portsmouth and Bournemouth University. The Authors would like to thank Steve Austen, Head of Engineering Support of the Royal National Lifeboat Institution for providing the marine composite panel used in this investigation.

## References

- [1] E. Bunting, M. Sheahan, Broken by design?, *Yachting World*, IPC INSPIRE Ltd, (2009) 66-79.
- [2] X. Guo-dong, D. Wen-yang, Review of prediction techniques on hydrodynamic impact of ships *J. Marine Science Application*, 8 (2009) 204-210.
- [3] J. Lee, P. A. Wilson, Experimental Study Of The Hydro-Impact Of Slamming In A Modern Racing Sailboat, *J. ships Marine Science Application*, 8 (2009) 204-210.
- [4] Phelps, B.P. Determination of Wave Loads for Ship Structural Analysis, DSTO Aeronautical and Maritime Research Laboratory, Australia, 1997.
- [5] T.V. Karman, The impact on seaplane floats during landing - Technical report, Aerodynamical Institute of the Technical High School, Aachen, 1929.
- [6] M. Ochi, L. Motter, Prediction of slamming characteristics and hull responses for ship design, *J. Transactions of SNAME*, 81(1973) 144-176.
- [7] O. Faltinsen, Hydroelastic slamming, *J. Marine Science and Technology*, 5(2) (2000) 49-65.
- [8] G. Kapsenberg, A. Veer, J. Hackett, M. Levadou, Aftbody slamming and whipping loads, *SNAME Annual Meeting*, 111 (2003) 213-231.
- [9] S. Heller, N. Jasper, On the structural design of planing craft, *Transaction of RINA*, 103 (1961) 49-65.
- [10] A. Stavovy, S. Chuang, Analytical determination of slamming pressures for high-speed vehicles in waves, *J. Ship Research*, 20(4) (1976) 190-198.
- [11] D. Savitsky, P. Brown, Procedures for hydrodynamic evaluation of planning hulls in smooth and rough water, *J. Marine Technology*, 13(4) (1976) 381-400.

- 
- [12] R. Allen, R. Jones, D. Taylor, A simplified method for determining structural design limit pressures on high performance marine vehicles, Proceedings of AIAA/SNAME Advanced Marine Vehicle Conference, San Diego, (1978).
- [13] P. Joubert, Strength of bottom plating of yachts, *J. Ship Research*, 26 (1) (1982) 45-49.
- [14] R. Reichard, The structural response of small craft to dynamic loading, Proceedings of the 14th AIAA Symposium on the Aero/Hydrodynamics of Sailing, 30 (1984) 105-110.
- [15] M. Hentinen, G. Holm, Load measurement on the 9.4m sailing yacht sail lab, 13th International Symposium Yacht Design and Yacht Construction, Netherlands, (1994) 131-161.
- [16] P. Joubert, Tests on yacht hull plating, *J. Marine Technology*, 33, (1996) 130-140.
- [17] ABS, Guide for building and classing offshore racing yachts, American Bureau of Shipping, 1994.
- [18] ISO, ISO 12215 Small craft – Hull construction and scantlings, Part 5: Design pressures for monohulls design stresses, scantlings determination. International Organization for Standardization, 2008.
- [19] BV, Rules for the Classification and Certification of Yachts. Bureau Veritas, 2008.
- [20] M.K. Ochi, L.E. Motter, Prediction of extreme values of impact pressures associated with ship slamming, *J. Ship Res*, 13(2) (1969).
- [21] M.K. Ochi, L.E. Motter, Prediction of slamming characteristics and hull responses for ship design, *J. Trans SNAME*, (1973).
- [22] D.W Chalmers, Design of ship's structures, HMSO Publications Centre, London, 1993.
- [23] N. Jones, Slamming damage, MIT Department of Ocean Engineering, Report No 72-4, 1972.
- [24] Ship Structure Committee, Hydrodynamic impact on displacement ship hulls, an assessment of the state of the art, SSC 385, 1995.
- [25] S.R Heller, N.H. Jasper. On the structural design of planing craft, *RINA Quarter Trans*, 1960.
- [26] W. Beukelman, Bottom impact pressures due to forced oscillation, *Int Shipbuilding Progress*, 27(309) (1980).
- [27] P. Manganelli, Experimental investigation of dynamic loads on offshore racing yachts, Ph.D thesis, University of Southampton, UK, 2006.
- [28] M.R. Ramazani, S. Noroozi, M. Koohgilani, B. Cripps, P. Sewell, Determination of static pressure loads on a marine composite panel from strain measurements utilising artificial neural networks, submitted to Proceedings of the Institution of Mechanical Engineers, Part M: *J. Eng. for the Maritime Environment*.
- [29] V.B. Rao, H.V. Rao, *C++ Neural Networks and Fuzzy Logic*, M & T Books, New York, 1995.
- [30] X. Cao, Y. Sugiyama, Y. Mitsui, Application of artificial neural networks to load identification *J. Computers and Structures*, 69 (1998) 63-67.
- [31] L. Ziemianski, G. Harpula, The use of neural networks for damage detection in eight storey frames, Proceedings of the 5th international conference: Engineering applications of neural networks, (1999) 292–297.
- [32] R. Amali, S. Noroozi, J. Vinney, The application of combined artificial neural network and finite element method in domain problems, Proceedings of the Sixth International Conference on Engineering Applications of Neural Networks (EANN 2000), (2000) 1-7.

- 
- [33] R. Amali, S. Noroozi, J. Vinney, P. Sewell, S. Andrews, Predicting interfacial loads between the prosthetic socket and the residual limb for below-knee amputees: a case study, *J. Strain*, 42(1) (2006) 3-10.
- [34] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain *Psychological Review*, 65(6) (1958) 386–408.
- [35] S. Shar, F. Palmieri, MEKA: a fast, local algorithm for training feed forward neural networks, *Proceedings of the International Joint Conference on Neural Networks*, (1990) 41–46.
- [36] K. Soo-Young, B.Y. Moon, D.E. Kim, S.C. Shin, Automation of hull plates classification in ship design system using neural network method, *Mechanical Systems and Signal Processing* 20(2) (2006) 493–504.
- [37] S. Xu, X. Deng, V. Tiwar, M.A. Sutton, W.L. Fourney, An inverse approach for pressure load identification, *International Journal of Impact Engineering*, 37(7) (2010) 865–877.
- [38] R. Amali, S. Noroozi, J. Vinney, P. Sewell, S. Andrews, Improvements in the accuracy of an Inverse Problem Engine's output for the prediction of below-knee prosthetic socket interfacial loads, *Engineering Applications of Artificial Intelligence*, 23(6) (2010) 1000–1.