

YEAR
2019



SANTA CATARINA STATE UNIVERSITY – UDESC
COLLEGE OF TECHNOLOGICAL SCIENCE – CCT
MASTER IN MECHANICAL ENGINEERING – PPGEM

LUÍSA ROSENSTOCK VÖLTZ | FAULT DIAGNOSIS IN COMPOSITE STRUCTURES
USING ARTIFICIAL NEURAL NETWORK AND PRINCIPAL COMPONENT ANALYSIS

MASTER THESIS

**FAULT DIAGNOSIS IN COMPOSITE
STRUCTURES USING ARTIFICIAL
NEURAL NETWORK AND
PRINCIPAL COMPONENT ANALYSIS**

LUÍSA ROSENSTOCK VÖLTZ

JOINVILLE, 2019

Artificial Neural Networks (ANNs) have emerged as one of the most useful tools in Artificial Intelligence. Composite materials are increasingly used in critical and demanding applications. However, the challenging of the current Structural Health Monitoring (SHM) methodologies is to detect damage in entire composite structure in real time and continuously, without time-consuming. In this way, this work aims to evaluate a SHM-methodology for fault diagnosis in composite materials. The methodology used includes the use of a piezoelectric system as input sensor, vibration-based methods for the analysis of the different structural states, Principal Component Analysis for reducing data and machine learning (ANNs) algorithms for classifying all the structural states.

Advisor: Prof. Dr. Ricardo de Medeiros

Co-advisor: Prof. Dr. Eduardo Lenz Cardoso

Joinville, 2019

LUÍSA ROSENSTOCK VÖLTZ

**FAULT DIAGNOSIS IN COMPOSITE STRUCTURES USING ARTIFICIAL NEURAL
NETWORK AND PRINCIPAL COMPONENT ANALYSIS**

Master thesis submitted to the Mechanical Engineering Department at the College of Technological Science of Santa Catarina State University in fulfillment of the partial requirement for the Master's degree in Mechanical Engineering.

Advisor: Prof. Dr. Ricardo de Medeiros

Co-advisor: Prof. Dr. Eduardo Lenz Cardoso

JOINVILLE - SC

2019

**Ficha catalográfica elaborada pelo programa de geração automática da
Biblioteca Setorial do CCT/UDESC,
com os dados fornecidos pelo(a) autor(a)**

Völtz, Luísa Rosenstock
Fault Diagnosis in Composite Structures Using Artificial
Neural Network and Principal Component Analysis / Luísa
Rosenstock Völtz. -- 2019.
201 p.

Orientador: Ricardo de Medeiros
Coorientador: Eduardo Lenz Cardoso
Dissertação (mestrado) -- Universidade do Estado de
Santa Catarina, Centro de Ciências Tecnológicas, Programa
de Pós-Graduação em Engenharia Mecânica, Joinville, 2019.

1. Artificial neural network . 2. Composite materials. 3.
Damage detection. 4. Principal component analysis. 5. Pattern
recognition. I. Medeiros, Ricardo de . II. Lenz Cardoso,
Eduardo. III. Universidade do Estado de Santa Catarina,
Centro de Ciências Tecnológicas, Programa de
Pós-Graduação em Engenharia Mecânica. IV. Título.

**FAULT DIAGNOSIS IN COMPOSITE STRUCTURES USING ARTIFICIAL NEURAL
NETWORK AND PRINCIPAL COMPONENT ANALYSIS**

por

Luisa Rosenstock Völtz

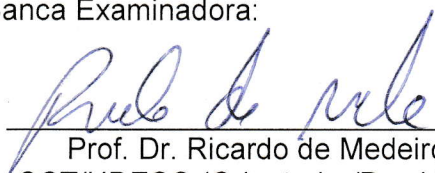
Esta dissertação foi julgada adequada para obtenção do título de

MESTRA EM ENGENHARIA MECÂNICA

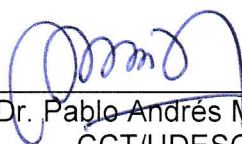
Área de concentração em “Modelagem e Simulação Numérica”
e aprovada em sua forma final pelo

CURSO DE MESTRADO ACADÊMICO EM ENGENHARIA MECÂNICA
DO CENTRO DE CIÊNCIAS TECNOLÓGICAS DA
UNIVERSIDADE DO ESTADO DE SANTA CATARINA.

Banca Examinadora:



Prof. Dr. Ricardo de Medeiros
CCT/UDESC (Orientador/Presidente)



Prof. Dr. Pablo Andrés Muñoz Rojas
CCT/UDESC



Profa. Dra. Viviana Meruane Naranjo
Universidad de Chile

Joinville, SC, 21 de março de 2019.

ACKNOWLEDGMENTS

The author would like to express the following acknowledgments.

To God, specially, for giving me health, peace and tranquillity to overcome this stage of my life.

To my advisor Prof. Dr. Ricardo De Medeiros for his support during my study, for his patience and for sharing his knowledge. Also, to my co-advisor Prof. Dr. Eduardo Lenz Cardoso for teaching, sharing ideas and supporting.

To the Santa Catarina State University, the Mechanical Engineering Department and Professors, the LAMEC, the Laboratory of Vibrations and the Laboratory of Composite Materials for the opportunity and the structure.

To my family, specially my parents, Vigando and Sandra for all the encouragement, love and education. And to my husband, Rafael, for the emotional support and for believing in my competence.

For the students that started before me, Doglas, Luiz, Luis and Eliseo, who helped me a lot at the beginning of the master's degree whose assistance helped me to progress. To my friends from LAMEC for all encouragement, learning and discussions.

To PROMOP (Programa de Bolsas de Monitoria de Pós-Graduação) of the Santa Catarina State University for the financial support.

Finally, I gratefully acknowledge the financial support of the Fundação de Amparo a Pesquisa e Inovação do Estado de Santa Catarina - Brazil (FAPESC grant: 2017TR1747 and 2017TR784).

ABSTRACT

ROSENSTOCK VÖLTZ, Luísa, FAULT DIAGNOSIS IN COMPOSITE STRUCTURES USING ARTIFICIAL NEURAL NETWORK AND PRINCIPAL COMPONENT ANALYSIS. 2019. 201 f. Master Thesis (Master in Mechanical Engineering - Area: Numerical Modeling and Simulation) – Santa Catarina State University. Master in Mechanical Engineering Joinville 2019.

Artificial Neural Networks (ANNs) have emerged as one of the most useful tools in Artificial Intelligence, being used in many applications. Due to their high capacity for learning, adaptation, and generalization, ANNs can fit linear and non-linear models that other methods are not able to describe. Composite materials are increasingly used in critical and demanding applications, mainly due to their high specific strength and stiffness. However, the challenging of the current Structural Health Monitoring (SHM) methodologies is to detect damage in entire composite structure in real time and continuously, without time-consuming and unnecessary expenses in maintenance. In this way, this work aims to evaluate a SHM-methodology for fault diagnosis in composite materials. The methodology used in this work includes the use of a piezoelectric system as input sensor, vibration-based methods for the analysis of the different structural states, Principal Component Analysis (PCA) for reducing data and machine learning (ANNs) algorithms for classifying all the structural states. To assess the applicability of the methodology, carbon fiber/epoxy composite plates are evaluated in healthy and damaged state conditions. Later glass-fiber/epoxy beams are manufactured by the modified Vacuum Assisted Resin Transfer Molding process and with the aid of a Teflon tape to simulate delamination damages in 3 different proportions: 5 mm, 10 mm and 19 mm. Vibration-based tests are performed to acquire Frequency Response Functions from healthy and damages conditions. PCA is used to reduce the dimension of the problem while maintaining its main characteristics. Next, a multi-layer neural network is developed, based on automatic differentiation and dual numbers for sensitivity analysis. A Particle Swarm Optimization algorithm is used to find the best topology of the ANNs maximizing the accuracy of the validation dataset. The last step is the evaluation of the ANN-classifier using a confusion matrix technique and its indicative parameters. Finally, a discussion is made with respect to the potentialities and limitations of the methodology for using in fault diagnosis systems.

Key-words: Artificial neural network (ANN), composites materials, damage detection, principal components analysis (PCA), pattern recognition.

RESUMO

ROSENSTOCK VÖLTZ, Luísa, DIAGNÓSTICO DE FALHAS EM ESTRUTURAS DE COMPÓSITOS UTILIZANDO REDE NEURAL ARTIFICIAL E ANÁLISE DE COMPONENTES PRINCIPAIS. 2019. 201 f. Dissertação (Mestrado em Engenharia Mecânica - Área: Modelagem e Simulação Numérica) – Universidade do Estado de Santa Catarina. Programa de Pós-Graduação em Engenharia Mecânica Joinville 2019.

As Redes Neurais Artificiais (RNAs) surgiram como uma das ferramentas mais úteis na Inteligência Artificial, utilizadas em diversas aplicações. Devido a sua grande capacidade de aprendizado, adaptação e generalização, as RNAs podem se ajustar a modelos lineares e não-lineares. Os materiais compósitos estão sendo cada vez mais utilizados em aplicações críticas e exigentes, devido à sua alta resistência e rigidez. No entanto, o desafio das atuais metodologias de Monitoramento da Integridade Estrutural (SHM em inglês) é a detecção de danos em estruturas de compósitos em tempo real e contínuo, sem perda de tempo e gastos desnecessários com manutenção. Diante disso, este trabalho tem como objetivo avaliar uma metodologia de SHM para diagnóstico de falhas em materiais compósitos. A metodologia utilizada inclui o uso de um sistema piezelétrico para a inspeção das estruturas, métodos baseados em vibrações para a análise dos diferentes estados estruturais, Análise de Componentes Principais (ACP) para redução dos dados e algoritmo de aprendizado de máquina (RNAs) para classificação. Visando verificar a aplicabilidade da metodologia, placas de fibra de carbono/epóxi são avaliadas em condições de estados intactas e danificadas. Posteriormente, vigas de fibra de vidro/epóxi são fabricadas pelo processo modificado de Moldagem por Transferência de Resina Assistida a Vácuo e com o auxílio de uma fita Teflon para simular os danos de delaminação em 3 diferentes proporções: 5 mm, 10 mm e 19 mm. Testes baseados em vibrações são executados para obter as Funções de Resposta em Frequência das condições estruturais. A ACP é implementada para reduzir a dimensão dos dados, mantendo suas principais características. É desenvolvida uma RNA multicamada, baseada em diferenciação automática e em números duais para análise de sensibilidade. Um algoritmo de otimização de enxame de partículas (PSO em inglês) é usado para obter a melhor topologia da RNA, maximizando a acurácia dos dados de validação. O último passo é a avaliação do classificador de RNA usando a matriz de confusão. Por fim, são discutidas as potencialidades e limitações da metodologia para uso em sistemas de diagnóstico de falhas.

Palavras-chave: Rede neural artificial (RNA), materiais compósitos, detecção de danos, análise de componentes principais (ACP), reconhecimento de padrões.

Table of Contents

List of Figures	17
List of Tables	21
1 Introduction	25
1.1 Background and Motivation	25
1.2 Objectives and Scope	31
1.3 Brief Outline of Thesis	31
2 Structural Health Monitoring - Review	33
2.1 Structural Health Monitoring	33
2.2 Composite Materials	37
2.2.1 Composite classification	37
2.2.2 Mechanisms of failure in composites materials	39
2.2.3 Damage tolerance of composites structures	41
2.2.4 Manufacturing processes	42
2.3 Vibration-based Methods	44
2.3.1 FRF measurement set-up	45
2.3.2 Damping ratio	48
3 Pattern Recognition - Review	51
3.1 Pattern Recognition	51
3.1.1 Confusion matrix	51
3.2 Artificial Neural Networks	53
3.2.1 Network architectures	55
3.2.2 Types of activation functions	58
3.2.3 Learning process	61
3.2.3.1 Supervised learning	61
3.2.3.2 Unsupervised learning	63
3.2.4 Multilayer Perceptrons networks	63
3.2.4.1 Network learning	63

3.2.5	Bias and variance tradeoff	69
3.2.5.1	Cross-validation	71
3.2.5.2	Training with noise	71
3.2.5.3	Regularization techniques	72
3.2.5.4	Early stopping	73
3.3	Principal Component Analysis	73
3.4	Particle Swarm Optimization	76
3.5	Automatic Differentiation and Dual Numbers	77
4	Materials and Methods	79
4.1	Damage Detection in Glass Fiber/Epoxy Beams	79
4.1.1	Manufacturing process	79
4.1.2	Dynamic tests: Experimental	85
4.1.3	Dynamic tests: Numerical	88
4.2	Damage Detection in Carbon Fiber/Epoxy Plates	91
4.3	Fault Diagnosis of Composite Structures: Methodology.	92
4.3.1	Principal Component Analysis	92
4.3.2	Artificial Neural Networks	94
5	Results	99
5.1	Damage Detection in Carbon-Fiber/Epoxy Composite Plates	99
5.1.1	Composite plates with stacking orientation $[0]_8$	100
5.1.1.1	Dynamic tests	100
5.1.1.2	Principal component analysis	101
5.1.1.3	Artificial neural networks and pattern recognition	104
5.1.2	Composite plates with stacking orientation $[0/15/-15/0/15/-15]_S$	106
5.1.2.1	Dynamic tests	107
5.1.2.2	Principal component analysis	108
5.1.2.3	Artificial neural networks and pattern recognition	110
5.2	Damage Detection in Glass-Fiber/Epoxy Composite Beams: Experimental	113
5.2.1	Dynamic and modal analysis	113
5.2.2	Principal component analysis	116
5.2.3	Artificial neural networks and pattern recognition: Case I	120
5.2.4	Artificial neural networks and pattern recognition: Case II	126
5.2.5	Artificial neural networks and pattern recognition: Case III	131
5.2.6	Artificial neural networks and pattern recognition: Case IV	135
5.2.7	Artificial neural networks and pattern recognition: Case V	140
5.3	Damage Detection in Glass-Fiber/Epoxy Composite Beams: Numerical	146
5.3.1	Dynamic and modal analysis	146
5.3.2	Principal component analysis	147

5.3.3	Artificial neural networks and pattern recognition: Case I-Numerical	149
5.3.4	Artificial neural networks and pattern recognition: Case II-Numerical	152

6	Conclusions	155
6.1	Conclusions	155
6.2	Suggestions for Future Works	157
	Bibliography	159
	Appendix	166
A	Graphics	167
B	Fault Diagnosis Methodology Algorithms	181
C	Ignition Loss Test in Glass-Fiber/Epoxy Composites	195
D	Damage Detection in Metallic Beams	197
E	Scientific Publications	201

List of Figures

1.1	Plane crash in Mississippi.	26
1.2	Machine and deep learning 2000s timeline.	28
1.3	The global composites market (value and volume).	30
1.4	The global composites market application.	30
2.1	An overview of the range of damage identification approaches, focusing in vibration-based methods.	35
2.2	Structural components of composite materials.	38
2.3	Woven fabric styles.	38
2.4	Types of composites cracks mechanisms (a) In-plane damage, (b) De-lamination.	39
2.5	Comparison of composite non-growing damage and metal fatigue crack damage (UL: Ultimate Load and LL: Limit Load).	42
2.6	Filament winding process.	43
2.7	VARTM process.	44
2.8	Single degree of freedom mass-spring-damper system.	44
2.9	Graphical display of an FRF: (a) Amplitude-phase plot and (b) Real-imaginary plot.	47
2.10	Coherence display plot.	48
2.11	Peak-peaking method.	49
3.1	Confusion matrix example.	52
3.2	ANNs potential areas of applicability.	53
3.3	Example of a single neuron.	54
3.4	Relation between the induced local field and linear combiners output.	55
3.5	Separability boundary illustration AND and OR-problems.	56
3.6	XOR problem.	57
3.7	XOR problem with MLP Networks.	57
3.8	Logistic function.	59
3.9	Hyperbolic tangent function.	59
3.10	Leaky-ReLU function.	60

3.11 Geometric interpretation of the Delta rule.	63
3.12 Multilayer Perceptron network.	64
3.13 Behavior of the cost function with different values of the learning rate.	67
3.14 Armijo-Goldstein condition.	68
3.15 Bias-variance tradeoff.	70
3.16 Examples of underfitting, optimal fitting and overfitting cases.	71
3.17 Example of 5-fold Cross-validation.	72
3.18 PCA-variance illustration.	75
3.19 PCA illustration.	75
4.1 VARTM manufacturing process layout.	80
4.2 VARTM manufacturing process: step 1.	81
4.3 VARTM manufacturing process: step 2.	81
4.4 VARTM manufacturing process: step 3.	82
4.5 VARTM manufacturing process: step 4 (a), 5 (b) and 6 (c).	82
4.6 VARTM manufacturing process: step 7.	82
4.7 Cutting machine.	83
4.8 Composite beams: four damages cases.	83
4.9 Beam experimental setup.	85
4.10 Impact hammer (a) and miniature accelerometer (b).	85
4.11 Signal converter.	86
4.12 Accelerometer position and excitation position on glass fiber/epoxy beam.	87
4.13 Modal Analysis Setup.	87
4.14 Conventional shell and continuum shell.	88
4.15 Double-beam FEM model with delamination.	89
4.16 Mesh convergence.	89
4.17 Dynamic experimental set-up.	92
4.18 Dataset split and PCA approaches.	93
4.19 Methodology flowchart.	96
5.1 FRFs from healthy and damaged plate for $[0]_8$ (a)real part and (b) mag- nitude.	100
5.2 Relative variance and accumulated variance using accelerance values and real part values ($[0]_8$).	101
5.3 FRFs reconstructed with 5 PCs (a) and 20 PCs (b) comparing with orig- inal FRF: real part values ($[0]_8$).	102
5.4 FRFs reconstructed with 5 PCs (a) and 20 PCs (b) comparing with orig- inal FRF: accelerance values ($[0]_8$).	102
5.5 PCs curves using (a) real part values and (b) accelerance - ($[0]_8$.)	103

5.6	FRFs from healthy and damaged plate for $[0/15/-15/0/15/-15]_S$ (a) real part and (b) accelerance.	107
5.7	Relative variance and accumulated variance using accelerance values and real part values ($[0/15/-15/0/15/-15]_S$).	108
5.8	FRFs reconstructed with 5 PCs (a), 20 PCs (b) and 30 PCs (c) comparing with original FRF: real part values ($[0/15/-15/0/15/-15]_S$).	109
5.9	FRFs reconstructed with 5 PCs (a) and 20 PCs (b) comparing with original FRF: accelerance values ($[0/15/-15/0/15/-15]_S$).	109
5.10	PCs curves using real part values (a) and accelerance (b) - ($[0/15/-15/0/15/-15]_S$).	110
5.11	H_{11} - FRFs curves (accelerance values).	114
5.12	H_{21} - FRFs curves (accelerance values).	115
5.13	Frequency range (Hz) for the first five mode shapes from the four damage states (H_{11}).	115
5.14	Positions H_{11} and H_{21} - FRFs curves (Beam 6).	116
5.15	Relative variance (%) for the 5-fold for the first 30 PCs	118
5.16	FRFs reconstructed using 20, 25 and 30 PCs for the 5-folds (sets).	119
5.17	Training error (dot line) and accuracy (%) convergence curves for ANN_{24} - Set 3: best result - Case I.	126
5.18	Training error (dot line) and accuracy (%) convergence curves for ANN_8 - Set 5: best result - Case II.	131
5.19	Training error (dot line) and accuracy (%) convergence curves for ANN_1 - Set 4: best result - Case III.	135
5.20	Training error (dot line) and accuracy (%) convergence curves for ANN_3 - Set 3: best result - Case IV.	139
5.21	Mode shapes 1 and 2.	146
5.22	Mode shapes 3 and 4.	146
5.23	H_{11} - Numerical FRFs curves (accelerance values) - 4 states conditions.	147
5.24	H_{21} - Numerical FRFs curves (accelerance values) - 4 states conditions.	147
5.25	Frequency range (Hz) for the first four modes shapes from the four damage states (H_{11}).	148
5.26	10 inputs correlation variables.	150
A.1	Total FRFs from healthy and damaged plates for $[0]_8$ (magnitude part).	168
A.2	Total FRFs from healthy and damaged plates for $[0]_8$ (real part).	169
A.3	Total FRFs from healthy and damaged plates for $[0/15/-15/0/15/-15]_S$ (magnitude part).	170
A.4	Total FRFs from healthy and damaged plates for $[0/15/-15/0/15/-15]_S$ (real part).	171

A.5	Total PCs curves from healthy and damaged plates for $[0]_8$ (magnitude part).	172
A.6	Total PCs curves from healthy and damaged plates for $[0]_8$ (real part).	173
A.7	Total PCs curves from healthy and damaged plates for $[0/15/-15/0/15/-15]_S$ (magnitude part).	174
A.8	Total PCs curves from healthy and damaged plates for $[0/15/-15/0/15/-15]_S$ (real part).	175
A.9	Experimental FRFs from all damaged cases: position H_{11} .	176
A.10	Experimental FRFs from all damaged cases: position H_{21} .	177
A.11	Scatter plot (0—healthy and 1—damaged level 1 classes) - numerical.	178
A.12	Scatter plot (0—healthy and 1—damaged level 1 classes) - experimental (case III).	179
D.1	Experimental setup for aluminum beams.	197
D.2	Modes shape of the beam.	198
D.3	FRFs for healthy, damaged with 2 mm crack size (D_2), damaged with 4 mm crack size (D_4) and damaged with 8 mm crack size (D_8) for the first excitation point in an aluminum beam.	198

List of Tables

1.1 Literature examples related to machine learning and mechanical engineering.	29
2.1 Summary of the techniques applied to damage detection in composite structures.	36
3.1 Algorithm Backtracking line search	68
3.2 PSO Algorithm	77
4.1 Characteristics of glass fiber shallow turkish.	80
4.2 Glass fiber/epoxy beams dimensions.	84
4.3 Glass fiber/epoxy beams mean and mean deviation dimensions.	84
4.4 Experimental setup.	86
4.5 Geometrical and mechanical properties: numerical simulation.	90
4.6 Mechanical properties variation: numerical simulation.	91
4.7 PCA Algorithm - approach 2	94
4.8 Sensitivity analysis algorithm	95
4.9 ANN algorithm - learning phase	97
4.10 ANN algorithm - validation set verification	97
5.1 PSO algorithm parameters.	104
5.2 ANNs simulations summary results for $[0]_8$ plates (100 times runs).	105
5.3 Confusion matrix after 100 runs $[0]_8$ plates (accelerance values).	105
5.4 Confusion matrix after 100 runs $[0]_8$ plates (real part values).	105
5.5 Confusion matrix parameters for $[0]_8$ plates (100 times runs).	106
5.6 ANNs best results for $[0]_8$ plates.	106
5.7 ANNs simulations summary results for $[0/15/-15/0/15/-15]_S$ plates (100 times runs).	111
5.8 Confusion matrix after 100 runs $[0/15/-15/0/15/-15]_S$ plates (accelerance values).	112
5.9 Confusion matrix after 100 runs $[0/15/-15/0/15/-15]_S$ plates (real part values).	112

5.10 Confusion matrix parameters for $[0/15/-15/0/15/-15]_S$ plates (100 times run).	112
5.11 ANNs best results for $[0/15/-15/0/15/-15]_S$ plates.	113
5.12 Modal analysis results - flexural modes (Beam 3).	116
5.13 Summary results for PCA-total and PCA-partial using FFT curves (500 times runs).	117
5.14 Total variance for each 5-folds with 20, 25 and 30 PCs.	118
5.15 PSO algorithm parameters.	120
5.16 ANNs simulations with different parameters - Case I.	121
5.17 ANNs simulations with different parameters - results (Set 1) - 150 times run - Case I.	122
5.18 ANNs simulations with different parameters - accuracy results (Sets 2, 3, 4 and 5) - 150 times run - Case I.	123
5.19 ANN two best accuracy results (ANN_{24} - Set 3 and 5) - Case I.	124
5.20 Confusion matrix for ANN_{24} - Set 3: best result - Case I.	124
5.21 Confusion matrix for ANN_{24} - Set 5: best result - Case I.	125
5.22 Confusion matrix parameters for ANN_{24} - Sets 3 and 5: best result - Case I.	125
5.23 Total variance for each 5-folds 30 PCs - Case II.	126
5.24 ANNs simulations with different parameters- Case II.	127
5.25 ANNs simulations with different parameters - accuracy results (Set 1) - 150 times run - Case II.	128
5.26 ANNs simulations with different parameters - accuracy results (Sets 2, 3, 4 and 5) - 150 times run - Case II.	129
5.27 ANN best accuracy results (ANN_8 - Set 5) - Case II.	130
5.28 Confusion matrix for ANN_8 - Set 5: best result - Case II.	130
5.29 Confusion matrix parameters for ANN_8 - Set 5: best result - Case II.	130
5.30 Total variance for each 5-folds 10 and 20 PCs - Case III.	131
5.31 ANNs simulations with different parameters - Case III.	132
5.32 ANNs simulations with different parameters - accuracy results (Sets 1, 2, 3, 4 and 5) - 150 times run - Case III.	133
5.33 ANN best accuracy results (ANN_1 - Set 4) - Case III.	134
5.34 Confusion matrix for ANN_1 - Set 4: best result - Case III.	134
5.35 Confusion matrix parameters for ANN_1 - Set 4: best result - Case III.	134
5.36 Total variance for each 5-folds for 25 PCs - Case IV.	135
5.37 ANNs simulations with different parameters - Case IV.	136
5.38 ANNs simulations with different parameters - accuracy results (Sets 1, 2, 3, 4 and 5) - 150 times run - Case IV.	137
5.39 ANN best accuracy results (ANN_3 - Set 3) - Case IV.	138

5.40	Confusion matrix for ANN_3 - Set 3: best result - Case IV.	138
5.41	Confusion matrix parameters for ANN_3 - Set 3: best result - Case IV. . .	138
5.42	Total variance for each 5-folds 30 PCs - Case V.	140
5.43	ANNs simulations with different parameters- Case V.	141
5.44	ANNs simulations with different parameters - accuracy results (Set 1) - 150 times run - Case V.	142
5.45	ANNs simulations with different parameters - accuracy results (Sets 2, 3, 4 and 5) - 150 times run - Case V.	143
5.46	ANN two best accuracy results (ANN_1 - Set 3) - Case V.	144
5.47	Multi-class confusion matrix for ANN_1 - Set 3 - Case V.	144
5.48	Total variance for each 5-folds 10 and 20 PCs - Numerical (healthy and damaged level 1 sates conditions).	148
5.49	Total variance for each 5-folds 10 and 20 PCs - Numerical (4 sates con- ditions).	149
5.50	ANNs simulations with different parameters - accuracy results (Sets 1, 2, 3, 4 and 5) - 150 times run - Numerical.	151
5.51	ANN simulation - the best accuracy results (Sets 1, 2, 3, 4 and 5) - 150 times run - Case II-N.	152
5.52	Numerical Multi-class confusion matrix - Case V.	153
C.1	Information of the specimens before and after the ignition loss test. . . .	195
C.2	Mechanical properties.	195
C.3	Mechanical properties of composite.	196
D.1	Frequency range (Hz) for the first five mode shapes.	199
D.2	ANNs simulations summary results for aluminum beams (200 times runs).199	
D.3	ANNs results: aluminum beams.	199

Chapter 1

Introduction

This chapter introduces the background and motivation for this work: the use of Artificial Neural Networks (ANNs) and Principal Component Analysis (PCA) as a tool for monitoring the structural integrity of composites structures.

1.1 Background and Motivation

Structural health monitoring (SHM) has played an import role in protecting equipment and structural components from performance degradation and failure. Such interest is driven by the hazard of human lives losses, due to unpredictable structural damages, like building collapses and airplane crashes (LOPES *et al.*, 2011).

In 2017, a tragic plane crash ended the lives of 16 soldiers, as shown in Fig. 1.1. A KT-130C plane crashed killing 16 people from Marine Reserve in the USA, due to sloppy maintenance work at an Air Force depot. According to Insinna (2018), the report shows "A corroded blade broke off of the aircraft, sliced through the fuselage, and set off a chain of events that ended with the plane splitting into three pieces and crashing into a Mississippi soybean field". In the report, it was found that the blade was last revised in 2011 by Warner Robins Air Logistics Complex, where the company was responsible for rooting out corrosion and fixing other problems. However, the maintainers did not correctly fix it. The auditors found proofs that small cracks and pits already existed in the propeller blade, resulting in increased damage into a long fracture.

Another plane crash due to poor maintenance occurred in 2005 in Miami (Chalk's Ocean Airways Flight 101). The possible cause was a fatigue failure in the right wing initialized by a crack. According to NTSB (2005), "The crack had been detected running through a slosh hole and seemingly repaired earlier, but the repair was eventually to prove ineffective".

A large diversity of highly effective local non-destructive evaluation (NDE) tools

Figure 1.1 – Plane crash in Mississippi.



Source: Insinna (2018).

are accessible for health monitoring. Although there are several NDE techniques that can be used to detect damage, they may be time-consuming or even require physical access to difficult places, increasing the risk of human error (FARRAR; WORDEN, 2007).

The majority of SHM investigation performed over the past years has persuaded to identify damage in structures on a global basis (FARRAR; WORDEN, 2007), in online conditions and continuous evaluation of the structure state (BENZONI *et al.*, 2013). As a result of many uncertainties in most of real-life applications, the use of statistical pattern recognition methodologies in SHM has gained much attention in the past decade (GUL; CATBAS, 2009). According to Webb (2003), statistical pattern recognition is an approach used in all the steps of a damage analysis, from problem definition and data acquisition, to discrimination and classification, evaluation of results and interpretation.

In past years, machine learning and deep learning have become a rapidly expanding research topic, redefining state-of-the-art performance in a broad range of areas, such as object recognition, image segmentation, modeling and predicting non-linear system behavior (ZHAO *et al.*, 2019). Artificial Neural Networks (ANNs) are deep learning technologies, suited to model complex processes. A classic ANN is a group of algorithms which model data using neurons for machine learning. The application of ANNs has proved to be a powerful tool for signal processing, system identification and pattern recognition and classification (PUSCASU; CODRES, 2011). Following the behavior of the biological nervous system, ANNs methodology is an attractive mathematical tool, which can be used to simulate a wide diversity of scientific and engineering problems. Like their biological counterparts, ANNs can learn from examples and can be

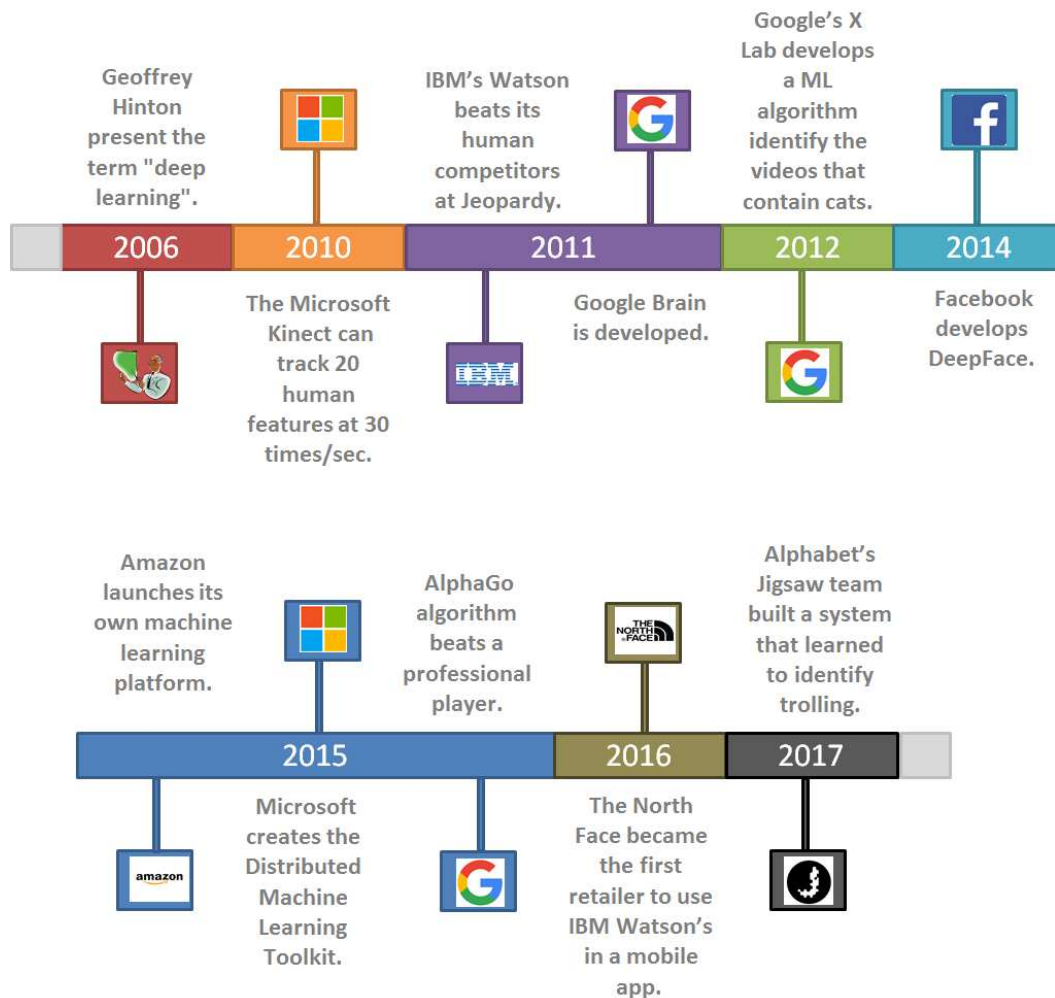
trained to find solutions to complex non-linear multi-dimensional function relationships without any prior assumptions about their nature (ZHANG; FRIEDRICH, 2003).

The modern era of ANNs started with McCulloch and Pitts in 1943. A significant improvement came in 1949 with the publication of Hebb's book, where an explicit statement of a physiological learning rule for synaptic modification was presented for the first time. In 1958, a new path to the pattern recognition problem was introduced by Rosenblatt in his work on the perceptron, an original procedure of supervised learning. In 1960, Widrow and Hoff proposed the least-mean-square algorithm and used it to formulate Adaline - Adaptive Linear Element and Madaline - Multiple Adaptive Linear Element in 1962. In 1967, Amari used the stochastic gradient method for adaptive pattern classification. A great achievement, done by Malsburg in the 1970s, was self-organizing maps using competitive learning. Followed by the development of the Hopfield networks and Kohonen in 1982. Ackley, Hinton, and Sejnowski in 1985 developed the first achievement of a multilayer neural network which become known as Boltzmann machine. The development of the back-propagation algorithm was reported by Rumelhart, Hinton, and Williams in 1986. In 1988 the Radial Basis Function was developed by Broomhead and Lowe. In the early 1990s Vapnik created a computationally powerful class of supervised learning networks known as support vector machines (HAYKIN, 2007). In 1993 the first version of the WEKA machine learning software was released by Waikato University. In 1997, IBM's deep blue computer beat the chess world champion Garry Kasparov, and at the end of the 1990s a Prototype Intelligent Workstation, developed at the University Chicago, reviewed 22.000 mammograms and detected cancer 52% more accurately than radiologists did (MARR, 2016; BUILD, 2017).

Entering the 2000s, Fig. 1.2 shows the main advances, in the industry, about machine learning and deep learning methods. In 2006, Geoffrey Hinton presented the term "deep learning" to clarify new algorithms that allow computers to distinguish objects and text in images. In 2010, the Microsoft Kinect could trail 20 human features at a rate of 30 times/second, grating people to interact with the computer by movements and gestures. In 2011, IBMs Watson beat two human champions in a Jeopardy competition, and in the same year, Google Brain was developed where a deep neural network can learn to discover and categorize objects. One year later, the same brand created a machine learning algorithm capable to identify videos that contained cats from YouTube videos. In 2014, Facebook developed a software algorithm capable to recognize individuals on photos to the same level as humans can, called DeepFace. One year later, another big brand released its own machine learning platform called Amazon SageMaker. In the same year, Microsoft developed the Distributed Machine Learning Toolkit, allowing the efficient distribution of machine learning problems across multiple computers. At the end of 2015, the AlphaGo algorithm from Google beat a professional player at the world's most complex board game - Chinese Board Game Go

(MARR, 2016). In 2016, The North Face became the first brand to use IBM Watson's in a mobile app, helping the costumers to find what they are looking for through a conversation. In 2017, the Alphabets Jigsaw team developed a system that learned to identify trolling by reading millions of website comments as part of anti-harassment efforts (BUILD, 2017).

Figure 1.2 – Machine and deep learning 2000s timeline.



Source: Authors production based on Marr (2016) and Build (2017).

ANNs have seen increasing application for the determination of material properties, as well as damage detection and localization, especially for challenging complex multiphase and composite materials (SHA; EDWARDS, 2007). Table 1.1 shows some literature examples using machine learning in the mechanical engineering area.

Due to an increasing demand for lightweight, corrosion and chemical materials in the industry, the composites market has a promising future. Figure 1.3 shows the global composites market from 2015 to 2021 (expected), in terms of value and volume (HACLALIOGLU, 2018), reaching 12.9 million tons and \$ 103.0 billion. According to Smith (2018), the last report about Global Advanced Composite Materials Market, the global

Table 1.1 – Literature examples related to machine learning and mechanical engineering.

Authors, Year	Paper	Material	ML Method
França, 2014	Detecção e localização de danos em materiais compósitos aplicado em aeronaves utilizando redes neurais artificiais	composite	BP-ANN
Bandara, Chan and Thambiratnam, 2014	Frequency response function based damage identification using principal component analysis and pattern recognition technique	metal	BP-ANN
Abdeljabber <i>et al.</i> , 2016	Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks (CNN)	metal	CNN
Kumar <i>et al.</i> , 2016	Failure strength prediction of glass/epoxy composite laminates from acoustic emission parameters using artificial neural networks	composite	RBFNN
Zhang <i>et al.</i> , 2017	A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load	metal	CNN
Yang <i>et al.</i> , 2018	An intelligent fault diagnosis approach based on transfer learning from laboratory bearings to locomotive bearings	metal	FTNN and CNN
Hoang and Kang, 2018	Rolling element bearing fault diagnosis using convolutional neural network and vibration image	metal	CNN
Glowacz, 2018	Acoustic based fault diagnosis of three-phase induction motor	metal	BP-ANN
Khan <i>et al.</i> , 2018	Structural vibration-based classification and prediction of delamination in smart composite laminates using deep learning neural network	composite	CNN
Chelliah <i>et al.</i> , 2018	Optimization of acoustic emission parameters to discriminate failure modes in glass–epoxy composite laminates using pattern recognition	composite	k-means; FCM and KSOM
Finotti <i>et al.</i> , 2018	An SHM approach using machine learning and statistical indicators extracted from raw dynamic measurements	metal	ANN and SVM
Jiang <i>et al.</i> , 2019	Multiscale convolutional neural networks for fault diagnosis of wind turbine gearbox	metal	CNN

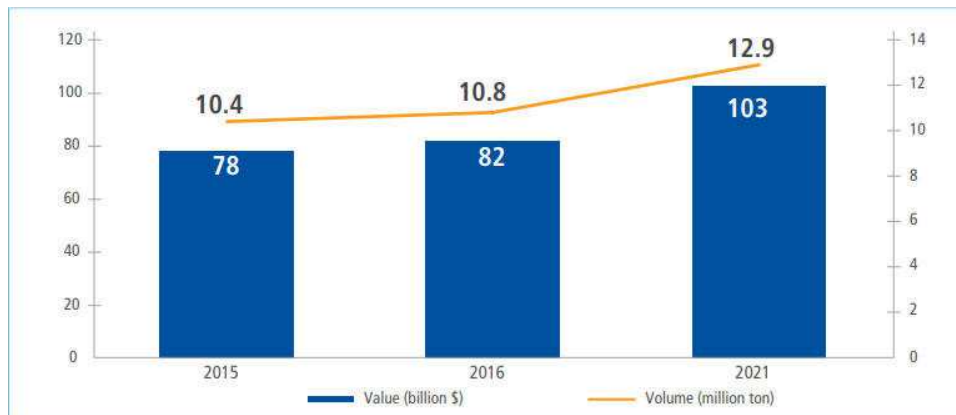
ML: machine learning; BP-ANN: backpropagation artificial neural networks; CNN: convolutional neural networks; FTNN: feature based transfer neural network; RBFNN: radial basis function neural network; FCM: fuzzy C-means; KSOM: Kohonen's self-organizing map; SVM: support vector machine.

Source: Author's production.

composites end product market is looking forward to reaching an estimated \$ 107.4 billion by 2023. Figures 1.4 show the volume-distribution (left) and value-distribution (right) of the composite materials applications in 2016, where the top three industries in terms of volume are logistics, construction and building, and electrical and electronic, and in term of value are logistics, electrical and electronic, and defense and aviation.

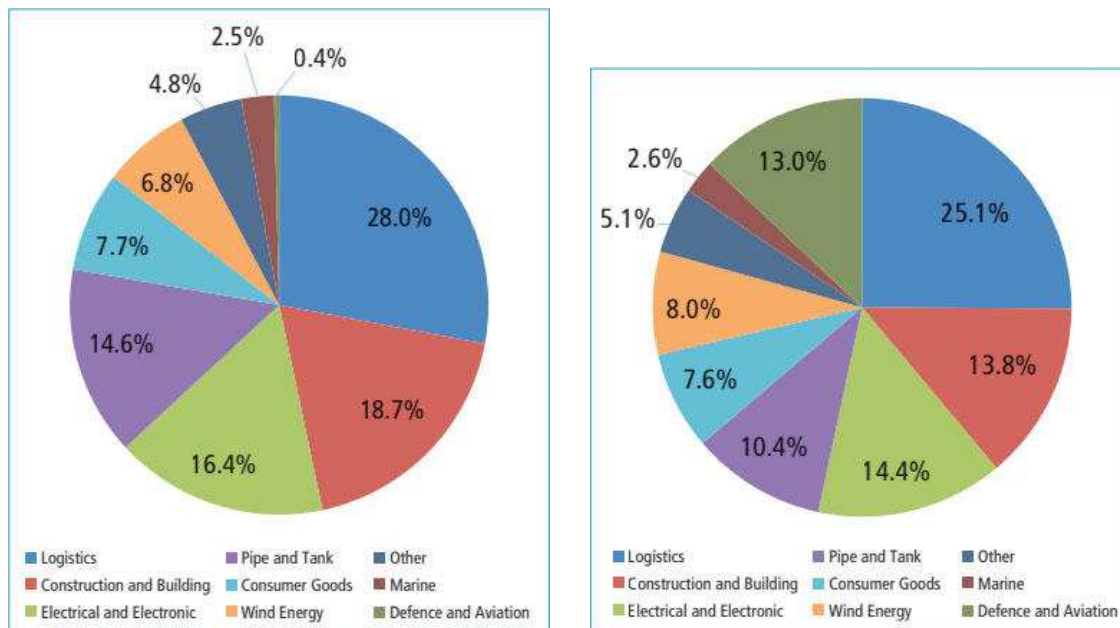
Nevertheless, the main drawback of composite materials is the difficulty in predicting the exact failure mode, reflecting in the reliability of manufactured structures. In the current context of aviation safety, it is crucial to prove that structures made of com-

Figure 1.3 – The global composites market (value and volume).



Source: Haclalioglu (2018)

Figure 1.4 – The global composites market application.



Source: Haclalioglu (2018)

posites are capable of sustaining loading even with damage, corrosion or errors during manufacture, without suffering accidents, failure or significant impacts until the damage is detected. This type of concept is called damage tolerance. The damage tolerance in metallic materials is already well developed and known, being possible to determine periods for inspection based on fatigue sensitivity evaluation and taking advantage of the slow growth rate of a crack. However, for composite materials, the knowledge behind the mechanical behavior is still in development (SILBERSCHMIDT, 2016).

The predominant damage types in laminated composites are interlaminar debonding, micro-cracks and micro-buckling. Delamination is undetected by visual procedures

and, as a result, is one of the most critical types of damage, being also the most common in aeronautical components (LOPES *et al.*, 2011). In addition, composite materials are almost insensitive to fatigue, especially for composites made of carbon fiber, a material broadly used in the aviation industry. This insensitivity to fatigue usually does not lead to increased damage in service, so it is difficult to predict the inspection intervals as in metals. Further, composite materials are highly sensitive to impact, which reduces the residual strength of the structure below a tolerable limit (SILBERSCHMIDT, 2016).

1.2 Objectives and Scope

Due to the expanding application of machine learning techniques, this work aims to evaluate the applicability of a methodology that contemplates Artificial Neural Networks (ANNs) and Principal Component Analysis (PCA) for the failure diagnosis in composite structures. Since the prediction of the use of these unconventional materials over the next 4 years in the global industry has a significant growth prospect. Due to the difficulty in predicting and diagnosing faults in composite materials, there is a need for a more in-depth study of a methodology capable of predicting in real time the presence of faults in these systems. Thus the principal objective can be split into small steps mentioned below:

- To propose a methodology to faults diagnosis using ANNs and PCA;
- To perform preliminary studies using conventional and non-conventional materials in order to evaluate the application of the methodology to some non-complex systems;
- To manufacture undamaged and damaged composite beams and carry out experimental analyses using vibration-based methods to acquire Frequency Response Functions (FRFs);
- To evaluate the potentialities and the limitations of the methodology in composite materials.

1.3 Brief Outline of Thesis

The thesis consists of six chapters as follow:

- **Chapter 1: Introduction** - describes a brief overview of the background and motivation as well the objectives and the scope of the study.

- **Chapter 2: Structural Health Monitoring - Review** - presents the principal concepts of the methods and areas involved in the study, an overview of Structural Health Monitoring (SHM), Composite Materials and Vibration-based Methods,
- **Chapter 3: Pattern Recognition - Review** - presents the concepts of Pattern Recognition, Artificial Neural Networks, Principal Component Analysis, Particle Swarm Optimization, Automatic Differentiation, and Dual Numbers.
- **Chapter 4: Materials and Methods** - explains the methodology for faults diagnosis step by step, the processes involved during the study, and the experimental analysis.
- **Chapter 5: Results** - presents the results obtained with the study as well as some discussions.
- **Chapter 6: Conclusions and Future Works** - describes the potentialities and limitations of the methodology and the recommendation of future works.

Chapter 2

Structural Health Monitoring - Review

The present chapter presents concepts and mathematical formulae as well as the methods used in this work.

2.1 Structural Health Monitoring

The goal of Structural Health Monitoring (SHM) is to identify as early as possible the changes introduced in a system that affect its performance. Thus, corrective actions can be taken in a way that minimizes time, operational and maintenance costs, and therefore prevents greater damages and losses (YUAN, 2016). The process of SHM comprehends the continuous observation of the structure using periodic measurements to analyze its structural characteristics. The extraction features, using these measures combined with statistical techniques, determine the actual health state of the system (FARRAR; WORDEN, 2007).

In the question of why there is much attention to SHM techniques, two quick answers come in mind: financial motivation and human life protection. Many structures undergo periodic inspections and maintenance to ensure the structural stability of the system, so an early stage of damage detection can have a considerable economic impact. The costs of these inspections could be reduced if these inspections are shown to be not required in case of a healthy system, and this could automatically be displayed by using an SHM system (DERVILIS, 2013).

Farrar and Worden (2007) define SHM as a technique in terms of four-step statistical pattern recognition:

1. Operational evaluation.
2. Data acquisition, normalization and cleansing.
3. Feature selection and information condensation.

4. Statistical model development for feature discrimination.

The operational evaluation consists in a study of the characteristics of the system and how the monitoring will be performed, especially if the features are unique to the system and then can be used as an advantage for damage identification. In this step, some questions are raised as life-safety justification for performing SHM, damage types possibilities and which cases are of the most concern, operational and environmental conditions, and the data acquisition limitations.

The data acquisition step encompasses selecting the excitation methods such as quantity and types of sensors, sensors location, sensors resolution, bandwidth, data acquisition systems software, and test conditions. For test conditions, it is necessary to evaluate the interval that the data should be acquiring (MONTALVAO *et al.*, 2006). Data normalization is the action of separating changes in sensor reading induced by damage from those caused by varying operational and environmental conditions. One of the most known methods is to normalize the measured responses by the measured inputs. When the variability is a problem, especially when changing environmental and test conditions, it is necessary to normalize the data in some temporal fashion to help comparing the acquired data to similar times in an operational cycle. Data cleansing is the procedure of selectively choosing data to pass on to or reject from the feature selection process, especially when the acquired data shows some noise coming from sensors or poor operational performance (FARRAR; WORDEN, 2007).

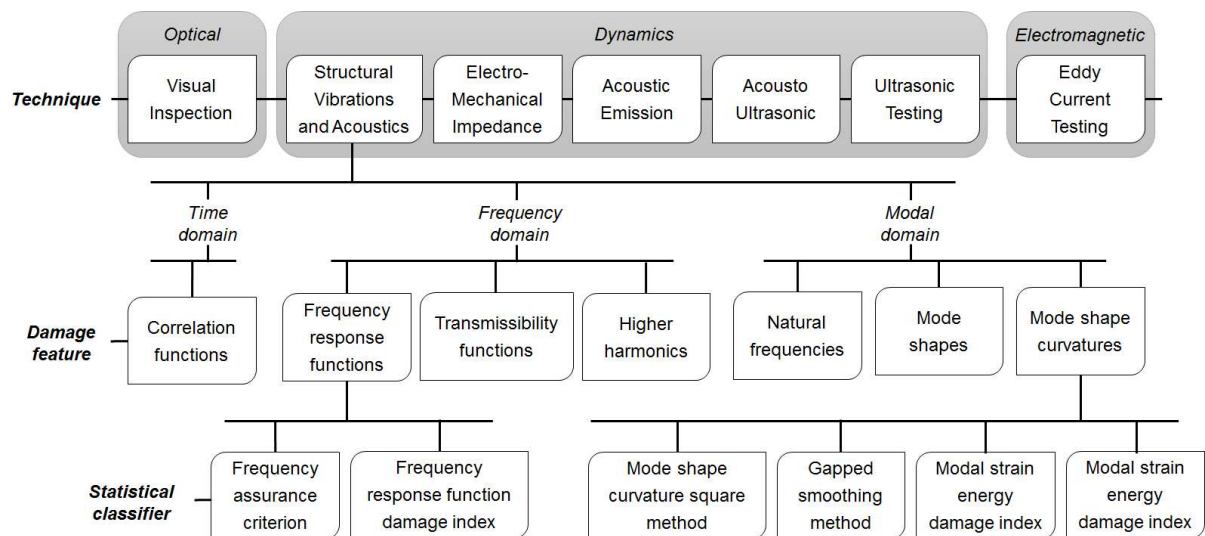
The data feature extraction receives the highest attention because it allows distinguishing between healthy and damaged cases. The most common feature extraction technique is based on mechanical system responses, such as the vibration-based method. Another technique, widely used in finite element computational simulation, is to apply a failure similar to real conditions to the system and evaluate which parameters are sensitive to the expected damage. A condensation of the data is needed when comparisons are made between data obtained during the life cycle of the system (MONTALVAO *et al.*, 2006).

The statistical model phase is the implementation of an algorithm that operates on the extracted features to quantify the damage state of the structure. Another traditional aim of feature selection and discrimination is to generate quantities with a low dimension. The sense for this is that the data requirements of learning algorithms usually grow exponentially with the dimension of the problem - the so-called curse of dimensionality (WORDEN; MANSON, 2006). The algorithms for statistical models are divided into three categories: classification, when the data from both intact and damaged are available; analysis of outliers, when one has only undamaged data, and regression analysis which refers to the process of correlating data features with particular types, locations or extents of damage (FARRAR *et al.*, 2001). The statistical models are used to answer the five questions regarding the damage state of the structure including dam-

age existence (I), damage location (II), damage type (III), damage extension (IV) and residual life of the structure (V). Statistical models are also used to study and minimize the possibility of false damage indications. Which are divided into (1) false-positive damage indications and (2) false-negative damage indications. The first one refers to damage indication when none is present. These errors can cause unnecessary downtime and operational resources, as well as a loss of confidence in the SHM. The second one refers to the cases when there is no indication of damage, but the damage is present. When this type of false indication is present, errors can cause catastrophic failure or even loss of life (FARRAR; WORDEN, 2007).

An overview of some damage identification approaches can be seen in Fig. 2.1, focusing on vibration-based methods and their damage feature types as well as the statistical classifier.

Figure 2.1 – An overview of the range of damage identification approaches, focusing in vibration-based methods.



Source: Adapted from Ooijevaar (2014).

A summary of some techniques applied to damage detection in composite structures is described in Tab. 2.1 with an overview of their advantages and limitations, damage state scale and damages composites types.

Table 2.1 – Summary of the techniques applied to damage detection in composite structures.

Technique	Advantages	Limitations	Damage scale	Types of damages
Vibration -based	Easy applicability; cost-effective; high sensitivity to damage; online monitoring; global area	Errors in measurements; environmental factors.	I,II, III and IV	Delamination and cracks.
Lamb waves	Travels over long distances without deviating; Monitoring large area from a single point; cost-effective; can detect internal damages in thin material; safe and no harmful radiation.	Cannot detect small damage; multiple wave-form in single frequency affect its application; requires skilled personnel for interpretation;.	I and II	Delamination; matrix cracking;broken fiber; impacts; adhesive defects on stiffness, and porosity.
Acoustic emission - AE	Can detect several types damages caused by fatigue loading; high sensitivity; fast results and global monitoring using multiple sensors; used for leak detection and location; online monitoring and global area.	Difficult to find damage; cannot characterise damage; requires load application to generate AE event; requires skilled personnel to correlate data to the specific damage mechanism; a crack that is not propagating cannot be detect.	I, II and IV	Translaminar cracks; fiber breakage; delamination; fiber matrix debonding and matrix microcracks.
Ultrasonic inspection	Easy to interpret; can detect early stage of damage initiation; cheap and readily available.	Limited in depth because of attenuation; variations in composite properties affects its performance.	I, II and III	Cracks; delamination and debonding.
Acousto ultrasonic	Assessment of non-critical damages; global area; a good indicator of accumulated damage due to impact damage.	Not useful for the detection of delamination and voids; mandatory setup and pre-calculations before testing; surface roughness and texture affect its performance; offline monitoring.	I, II, III and IV	Translaminar cracks and debonding.
Eddy currents	Convenient to apply; no contact needed.	Not applicable for all composites; offline monitoring; local detection.	I, II, III and IV	Cracks; delamination and porosity.
Digital Image Correlation	Simple to use; non-contact needed; images in 3D.	Needs high-resolution measurements; requires output post-processing; requires sufficient black and white contrast.	I, II and III	Microcracks, delamination, pullout, progressive debonding of interfaces; crack along interfaces.

Source: Based on Amafabia *et al.* (2018), Speckmann and Henrich (2004) and Hild *et al.* (2015).

2.2 Composite Materials

According to Reddy (2004), composite materials consist of two or more combined constituents, which produce desirable properties, such as low density, high strength and stiffness, excellent fatigue and corrosion resistance. One constituent is the reinforcing phase and can be in the form of fibers, particles or flakes. Another one is called matrix or continuous phase. The reinforcement provides strength and stiffness, while the matrix provides protection. The mechanical performance depends on the reinforcement and matrix properties, and the amount and orientation of the reinforcement (CHUNG, 2010). The reinforcement arrangement is led by structural concerns and the fabrication process. Usually, composite materials are made of thin layers called laminate or plies.

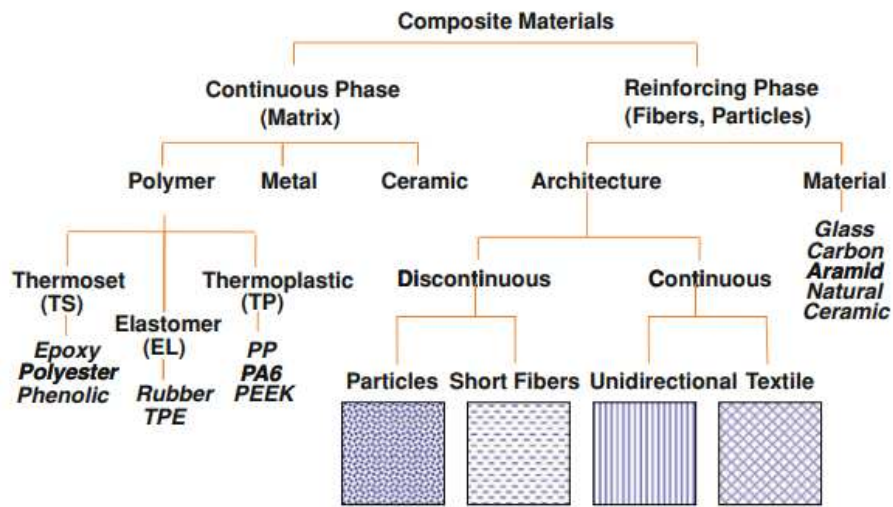
According to Kaw (2005), the main drawbacks and limitations when working with composite materials include: (1) high cost of composites fabrication; (2) the mechanical characterization is more complex compared with isotropic materials; (3) the damaged detection in composite materials is not a simple process, especially when some critical cracks may be undetected; and (4) composites do not necessarily give the best performance in all the mechanical properties.

2.2.1 Composite classification

Figure 2.2 shows the classification of the two phases of composite materials. Some materials usually used as the continuous phase are polymers, metals, and ceramics. The polymer material can be classified into three other categories: thermoset, elastomer, and thermoplastic. Architecture can be continuous or discontinuous. Discontinuous composites contain particles and short fibers, nanotubes or whiskers that can be either oriented in some directions or randomly. Particulate composites consist of particles of different sizes and shapes randomly dispersed within the matrix. Continuous fiber can be aligned in the same direction, called unidirectional continuous-fiber composite or in different directions, called multidirectional continuous-fiber composite or textile (KOLLÁR; SPRINGER, 2003). The material used as fiber can be Glass, Aramid, Carbon, among others.

Fibers are a singular class of materials because of their anisotropy. Usually, fibers have a high length compared to their diameter. This so-called high aspect ratio is responsible for the singular properties compared to a bulk material. Common fibers for composite employments have high strength and stiffness in fiber direction, while they are weak and flexible perpendicular to it. Another particular property is the high surface area of fibers, induced by their small diameter compared to their length. The fiber surface, with its topography and chemistry, is the interface to the matrix material.

Figure 2.2 – Structural components of composite materials.

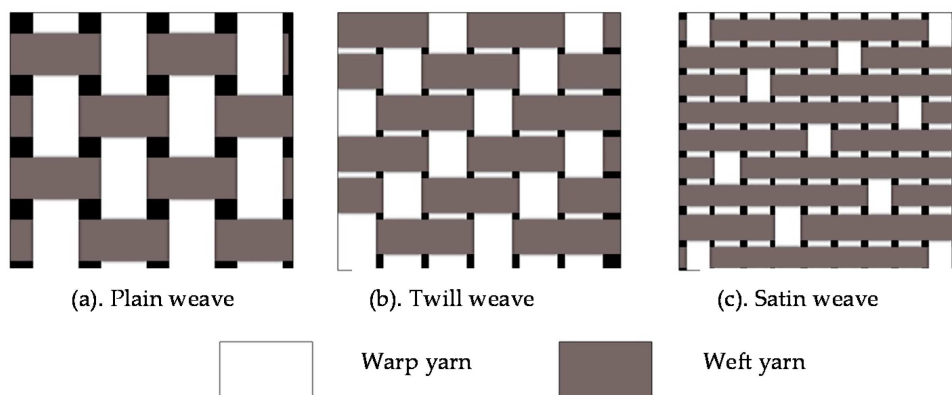


Source: Nicolais *et al.* (2011).

Consequently, the interface between fiber and matrix is responsible for the load transfer to the fiber (RANA; FANGUEIRO, 2016).

Textile structural composites are widely applied in many industrial applications, as they have better specific properties in comparison to basic materials (metal and ceramics)(RANA; FANGUEIRO, 2016). Woven fabrics have a structure in which warp yarns and weft yarns are interlaced. The possibilities of woven styles in textile reinforcement are categorized, basically, in three types of patterns, which can be assembled in increasingly complex configurations. These patterns are plain weave, harness satin weave and twill weave (FAZITA *et al.*, 2016), and can be seen in Fig. 2.3. In the plain weave

Figure 2.3 – Woven fabric styles.



Source: Fazita *et al.* (2016).

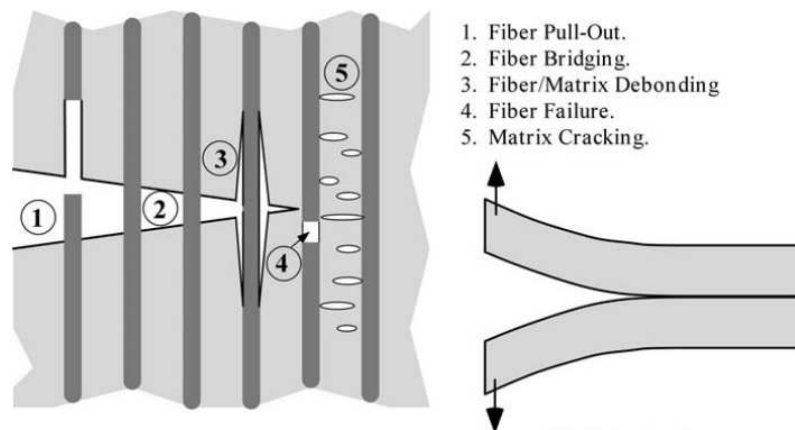
pattern the warp and weft yarns are interlaced in a regular sequence of one under and one over; in the twill weave pattern the yarns are interlaced to produce a pattern of

diagonal lines, and the satin weave pattern is characterized by sparse positioning of interlaced yarns.

2.2.2 Mechanisms of failure in composites materials

The diversification of microstructure and the anisotropy properties provide notably distinct features to composite materials in how they deform and fail when compared to other materials. Furthermore, when an interface is present, the stress transfer by interfaces provides conditions for multiple cracking (TALREJA; SINGH, 2012). Different manufacturing methods may promote different process-related defects including porosity, shrinkage cracking and fiber-matrix debonding due to resin shrinkage. Figure 2.4 shows different types of failure mechanisms: (a) tensile load can produce matrix cracking, fiber bridging, fiber rupture, fiber pull-out, and fiber-matrix debonding. (b) out-of-plane stress can lead to delamination.

Figure 2.4 – Types of composites cracks mechanisms (a) In-plane damage, (b) Delamination.



Source: Anderson (2017).

- **Fiber-matrix debonding.**

Fiber debonding due to shrinkage arises as a result of low adhesion between fiber and matrix, causing a weak interface. Another reason for the appearance of this type of crack is thermomechanical loading (TALREJA; VARNA, 2015). If the fiber-matrix interaction is weak, the composite begins to form a matrix crack even at low stress. If the interaction is strong, this type of cracking is delayed, but the composite will fail drastically due to the fiber breakage that will lead to the pull-out phenomenon as the matrix cracks. Pull-out is characterized by the fiber pulling from the matrix. Before the pullout, there may be the formation of Fiber Bridging, where the crack propagates through the matrix, and the fiber forms a kind

of bridge interconnecting the two surfaces of the crack matrix. In a unidirectional composite, the interfacial debonding occurs when the fibers are poorly held by the matrix. When the fiber fracture strain is larger than the matrix, the crack originated at a stress point concentration in the matrix is interrupted by the fiber in the low-stress case. On the other hand, when the load applied increases, there is a local stress accumulation in the fiber, causing a local Poisson contraction, and meanwhile, the shear stress developed at the interface surpasses the interfacial shear strength. Thus, fiber debonds growing from the ends of the fractured fiber along the fiber length (TALREJA; SINGH, 2012).

- **Matrix cracking.**

Generally caused by tensile loading, fatigue loading or by changes in the temperature. It leads to stiffness reductions in composite laminates, not causing structural failure, but generating more fatal types of damage such as delamination and fiber breakage. They may originate from the fiber-matrix debonding or due to defects in the manufacturing process (TALREJA; VARNA, 2015). The mechanical properties of the composite materials are generally low in the transverse direction, which results in cracks along the fibers. In laminates of different plies orientations, these cracks can appear from imperfections in a ply and extend across the thickness of the ply and begin to run parallel to the fiber in that ply (TALREJA; SINGH, 2012).

- **Fiber breakage.**

Fiber-reinforced composites are made from bundles of fibers, and the failure strain of each fiber within the bundle will not be the same, mainly because of the different size of fibers or imperfection during fiber manufacture. If these fibers are used for a manufacturing composite, they will crack at different values of the applied strain, usually generating isolated fiber fractures. At higher strains, the local stress concentrations produced by isolated fiber fractures can cause failure in adjacent fibers, leading to an increase in fiber fractures and composite failure (TALREJA; VARNA, 2015). Compressive loading can produce fibers micro-buckling (TALREJA; SINGH, 2012).

- **Delamination.**

Due to mismatch of the elastic properties of interlaminar plies, cracking in the interfacial plane (between two adjacent plies in a laminate) causes separation of the plies, as delamination. This phenomenon can happen at free edges or

at an exposed surface through the thickness. Also, it can be formed as a consequence of low-velocity impact. Delamination accompanies most failure processes in composites, but it is most critical under compressive stress, either from direct compressive loading or as induced by bending loads. The main problem of delamination is that it can decrease the performance of strong fibers and make the properties of the weaker matrix control the structural strength (TALREJA; SINGH, 2012). The general analytical treatment of delamination is to partition it into modes: mode I (peel), mode II (in-plane shear) and mode III (out-of-plane shear). The existence of delamination causes a redistribution of stress within the laminate, which may influence the initiation of fiber breakage in the primary load-bearing plies and reduce the fatigue life of the laminate (MALLICK, 2007).

2.2.3 Damage tolerance of composites structures

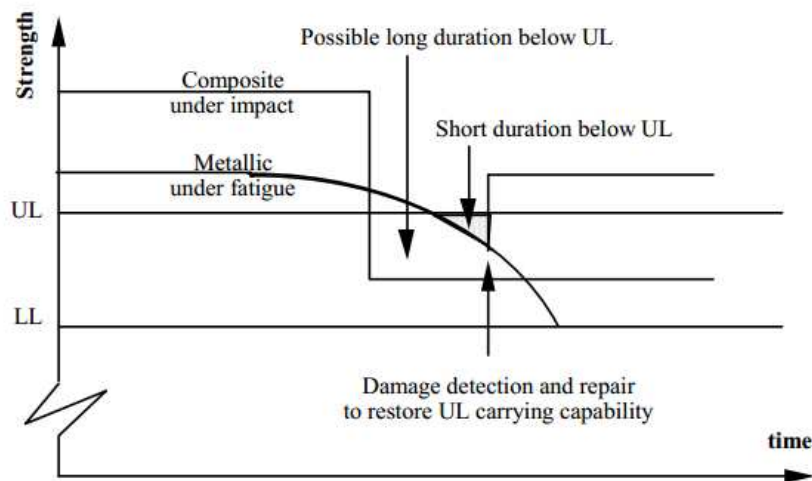
According to Silberschmidt (2016), the damage tolerance concept, introduced in the 1970s for civil aircraft structures, corresponds of certifying that the structure is able to sustain acceptable loads, without significant deformations, failures and break until the damage is detected. Damage tolerance means:

- Even with damage, the residual strength of the structure must stay higher than the Limit Load (LL).
- Any damage advancing to residual strength below the Ultimate Load (UL) must be immediately detected and fixed, restoring the strength up to the UL.

The type of in-service damage that most significantly affects the strength of composite structures is the damage caused due to the impact. The delaminations in the impact area have an unfavorable effect, especially on the compression strength. According to Handbook-MIL-HDBK (2002), a laminate can lose 60 to 65% of its undamaged static strength by impact damage that is essentially non-visible (apud Heida and Platenkamp (2012)). In addition, delaminated areas can grow in-service due to the moisture uptake that undergoes a repeated freezing-thawing period meanwhile consecutive flight cycles (HEIDA; PLATENKAMP, 2012).

Figure 2.5 compares a composite non-growing damage and metal fatigue crack damage. The damage tolerance of metallic materials is based on a slow-growth approach. The damage growth in metallic structures is relatively slow and generally well controlled, enabling the determination of inspection intervals to guarantee that damage does not grow too much undetected. Initially, the residual strength is constant until the existence of the damage. As the damage increases the residual strength decreases, passing below the UL and if the damage is not detected passing the LL. This damage

Figure 2.5 – Comparison of composite non-growing damage and metal fatigue crack damage (UL: Ultimate Load and LL: Limit Load).



Source: Heida and Platenkamp (2012).

is known as critical damage. The objective is to decrease the time spent below the UL (SILBERSCHMIDT, 2016).

In contrast, strength in composites reduces not continuously - even with damage there is generally no damage growth due to their insensitivity to fatigue - then it becomes impossible to determine maintenance intervals based on the concept of slow growth (SILBERSCHMIDT, 2016). However, an impact can abruptly drop the strength to an unwanted level - below UL. The requirement then is that as long as damage cannot be detected visually, it should never drop the structural strength below UL. Only detectable damage may cause structural degradation below UL (but never below LL), and should be properly detected by visual inspection or more advanced NDI methods. The inspection interval should be related to the probability of damage. After detection, the damage must be fixed to recover UL capability or the component should be replaced (HEIDA; PLATENKAMP, 2012).

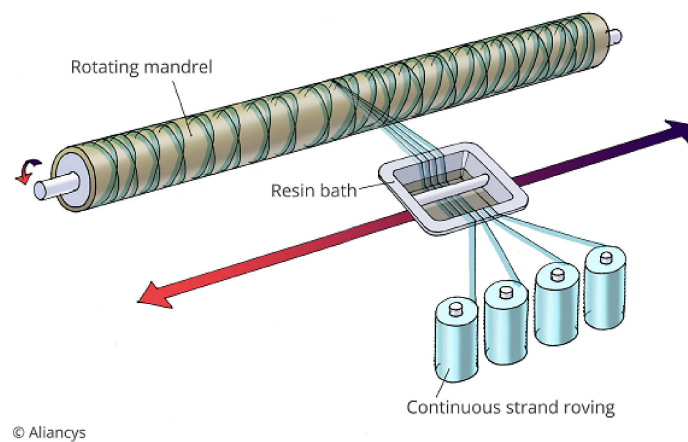
2.2.4 Manufacturing processes

The most widely employed manufacturing processes used for high-performance composites includes automated ply cutting, manual lay-up, and autoclave curing. Most of them are very expensive. Considering the high cost is a problem, a number of low cost processes such as automated tape laying, filament winding, and fiber placement are adopted for certain classes. Another low-cost fabrication process is in limited production and has received significant attention, such as liquid composite molding and pultrusion (JUNIOR, 2011).

In the filament winding process, the filaments (fibers) are properly impregnated

by a polymer and subjected to winding on a rotating mandrel, as shown in Fig. 2.6. The cure can be carried out at room temperature, but a post-cure stage is generally carried out at elevated temperatures. After the healing process, the mandrel is removed (LEVY-NETO *et al.*, 2016). The positioning and angle of the fiber are defined by controlling the mandrel, and carriage speeds and the relative volume of fibers are controlled by their tensioning. Despite the high initial cost, one of the main advantages of the method is the fast production of parts with high fiber volume (about 70%) (MENDONÇA, 2005).

Figure 2.6 – Filament winding process.

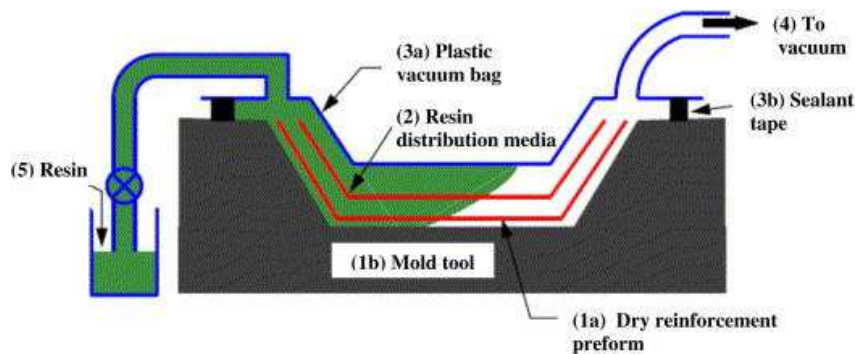


Source: Aliancys (2019).

Liquid composite molding (LCM) is a manufacturing process capable of building complex and accurately dimensional parts. One of the most used liquid molding processes is the resin transfer molding (RTM). This technique consists of fabricating a dry fiber which is placed in a mold, impregnated with resin and then cured in the mold. Another variation of RTM is the VARTM (Vaccum Assisted Resin Transfer Molding), where vacuum is used in addition to the resin injection to pull the liquid resin into the preform (JUNIOR, 2011), as shown in Fig. 2.7.

For the LCM process it is import to consider the viscosity of the resin. The viscosity increases with increasing cure time and temperature. The rate of viscosity increase is small at the initial phase of curing. After a threshold degree of cure is complete, the resin viscosity increases at a quick rate. The time at which this happens is known as the gel time. The gel time is a crucial molding parameter since the flow of resin in the mold becomes progressively difficult at the end of this time cycle (MALLICK, 2007).

Figure 2.7 – VARTM process.

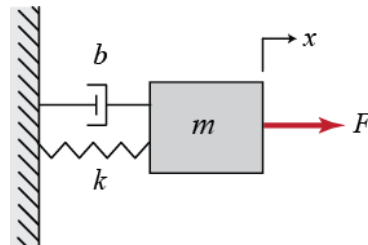


Source: Kuentzer *et al.* (2007).

2.3 Vibration-based Methods

According to Fu and He (2001) vibration can be described as time transfer between the kinetic energy and potential energy. A vibratory system has to contain a means of storing and releasing both energies, as the single degree of freedom mass-spring-damper system showed in Fig. 2.8.

Figure 2.8 – Single degree of freedom mass-spring-damper system.



Source: Matlab (2017).

Most classes of damage detection methods use structural dynamic characteristics, including Frequency Response Functions (FRF), natural frequencies and mode shapes (BANDARA *et al.*, 2014). In addition, the main idea behind damage detection techniques based on structural dynamic changes, is the fact that the modal parameters are functions of the physical parameters like mass, stiffness, and damping, and thus, it is reasonable to assume that the existence of damage leads to changes in the modal properties of the structure (CAMPBELL, 2010). On the other hand, the low sensitivity of natural frequency to damage requires high levels of damage and the experimental analysis should be made with high accuracy for reliability (MONTALVAO *et al.*, 2006). Another problem is that the natural frequencies are easily overwhelmed by environmental changes such as humidity and temperature, generating unrealistic results. The mode shapes are also influenced by environmental changes, and additionally affected by the number and location of sensors (BANDARA *et al.*, 2014). In recent years, the use of FRF

data has grown for damage identification and detection, mainly due to the fact that it requires a small number of sensors, simple application and, beyond that, the fact that it can be obtained in real time. However, FRF approaches have some disadvantages such as the large size of FRF data, and that as natural frequencies, they are sensitive to environmental fluctuations and to measurement noise. An improper selection of the frequency windows from which the data are drawn results in the loss of important information and introduces errors to the damage identification strategy (DACKERMANN, 2009).

Rao and Yap (2011), define the motion equation of a structure with N degree of freedom as,

$$M\ddot{x}(t) + B\dot{x}(t) + Kx(t) = f(t), \quad (2.3.1)$$

where M , B and K represent the $N \times N$ mass, damping and stiffness matrices, respectively. If an harmonic input is assumed, the external force can be expressed as,

$$f(t) = F(\omega)e^{i\omega t}, \quad (2.3.2)$$

and displacement as,

$$x(t) = X(\omega)e^{i\omega t}. \quad (2.3.3)$$

Substituting Eq. (2.3.2) and Eq. (2.3.3) into Eq. (2.3.1)

$$(-\omega^2 M + j\omega B + K)X(\omega)e^{i\omega t} = F(\omega)e^{i\omega t}. \quad (2.3.4)$$

Then Eq. (2.3.4) can be expressed as,

$$X(\omega) = H(\omega)F(\omega). \quad (2.3.5)$$

Where the FRF matrix, $H(\omega)$, is defined as,

$$H(\omega) = (-\omega^2 M + j\omega B + K)^{-1}. \quad (2.3.6)$$

2.3.1 FRF measurement set-up

Frequency response function (FRF) measurement set-up should have three steps. The first step is responsible for producing the excitation force, which applies a force of enough amplitude and frequency contents to the structure, that can be done using a hammer or a shaker. The hammer is made of a tip, a force transducer, a balancing mass and a handle. The hammer tip can be altered to change the hardness according to the material of the structure, and it is related to the frequency range of the input pulse force. The second step is to measure the data, by using a transducer,

such as an accelerometer. It measures the acceleration of a test structure and the output signal in the form of voltage. There are two considerations in the acceleration measurement that a sensor must be able to deal with: frequency and amplitude. Both are reflected in the input-output relationship of an accelerometer. The main parameters influencing the execution of a piezoelectric accelerometer are:

- frequency range property: decides the linearity of the sensor;
- sensitivity: determines the signal to noise ratio. Large and stable sensitivity means accurate measurement;
- cross-axial sensitivity: generates inaccuracy in measurement;
- base strain: is caused by the flexure of the accelerometer base interacting within a non-rigid structure surface.

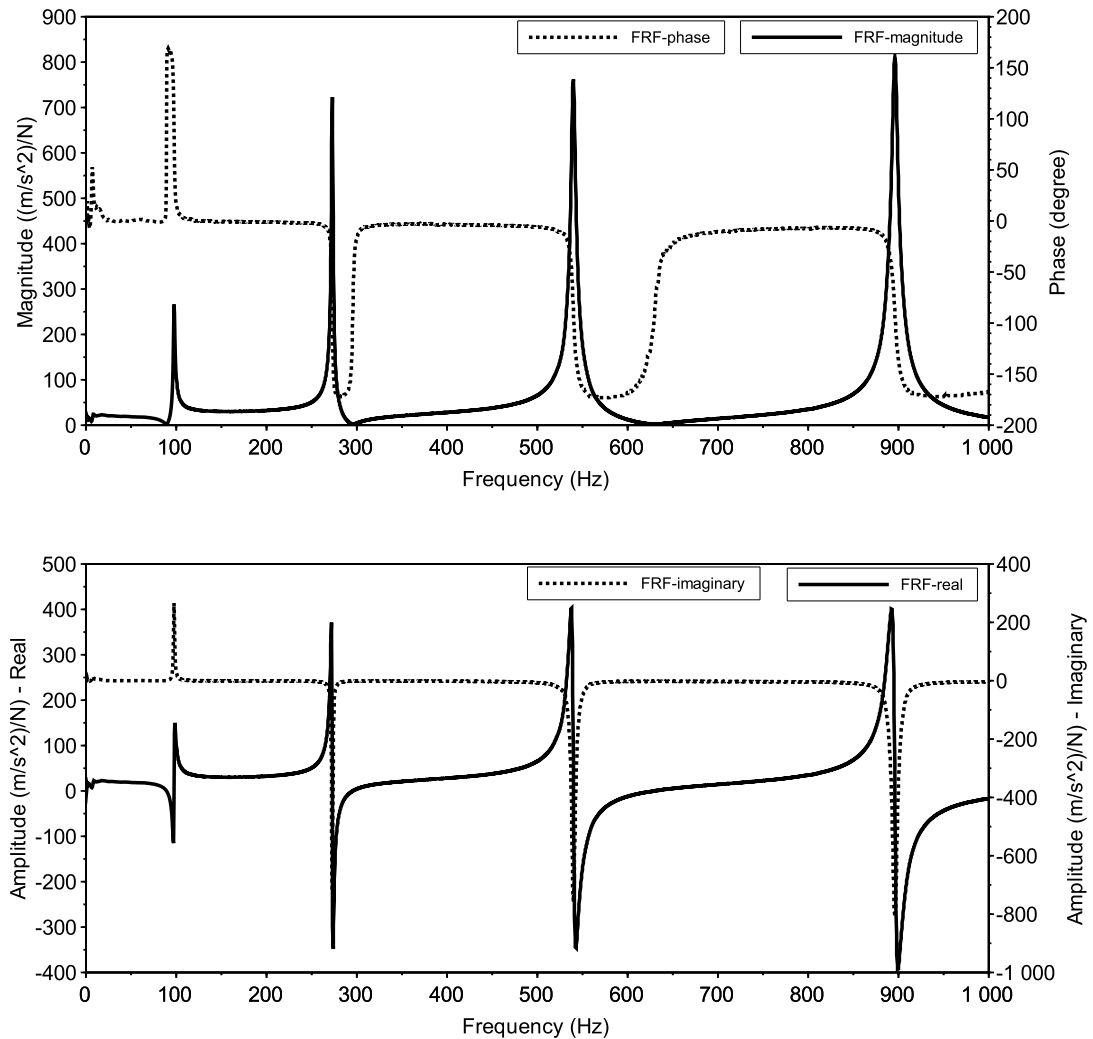
The accelerometer mass has the potential to change the characteristics of the test structure. Finally, the third step provides the signal processing capacity to derive FRF data from the measured force and response data (FU; HE, 2001).

During the FRF measurement, some adjustments must be made to get clean data. One of them is to do the pre-trigger with the hammer. As the impulse signal exists for a short period of time, it is essential to catch all of it in the sampling window, *i.e.*, the analyzer must be able to capture the impulse, and the response signals prior to the occurrence of the impulse, which is often set to a small percentage of the peak value of the impulse. Another setting is the force and exponential windows. The force window is applied to remove noise from the impulse signal. The exponential window is used in the impulse response signal to reduce leakage in the spectrum of the response (SCHWARZ; RICHARDSON, 1999). Leakage occurs from the transformation of time data to the frequency domain using the Fast Fourier Transform (FFT). The FFT calculations need that the sampled data consist of a complete representation of the data for all time or contain a periodic repetition of the measured data. When this is satisfied, then the FFT produces a proper representation of the data in the frequency domain. However, when this is not the case, then leakage will cause severe distortion of the data in the frequency domain (AVITABILE, 2001). Another problem in Fourier analysis is known as aliasing. The existence of very high frequencies in the original signal may well be misinterpreted if the sampling rate is too slow (EWINS, 2000). A signal must be sampled at a rate higher than twice the frequency of the highest frequency of the interest, according to the Nyquist sampling theorem (CRAIG; KURDILA, 2006).

Graphical display of an FRF plays a vital role in the modal analysis. A different graphical display highlights different information an FRF carries. They are displayed as an amplitude-phase plot, a real-imaginary plot, a Nyquist plot, and a dynamic stiffness

plot (FU; HE, 2001). Figure 2.9 shows an example of an amplitude-phase plot and an example of the real-imaginary plot.

Figure 2.9 – Graphical display of an FRF: (a) Amplitude-phase plot and (b) Real-imaginary plot.



Source: Author's production.

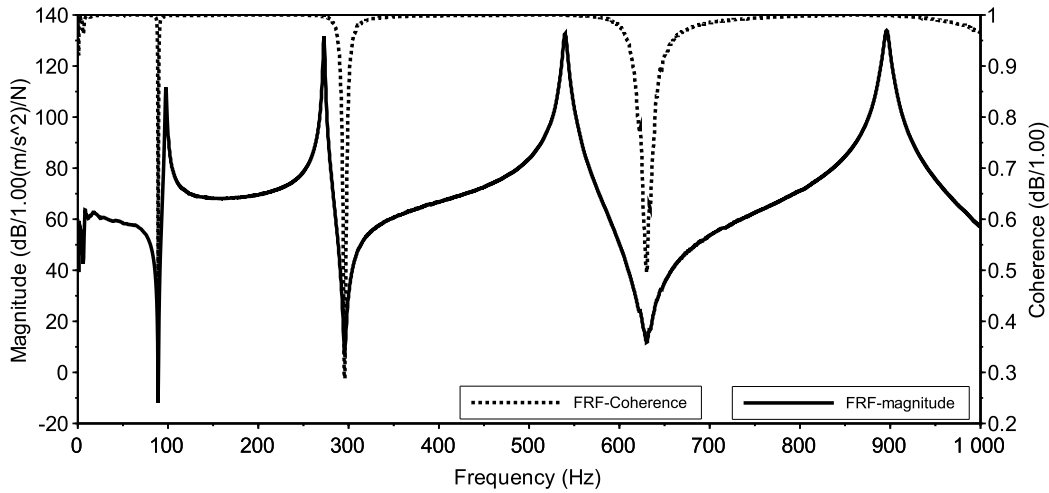
A measure of the amount of the output signal that is due to the input signal is the coherence, known as γ^2 , defined as,

$$\gamma^2(\omega) = \frac{H_1(\omega)}{H_2(\omega)}, \quad (2.3.7)$$

where H_1 (Noise on the output) and H_2 (Noise on the input) are the FRF estimators (CRAIG; KURDILA, 2006). The coherence function is used as a data quality assessment tool, which identifies how much of the output signal is related to the measured input

signal. Poor coherence is indicative of a poor signal to noise ratio, measurement errors, nonlinear or time-variant, the behavior of the structure or a combination of them (FU; HE, 2001). An example of a coherence plot is shown in Fig. 2.10.

Figure 2.10 – Coherence display plot.



Source: Author's production.

2.3.2 Damping ratio

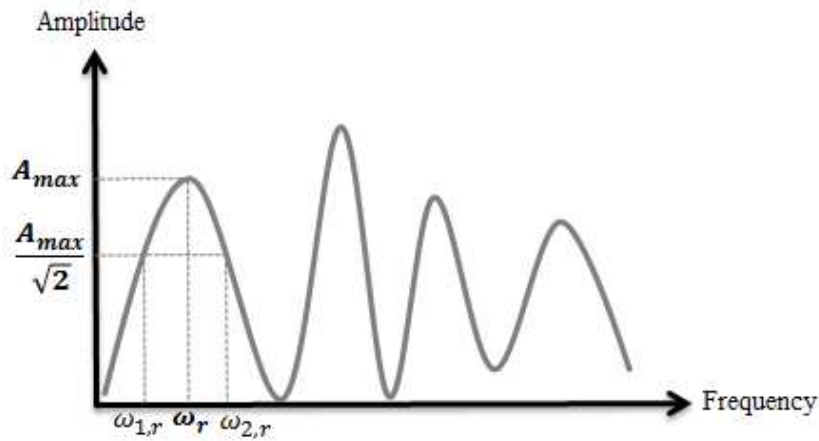
According to Fu and He (2001) the peak-picking method, also called the half-power method, is the simplest single degree of freedom (SDoF) method for estimating modal parameters, such as resonance frequency and damping ratio (ζ). The method is based on the FRF observation, looking for a point of maximum amplitude, *i.e.*, a peak (A_{max}). The frequency at which the peak value is observed is known as the resonance frequency (ω_r). The damping factor can be estimated by finding the points $\omega_{1,r}$ and $\omega_{2,r}$ on both sides of the FRF peak, which correspond to amplitude $\frac{A_{max}}{\sqrt{2}}$, as shown in Fig. 2.11, defined as,

$$\zeta_r = \frac{\omega_{2,r} - \omega_{1,r}}{\omega_r} \quad (2.3.8)$$

There are four common types of damping mechanisms used in the construction of vibratory models, such as: viscous damping, Coulomb damping, hysteretic damping, and fluid damping (FU; HE, 2001). Nevertheless, according to Amafabia *et al.* (2018) "proportional damping can be assumed as a special case of damping and it suggests that the damping matrix is a linear combination of the mass and stiffness matrices".

Proportional damping has found several applications in finite element analysis

Figure 2.11 – Peak-peaking method.



Source: Author's production.

where damping needs to be included to carry out significant response analysis and prediction. Considering again the Eq. (2.3.1), in general, the viscous damping matrix **B** cannot be mathematically constructed from the damping matrices of the elements, as is done for the mass matrices and stiffness through the classical Finite Element Method formulation. A usual alternative is the use of Proportional Damping (or Rayleigh), defined as

$$\mathbf{B} = \alpha \mathbf{M} + \beta \mathbf{K}, \quad (2.3.9)$$

where α and β are proportional constants (FU; HE, 2001), obtained using

$$\zeta = \frac{\alpha}{2\omega_n} + \frac{\beta\omega_n}{2}. \quad (2.3.10)$$

So, it is necessary to know two natural frequencies and two damping ratios, so that it is possible to compose the linear system of two equations and two unknowns (NEGRI, 2018).

Chapter 3

Pattern Recognition - Review

3.1 Pattern Recognition

Pattern recognition is a process where a signal is referred to as one of a given set of categories. Patterns recognition carried out by an ANN is statistical in its nature, with the patterns being points in a multidimensional decision space, which is divided into regions related to each category, according to their characteristics. Haykin (2007) complemented that the existence of a large number of input variables can present some serious problems for pattern recognition systems. Data pre-processing is one of the essential steps in the development of the solution, and the selection of pre-processing methods can often have an important consequence on generalization performance. One essential form of the pre-processing involves the reduction in the dimensionality of the input data. Another approach involves forming linear (or non-linear) combinations of the original variables to make a small number of new variables called features (BISHOP, 1995a). Many techniques can be used for pre-processing such as Principal Component Analysis, Factor Analysis, Clustering, among others.

3.1.1 Confusion matrix

The confusion matrix is used to find out how the errors are distributed across the patterns (KUNCHEVA, 2004). The method is also commonly used to measure over- and underestimates of a particular category according to Hay (1988). Figure 3.1 shows an example of a confusion matrix, where true positive (TP) is the number of samples of the damaged class that were correctly classified as damaged and false positive (FP) is the number of samples not belonging to damaged class but misclassified into damaged class. False negative (FN) is the number of damaged samples class misclassified as an undamaged class, and true negative (TN) is the number of samples of undamaged class that were classified correctly. Recall (R), or sensitivity, is the proportion of posi-

Figure 3.1 – Confusion matrix example.

		Actual Class	
		positives	negatives
Predicted Class	positives	TP true positive	FP false positive
	negatives	FN false negative	TN true negative

Source: Author's production.

tive cases that were correctly identified (ZHU *et al.*, 2010). It indicates, how good a test is at detecting the positives. The recall is defined as,

$$R = \frac{TP}{(TP + FN)}. \quad (3.1.1)$$

Precision (P), also known as positive predictive value, is the proportion of the predicted positive cases that were correct. This measure is defined as,

$$P = \frac{TP}{(TP + FP)}. \quad (3.1.2)$$

Specificity (E) indicates how good the test is at avoiding false alarms, which means, the proportion of negative cases correctly classified. It is defined as,

$$E = \frac{TN}{(TN + FP)}. \quad (3.1.3)$$

Matthews correlation coefficient (MCC) has a range of -1 to $+1$, where -1 indicates a completely wrong binary classifier, 0 indicates no better than random prediction and $+1$ indicates a completely correct binary classifier, defined as,

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}. \quad (3.1.4)$$

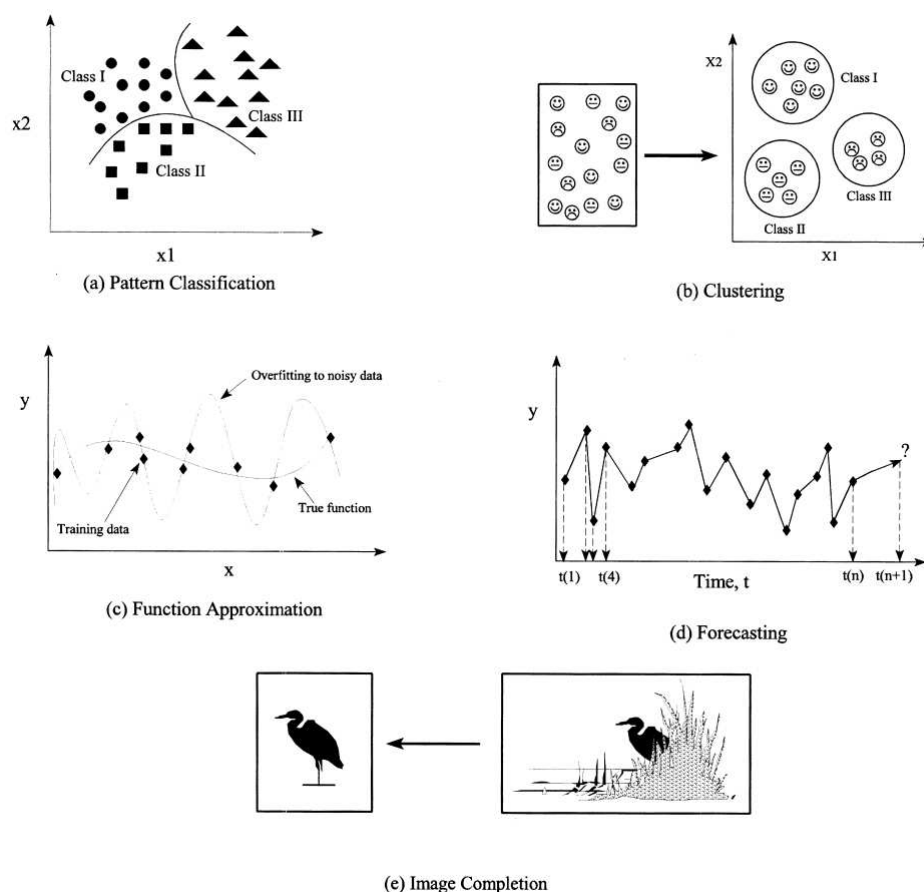
The accuracy of the classifier can be estimated using the trace of the matrix divided by the total sum of the entries. This measure is highly susceptible to an imbalance of the data set and can easily lead to a wrong conclusion about the performance of the system. In practice, sensitivity and specificity vary in opposite directions. This usually happens when a method is very sensitive to positives, tends to generate many false positives, and vice versa (CLESIO, 2014). Because of that, it is essential to eval-

uate the efficiency of the system, that is, the arithmetic mean of both sensitivity and specificity. One important information that the confusion matrix provides is where the misclassifications have occurred.

3.2 Artificial Neural Networks

Bishop (1995a) defined that the role of Artificial Neural Networks (ANNs) is to provide general non-linear parameterized mappings between a set of inputs and outputs variables. ANNs can be employed in several problems related to engineering and science. The potential areas of applicability are signal processing, pattern recognition, pattern association, clustering, function approximation, speech recognition, business forecasting, and others. Examples are shown in Fig. 3.2.

Figure 3.2 – ANNs potential areas of applicability.

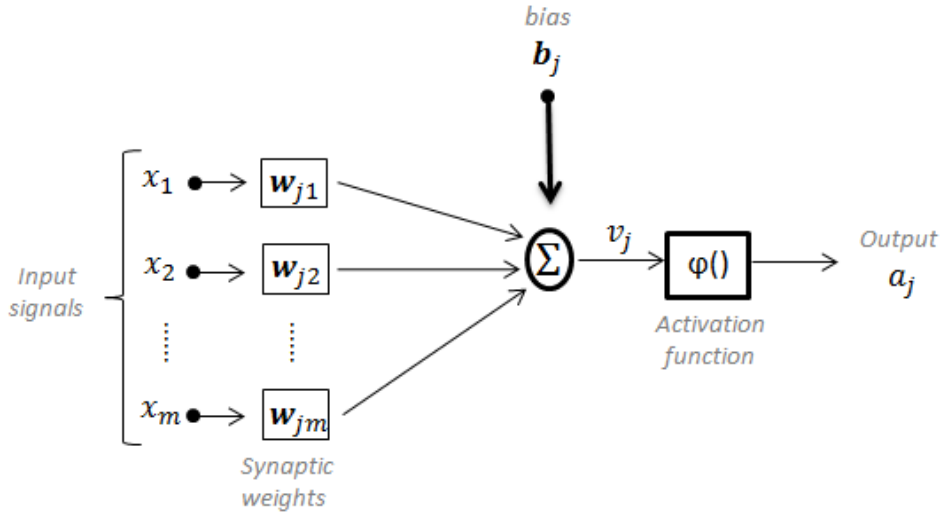


Source: Basheer and Hajmeer (2000).

The simplest neuron model that encompasses the main features of a neural network was proposed by McCulloch & Pitts. In this representation, each artificial single neuron, known as *Perceptron*, can be implemented as shown in the Fig. 3.3. The

input signals are represented by x_m , and w_{jm} are the connective weights responsible for weighting each input variable, allowing to quantify its relevance to the functionality of the respective neuron. The total input, *i.e.*, the sum of the input signals weighted by

Figure 3.3 – Example of a single neuron.



Source: Adapted from Haykin (1999).

the respective weights is described by

$$I_j = \sum_{k=1}^m x_k w_{jk}, \quad (3.2.1)$$

Usually, a neuron cannot produce an output until the total input exceeds a certain value, vitalizing the neuron. This value is known as bias b_j , or threshold, of the neuron. Depending on the signal of the bias, the relation between the induced local field (or activation potential) v_j of the neuron j , defined as

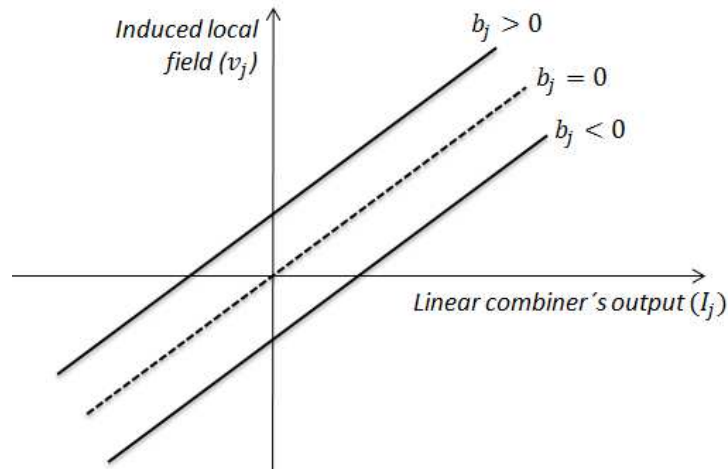
$$v_j = I_j + b_j, \quad (3.2.2)$$

and the linear combiner output j is modified, and the form is illustrated in Fig. 3.4. It can be noticed that depending on whether the bias is positive or negative, the line no longer passes through the origin. $\varphi()$ is the activation function, whose purpose is to limit the neuron output within a range of reasonable values and a_j is the output signal of the neuron, (HAYKIN, 2007), defined as

$$a_j = \varphi(v_j). \quad (3.2.3)$$

Typically, the activation functions used in the *Perceptron* are the step function or bipolar step, having only two possible values to be produced by its output (0 or 1) or (−1 or

Figure 3.4 – Relation between the induced local field and linear combiners output.



Source: Adapted from Haykin (1999).

1), respectively. Considering a *Perceptron* with two inputs (x_1 and x_2) and the step activation function, in mathematical terms, the output of the *Perceptron* will be given by:

$$a = \begin{cases} 1, & \text{if } \sum w_j x_j + b > 0 \\ 0, & \text{otherwise} \end{cases} \quad (3.2.4)$$

Since the inequalities represented by a first degree expression, the decision boundary for the two-input *Perceptron* will then be a line defined by

$$w_1 x_1 + w_2 x_2 + b = 0. \quad (3.2.5)$$

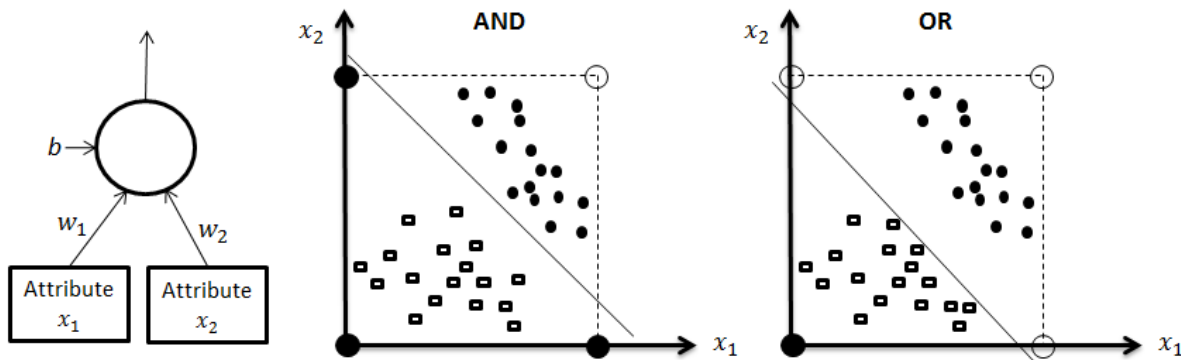
Therefore, the *Perceptron* behaves as a pattern classifier whose function is to divide classes that are linearly separable, *i.e.*, the *Perceptron* can only solve linear problems such as the AND or the OR-problems, as shown in Fig. 3.5, a line positioned at the separability boundary.

3.2.1 Network architectures

The ANN can be divided into three parts denominated layers:

- **Input layer:** as the name says, the input layer receives the input data, whose values are usually normalized. Is the only layer that is not constituted of neurons.
- **Hidden layer(s):** the main function is the feature extraction associated with the problem.
- **Output layer:** as the name says, it is responsible for the network output.

Figure 3.5 – Separability boundary illustration AND and OR-problems.



Source: Author's production.

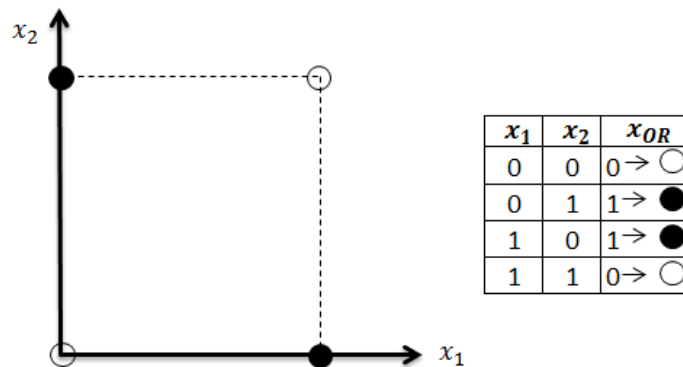
Haykin (1999) presented three different classes of network categories. The first one is Single-Layer Feedforward Networks, and the neurons are organized in the form of layers. Since it is composed of a single layer, there are no hidden layers, just input-output layers. The main types of single-layer forward networks are *Perceptron* and *Adaline*. The second class of network category is named Multilayer Feedforward Networks, as the previous one the neurons are organized into layers, but now hidden layers are present, either only one or multiple. The neural network can be fully connected, where every neuron in each layer is connected to every neuron in the adjacent forward layer, or partially connected where some of these connections are missing. Among the main types of multilayer forward networks are Multilayer Perceptron - MLP and Radial Basis Function - RBF. The last network category is called Recurrent Networks and distinguishes from a feedforward neural network because it has at least one feedback loop, which consists of a single layer (or multiple) of neurons with each neuron feeding its output signal back to the inputs. The examples of Recurrent Networks are Hopfield Network and MLP with feedback.

Network architecture or topology plays a critical role in neural net classification, and the optimal topology will depend on the problem. The number of hidden neurons determine the total number of weights and biases in the neural network, which is the number of degrees of freedom, and thus it is reasonable that we should not have more weights (and biases), than the total number of training samples (DUDA *et al.*, 1973).

The Single-Layer Feedforward Networks can only converge if the two classes involved with the problem are linearly separable. A classic case known as the XOR problem (OR-exclusive), involving a boolean logic, can be seen in Fig. 3.6. Looking at the XOR problem is possible to note that it would be impossible to position a single

line that would separate the two classes. This situation can be resolved using a MLP

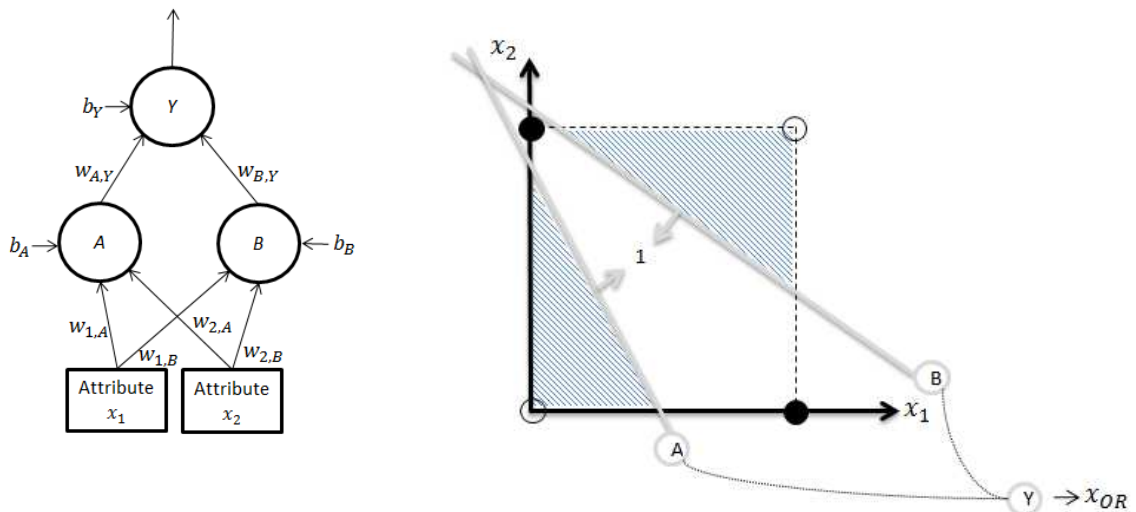
Figure 3.6 – XOR problem.



Source: Adapted from Silva *et al.* (2010)

network with one hidden layer, such as $[(x_1, x_2), (A, B), (Y: \text{output})]$. The separability result problem is shown in Fig. 3.7. Assuming logistic function as activation function, note that neuron A will have its exit equal to one only for those patterns that are above its straight line, while the neuron B will provide one for all those below (SILVA *et al.*, 2010).

Figure 3.7 – XOR problem with MLP Networks.



Source: Adapted from Silva *et al.* (2010)

Considering a MLP network with only one hidden layer, it can classify patterns that are arranged in a convex region, it can also be deduced that a network with two hidden layers is capable to classify patterns that are in any type of regions, including non-convex regions (SILVA *et al.*, 2010).

For classifying problems with two or more classes, it is necessary to add more

neurons in the output layer, because a MLP network with only one neuron in the output layer is able to classify only two classes. A MLP network with n output-neurons is capable to classify till 2^n classes. However, due to the complexity of some problems the adoption of this codification can make MLP network much more difficult to training because the classes would be represented by points that are spatially close to each other, demanding an increase in the number of neurons in the intermediate layers. One of the most used methods, known as one of the c-classes, is to associate the output of each neuron directly to the corresponding class (SILVA *et al.*, 2010).

3.2.2 Types of activation functions

Silva *et al.* (2010) classify the activation functions in two categories:

- **Partially differentiable functions:** such as the Heavyside or hard limiter function, the symmetric hard limiter function and the symmetrical ramp function.
- **Fully differentiable functions:** for example the logistic function, the hyperbolic tangent function, the Gaussian function and the Identity function.

The logistic function is defined as

$$\varphi(v_j) = \frac{1}{1 + \exp^{-\delta v_j}}, \quad (3.2.6)$$

where δ is a constant and represents the slope of the function. The logistic function assumes a continuous range of values from 0 to 1, as shown in Fig. 3.8. The logistic function derivative assumes small values in the interval between 0 and 0.25.

The hyperbolic tangent (*tanh*) activation function is defined as

$$\varphi(v_j) = \frac{\exp^{2v_j} - 1}{\exp^{2v_j} + 1}. \quad (3.2.7)$$

The *tanh* function assumes a continuous range between -1 and 1 , and the derivative of the function assumes values between 0 and 1, as shown in Fig. 3.9.

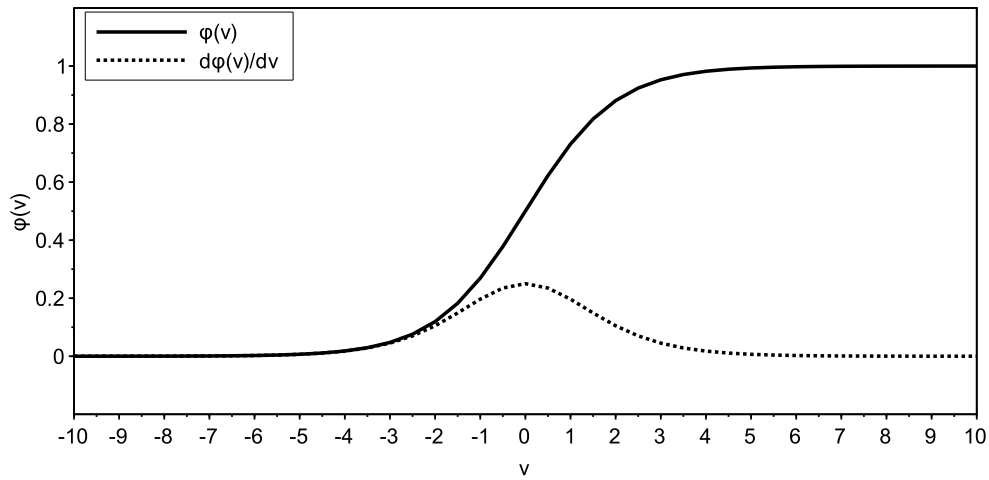
The logistic function is quite useful in probabilities and classification problems because it transforms an input value into an output range of $[0,1]$. However, in deep networks logistic and *tanh* functions suffer from the vanishing gradient problem, when the gradient becomes very small, and the learning process gets stuck (NIELSEN, 2018).

The Rectified Linear Unit (ReLU) is the most used activation function nowadays, (SHARMA, 2017), defined as

$$\varphi(v_j) = \max(0, v_j). \quad (3.2.8)$$

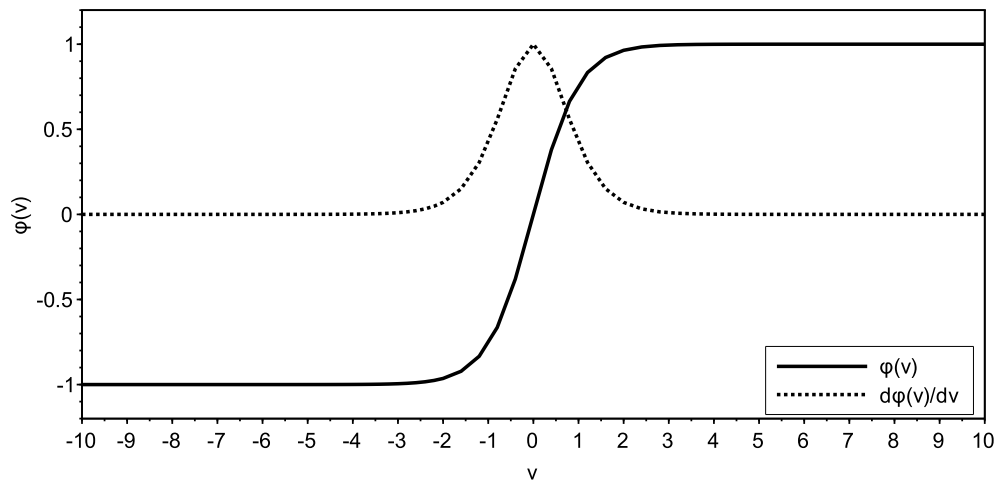
That is, when the input is smaller than zero, the function will output zero and the amount of change of the function is zero. When the input is greater or equal to zero, the output

Figure 3.8 – Logistic function.



Source: Author's production.

Figure 3.9 – Hyperbolic tangent function.



Source: Author's production.

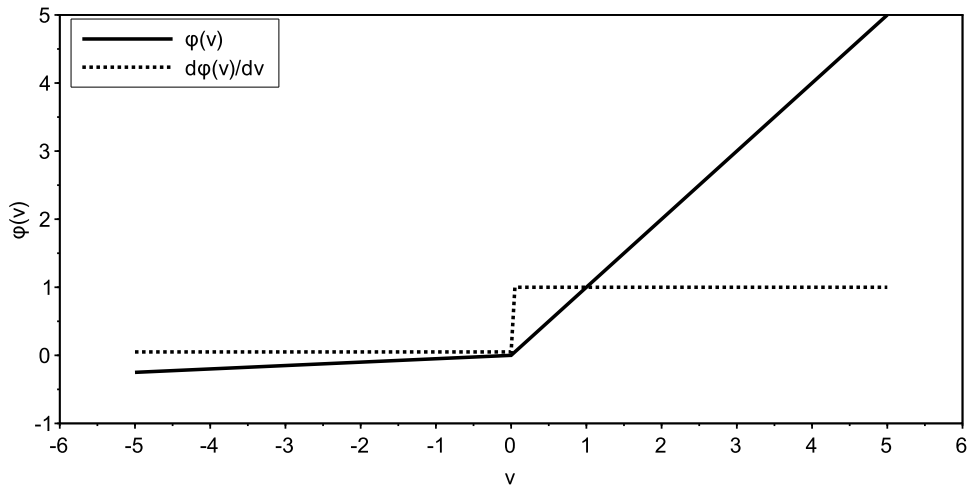
is directly the input, and thus the derivative is equal to one. However, a problem when using ReLU is the function output zero when the input to it is negative, blocking the learning process due to gradients dying off. To solve this problem another activation function called Leaky-ReLU is used. Instead of giving zero gradients the function give a very small gradient when the input is negative, giving the chance for the network to

continue its learning, defined as

$$\varphi(v_j) = \max(\kappa v_j, v_j), \quad (3.2.9)$$

where κ is a small number, such as 0.01 (KAPUR, 2016). Figure 3.10 presents a Leaky-ReLU function and its derivative. Nevertheless, ReLUs and leaky ReLUs are not nec-

Figure 3.10 – Leaky-ReLU function.



Source: Author's production.

essarily always optimal results, because the output of ReLU is not bounded between 0 and 1 or -1 and 1 , the activations (*i.e.*, neurons output, not the gradients) can, in fact, explode with extremely deep neural networks. During the learning process, the whole network becomes fragile and unstable in that, if the weights are updated in the wrong direction (even the slightest) the activations can blow up (KAPUR, 2016).

Another particularly useful activation function is known as Softmax and is used in an output layer (L), defined as

$$\varphi(v_j^L) = \frac{\exp(v_j^L)}{\sum_k \exp(v_k^L)}, \quad (3.2.10)$$

which is a set of positive numbers which sum up to 1, *i.e.*, a probability distribution making it useful for finding the most probable occurrence of output with respect to other outputs.

3.2.3 Learning process

One of the most significant highlights of the ANN is the ability to learn from its environment. After the network has learned the relation between inputs and outputs, it is able to generalize solutions (SILVA *et al.*, 2010). The learning process is the ability of the ANN to adapt free parameters through the process of stimulation by the environment in which the network is working, being the type of learning process defined in a particular way as the adjustments are made (BRAGA *et al.*, 2000).

To achieve good results in prediction problems, the ANN requires the division of the dataset into three subsets: training, validation, and testing. The training subset should be adequately large to cover the possible known variation in the problem domain and is used in the training phase to update the weights and bias of the network. The validation subset is used during the learning process to find the best hyperparameters, which include the number of hidden neurons and layers and their connectivity, the form of activations functions and parameters of the learning algorithm itself. The data contained in this subset should be distinct from those used in the training but lie within the training data boundaries. The testing subset should include different samples from those in the other subsets and is used to confirm the accuracy and efficiency of the ANN (BASHEER; HAJMEER, 2000). Looney (1996) recommended partition the database with 65% for the training set, 10% for the validation set and 25% for the verification testing set. Silva *et al.* (2010) divided into two subsets, 60-90% for the training (and validation) set a 10-40% for the testing set. Pereira and Bezerra (2007) used in their work 70%, 20% and 10% for the training, the validation, and the testing sets, respectively.

According to Bose and Liang (1996), the performance of learning does depend on the distribution of the training exemplars, since the way which the N samples are split from the entire exemplars influences the generalization of the future samples. In a classification problem, samples close to the decision surface known as boundary samples, carry more information for training the network. Indeed, the boundary samples tell us about the shape of the decision surface to be learned. This not only cuts down the training time since the number of samples is smaller, but also, in general, should yield a good generalization after the boundary samples are successfully learned. These desired properties are achieved because of the prior knowledge about the samples that are at the boundary. However, this prior knowledge may not be available.

Two learning techniques are known in the literature as follow.

3.2.3.1 Supervised learning

The strategy of this type of learning consists of providing known samples with known desired (target) output. The weights and biases are adjusted by comparing the

output produced by the network and the desired output, performed by the learning algorithm. The difference between the two values is used in the adjustment procedure. It can be implemented in three ways: off-line (or batch), where the adjustment process are made only after the presentation of the entire training set; stochastic, where the patterns are chosen randomly from the training set and the network weights are updated for each pattern presentation and online, where adjustments are made after the presentation of each training sample (DUDA *et al.*, 1973). Haykin (2007) related two main methods used in supervised learning algorithms: Error-correction learning and Memory-based learning.

- **Error-correction learning:** A type of supervised learning that aims at the minimization of the cost function defined in terms of the error signal. The error signal is defined as, (HAYKIN, 2007)

$$e_j(t) = z_j(t) - a_j(t), \quad (3.2.11)$$

where z_j is the target output, and a_j is the output signal from the neuron j . The cost function is defined as

$$E(t) = \frac{1}{2} e_j^2(t). \quad (3.2.12)$$

The minimization of the cost function leads to a learning rule usually assigned to as the Delta rule or Widrow-Hoff rule, named in honor of its creators, or also Gradient-descent method. The basic idea with the application of the Delta rule in order to adjust the values of the neuron design variables lies in the minimization of the error function. For the variable \mathbf{w} , the delta rule can be defined as

$$\Delta \mathbf{w}(t) = -\eta(t) \nabla E(\mathbf{w})(t), \quad (3.2.13)$$

where $\nabla E(\mathbf{w})$ is the gradient operator of the error with respect to the vector \mathbf{w} , defined by

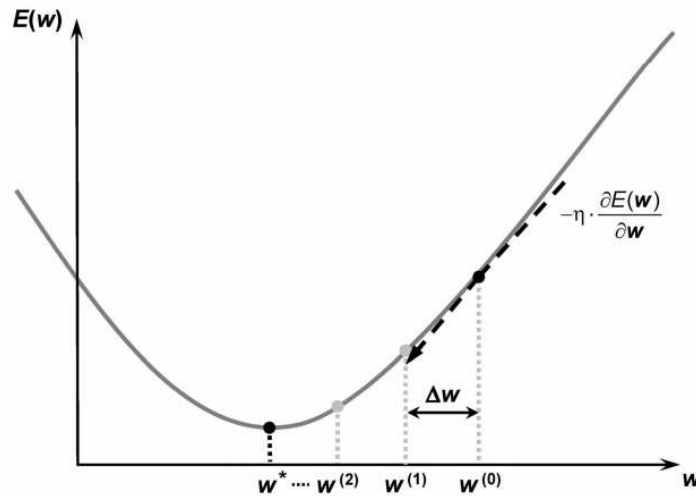
$$\nabla E(\mathbf{w})(t) = \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}(t), \quad (3.2.14)$$

η is called the learning rate. Having calculated the synaptic adjustment the updated value of the synaptic weight, \mathbf{w} , is driven by

$$\mathbf{w}(t) = \mathbf{w}(t-1) + \Delta \mathbf{w}(t). \quad (3.2.15)$$

Figure 3.11 shows the geometric interpretation of the Delta rule.

Figure 3.11 – Geometric interpretation of the Delta rule.



Source: Silva *et al.* (2010).

3.2.3.2 Unsupervised learning

As the name suggests, there is not a supervisor to oversee the learning process, *i.e.*, there are no marked samples of the function to be learned by the network. The network then establishes a harmony with the statistical regularities of the input data developing an ability to form internal patterns to encode the characteristics of the input data and thus create new classes automatically. The network must self-organize in relation to these characteristics existing between the samples identifying clusters that contain similarities. The synaptic weights are adjusted by the learning algorithm to reflect this internal representation of the network (SILVA *et al.*, 2010). The methods for unsupervised learning implementations, commonly used are Hebbian learning, Competitive learning, Boltzmann learning and Reinforcement learning (HAYKIN, 1999).

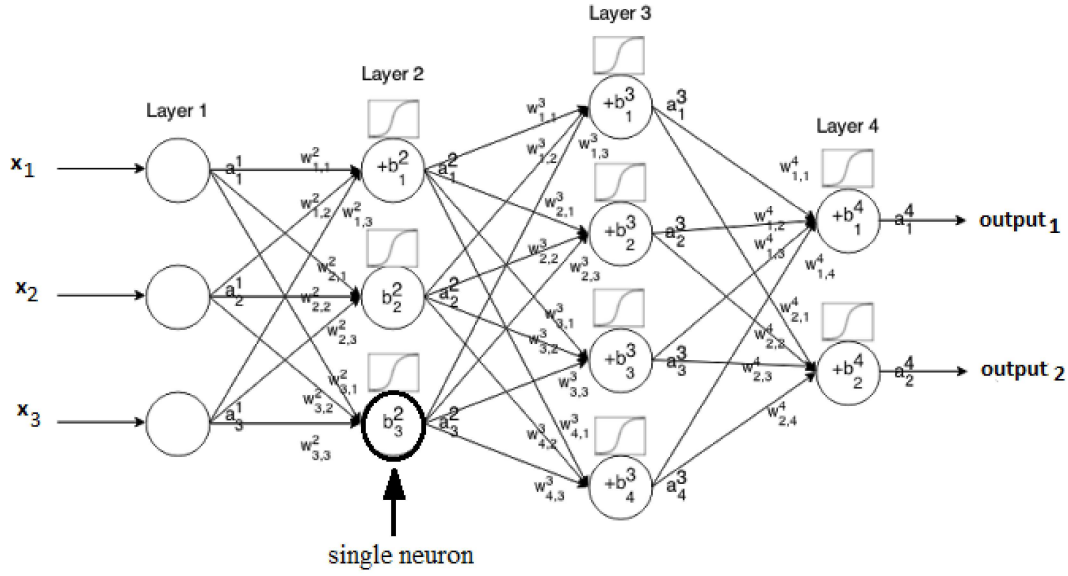
3.2.4 Multilayer Perceptrons networks

The Multilayer Perceptrons (MLP) networks are used in most applications as pattern recognition, process identification and control, time series forecasting and are also considered one of the most versatile architectures, (SILVA *et al.*, 2010).

3.2.4.1 Network learning

The basic approach in learning processes is to provide a training pattern to the input layer, propagate the signal through the layers and calculate the output at the output layer, as can be seen in Fig. 3.12. The outputs are compared to the target values and the difference corresponds to an error. This error is a function of the weights and

Figure 3.12 – Multilayer Perceptron network.



Source: Adapted from (PREETHAM, 2016).

bias and is minimized when the network outputs match the target values. The weights and bias (design variables) are adjusted to reduce this error. The error function, also known as cost function (or loss function), is based on the supervised learning seen in the previous section based on Widrow-Hoff rule, being the sum over outputs units of the squared difference between the target value z and the computed output a , (DUDA *et al.*, 1973)

$$E(\mathbf{w}, \mathbf{b}) = \frac{1}{2} \sum_{k=1}^n (z_k - a_k)^2 = \frac{1}{2} \|\mathbf{z} - \mathbf{a}\|^2, \quad (3.2.16)$$

where \mathbf{w} and \mathbf{b} represent all the weights and biases respectively and n the length of the output vector. This type of error function is known as the quadratic function. Another type of the error function known in the literature is the L_p distance where the p -norm (for p even), (CARLINI; WAGNER, 2017), is defined as

$$E(\mathbf{w}, \mathbf{b}) = \left(\sum_{k=1}^n (z_k - a_k)^p \right)^{1/p} = \|\mathbf{z} - \mathbf{a}\|_p. \quad (3.2.17)$$

when $p=2$, the equation is known as L_2 distance measured the Euclidian distance between \mathbf{z} and \mathbf{a} . An alternative approach for error function is the Cross-entropy error function, (BISHOP, 2006), defined as

$$E(\mathbf{w}, \mathbf{b}) = - \sum_{k=1}^n (z_k \ln a_k + (1 - z_k) \ln(1 - a_k)). \quad (3.2.18)$$

Simard *et al.* (2003) found that the cross-entropy function trained faster than the quadratic function for classification problems. The use of the quadratic function is that sometimes the network learning process is very slow, which means, the partial derivatives $\frac{\partial E}{\partial w} \frac{\partial E}{\partial b}$, are small. Considering the quadratic cost function presented in Eq. (3.2.16), where a is the neurons output when the training input $x=1$ is used, and $z=0$ is the desired output and recall equations (3.2.2) and (3.2.3). Using the chain rule

$$\frac{\partial E(z, a)}{\partial w} = \frac{\partial E(z, a)}{\partial a} \frac{\partial a}{\partial v} \frac{\partial v}{\partial w}, \quad (3.2.19)$$

differentiating with respect to the weight and bias, and replacing $x=1$ and $a=0$, the result is

$$\begin{aligned} \frac{\partial E}{\partial w} &= (z - a)\varphi'(v)x = z\varphi'(v), \\ \frac{\partial E}{\partial b} &= (z - a)\varphi'(v) = z\varphi'(v). \end{aligned} \quad (3.2.20)$$

Thus, the magnitude of the sensitivity, depends on of $\varphi'(v)$, which is close to zero. Now do the same procedure but instead of using quadratic function replacing it for cross-entropy function. Substituting Eq. (3.2.3) into Eq. (3.2.18) and applying the chain rule twice, obtaining

$$\frac{\partial E(z, a)}{\partial w} = - \sum_x \left(\frac{z}{\varphi(v)} - \frac{(1-z)}{1-\varphi(v)} \right) \frac{\partial \varphi}{\partial w}, \quad (3.2.21)$$

$$\frac{\partial E(z, a)}{\partial w} = - \sum_x \left(\frac{z}{\varphi(v)} - \frac{(1-z)}{1-\varphi(v)} \right) \varphi'(v)x. \quad (3.2.22)$$

Placing everything over a common denominator and simplifying

$$\frac{\partial E(z, a)}{\partial w} = \sum_x \frac{\varphi'(v)x}{\varphi(v)(1-\varphi(v))} (\varphi(v) - z). \quad (3.2.23)$$

Using the logistic function - Eq. (3.2.6) - and some algebra

$$\frac{\partial E(z, a)}{\partial w} = \sum_x \frac{\varphi(v)(1-\varphi(v))x}{\varphi(v)(1-\varphi(v))} (\varphi(v) - z). \quad (3.2.24)$$

After some simplification, the resultant equation is presented as

$$\frac{\partial E(z, a)}{\partial w} = \sum_x x(\varphi(v) - z). \quad (3.2.25)$$

So Eq. (3.2.25) describes the rate at which the weight learns is controlled by $\varphi(v) - z$, that means the error in the output. The larger the error, the faster the neuron will learn, and it avoids the learning slowdown caused by the derivative term $\varphi'(v)$ as shown when

using quadratic function (NIELSEN, 2018).

Assuming a training set composed of s samples, the measurement of the overall performance of the learning process is done by

$$E_{total} = \frac{1}{s} \sum_{k=1}^s E(k), \quad (3.2.26)$$

The learning rule is based on gradient descent. The principal objective of the iterative optimization process based in the gradient descent is to reach a minimum point for the error function E , finding the desirable search direction d to reach this point (ARORA, 2004). Assorted optimization methods can be employed such as Steepest Descent, Newton, Conjugate Gradient, Levenberg-Marquardt, etc. The search direction can be determined, for each iterations t , (ARORA, 2004), as

$$d^t = -\nabla E^t, \quad (3.2.27)$$

where

$$\nabla E = \left\{ \begin{array}{c} \frac{\partial E}{\partial \mathbf{w}} \\ \frac{\partial E}{\partial \mathbf{b}} \end{array} \right\}. \quad (3.2.28)$$

The gradients can be evaluated using the backpropagation algorithm. It is one of the most popular technique for training MLP based on gradient descent in error (DUDA *et al.*, 1973). An alternative approach is the use of finite differences combined with weight perturbation, that estimates the gradients by independently perturbing synaptic weights and observing the change in the cost function (JABRI; FLOWER, 1992). Another approach evaluates the gradient using automatic differentiation, which works by systematically applying the chain rule of differential calculus at the elementary operator level (KEDEM, 1980).

The weights and biases are initialized with random values, and then adjusted in a direction that will reduce the error, opposite of the gradient, (HAYKIN, 2007), as

$$\Delta \mathbf{w} = -\eta \frac{\partial E}{\partial \mathbf{w}}, \quad (3.2.29)$$

and

$$\Delta \mathbf{b} = -\eta \frac{\partial E}{\partial \mathbf{b}}, \quad (3.2.30)$$

where η is the learning rate, that expresses how fast the network training process is being conducted towards its convergence. Substituting Eq. (3.2.27) into Eq. (3.2.29) and Eq. (3.2.30), the adjustment of the synaptic weight and biases, for t iterations, are done by

$$\Delta \mathbf{w}^t = \eta^t d^t, \quad (3.2.31)$$

$$\Delta \mathbf{b}^t = \eta^t \mathbf{d}^t, \quad (3.2.32)$$

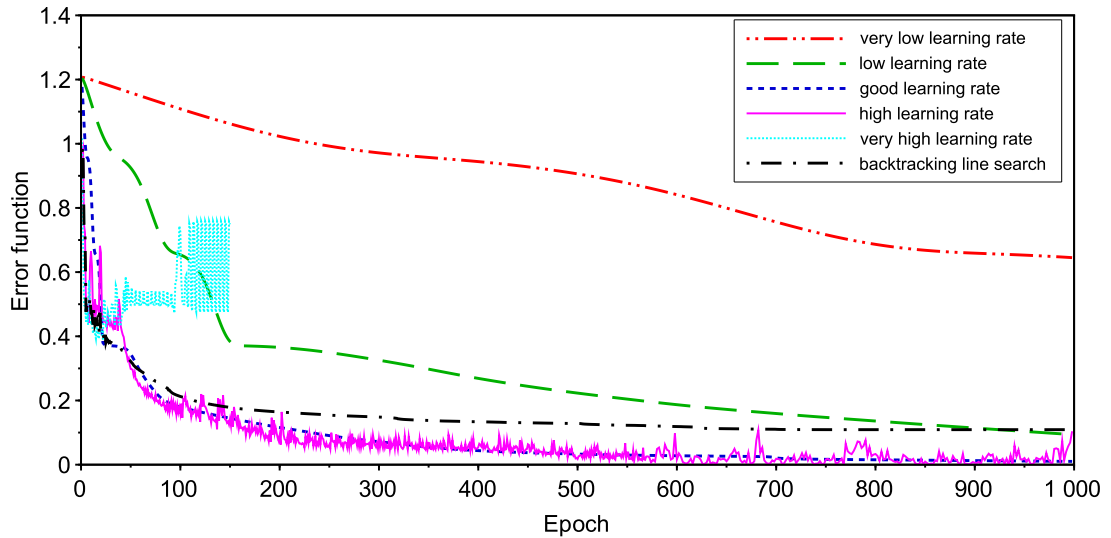
such that

$$\mathbf{w}^t = \mathbf{w}^{t-1} + \Delta \mathbf{w}^t, \quad (3.2.33)$$

$$\mathbf{b}^t = \mathbf{b}^{t-1} + \Delta \mathbf{b}^t. \quad (3.2.34)$$

The learning rate can also be seen as a step size to reach the minimum of the error (cost) function along the search direction (SILVA *et al.*, 2010). The behavior of the cost function during iterations with different values of the learning rate can be seen in Fig. 3.13. The smaller we make it, the smaller the changes to the weights (and biases) in the network will be from one iteration to the next, and the smoother will be the trajectory in weight space. If η is too big to accelerate the learning process, the changes in the weights (and bias) may assume a form that the network becomes unstable.

Figure 3.13 – Behavior of the cost function with different values of the learning rate.

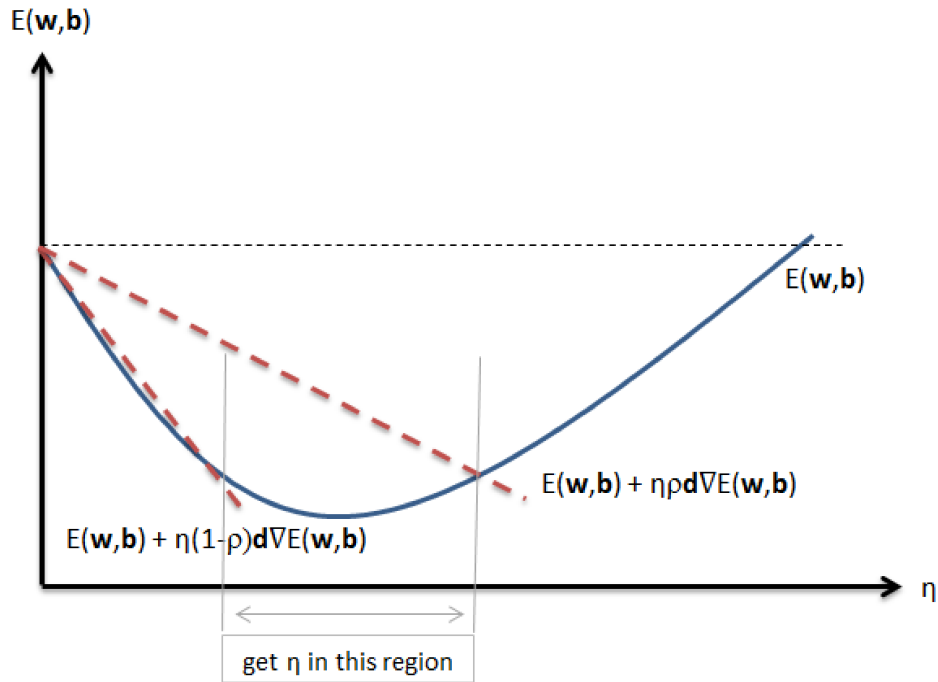


Source: Author's production.

The learning rate can be maintained constant (HAYKIN, 1999) or variable and calculated using some line search method. One approach is called Backtracking (or Armijo-Goldstein) line search. The algorithm starts with a fixed value of step size and repeatedly shrinks it by a factor τ , called the relaxation step until the Armijo-Goldstein condition is fulfilled. Figure 3.14 shows that condition, the region between the two lines is the region where η can be taken from. The structure of the algorithm is shown in Tab. 3.1, where ρ is the relaxation of the initial slope (ARORA, 2004).

A simple method of increasing the learning rate avoiding instability is to modify

Figure 3.14 – Armijo-Goldstein condition.



Source: Author's production.

Table 3.1 – Algorithm Backtracking line search

Algorithm Backtracking line search

- 1 being initialize $\eta, \tau, \rho, t \leftarrow 0$
- 2 do $t \leftarrow t + 1$ (increment epoch)
- 3 until $E(\mathbf{w}, \mathbf{b}) - E((\mathbf{w}, \mathbf{b}) + \eta^t \mathbf{d}) < -\eta^t \rho \mathbf{d}^T \nabla E(\mathbf{w}, \mathbf{b})$
- 4 $\eta^t = \tau \eta^{t-1}$
- 5 return η^t
- 6 end

Source: Author's production.

the delta rule - Eq. (3.2.31) and Eq. (3.2.32), including the term known as momentum γ , as follow

$$\Delta \mathbf{w}^t = \gamma \Delta \mathbf{w}^{t-1} + \eta^t \mathbf{d}^t, \quad (3.2.35)$$

$$\Delta \mathbf{b}^t = \gamma \Delta \mathbf{b}^{t-1} + \eta^t \mathbf{d}^t. \quad (3.2.36)$$

The momentum term value is between 0 and 0.9 (SILVA *et al.*, 2010). When the current solution is far from the final (minimum) solution, the variation in the opposite direction of the gradient between two successive iterations will also be large. This implies that the difference between the matrices of weights and bias of these iterations will be

significantly large, making a larger increment step for the weights and bias toward the minimum of the error function. The execution of such a task is made by the momentum term. When the actual solution is very close to the final solution, the contribution of the momentum term to the convergence process is very small, and the adjustments in the matrices of weights and bias are driven only by the learning term (SILVA *et al.*, 2010). One problem with simple gradient descent with momentum term is that it contains two parameters, η and γ , whose values must be selected by trial and error (BISHOP, 1995a). An approach to setting the optimal learning rate parameter was introduced by LeCun *et al.* (1993).

Due to the fact the error surface produced from MLP is non-linear, there is a possibility that the learning process directs the weights (and biases) matrix to a local minimum, which may not correspond to the most appropriate values for the good network generalization. The convergence for a given minimum local is then conditioned to the spatial position, which the weight (and bias) matrices are started. Aiming to avoid the convergence of the MLP to inappropriate local minimum, one of the procedures adopted is to execute the MLP-candidates many times during the training process, with distinct initial weights (and bias) matrices (SILVA *et al.*, 2010). A good choice for the initial weights (and biases) values improves the success of the learning process. When large initial weight values are assigned, it is highly likely that the neuron will be saturated, consequently, the local gradients assume small values and the learning process slows down. On the other hand, small weight corresponds to a very flat region of the error surface (LECUN, 1993). Thus, it is desirable an uniform distribution, from which the weights are selected to have a mean of zero and a variance equal to the reciprocal of the number of connections of a neuron (HAYKIN, 2007).

According to Haykin (2007), each input variable should be preprocessed so that its mean value is close to zero or it is small compared to its standard deviation. In order to accelerate the learning process, the normalization of the inputs should also include two other measures:

- The input variables contained in the training set should be uncorrelated;
- The decorrelated input variables should be scaled so that their covariances are similar, ensuring that the different synaptic weights in the network learn at approximately the same speed.

3.2.5 Bias and variance tradeoff

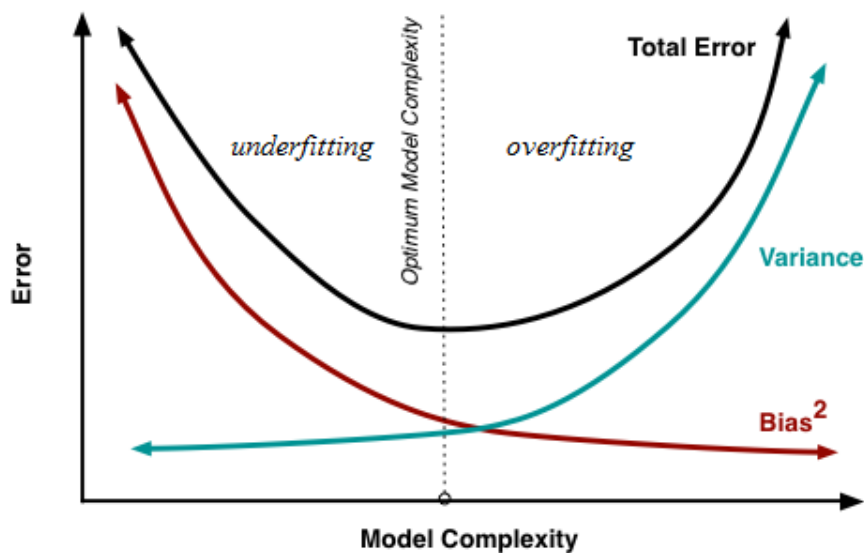
First, it is necessary to introduce two concepts, according to Bishop (1995a).

- **Bias:** is an error from the erroneous assumption in the learning algorithm. It is the difference between the expected prediction and the target value.

- **Variance:** is an error from the variability of a model prediction for a given data point, that is, is how the predictions for a given point vary between the different realization of the model.

A model, which is too simple (or inflexible) will have a large Bias while one which too much flexibility in relation to the particular data set will have a large variance. The best generalization is obtained when there is the best compromise between the conflicting requirements of small Bias and small variance, and then the Bias-variance tradeoff is in which the generalization error is decomposed into the sum of the Bias squared plus the variance. Figure 3.15 shows the Bias-variance tradeoff (BISHOP, 1995a).

Figure 3.15 – Bias-variance tradeoff.

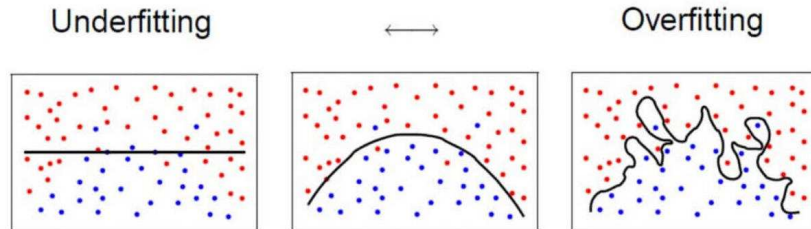


Source: Fortmann-Roe (2012).

Increasing the quantity of the neurons, and also the hidden layers do not ensure the generalization of the MLP. Such actions tend to lead the MLP to the memorization (called overfitting), where the network ends up to memorize the response against the presented input pattern. The cost function during the learning phase tends to be small, otherwise, during the generalization phase, when the testing set is used, the cost function tends to be larger. When overfitting, the algorithm has low Bias and cannot generalize because it has more complexity than necessary, considering features that should not (like noise). However, a network with a very small number of neurons may be insufficient for the feature extraction that allows the network to implement the hypotheses regarding the behavior of the process. This situation is called underfitting. In this case, the cost function during learning and generalization phases will be high. When we are facing an underfitting case, the problem has a high Bias and can cause it to miss the relevant relations between features and targets output (DUDA *et al.*, 1973).

Figure 3.16 shows an example of underfitting, good fitting and overfitting representations.

Figure 3.16 – Examples of underfitting, optimal fitting and overfitting cases.



Source: Robertshaw (2015).

Some solutions can be implemented to overcome the high variance and the overfitting problem. This includes selecting a smaller set of variables, implement regularizations models, use cross-validation and early-stopping techniques. To minimize the problem of high Bias and consequently underfitting it is recommended the use of more training samples (BISHOP, 1995a). According to Duda *et al.* (1973), in order to achieve the desired low generalization error it is more important to have low variance than to have low Bias. Bias and variance can be lowered with large training size.

3.2.5.1 Cross-validation

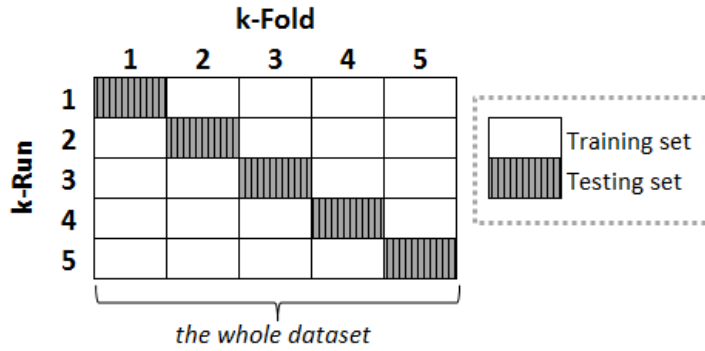
One of the most used statistical techniques to select the best candidate topologies is the cross-validation (SILVA *et al.*, 2010). The k -fold cross-validation technique is a common technique to evaluate the performance of the classifier. The training set is randomly split into k sets of approximately equal size s/k , where s is the total number of samples in the dataset, as seen in Fig. 3.17. The classifier is trained k times, each time with a different set held out as a validation/testing set. The estimated performance is the mean of these k errors (DUDA *et al.*, 1973).

3.2.5.2 Training with noise

According to Bishop (1995b), one way to control the trade-off Bias against variance involves the addition of artificial noise to the input data during training. Adding small quantities of artificial noise can increase generalization power as long as the amount of noise is maintained small to have little effect on the desired output. One technique consists of the addition of noise with Gaussian distribution to the uncorrelated input patterns (SILVA; ADEODATO, 2011). The expression used to generate the noise values is defined as

$$c = \psi G(0, 1), \quad (3.2.37)$$

Figure 3.17 – Example of 5-fold Cross-validation.



Source: Author's production.

where c is the noise added to every input variable, ψ is the amount of noise and G is the Gaussian distribution with zero mean and unitary variance.

3.2.5.3 Regularization techniques

The regularization techniques encourage smoother network mapping by adding a penalty Ω to the error function defined as

$$E = E + \xi\Omega, \quad (3.2.38)$$

where E is the error function defined previously, ξ is a control parameter that extent the penalty term and influences the solution, and Ω a function of the weights (BISHOP, 2006).

A network with large weights can be a sign of an unstable network where small changes in the input can lead to large changes in the output. The effect of regularization is to make the network prefers to learn small weights, all other things being equal. Large weights will only be allowed if they considerably improve the first part of the cost function. If the penalty values is too large, the model will underestimate the weights and underfit the problem, on the other hand, small penalty parameter value can lead to an overfitting problem (BROWNLEE, 2018).

One of the most known regularization technique is called weight decay (or L_2 regularization) and consists of the sum of the squares of the all weights in the network

$$\Omega = \frac{1}{2} \sum_i w_i^2. \quad (3.2.39)$$

Another technique is called L_1 regularization, defined as

$$\Omega = \sum_i |w_i|. \quad (3.2.40)$$

In both expressions the effect of regularization is to shrink the weights. In L_1 regularization, the weights shrink by a constant amount toward 0 (because of the derivative of $|w|$). In L_2 regularization, the weights shrink by an amount, proportional to w (because the derivative of w^2). When a specific weight has an absolute large magnitude, L_2 regularization shrinks the weight much more than L_1 regularization does. When $|w|$ is small, L_2 regularization shrinks the weight much less than L_1 regularization (NIELSEN, 2018).

3.2.5.4 Early stopping

A different strategy for controlling the effective complexity of a network is the procedure of early stopping. As defined previously, the training of neural networks is the iterative reduction of the cost function, in which this value decreases as a function of the number of iterations. However, the error measured with respect to another dataset, such as the validation dataset, generally exhibits a decrease at first, trailed by an increase as the networks start to over-fit. Such problems can be solved stopping the training process at the point of the smallest error in the validation dataset (BISHOP, 2006).

3.3 Principal Component Analysis

Pre-processing is often one of the most significant steps in the development of reliable ANN, and the choice of pre-processing steps can often have an important consequence on generalization. One of the most important models of pre-processing involves a reduction in the dimensionality of the input data. At the simplest level, this could involve rejecting a subset of the original inputs. The main reason for dimensionality reduction is that it can support to mitigate the worst effects of the curse of dimensionality, which was introduced by Richard Bellman (BISHOP, 1995a). The phenomenon can be explained basically as much as the number of training elements required for a classifier to have good performance is an exponential function of the dimension space of the characteristics (HAYKIN, 1999). Furthermore, Zang and Imregun (2001b) described the relationship between the number of input variables m , output variable n , training samples s , and hidden neurons h_n ,

$$s = 1 + \frac{h_n(m + n + 1)}{n}. \quad (3.3.1)$$

For example, take $n=2$ (undamaged and damaged classes), $m=2048$ input variables and assuming that the number of hidden neurons is half of the input variables. In this case, the required number of training samples becomes 1.050.113.

Principal Component Analysis (PCA) is a statistical technique that uses an orthogonal projection, transforming a set of correlated variables into a set of uncorrelated variables, called principal components (PCs). The aim of the PCA is a dimensional reduction and elimination of the noise present in the original data (DACKERMANN, 2009). Let the matrix $\mathbf{H}(\omega)_{s \times m}$ be formed with all frequency response functions (FRFs) data, where s is the number of training samples and m the frequency points. The first step for PCA is standardizing the data, which depends on the measurement scales of the original features. The mean response vector \bar{h} is defined as

$$\bar{h}_j = \frac{\sum_{i=0}^s h_{ij}}{s}, \quad (3.3.2)$$

where i are lines, and j are columns. Then, the determination of standard deviation S^2 can be defined as

$$S_j^2 = \frac{\sum_{i=0}^s (h_{ij} - \bar{h}_j)^2}{s}. \quad (3.3.3)$$

A single element of the FRF matrix can then be replaced by

$$\tilde{h}_{ij}(\omega) = \frac{h_{ij} - \bar{h}_j}{S_j \sqrt{s}}, \quad (3.3.4)$$

and a response variation matrix $\tilde{\mathbf{H}}$ can be found. The correlation matrix \mathbf{C} can be defined as

$$\mathbf{C}_{s \times m} = \tilde{\mathbf{H}}(\omega)_{m \times s}^T \tilde{\mathbf{H}}(\omega)_{s \times m}. \quad (3.3.5)$$

Typically, the correlation matrix is used when variables are on different scales and the co-variance matrix is used when the variables scales are similar. Finally, the PCs are obtained using

$$\mathbf{C} \phi_l = \lambda_l \phi_l, \quad (3.3.6)$$

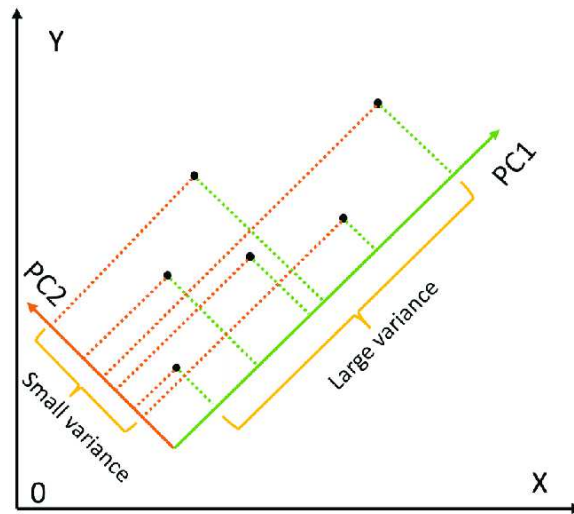
where l is the principal component's index with λ_l being the eigenvalue (PC) and ϕ_l the corresponding eigenvector. The first PCs are the highest eigenvalues corresponding to the direction and amount of the maximum variability in the raw data (ZANG; IMREGUN, 2001a). A schematic illustration of how PCA takes the high variance is shown in Fig. 3.18.

The projection matrix \mathbf{A} can be defined as

$$\mathbf{A}_{s \times p} = \tilde{\mathbf{H}}(\omega)_{s \times m} \phi_{m \times p}, \quad (3.3.7)$$

where p is the number of the PCs retained to achieve the maximum variance of the

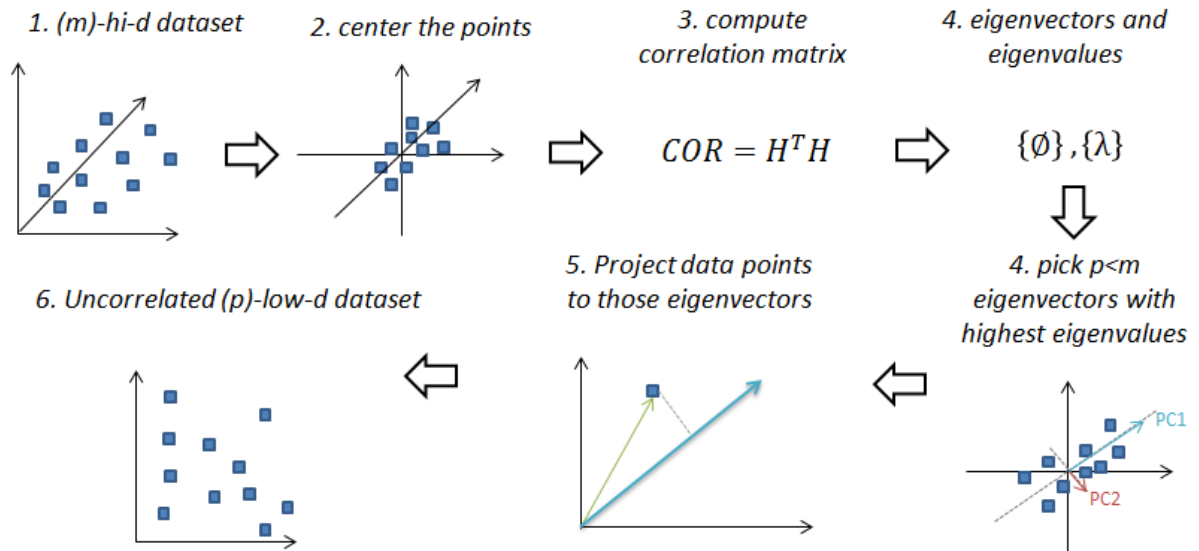
Figure 3.18 – PCA-variance illustration.



Source: Zhang and Castelló (2017).

problem. This linear dimensionality reduction procedure is also called the Karhunen-Loève transformation, summarized in Fig. 3.19.

Figure 3.19 – PCA illustration.



Source: Author's production.

The response variation matrix can be reconstructed from p components by

$$\tilde{H}_R = A_{s \times p} \phi_{p \times s}^T. \quad (3.3.8)$$

The reconstruction FRF can be calculated as

$$\mathbf{H}_R = \mathbf{S}\sqrt{s}\tilde{\mathbf{H}}_R + \bar{\mathbf{h}}. \quad (3.3.9)$$

Therefore the projection matrix \mathbf{A} is used as input to the Neural Networks.

3.4 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population-based stochastic optimization technique, based on the behavior of a colony or swarm of insects, a school of fish or a flock of birds. The word particle denotes, for example, a bird in a flock. Each particle behaves in a distributed manner using its own intelligence and the collective intelligence of the flock (swarm). If one particle discovers a good track to food, the rest of the flock will also be able to follow a good track immediately even if their location is far away in the flock (RAO, 2009). Each particle is assumed to have two characteristics: a position (X) and a velocity (V), both defined as

$$V(t) = W\mathbf{rand}V(t-1) + C_1\mathbf{rand}(pBest - X(t-1)) + C_2\mathbf{rand}(gBest - X(t-1)), \quad (3.4.1)$$

$$X(t) = X(t-1) + V(t). \quad (3.4.2)$$

By randomly initializing the algorithm with candidate solutions, the PSO tries to lead to a (more suitable) local optimum which is the best objective function of the individual ($pBest$) and of the group ($gBest$) (ARORA, 2004). W , C_1 , and C_2 are PSO parameters known as inertia weight, self confidence factor and swarm confidence factor, respectively (HASSAN *et al.*, 2004). The main lines of the code are shown on Tab. 3.2.

Table 3.2 – PSO Algorithm

*PSO Algorithm*Maximize $f(\mathbf{X})$ with $\mathbf{X}^{low} \leq \mathbf{X} \leq \mathbf{X}^{upper}$

```

1 being initialize the size of particle and others PSO parameters,  $t \leftarrow 1$ 
2   initialize the positions and velocities for all particles
3   calculate the fitness value of each particle
4   if ( $currentSolution^i < pBest^i$ ) then  $pBest^i = currentSolution^i$ 
5   update particles position and velocity (equations 3.4.1 and 3.4.2)
6   end if
7    $gBest^i = argmin(pBest^i, gBest^{i-1})$ 
8    $t \leftarrow t + 1$ 
9   until criterion stop
10  return gBest
11 end

```

Source: Author's production.

3.5 Automatic Differentiation and Dual Numbers

Automatic differentiation (AD) performs a non-standard interpretation of a given computer program by replacing the domain of the variable to incorporate derivative values. Redefining the meaning of the operators to propagate derivatives per the chain rule of differential calculus according to Fike and Alonso (2011). According to Manzyuk *et al.* (2012) the meaning of Forward AD is to attach perturbations to each input number, and propagate these through the computation.

Consider an ordered pair of real numbers (u, \tilde{u}) . A dual number can be written as,

$$\tilde{u} = u + \epsilon \tilde{u}, \quad (3.5.1)$$

with $\epsilon^2 = 0$ (EASTHAM, 1961) and the coefficient of ϵ is called a perturbation. The fact that ϵ^2 , in the Taylor expansion, vanishes facilitated the differentiation operation for functions of dual numbers (LEUCK; NAGEL, 1999)

$$\begin{aligned}
 f(\tilde{u}) &= f(u + \epsilon \tilde{u}) \\
 &= f(u) + \frac{1}{1!} \frac{\partial f}{\partial u}(u) \epsilon \tilde{u} + \frac{1}{2!} \frac{\partial^2 f}{\partial u^2}(u) (\epsilon \tilde{u})^2 + \dots \\
 &= f(u) + \epsilon \left(\tilde{u} \frac{\partial f}{\partial u}(u) \right).
 \end{aligned} \quad (3.5.2)$$

Arithmetic over dual numbers is again defined by simple rules derived algebraically, as

$$(u_1 + \epsilon \tilde{u}_1) + (u_2 + \epsilon \tilde{u}_2) = (u_1 + u_2) + (\tilde{u}_1 + \tilde{u}_2) \epsilon, \quad (3.5.3)$$

$$(u_1 + \epsilon \tilde{u}_1)(u_2 + \epsilon \tilde{u}_2) = u_1 u_2 + (u_1 \tilde{u}_2 + \tilde{u}_1 u_2) \epsilon + \tilde{u}_1 \tilde{u}_2 \epsilon^2 = u_1 u_2 + (u_1 \tilde{u}_2 + \tilde{u}_1 u_2) \epsilon. \quad (3.5.4)$$

The value of a function of an independent dual-number variable is also a dual-number and is given by the first derivative of the function with respect to the real part of the independent variable. The chain rule works as expected

$$\begin{aligned} (f \circ g)(\tilde{u}) &= (f \circ g)(u) + \epsilon \tilde{u} \frac{\partial(f \circ g)}{\partial u}(u) \\ &= f(g(u)) + \epsilon \tilde{u} \frac{\partial f}{\partial g} \left(g(u) \right) \frac{\partial g}{\partial u}(u) \\ &= f(g(u) + \epsilon \tilde{u} \frac{\partial g}{\partial u}(u)) \\ &= f(g(\tilde{u})). \end{aligned} \quad (3.5.5)$$

The method for computing derivatives of functions expressed as computer programs can be described in a few steps, (MANZYUK *et al.*, 2012):

- One organizes for the programming language to support dual numbers and the arithmetic on dual numbers;
- To compute the derivative f' at a point u :
 1. re-write as $u + 1\epsilon$,
 2. applied f to $u + 1\epsilon$ to obtain a result $f(u) + f'(u)\epsilon$, and
 3. extract $f'(u)$ from the result $f(u) + f'(u)\epsilon$.

Chapter 4

Materials and Methods

In this chapter, the development of the work is presented, starting with the description of the manufacturing process of glass-fiber/epoxy and the vibration-based tests, followed by the details of the dynamic tests using carbon-fiber/epoxy plates, and the whole presentation of the methodology using Principal Component Analysis (PCA) and the Artificial Neural Networks (ANNs) algorithms.

4.1 Damage Detection in Glass Fiber/Epoxy Beams

A set of (healthy and damaged) glass fiber/epoxy beams is built in order to study the applicability of the proposed methodology. The modified-VARTM (Vacuum Assisted Resin Transfer Molding) technique is used, and different delamination sizes are introduced during the lamination. Then, dynamic tests are performed to obtain the Frequency Response Function (FRFs) data followed by the application in PCA and ANN.

4.1.1 Manufacturing process

The epoxy resin used is the AR-260, combined with a hardener epoxy AH-150 in the proportion 3:1, part both from Barracuda Advanced Composites. The gel time is 50 minutes at 25 °C. The glass fibers fabric (WRU 221 RT) was manufactured by Texiglass. This woven type is a kind of satin weave known as Shallow Turkish, and its characteristics are shown in Tab. 4.1.

The modified-VARTM layout scheme is shown in Fig. 4.1. Instead of using a resin injection pump, the epoxy resin is directly placed in the mold. To reduce the moisture absorption from the epoxy resin, the fibers and the resin are stored inside the kiln around 100 °C before starting the manufacturing process. The plates (40 × 40 cm) have twelve layers with stacking sequence $[0]_{12}$. It is important to note that fabric

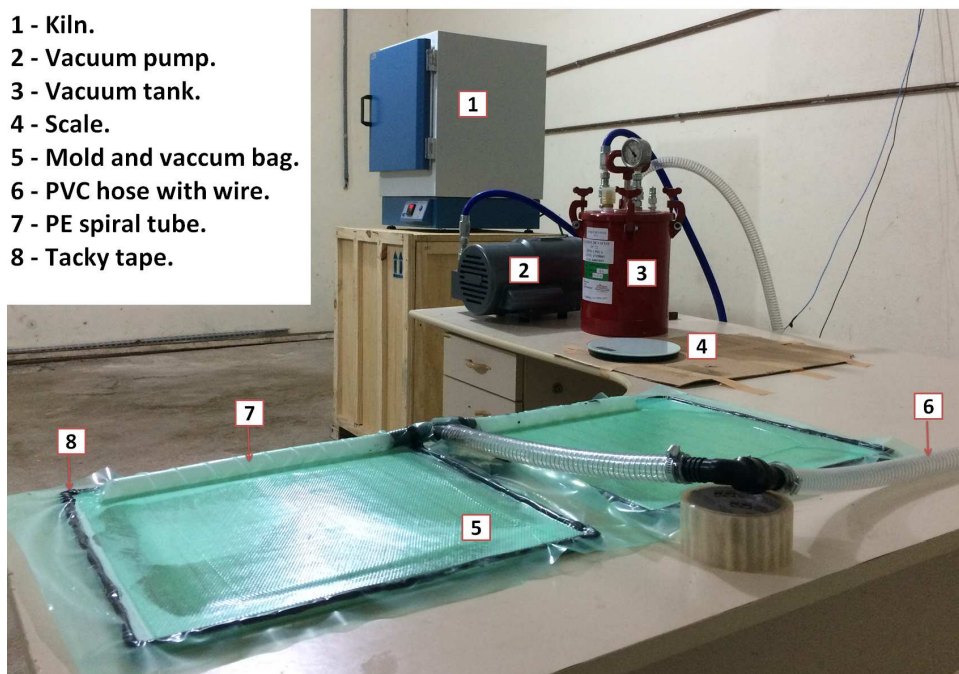
Table 4.1 – Characteristics of glass fiber shallow turkish.

Dimension	Unity	Value
Weight	g/m^2	230.2
Warp number	yarn/cm	18.0
Weft number	yarn/cm	6.0
Thickness	mm	0.24

Source: Author's production.

weave presents 75% of the fibers aligned in 0° direction and 25% in 90° direction.

Figure 4.1 – VARTM manufacturing process layout.



Source: Author's production.

The manufacturing process is summarized in a few steps:

- *Step 1:* put the tacky tape around the mold (glass), and spread a thin layer of carnauba wax. The carnauba wax helps at the demolding time, so the resin is not glued to the glass mold - Fig. 4.2;
- *Step 2:* put a peel ply layer, to help during demolding - Fig. 4.3;
- *Step 3:* spread a mixture of resin and hardener above the peel ply - Fig. 4.4;
- *Step 4:* position the glass fibers aligned. To simulate the delamination damage, a Teflon tape is placed above the fiber fabric - Fig. 4.5 (a);

- *Step 5*: put another peel ply layer - Fig. 4.5 (b);
- *Step 6*: place a flow media, to help the resin to flow - Fig. 4.5 (c);
- *Step 7*: close the bag vacuum with a vacuum film - Fig. 4.6;
- *Step 8*: start the vacuum pump, and monitor the flow of the resin;
- *Step 9*: turn off the pump after 5 hours (depend the gel time);
- *Step 10*: unmold after 24 hours of curing. Sometimes it is required the use of a pos-curing method.

Figure 4.2 – VARTM manufacturing process: step 1.



Source: Author's production.

Figure 4.3 – VARTM manufacturing process: step 2.



Source: Author's production.

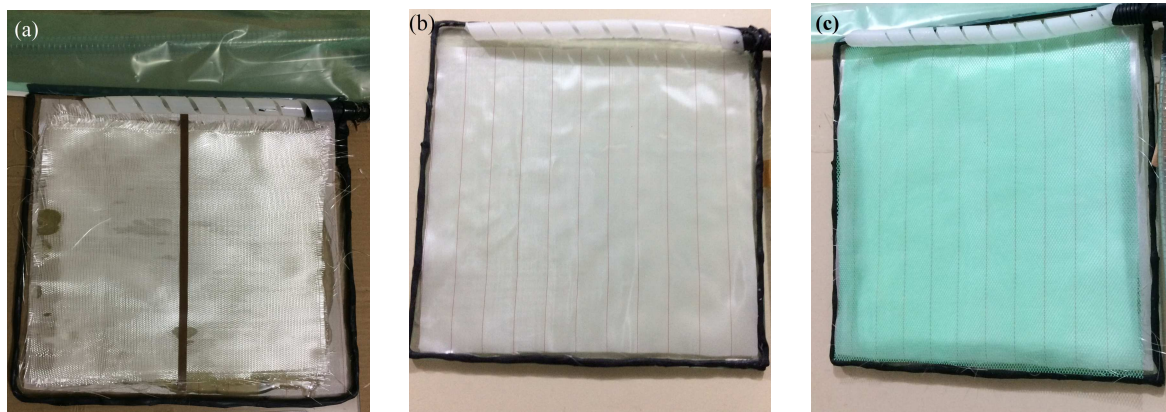
Different damage patterns are generated to evaluate the performance of the methodology. The set is composed of 3 undamaged plates, 2 plates with 5 mm delamination, 2 plates with 10 mm delamination, and two plates with 19 mm delamination. To simulate the delamination damage, a brown PTFE (polytetrafluoroethylene) Teflon tape is introduced in the middle of the plate (between plies 6 and 7). With the aim of

Figure 4.4 – VARTM manufacturing process: step 3.



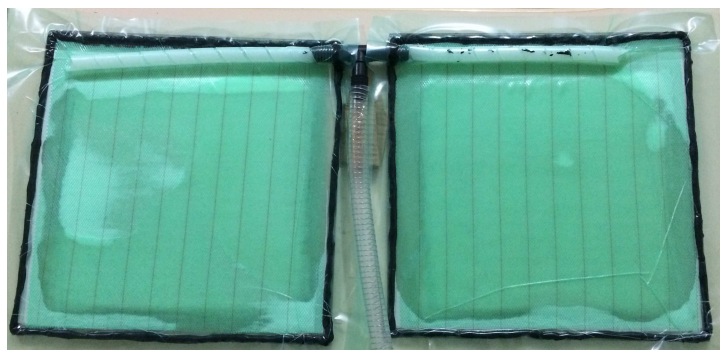
Source: Author's production.

Figure 4.5 – VARTM manufacturing process: step 4 (a), 5 (b) and 6 (c).



Source: Author's production.

Figure 4.6 – VARTM manufacturing process: step 7.



Source: Author's production.

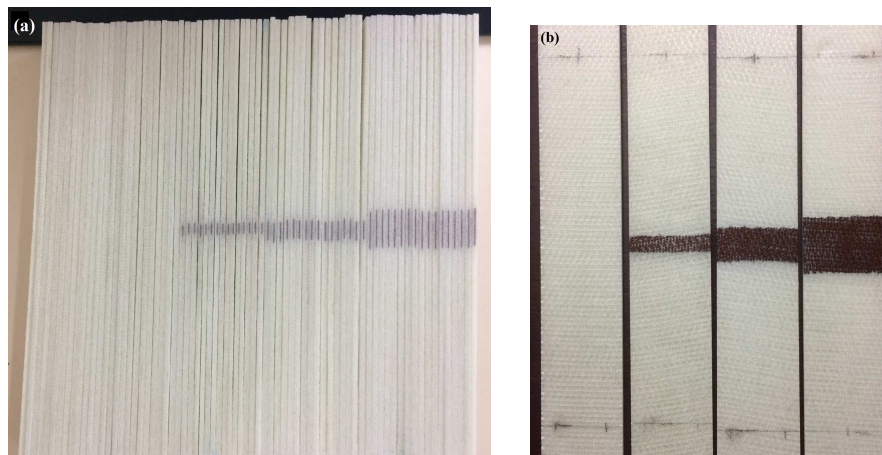
obtaining the largest number of samples, the plates are transversely cut into composite beams by using a diamond disk to avoid another delaminations cases as shown in Fig. 4.7. Figure 4.8 shows the composite beams with the four damages cases. In total, there are 25 beams for healthy case H , 15 beams for damage Level 1 $D-L_1$ (5 mm), 16 beams for damage Level 2 $D-L_2$ (10 mm) and 17 beams for damage Level 3 $D-L_3$ (19 mm). The dimensions of all beams are shown in Tab. 4.2 and Tab. 4.3 shows the mean dimensions and the mean deviation dimensions from all beams.

Figure 4.7 – Cutting machine.



Source: Author's production.

Figure 4.8 – Composite beams: four damages cases.



Source: Author's production.

Table 4.2 – Glass fiber/epoxy beams dimensions.

Beam	Width (mm)	Length (m)	Thickness (mm)	Beam	Width (mm)	Length (m)	Thickness (mm)
1	28.10	0.229	3.00	38	28.70	0.227	2.60
2	28.40	0.229	2.90	39	28.30	0.226	3.10
3	28.50	0.228	3.00	40	28.55	0.227	3.05
4	28.55	0.228	3.00	41	28.40	0.227	3.20
5	28.55	0.227	2.90	42	28.55	0.227	3.00
6	28.30	0.229	2.90	43	28.35	0.227	3.00
7	28.30	0.228	3.10	44	28.15	0.226	3.20
8	28.40	0.228	2.95	45	28.40	0.226	3.10
9	28.30	0.228	2.90	46	28.05	0.227	3.10
10	28.50	0.228	2.90	47	28.30	0.227	3.20
11	28.30	0.229	2.95	48	28.30	0.227	3.00
12	28.30	0.227	2.65	49	28.30	0.227	3.00
13	28.45	0.228	2.90	50	28.40	0.227	3.15
14	28.55	0.227	2.95	51	28.20	0.226	3.15
15	28.55	0.228	2.95	52	28.20	0.227	3.15
16	28.35	0.228	2.80	53	28.35	0.227	3.10
17	28.35	0.229	2.90	54	28.55	0.227	3.20
18	29.40	0.228	2.95	55	28.30	0.227	3.00
19	28.40	0.228	2.90	56	28.30	0.227	2.95
20	28.50	0.227	2.90	57	28.30	0.229	3.15
21	29.30	0.228	2.90	58	28.00	0.228	3.20
22	28.70	0.228	2.90	59	28.15	0.229	3.15
23	28.00	0.229	2.95	60	28.05	0.229	3.35
24	28.60	0.227	2.80	61	22.35	0.230	3.15
25	28.60	0.227	3.05	62	28.30	0.230	3.35
26	28.40	0.227	3.05	63	28.05	0.230	3.35
27	28.55	0.226	3.05	64	27.90	0.229	3.00
28	28.55	0.226	3.05	65	28.00	0.229	3.15
29	28.55	0.226	2.70	66	28.45	0.230	3.00
30	28.55	0.227	2.65	67	28.35	0.229	3.10
31	28.60	0.226	3.00	68	28.30	0.230	3.05
32	28.50	0.227	2.60	69	28.05	0.229	3.35
33	28.60	0.226	2.95	70	28.10	0.229	3.35
34	28.60	0.226	2.65	71	28.30	0.230	3.05
35	28.60	0.227	2.55	72	28.40	0.230	3.05
36	28.30	0.226	2.95	73	28.40	0.230	3.10
37	28.55	0.227	2.60	-	-	-	-

Source: Author's production.

Table 4.3 – Glass fiber/epoxy beams mean and mean deviation dimensions.

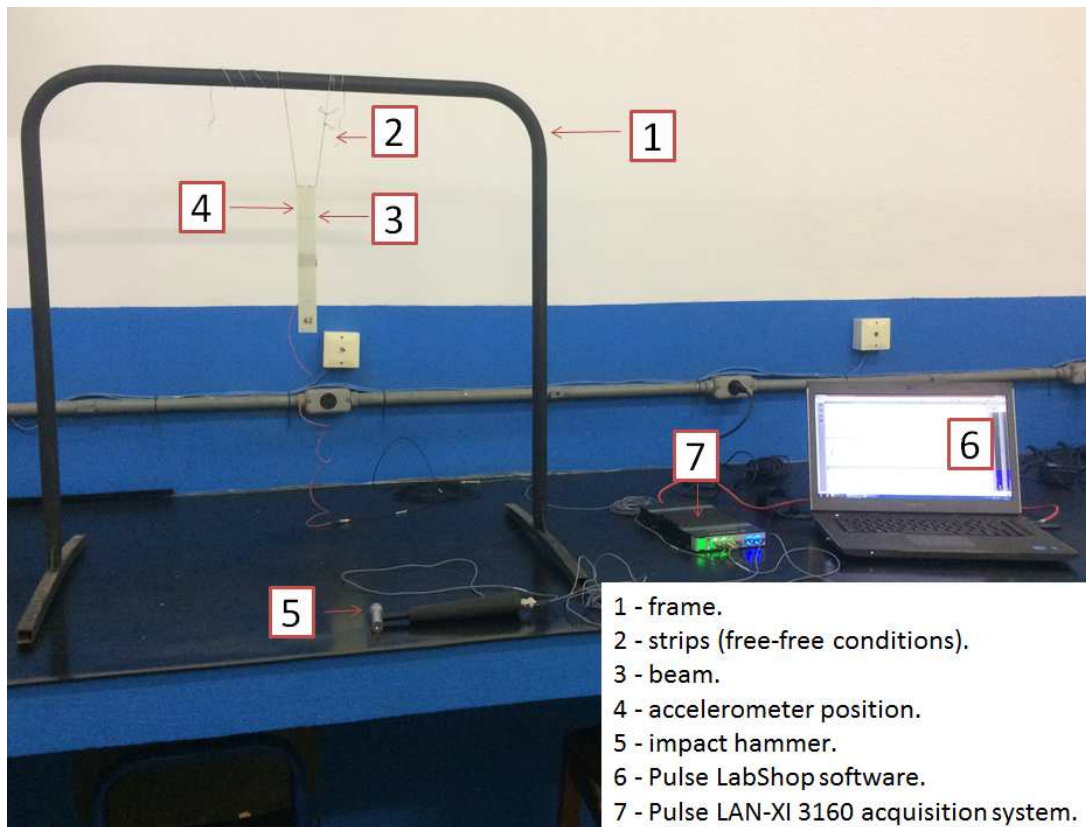
	Width (mm)	Length (mm)	Thickness (mm)
Mean	28.35	227	3.00
Mean Deviation	0.146	0.097	0.136

Source: Author's production.

4.1.2 Dynamic tests: Experimental

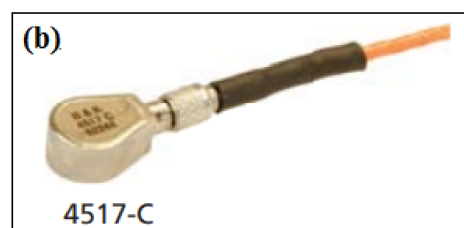
Dynamics tests are performed in each beam in free-free conditions as shown in Fig. 4.9. The excitation is provided by an impact hammer model 8206-003 (sensitivity 1.14 mV/N) with an aluminum tip from Brüel & Kjaer (B&K), and the response is measured by using a miniature accelerometer model 4517-C (sensitivity 0.18 pC/m^2 and weight 0.6 g) from B&K, as shown in Fig. 4.10. The PULSE LabShop software from B&K is used.

Figure 4.9 – Beam experimental setup.



Source: Author's production.

Figure 4.10 – Impact hammer (a) and miniature accelerometer (b).



Source: B&K

To connect the miniature accelerometer signal to CCLD inputs, a signal converter must be used. The converter model is 2647-B - fig. Fig. 4.11 - with a fixed sensitivity of 10 mV/pC. The experimental setup is shown in Tab. 4.4. The analyzed frequency range was 0 – 3200 Hz, in a total of 6400 spectral points.

Figure 4.11 – Signal converter.



Source: B&K

Table 4.4 – Experimental setup.

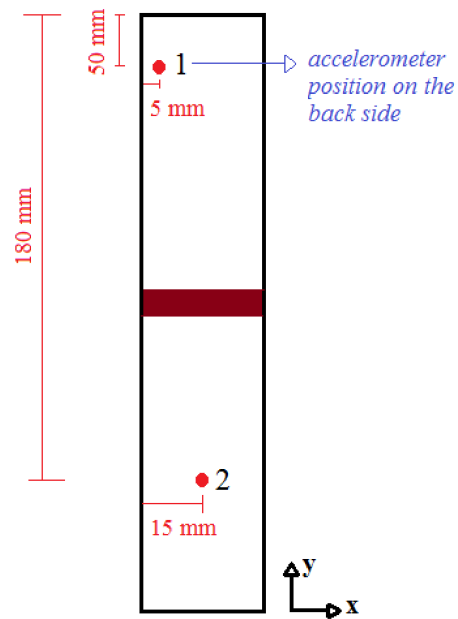
Parameter	Value
Bandwidth	3.2k Hz
Spectral lines	6400
Acquisition time	2 seconds
Resolution	0.5 Hz
Averages	3
Window input	Force - Transient
Window response	Exponential
Frequency response	H1
Test variance	2 days

Source: Author's production.

The accelerometer position is shown in Fig. 4.12, it is glued on the back side of the position 1, the forces are applied in position 1 (FRF H_{11}) and 2 (FRF H_{21}).

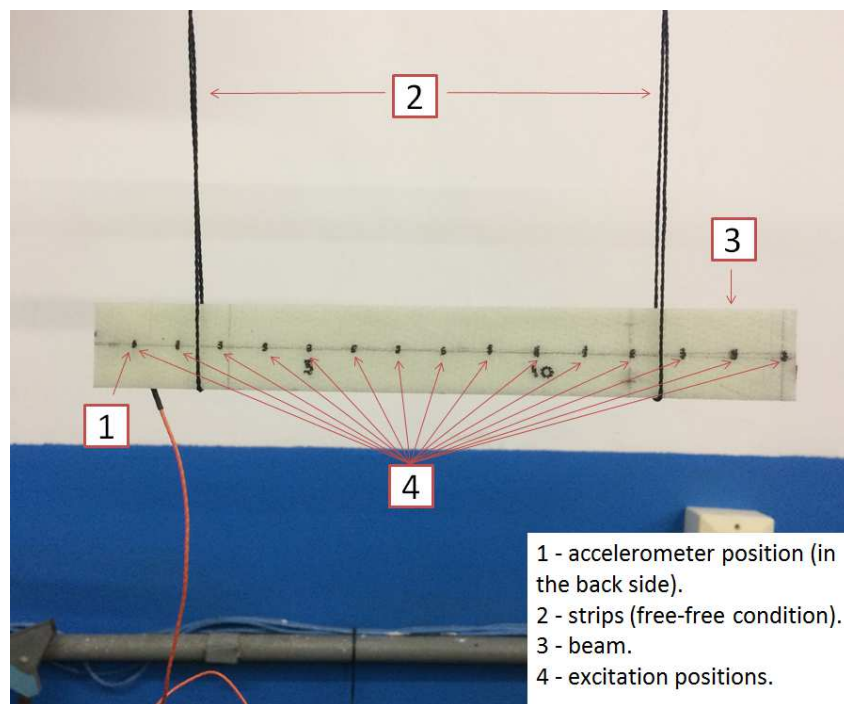
To analyze which modes correspond to flexural and torsional modes a modal analysis is performed in an undamaged beam using the same experimental system mentioned above as shown in Fig. 4.13, considering 15 excitation points.

Figure 4.12 – Accelerometer position and excitation position on glass fiber/epoxy beam.



Source: Author's production.

Figure 4.13 – Modal Analysis Setup.

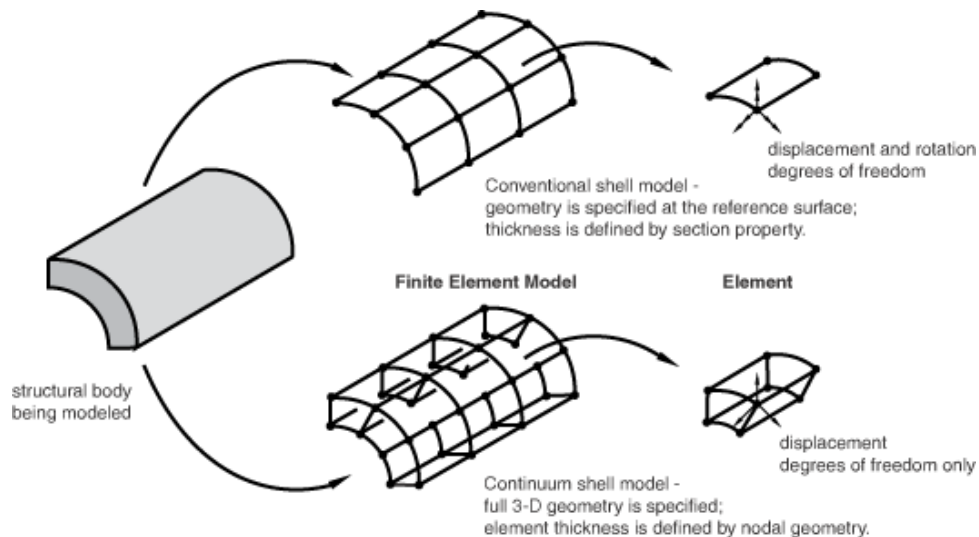


Source: Author's production.

4.1.3 Dynamic tests: Numerical

A numerical example using a finite element method (FEM) of a simple composite beam in free-free condition is selected to validate the effectiveness of the proposed methodology. The FEM model is done in ABAQUS 6.12, and consists of 25938 (mesh size 0.001) quadrilateral continuum shell elements (SC8R), with 8 nodes with three degrees-of-freedom (displacement). Shell elements are used to model structures in which one dimension, the thickness, is significantly smaller than the other dimensions. Conventional shell elements apply this condition to discretize a body by delimited the geometry at a reference surface. In the other hand, continuum shell elements discretize an entire three-dimensional body. The thickness is defined from the element nodal geometry (SIMULIA, 2013), as explained in Fig. 4.14.

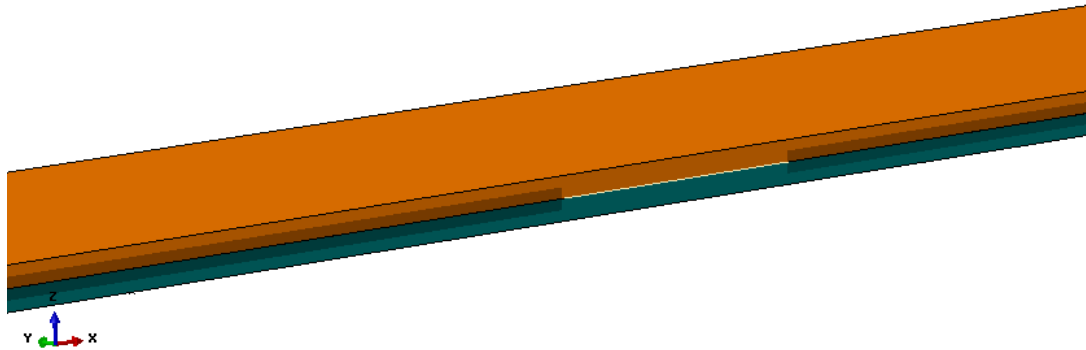
Figure 4.14 – Conventional shell and continuum shell.



Simulia (2013)

The double-beam can be seen in Fig. 4.15 where the line between the beams is characterized by the delamination.

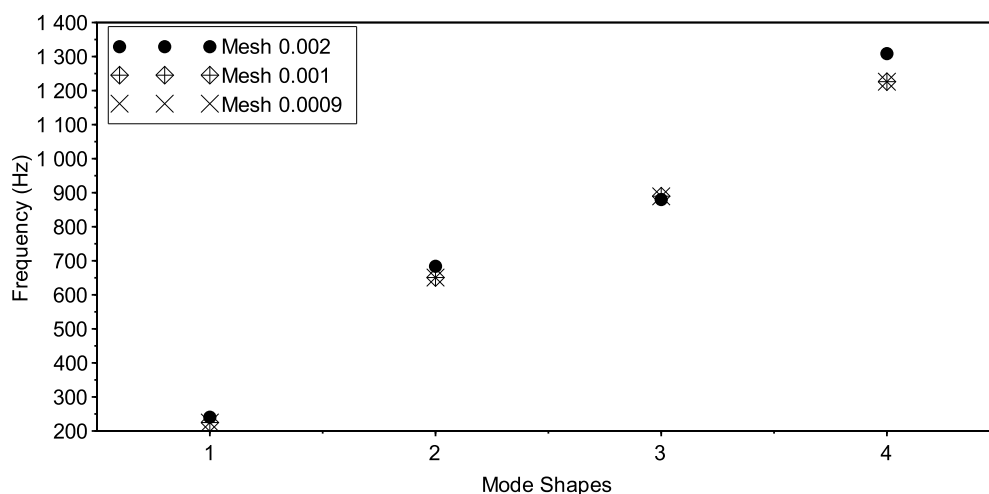
Figure 4.15 – Double-beam FEM model with delamination.



Source: Author's production.

The total number of elements is chosen according to the error between the frequencies of a very refined mesh (mesh size: 0.0009) and another two (0.002 and 0.001), as shown in Fig. 4.16. The maximum error between the frequencies of the mesh of 0.002 compared to the most refined is 6.99% (first mode), and the maximum error between the frequencies of the 0.001 mesh is 0.047% (third mode). Thus, as decides to use a mesh size of 0.001.

Figure 4.16 – Mesh convergence.



Source: Author's production.

The geometrical and the mechanical properties are shown in Tab. 4.5, and they are chosen according to the calculations done in the Ignition loss test of manufactured specimens (Appendix C), and also through the model updating comparing with the

experimental natural frequencies.

Two accelerometers and a force excitation are added in the same position as in the experimental procedure (Fig. 4.12), and the vibration signals are simulated with a sampling frequency of 1.0 Hz. The frequency range studied is 1500 Hz using the direct steady-state dynamic analysis, and the state conditions are simulated as in the experimental phase. The Rayleigh damping is used and its proportional parameters (α and β) are introduced in the simulation. The parameters are calculated using the critical damping found using peak-picking method from the experimental beams.

Table 4.5 – Geometrical and mechanical properties: numerical simulation.

Property	Value
Length	227 mm
Width	28.35 mm
Thickness	3.00 mm
Longitudinal elasticity modulus - E_{11}	30.0 GPa
Transversal elasticity modulus - E_{22}	15.0 GPa
Composite density - ρ_c	1260 kg/m^3
Shear modulus - G_{12}	5.9 GPa
Shear modulus - G_{13}	5.9 GPa
Shear modulus - G_{23}	4.0 GPa
Poisson ratio - ν_{12}	0.18
Stacking orientation	$[0]_{12}$

Source: Author's production.

In this study different dynamic tests are simulated for each stage of damage, varying the mechanical properties and the stacking orientation as presented in Tab. 4.6. The values are chosen varying the properties between $\pm 10\%$. The procedure is implemented as a Python script in order to automate the process. So for example, varying the longitudinal modulus from 27.0 GPa to 33.0 GPa at a 0.5 GPa step gets in total 26 FRFs for position H_{11} and H_{21} . In total 94 FRFs are simulated for state conditions healthy and damaged level 1, 72 FRFs for damaged level 2 and 70 FRFs for damaged level 3. In the case of damaged level 2 and 3 the changes in the stacking orientation were not considering, because that the difference of the numbers of FRFs comparing with the first two states conditions. And, also, in damaged level 3, the changes in the shear modulus were only from 5.3 GPa and 5.9 GPa, because that there are 70 FRFs instead of 72 FRFs.

Table 4.6 – Mechanical properties variation: numerical simulation.

Property	Minimum Value	Maximum Value	Step	Samples
E_{11}	27.0 GPa	33.0 GPa	0.5 GPa	26
E_{22}	13.5 GPa	16.5 GPa	0.5 GPa	14
ρ_c	1160 kg/m^3	1380 kg/m^3	20 kg/m^3	24
G_{12}	5.3 GPa	6.2 GPa	0.3 GPa	8
Stacking orientation	$[-5]_{12}$	$[+5]_{12}$	1.0 °	22

Source: Author's production.

4.2 Damage Detection in Carbon Fiber/Epoxy Plates

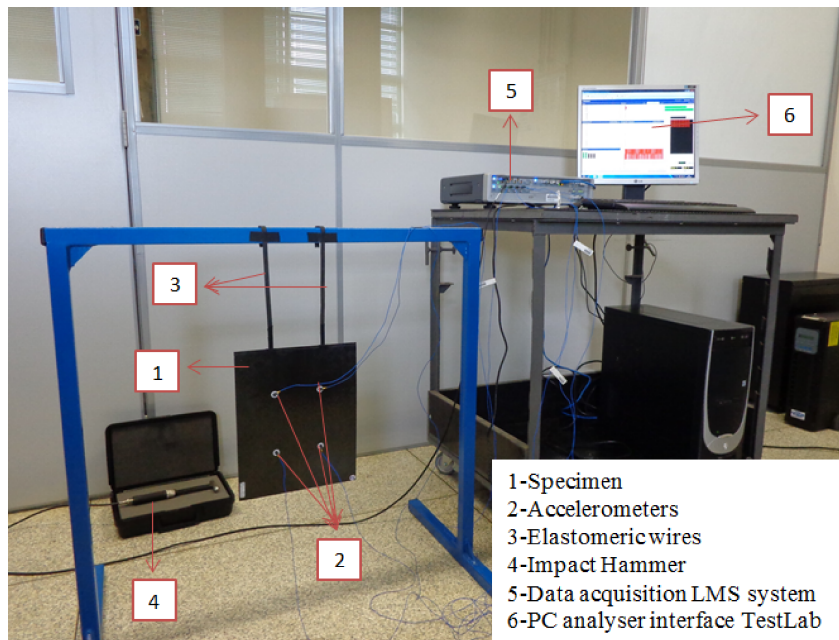
Sixteen composite plates produced from carbon fibers with epoxy resin through the filament winding process are studied. The specimens were manufactured by the Brazilian Navy Technology Centre in São Paulo and made available for previous studies made by Medeiros (2016). The carbon fiber composites are split into two groups: eight plates with twelve layers and stacking orientation $[0/15/-15/0/15/-15]_S$, and eight plates with eight layers and stacking orientation $[0]_8$. The average thickness for the first group is 3.39 mm (standard deviation = 0.07) and for the second 2.23 mm (standard deviation = 0.02). The experimental procedure was previously carried out by Medeiros (2016).

Initially, the dynamic response of each healthy laminated plate was obtained by using four accelerometers (models 352A24) in free-free boundary conditions, as shown in Fig. 4.17. The impulse input by impact hammer was applied on a fixed point on the back side of the plate. In the vibration tests, the analyzed frequency range was 0 – 1024 Hz, in a total of 2048 spectral points. Each Frequency Response Function (FRF) was obtained through an average of five samples, aiming to reduce the effects of variation (MEDEIROS, 2016).

The second stage consists of the damage test phase. For the $[0]_8$ stacking orientation group, five plates were damaged by impact loading, one was damaged by drilling a center hole, and the other two kept undamaged. For the $[0/15/-15/0/15/-15]_S$ plates, four were damaged by impact loading, two by delamination and other two kept undamaged. The damaged plates were analyzed again through the use of vibration methods monitored by accelerometers. In total 56 FRFs for $[0]_8$ orientation plates were obtained, 32 FRFs for healthy cases and 24 FRFs for damaged cases. For $[0/15/-15/0/15/-15]_S$ plates, a total of 48 FRFs were obtained, 24 for undamaged cases and 24 for damaged cases (MEDEIROS, 2016).

Due to a large number of inputs associated with each FRFs, and the consequently large number of connections required to form a suitable ANN topology, the PCA is used to reduce the dimension of the experimental data. After using the PCA,

Figure 4.17 – Dynamic experimental set-up.



Source: Medeiros (2016)

the data is split into three data sets: training, validation, and testing. The training set is used as input into the neural network, and the error calculation was performed. The validation set is then used to monitor the behavior of the network simultaneously with the training set during the learning phase, to monitor the best values of the hyperparameters and topology. After the learning phase, the network is then tested by using the last set of data, data not used during training. A confusion matrix is performed in the testing set to evaluate the performance of the ANN classifier.

4.3 Fault Diagnosis of Composite Structures: Methodology.

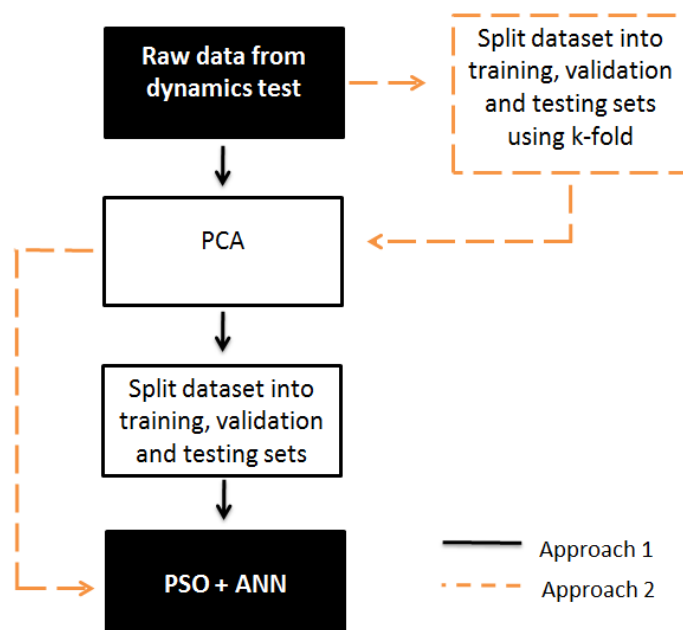
4.3.1 Principal Component Analysis

Considering a large number of spectral lines acquired in the vibration tests, the use of this data directly applied in the neural network would produce slow and inefficient learning. For example, if 3201 spectral points (raw data) were used as the input to a single large neural network, which would give 3202 adaptive weights (including the bias) for every unit in the first hidden layer. This suggests that a very large training set would be needed to ensure that the weights and bias were well determined, as well as huge computational resources would be necessary to find an appropriate minimum of the error function and the problem becomes infeasible. To try to ensure network

convergence the size of the original data must be reduced.

To reduce the dimension of the problem, a Principal Component Analysis (PCA) is implemented in the raw data to get the first principal components (PCs) with the maximum variance. Discussions between machine learning and data extraction techniques have been carried out around the order of the PCA application and the division of the data set into training, validation, and testing. Authors like Bandara *et al.* (2014), Li *et al.* (2011) and Zang and Imregun (2001a) use the PCA before splitting the dataset into three subsets. Other authors, Karpathy (2017), Ng (2018) and Vidhya (2016), first split the three subsets and then apply the PCA only in the data set called training after applying them to the two subsequent sets. According to them, starting with the PCA and then splitting the dataset the validation and testing sets samples would be contaminated with training data, since ANN's approach is that the validation and testing sets contain unseen samples, that is, not used during neural network training. The two approaches are used in this work, as can be seen in Fig. 4.18, in approach 1 the PCA is first applied and then the dataset is divided whereas in approach 2 the dataset is split using cross-validation (k-fold) before using PCA.

Figure 4.18 – Dataset split and PCA approaches.



Source: Author's production.

Table 4.7 shows the main lines of the PCA-2 to get a better idea of Approach 2. First, the raw dataset is split into 5-folds containing training set and testing set, using *kfold.jl* subroutine. For example, if the dataset is composed by 90 observations, 18 observations go to the testing set and 72 go to the training set. After that, each 5-fold of the training sets are again split by 85% for a new training set and 15% for validation set

using *stratifiedobs.jl* subroutine. The use of *stratifiedobs* will try to make sure that both subsets are both appropriately distributed. After PCA calculations on the training set, the change of space of the validation and the testing sets are performed by multiplying the eigenvectors (which means the new base) found during PCA calculations using only the training set.

Table 4.7 – PCA Algorithm - approach 2

```

0 begin
1  Split the raw data into  $\mathbf{X}_{train}$  and  $\mathbf{X}_{testing}$ : using kfold(data,k=5)
2  Split the  $\mathbf{X}_{train}$  into  $\mathbf{X}_{train_{new}}$  and  $\mathbf{X}_{validation}$ , using stratifiedobs(data),  $p = 0.85$ )
3  PCA calculations:
4  for  $\mathbf{X}_{train_{new}}$  data
5    perform the correlation matrix calculation
6    get eigenvalues  $\lambda$  and eigenvectors  $\phi$  from the first P-principal components
7    perform new space matrix calculation  $\mathbf{A}_{train}$ 
8    return  $\mathbf{A}_{train}$ ,  $\lambda$  and  $\phi$ 
9  end
10 for  $\mathbf{X}_{validation}$  and  $\mathbf{X}_{testing}$ 
11   perform new spaces matrices calculations with  $\phi$ 
12   return  $\mathbf{A}_{validation}$  and  $\mathbf{A}_{testing}$ 
13 end
14 end

```

Source: Author's production.

4.3.2 Artificial Neural Networks

A Multilayer Neural Network Perceptron (MLP) program is developed in the Julia language (version 0.6) (BEZANSON *et al.*, 2017). Sensitivity analyses is performed by automatic differentiation by means of dual numbers (REVELS *et al.*, 2016), as shown in Tab. 4.8.

One issue when working with ANNs is the difficulty in estimating the hyperparameters, such as the number of hidden layers, the number of the neurons in each hidden layers, the value for the momentum term, the initial value for the line search (or learning rate), as well as what type of activation function and cost function to use and the application of some technique to prevent overfitting. To solve the first issue, an optimization program using Particle Swarm Optimization (PSO) method is used to find the best topology to be used in the function of the validation set accuracy during the learning phase. Other optimization methods can be applied but due to robustness and ease of use, the PSO is chosen.

Being $f(\mathbf{X}_{top})$ a function of the validation set accuracy:

Maximize $f(\mathbf{X}_{top})$,

Table 4.8 – Sensitivity analysis algorithm

```

0 begin
1  for  $t = 1 : N_{iterations}$ 
2      convert ( $weights$ ,  $bias$ ,  $input$  and  $output_{target}$ ) real numbers into dual numbers
3      return  $w_{u'}$ ,  $b_{u'}$ ,  $x_{u'}$  and  $y_{u'}$ 
4      for  $w = 1 : N_{weights}$ 
5          for  $b = 1 : N_{bias}$ 
6              apply perturbation to the current weight/bias
7              for  $s = 1 : N_{samples}$ 
8                  for  $k = 1 : N_{layers}$ 
9                      for  $j = 1 : N_{neurons}$ 
10                         perform an activation function calculation
11                         return  $output_{jk}$ 
12                     end j
13                 end k
14                 perform a cost function calculation between  $output_{network}$  and  $output_{target}$ 
15                 return error  $E(s)$  and  $output_{network}$ 
16             end s
17             perform an error cost function calculation
18             return error  $\sum E(s)$ 
19             stores  $\nabla w$  and  $\nabla b$ 
20             remove perturbation
21         end w
22     end b
23     return  $\nabla w$  and  $\nabla b$ 
24 end

```

Source: Author's production.

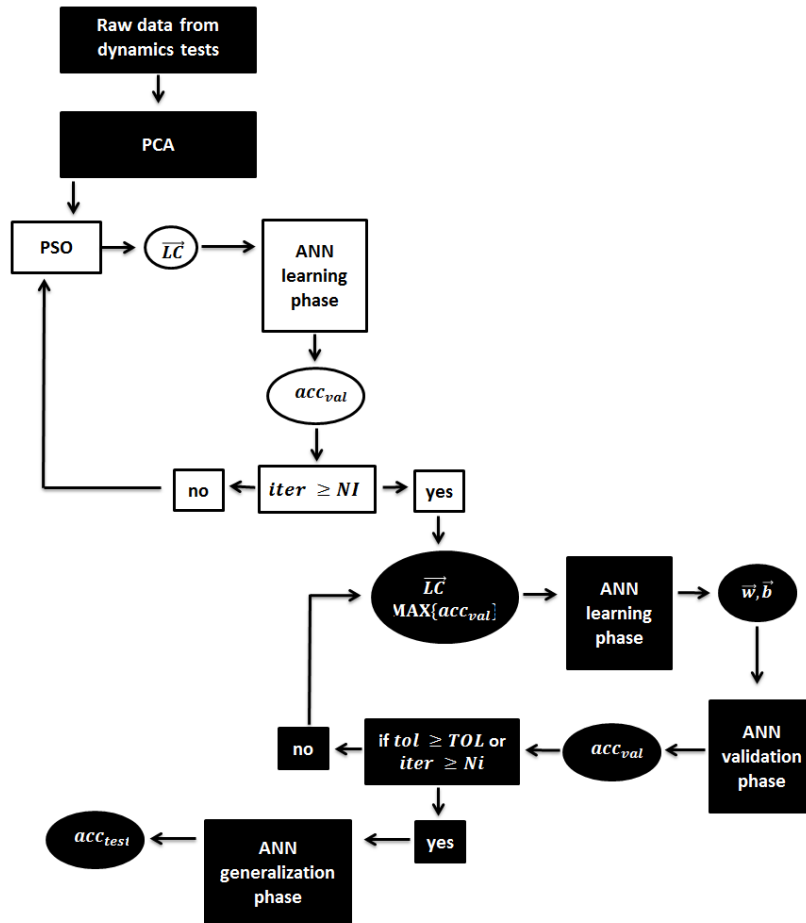
$$\text{with } \mathbf{X}^{low} \leq \mathbf{X}_{top} \leq \mathbf{X}^{upper},$$

where \mathbf{X}_{top} is a vector composed by three design variables, such as x_1 : number of the hidden layers; x_2 : number of the neurons from the first hidden layer, and x_3 : number of the neurons from the second hidden layer. \mathbf{X}^{low} are the lower bound and \mathbf{X}^{upper} the upper bound on \mathbf{X}_{top} . The lower bound for each variable is 1 and the upper bound depends of the problem, except for x_1 , where the upper bound is 2, *i.e.*, the maximum of two hidden layers is studied. The project variables are allocated within the topology vector represented by \vec{LC} . The PSO works with continuous variables so, as in the topology only integer variables can be entered, a rounding criterion is assumed. For example, if the PSO found a topology of 1.51 layers then it is leased to 2 layers; if a topology of 1.49 has been found, then it is rounded to 1 layer. The same is true for numbers of neurons.

The whole methodology is shown in Fig. 4.19: first the raw data is introduced to PCA analysis, and the new space data introduced to ANN. A PSO algorithm is run in the learning phase to maximize $f(\mathbf{X}_{top})$, with different topologies. After the maximum number of iterations (NI) the best solution (the most suitable topology) is introduced again to ANN learning phase, as \vec{LC} to find the best values of the weights and the

biases (\vec{w}, \vec{b}). In order to verify the hyperparameters, the validation data set is run in the learning phase to get the maximum validation accuracy. If the number of iterations (Ni) or the tolerance (TOL) between the two consecutive error functions are exceeded the learning process stops. The final step is to evaluate the generalization behavior of the ANN with testing set accuracy (acc_{test}).

Figure 4.19 – Methodology flowchart.



Source: Author's production.

The main lines of the Artificial Neural Network (ANN) algorithm are shown on Tab. 4.9. The input data are normalized to accelerate the learning process. In order to update the weights and biases values the gradient descent method is used. In this way, the steepest descent method is used to generate the direction toward the minimum of the cost function and backtracking line search to determine the step size (learning rate). When the learning rate is a fixed number, the backtracking line search is replaced by a constant number. Different types of activation (logistic, tanh, leaky-ReLU and SoftMax) and cost functions (quadratic, norm-2 and cross-entropy) are studied in this work.

To verify the learning process, the validation set is used. The main lines of the

Table 4.9 – ANN algorithm - learning phase

```

0 begin
1  inform hyperparameters
2  initialize weights and biases
3  normalize the input data
4  for  $t = 1 : N_{iterations}$ 
5    perform sensitivity analysis (Tab. 4.8)
6    return  $\nabla w$  and  $\nabla b$ 
7    define the search direction: steepest descent
8    return  $d_w$  and  $d_b$ 
9    perform line search calculation: backtracking or fixed
10   return  $\alpha$ 
11   perform synaptic weights and synaptic biases calculation: gradient descent method
12   return  $\Delta w$  and  $\Delta b$ 
13   update the weights and biases
14   return  $w$  and  $b$ 
15   perform a new error cost function calculation (step 7 to 17)
16 end t
17 perform a network total error calculation (step 7 to 17)
18 return network total error
19 end

```

Source: Author's production.

code are shown on Tab. 4.10.

Table 4.10 – ANN algorithm - validation set verification

```

0 begin
1  initialize weights and biases (according to the learning phase)
2  normalize the input data
3  Feedforward propagation:
4  for  $s = 1 : N_{samples}$ 
5    for  $k = 1 : N_{layers}$ 
6      for  $j = 1 : N_{neurons}$ 
7        perform an activation function calculation
8        return  $output_{jk}$ 
9      end j
10     end k
11     perform a cost function calculation between  $output_{network}$  and  $output_{target}$ 
12     return error  $E(s)$  and  $output_{network}$ 
13   end s
14   perform an error cost function and accuracy calculations
15   return error  $\sum E(s)$  and accuracy
16 end

```

Source: Author's production.

Chapter 5

Results

In this section, the results obtained from the studies using carbon fiber-epoxy plates and glass fiber-epoxy beams are presented. The first section discusses the application of the methodology, in the context of damage existence, in carbon fiber-epoxy plates with two different stacking orientations, and also the evaluation of the classifier performance when using two types of the Frequency Response Functions (accelerance and real part curves). In the second section, the study of glass fiber-epoxy beams are presented with experimental tests, split in five cases. Cases I to IV address the use of the classification methodology in the context of damage existence, question I in the Rytter's scale (damage state). Case V addresses the use of the methodology regarding to question I and also question IV about damage extension, classifying among the four conditions states. In the third section, a numerical study using glass-fiber/epoxy beams modeling in a Finite Element Model (FEM) is presented. The curves of the dynamic responses acquired in the simulation are introduced in the methodology to evaluate the effectiveness of the methodology compared to the experimental study.

5.1 Damage Detection in Carbon-Fiber/Epoxy Composite Plates

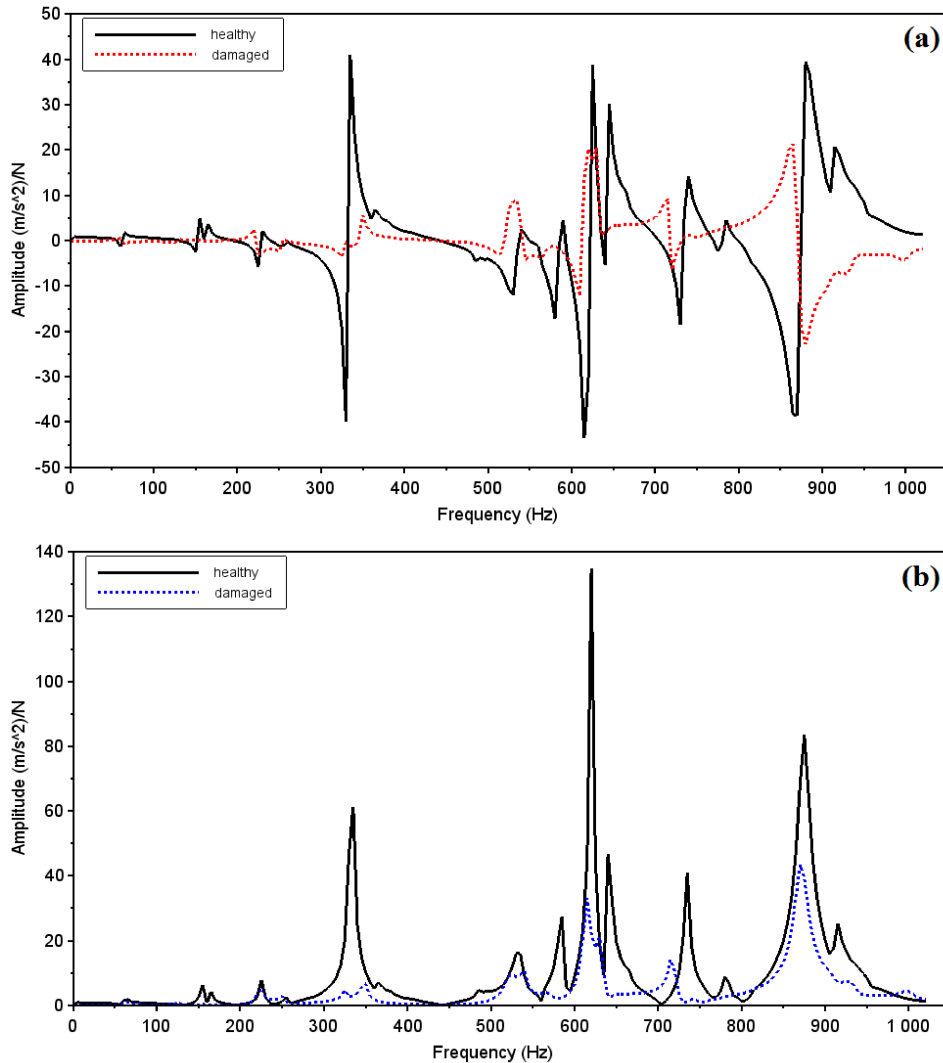
As mentioned in the previous chapter, the type of the Frequency Response Function (FRF) curves brings different pieces of information about the dynamic system, such as magnitude curves, phase curves, real part curves, and imaginary part curves. Each stacking orientation group is studied separately, due to the large differences in dynamic behavior among them.

5.1.1 Composite plates with stacking orientation $[0]_8$

5.1.1.1 Dynamic tests

The real part values of FRFs, considering a healthy and a damaged plate are shown in Fig. 5.1 (a), and the magnitude (accelerance) values of the same FRFs, *i.e.*, for the same position and plate are shown in Fig. 5.1 (b). A large difference can be observed between healthy and damaged cases, especially when working with real part values, where the inversions of the phases are quite remarkable. To visualize a whole set of FRFs and see the differences between the samples, FRFs curves of all plates are shown in Appendix A, in Fig. A.1 for magnitude values, and in Fig. A.2 for real part values of FRF.

Figure 5.1 – FRFs from healthy and damaged plate for $[0]_8$ (a) real part and (b) magnitude.

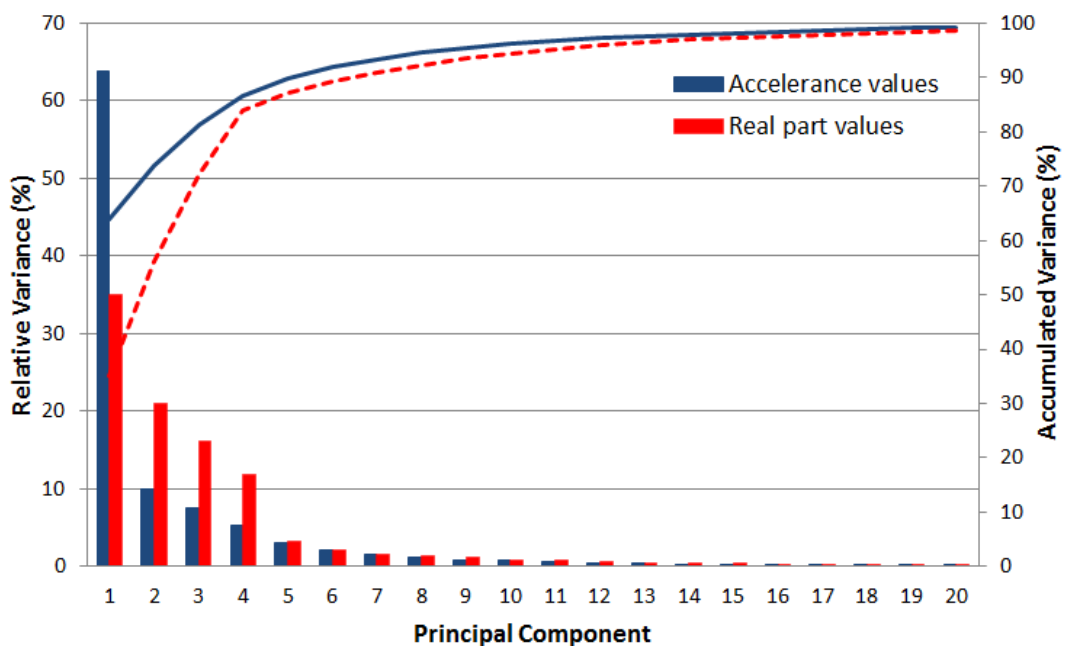


Source: Author's production.

5.1.1.2 Principal component analysis

The dimension problem is very high, with 2048 spectral lines. Thus, the Principal component analysis (PCA) is used to reduce the input size of the Artificial Neural Network (ANN) problem. The relative and the accumulated variance for the first 20 Principal Components (PCs) are shown in Fig. 5.2. As observed, the first PCs bring more information about the data, which means, more variance. It is possible to see that around the 10th PC the variance is less than 1%.

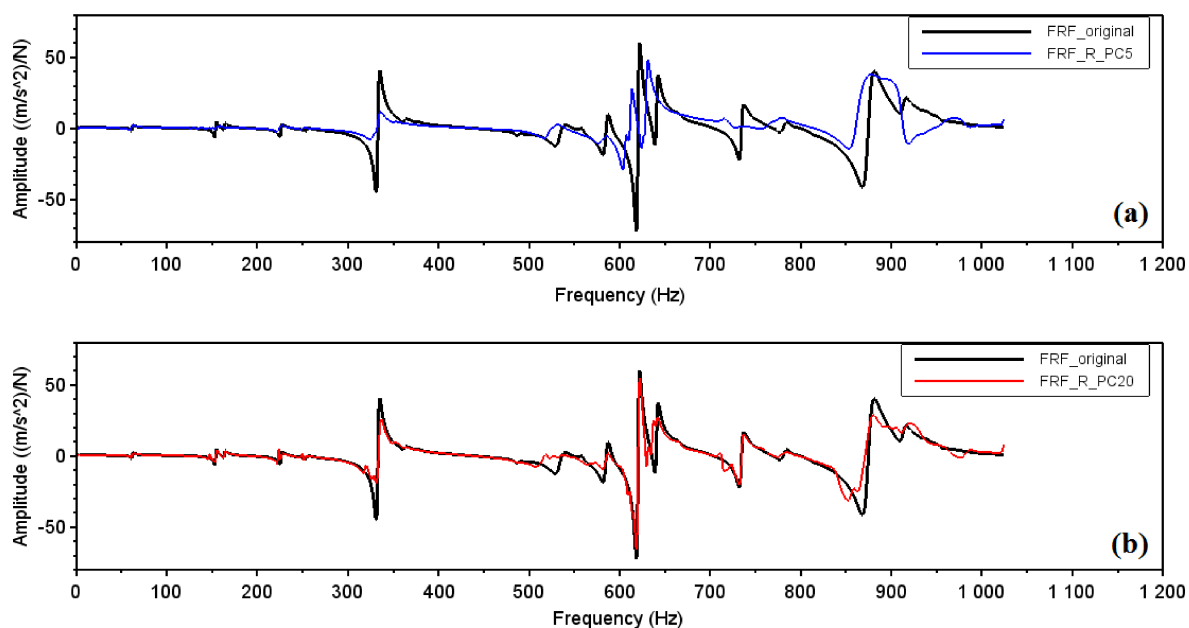
Figure 5.2 – Relative variance and accumulated variance using acceleration values and real part values ($[0]_8$).



Source: Author's production.

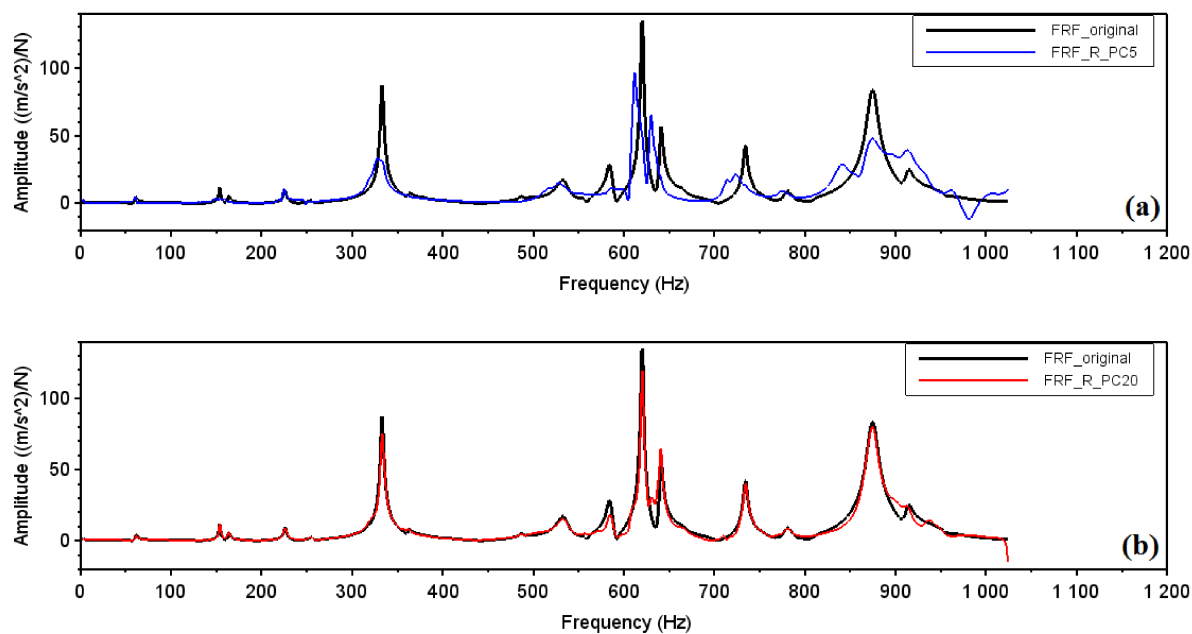
To evaluate how much percentage of variance is necessary to have a proper representation of the data, the FRFs are reconstructed using the first 5 PCs giving 87.17%, and using the first 20 PCs giving 98.55% for real part values, as shown in Fig. 5.3. It can be seen that 20 PCs better represent the original curve, when compared to the result obtained with just 5 PCs. The same procedure is performed using acceleration values, the first 5 PCs bring 89.73% of the variance, and the first 20 PCs bring 99.20% of the total variance. The reconstructed FRFs are shown in Fig. 5.4 using acceleration values.

Figure 5.3 – FRFs reconstructed with 5 PCs (a) and 20 PCs (b) comparing with original FRF: real part values ($[0]_8$).



Source: Author's production.

Figure 5.4 – FRFs reconstructed with 5 PCs (a) and 20 PCs (b) comparing with original FRF: accelerance values ($[0]_8$).

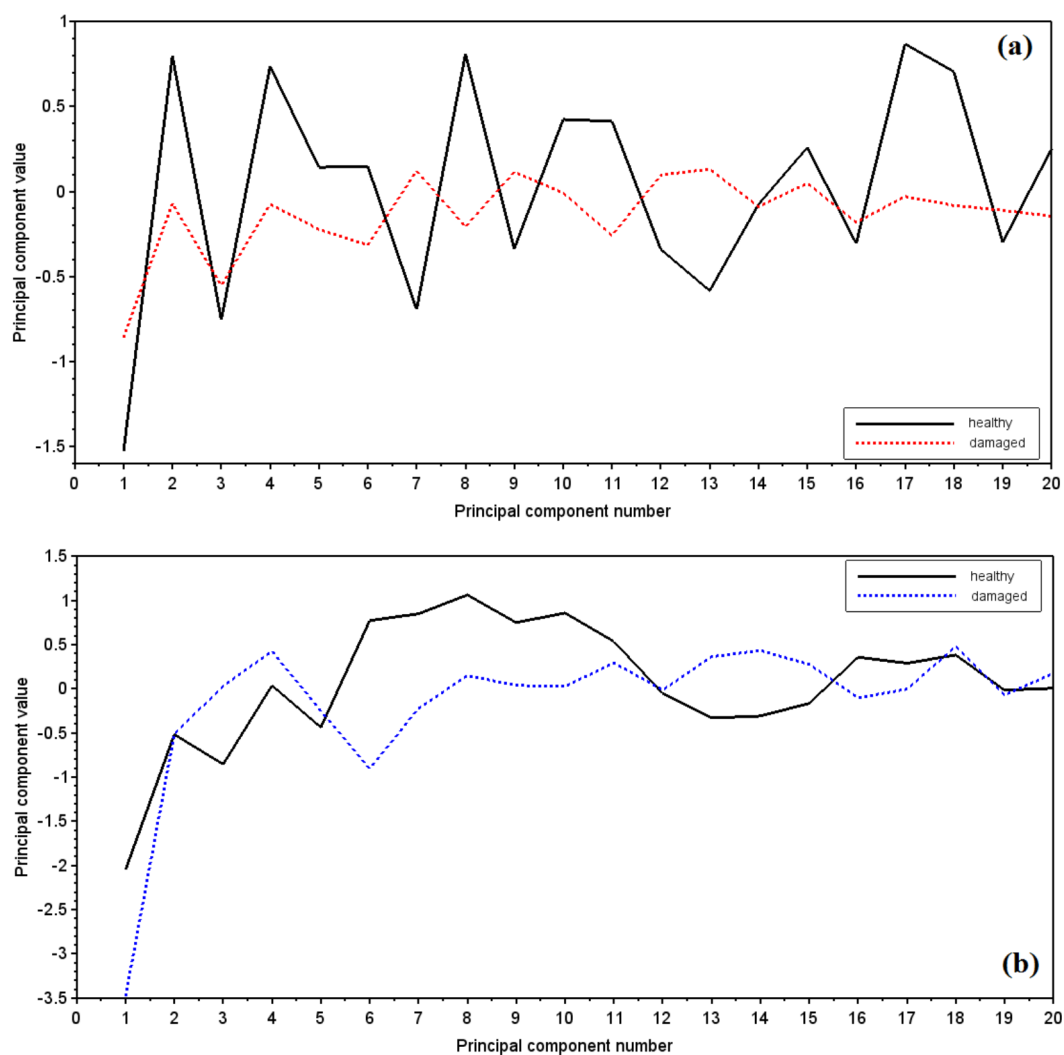


Source: Author's production.

As mentioned in previous chapters, as the problem size increases, the more samples are needed to train the network. So it is important to take a look of the samples and the reconstructed FRFs to evaluate if the PCs is not carrying some noise or irrelevant information about the data.

Figure 5.5 shows the PCs curves for healthy and damaged cases, using the real part values (a) and the PCs curves using the accelerance values (b) for the same specimen. No physical meaning is observed as expected. To visualize a whole set of PCs curves, Appendix A brings the graphics of PCs using accelerance values in Fig. A.5 and using real part values in Fig. A.6.

Figure 5.5 – PCs curves using (a) real part values and (b) accelerance - $([0]_8)$.



Source: Author's production.

5.1.1.3 Artificial neural networks and pattern recognition

The data are split into 61% for the training set, 21% for the validation set and 18% for the testing set. A Particle Swarm Optimization (PSO) is used to find the approximate topology of the ANN, in which the program finds the best results, maximizing the accuracy of the validation set according to three design variables, as explained in the previous chapter. The PSO parameters used are shown in Tab. 5.1.

Table 5.1 – PSO algorithm parameters.

Parameter	Value
C_1	1.2
C_2	1.2
w	0.5
Iterations	10
Particles number	10
Variables number	3
Lower bound values	[1;1;1]
Upper bound values	[2;24;24]

Source: Author's production.

The PSO results for accelerance values is $X_{top}=[1.51;9.05;2.10]$ with 91.67% of accuracy in the validation set and for real part values is $X_{top}=[1.48;5.97;5.59]$ with 100% of accuracy in the validation set.

The first simulation is performed using only accelerance values, the ANN topology is [20-(9,2)-1], *i.e.*, 20 inputs, 9 and 2 neurons for the first and second hidden layers, respectively, and one output neuron (0 for healthy and 1 for damaged). The logistic function is used as an activation function and L_2 function as a cost function. Steepest descent method and backtracking line search are used for the descent direction calculations. Also, a momentum term of 0.7 is used.

The second simulation is performed using only the real part values of the FRFs. The topology consists of 20 inputs, 6 neurons in the hidden layer and one output neuron (0 for healthy and 1 for damaged) – [20-(6)-1]. The same activation and cost functions, and also the same parameters (backtracking line search and momentum term) are used as before.

Each simulation is run 100 times to assess the convergence of the method. For each run the weights and biases are initialized with different values. The mean results are shown in Tab. 5.2.

Table 5.2 – ANNs simulations summary results for $[0]_8$ plates (100 times runs).

Data set	Accelerance values	Real part values
Training: mean accuracy (mean deviation)	100.0% (0.000)	100.0% (0.035)
Validation: mean accuracy (mean deviation)	83.3% (0.025)	83.3% (0.046)
Testing: mean accuracy (mean deviation)	70.0% (0.073)	93.5% (0.091)

Source: Author's production.

As observed both simulations reach 100.0% of accuracy in the training set, showing no signal of underfitting problems, and 83.3% of accuracy in the validation set. However, using real part values show better generalization than accelerance values.

Table 5.3 shows the confusion matrix after 100 runs, considering the accelerance values of the FRFs. For real part values, the confusion matrix is shown in Tab. 5.4. The confusion matrix parameters for both simulations are shown in Tab. 5.5.

Table 5.3 – Confusion matrix after 100 runs $[0]_8$ plates (accelerance values).

		Actual Class		Total
		Damaged	Healthy	
Predict Class	Damaged	89	89	178
	Healthy	211	612	823
Total		300	701	1001

Source: Author's production.

Table 5.4 – Confusion matrix after 100 runs $[0]_8$ plates (real part values).

		Actual Class		Total
		Damaged	Healthy	
Predict Class	Damaged	295	60	355
	Healthy	5	641	646
Total		300	701	1001

Source: Author's production.

Table 5.5 – Confusion matrix parameters for $[0]_8$ plates (100 times runs).

Parameters	Accelerance values	Real part values
Recall	29.7%	98.3%
Specificity	87.3%	91.4%
Accuracy	70.0%	93.5%
Matthews correlation coefficient (MCC)	0.20	0.86
Efficiency	58.5%	94.9%

Source: Author's production.

It can be noted that using the data from real part values returns better results than if using accelerance values, as observed previously. The data from accelerance values has more false-negatives (FN) than true-positives (TP), *i.e.*, the classifier is not good to predict damaged conditions, since only 29.7% was correctly classified as damaged. The classifier is able to hit 87.3% of the cases of absence of the damage condition (specificity), and 12.7% of false alarms. The Matthew correlation coefficient - MCC - is much smaller than +1, showing almost an average random prediction. Taking into account the data balanced, the efficiency of the classifier is 58.5%. On the other side, data from real part values brings more positive results. The classifier is 98.3% able to predict TP conditions, where the damage is present. The MCC is almost one, showing an excellent prediction and the efficiency is 94.9%.

The best results achieved after 100 times runs for both simulations are shown on Tab. 5.6.

Table 5.6 – ANNs best results for $[0]_8$ plates.

Simulation cases	Training iterations	Training accuracy (%)	Validation accuracy (%)	Testing accuracy (%)
Accelerance	300	100.0	83.3	90.0
Real part	150	100.0	100.0	100.0

Source: Author's production.

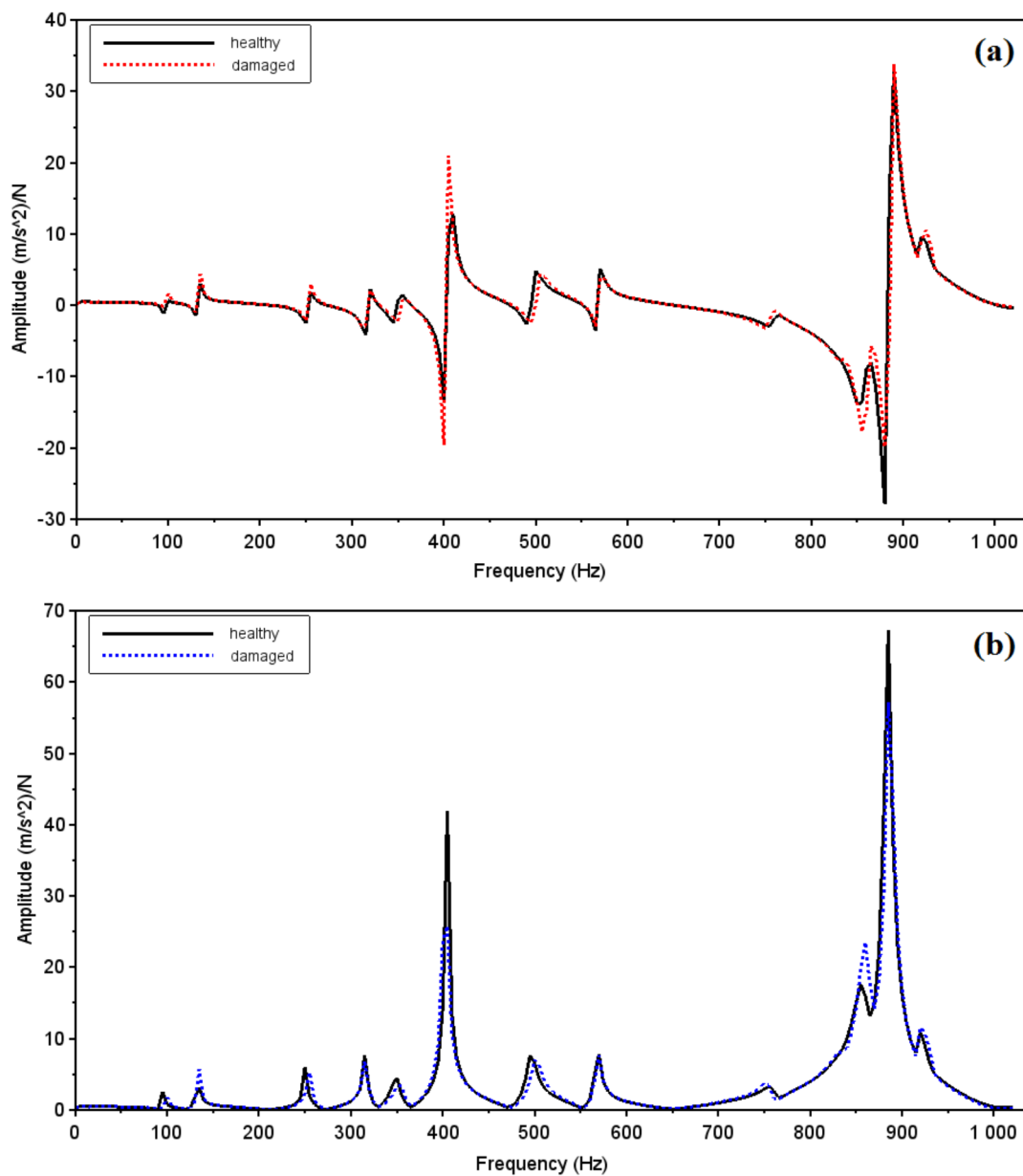
5.1.2 Composite plates with stacking orientation $[0/15/-15/0/15/-15]_S$

To verify the performance of the methodology when the differences between healthy and damaged structures are small, another stacking orientation is studied.

5.1.2.1 Dynamic tests

The real part values of FRFs are shown in Fig. 5.6 (a), and (b), the accelerance values of the same FRFs. As can be seen, there is a small difference between the resonant frequencies in some modes only. To visualize a whole set of FRFs and see the differences between the samples, FRFs curves of all plates are shown in the Appendix A, in Fig. A.3 for accelerance values and in Fig. A.4 for real part values.

Figure 5.6 – FRFs from healthy and damaged plate for $[0/15/-15/0/15/-15]_S$ (a) real part and (b) accelerance.



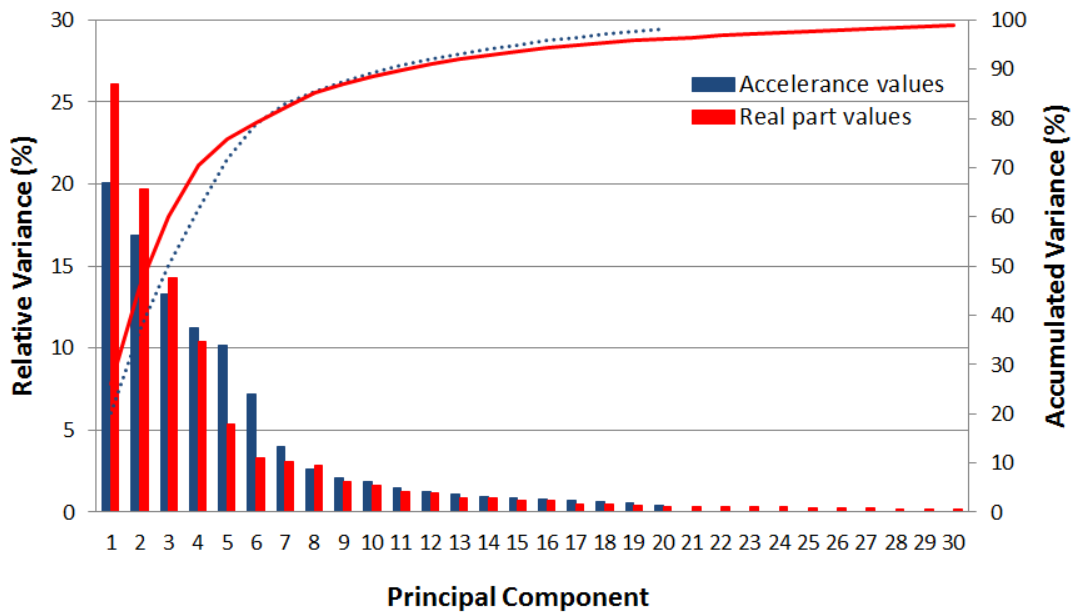
Source: Author's production.

5.1.2.2 Principal component analysis

The relative and the accumulated variance for the first PCs are shown in Fig. 5.7. Considering the accelerance values, the first 20 PCs bring 98.11% and for real part values, the first 30 PCs bring 98.85% of the total variance. As noted, at this time, to recover the approximate value of the total variance the data from real part values need more PCs than the data from accelerance values.

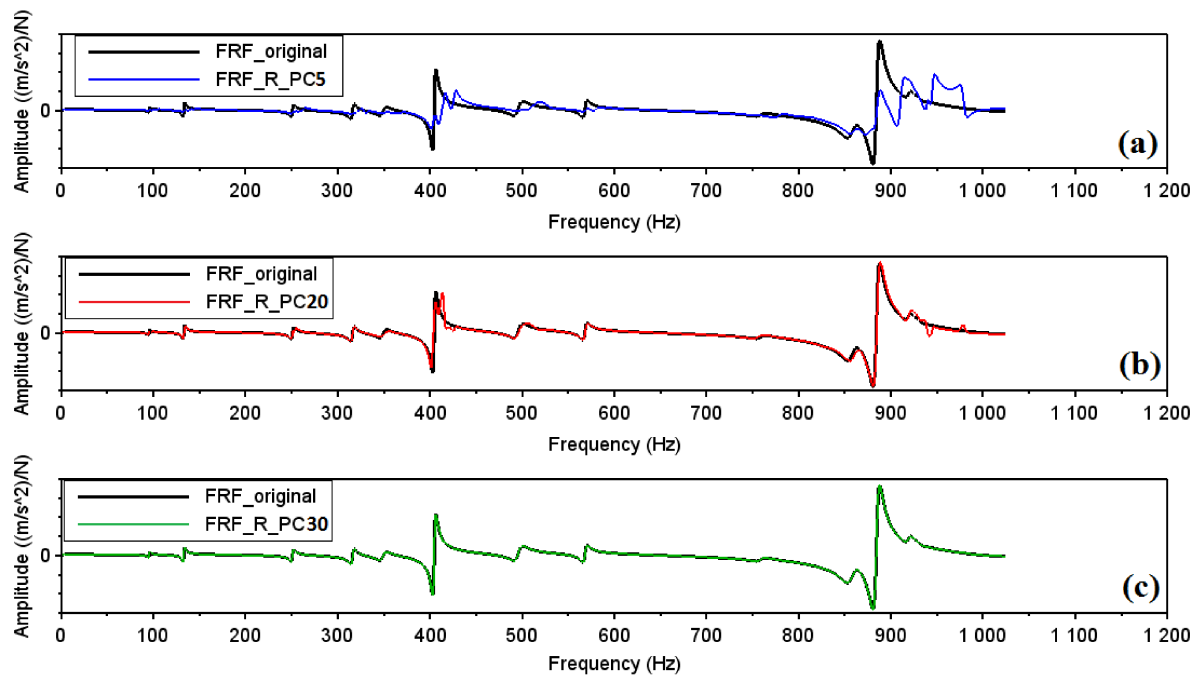
Figure 5.8 shows the original FRFs and the reconstructed FRFs using the first 5 PCs (75.90% of variance), the first 20 PCs (96.18% of variance) and the first 30 PCs, when working with real part values. It can be seen a correlation among the total variance and the FRF reconstruction. In the same way, but for accelerance values, Fig. 5.9 show the original FRFs comparing with reconstructed FRFs with the first 5 PCs (71.70% of variance) and with the first 20 PCs.

Figure 5.7 – Relative variance and accumulated variance using accelerance values and real part values ($[0/15/-15/0/15/-15]_S$).



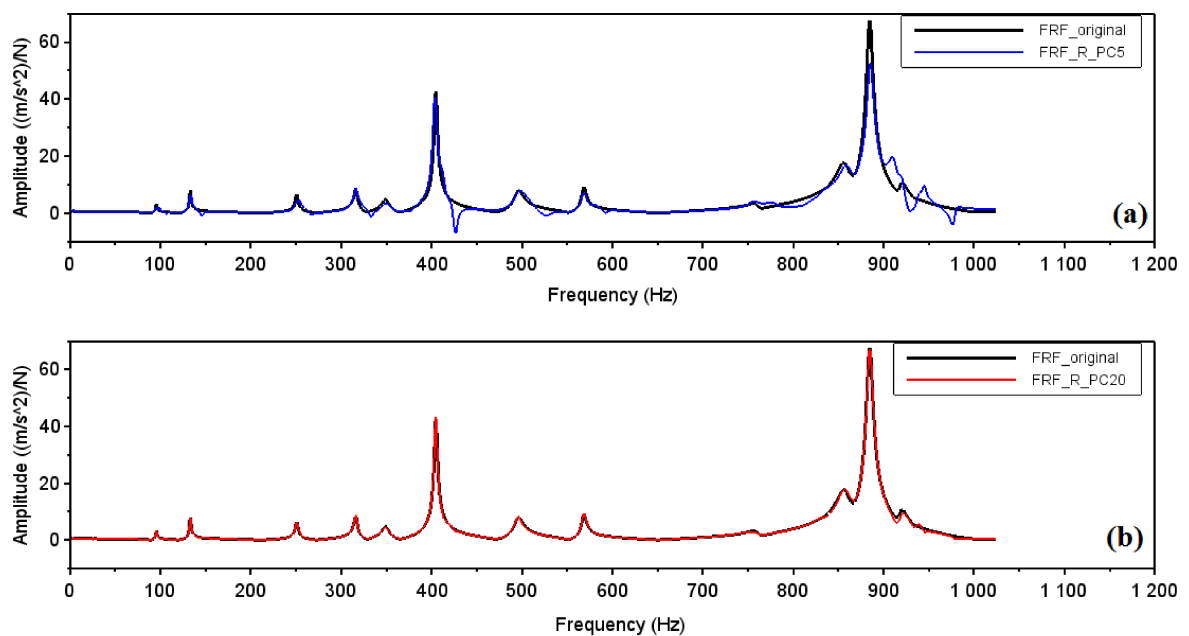
Source: Author's production.

Figure 5.8 – FRFs reconstructed with 5 PCs (a), 20 PCs (b) and 30 PCs (c) comparing with original FRF: real part values ($[0/15/-15/0/15/-15]_S$).



Source: Author's production.

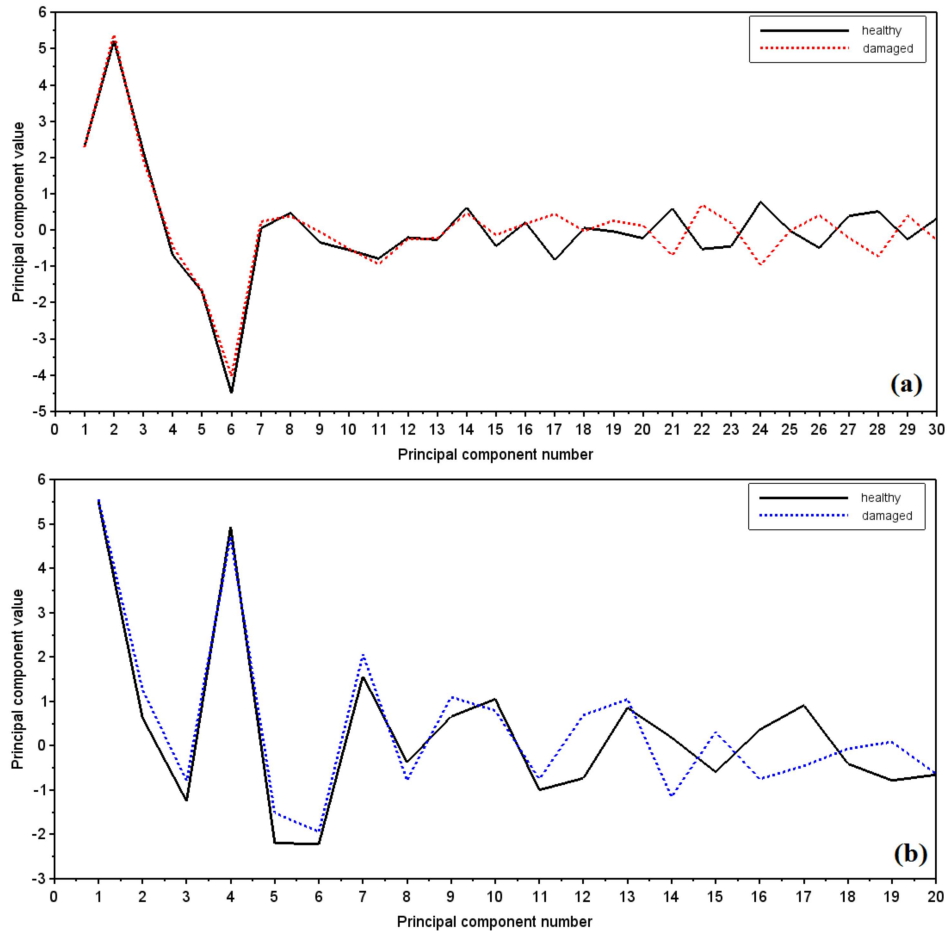
Figure 5.9 – FRFs reconstructed with 5 PCs (a) and 20 PCs (b) comparing with original FRF: accelerance values ($[0/15/-15/0/15/-15]_S$).



Source: Author's production.

Figure 5.10 shows the PCs curves for the two cases, healthy and damaged, using real part values (a) and (b) accelerance values of the FRFs. It can be noted, differently than the previous plate [0]₈, here for the first PCs, the curves between healthy and damaged are almost identical. In Appendix A, Fig. A.7 shows all the PCs curves for accelerance values and Fig. A.8 for real part values.

Figure 5.10 – PCs curves using real part values (a) and accelerance (b) - ([0/15/-15/0/15/-15]_S).



Source: Author's production.

5.1.2.3 Artificial neural networks and pattern recognition

The data is split into 62% for the training set, 21% for the validation set and 17% for the testing set. First, the PSO program is run to get the best topology maximizing the accuracy of the validation set. The PSO parameters used are the same as shown in Tab. 5.1, except for the upper bound variables values when using real part values, as the number of the inputs are greater (30 PCs) the upper bound values are [2;36;36].

The PSO results for accelerance values is $X_{top}=[1.07;11.94;12.10]$ with 80.00% of accuracy in the validation set and for real part values is $X_{top}=[1.84;19.91;9.55]$ with 70.0% of accuracy in the validation set.

The first simulation is performed using only accelerance values, the ANN topology is [20-(12)-1], *i.e.*, 20 inputs, 12 neurons for the hidden layer and one output neuron (0 for healthy and 1 for damaged). The logistic function is used as activation function and L_2 function as a cost function. Steepest descent and backtracking line search are used for the gradient direction calculations. Also, the momentum term of 0.7 is used.

The second simulation is performed using only real part values of FRFs, the topology consists of 30 inputs, 20 neurons in the first hidden layer, 10 neurons in the second hidden layer and one output neuron – [30-(20,10)-1]. The logistic function is used as activation function and cross-entropy function as a cost function. Steepest descent, fixed learning rate (0.1) and also the momentum term (0.1) are used. The mean results for both simulations, after 100 times runs, are shown on Tab. 5.7. For the training set, both simulations achieved 100% of accuracy; for the validation set, using real part values performed better than accelerance values. Although, for the testing set simulation using accelerance values the accuracy is 94.10%, better than real part values.

Table 5.7 – ANNs simulations summary results for $[0/15/-15/0/15/-15]_S$ plates (100 times runs).

Data set	Accelerance values	Real part values
Training: mean accuracy (mean deviation)	100.00% (0.0)	100.00% (0.0)
Validation: mean accuracy (mean deviation)	62.50% (0.078)	70.00% (0.041)
Testing: mean accuracy (mean deviation)	94.10% (0.065)	73.75% (0.027)

Source: Author's production.

The confusion matrix, for accelerance values after 100 times runs, is shown in Tab. 5.8, and for real part values in Tab. 5.9. Also, their parameter's values in Tab. 5.10. Opposite to the results obtained from the previous plates ($[0]_8$), the data from accelerance values show better results than real part values of FRF. With 88.2% of the capacity to predict correctly the samples with damaged against 47.5% for real part values. Also, for accuracy, MCC and efficiency. The accelerance values performed superior results when compared with real part values. The reason may be that when using accelerance values some characteristics can help the network to learn some important relationship between a certain class and its main features. Another reason may be that with the accelerance values the region becomes less non-convex, being easier the separation between classes. Can be observed in Fig. 5.10 that the difference be-

tween healthy curve and damaged curve using accelerance values is around the fifth principal component. On the other hand using real part values this difference can only be seen around principal component 15. So using accelerance values the differences between the classes can be seen more quickly. The MCC get close to +1, showing the excellent prediction. One important thing to note is that there is no misunderstanding between the actual healthy class and predicted damaged class, that is, the specificity is 100% for both simulations.

Table 5.8 – Confusion matrix after 100 runs [0/15/-15/0/15/-15]_S plates (accelerance values).

		Actual Class		Total
		Damaged	Healthy	
Predict Class	Damaged	353	0	353
	Healthy	47	400	447
Total		400	400	800

Source: Author's production.

Table 5.9 – Confusion matrix after 100 runs [0/15/-15/0/15/-15]_S plates (real part values).

		Actual Class		Total
		Damaged	Healthy	
Predict Class	Damaged	190	0	190
	Healthy	210	400	610
Total		400	400	800

Source: Author's production.

Table 5.10 – Confusion matrix parameters for [0/15/-15/0/15/-15]_S plates (100 times run).

Parameters	Accelerance values	Real part values
Recall	88.2%	47.5%
Specificity	100.0 %	100.0%
Accuracy	94.1%	73.75%
Matthews correlation coefficient (MCC)	0.89	0.56
Efficiency	94.1%	73.75%

Source: Author's production.

The best results achieved after 100 times runs for both simulations are shown on Tab. 5.11. With just 150 iterations accelerance values get maximum accuracy for validation set of 90% and 100% of generalization.

Table 5.11 – ANNs best results for $[0/15/-15/0/15/-15]_S$ plates.

Simulation cases	Training iterations	Training accuracy (%)	Validation accuracy (%)	Testing accuracy (%)
Accelerance	150	100.0	90.0	100.0
Real part	400	100.0	80.0	87.7

Source: Author's production.

Finally, as observed, all the ANNs showed good learning and generalization. The best result for each orientation agreed to what is observed in the originals FRFs. Large differences can be observed when using the real parts and accelerance values for $[0]_8$ stacking orientation. During ANN simulations, it is verified that when using only real parts values as input the behavior of the network is better than when using accelerance values. It is probably due to the fact that in the real part of FRFs there are phase shifts in almost all the modes between intact and damaged plates. This situation is hidden when working with accelerance values. The phase shifts, differentiating the intact from the damaged ones, can contribute to feature extraction during the learning and then to the classification performed by the network. For $[0/15/-15/0/15/-15]_S$ laminates, the best result is obtained when the accelerance values are used. This is probably due to the fact that values in accelerance carry more information about each integrity condition. As the FRFs are similar, the features acquired only by the real part created a more complex decision boundary between healthy and damaged cases. Transforming the features in accelerance values, the curves changed, differentiating the dynamic behavior between them. Due to this feature addition, the network begins to rank better and evaluates the presence or not of the damage and thus make its decision correctly.

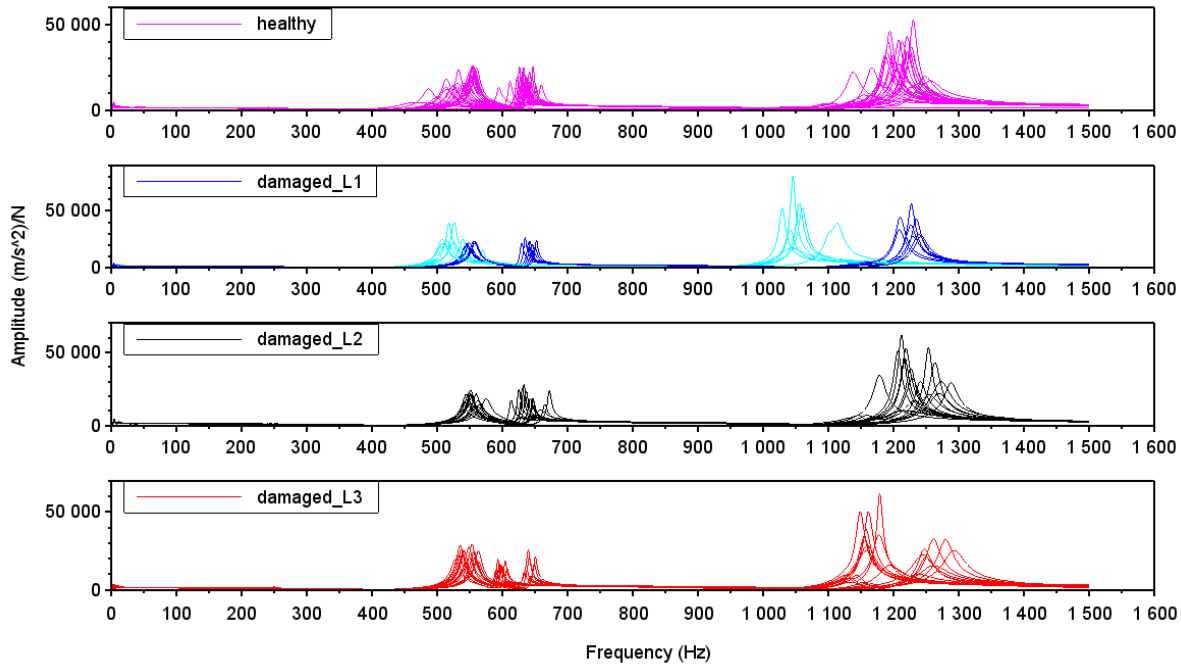
5.2 Damage Detection in Glass-Fiber/Epoxy Composite Beams: Experimental

5.2.1 Dynamic and modal analysis

A total 25 beams are manufactured without damage (H), 15 beams with 5 mm delamination size (Level 1: D- L_1), 16 beams with 10 mm delamination size (Level 2: D- L_2) and 17 beams with 19 mm delamination size (Level 3: D- L_3). As the vibration-based tests are performed in two points of each beam, there are in total 73 FRFs for

each position (H_{11} and H_{21}). The H_{11} - FRFs can be seen in Fig. 5.11 and H_{21} - FRFs in Fig. 5.12, both using accelerance FRFs. In Appendix A the Fig. A.9 shows the FRFs from all damaged cases from position H_{11} and Fig. A.10 from position H_{21} .

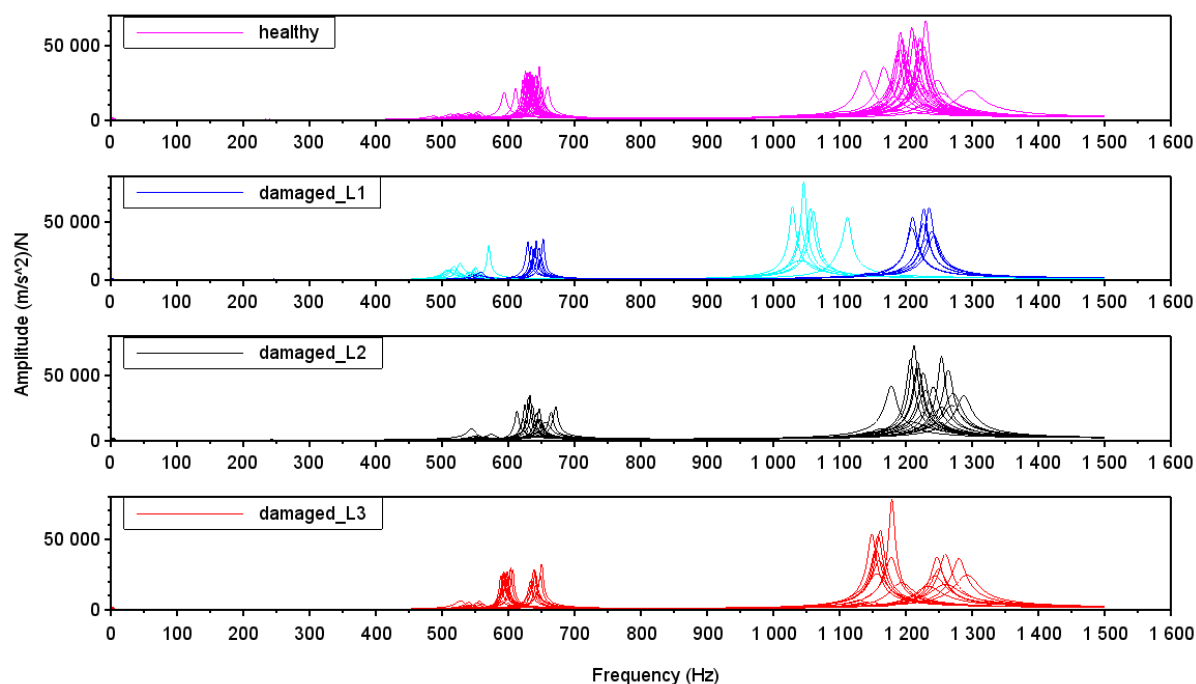
Figure 5.11 – H_{11} - FRFs curves (accelerance values).



Source: Author's production.

As one of the four-step of statistical pattern recognition, the feature extraction and selection has been a very important and critical procedure to get a very good generalization in classification. Taking this into account and observing the FRFs from the dynamics tests, it can be observed that some FRFs from damaged Level 1 (D- L_1) are distinct from the pattern. They are signalized with a light blue color curves. Because this difference in the pattern those curves are removed from the dataset. The reason is the difference in the dimensions of the beams, in thickness, due to the manufacturing process.

The range between the minimum frequency and maximum frequency, for the first five mode shapes from the four states conditions are shown in Fig. 5.13. It may be noted that the interval of each mode for a particular state is large. And when compared to the mode and the states it is noted that many samples are within the same range. Thus, the separation of the classes becomes more difficult, since the resonant frequencies are very close to each other when compared to the structural conditions of the system. Figure 5.14 shows an example of undamaged-FRF (accelerance and phase) and the coherence from the positions H_{11} and H_{21} .

Figure 5.12 – H_{21} - FRFs curves (accelerance values).

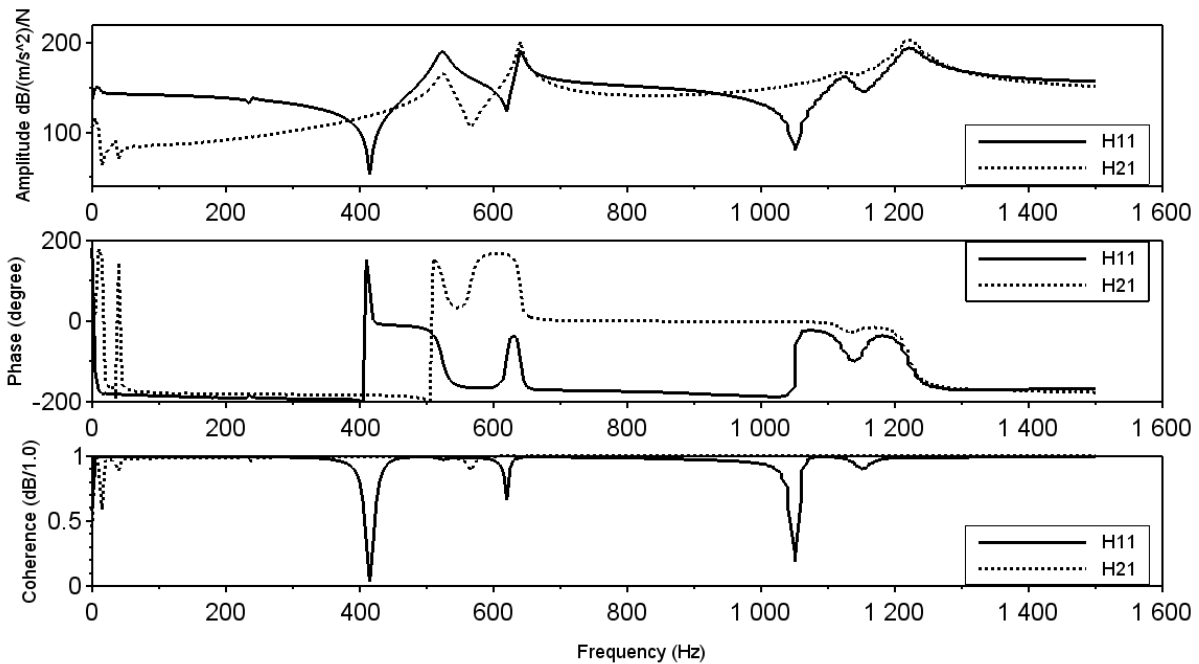
Source: Author's production.

Figure 5.13 – Frequency range (Hz) for the first five mode shapes from the four damage states (H_{11}).

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
Damaged 3	231 250	534 563.5	593 651.5	1125 1167	1149 1294
Damaged 2	237 256	542.5 575.5	613.5 672.5	1147 1186	1179 1289
Damaged 1	239.5 247	545.5 558.5	630.5 652	1147 1170	1209.5 1242
Healthy	223 251	467 572	594.5 659.5	1048 1201	1138 1300

Source: Author's production.

Results for the modal analysis performed in one undamaged beam are shown in Tab. 5.12.

Figure 5.14 – Positions H_{11} and H_{21} - FRFs curves (Beam 6).

Source: Author's production.

Table 5.12 – Modal analysis results - flexural modes (Beam 3).

Flexural Mode	Frequency (Hz)
1	232.5
2	644.5
3	1253.5

Source: Author's production.

5.2.2 Principal component analysis

The number of spectral lines, 1500, is very large. Thus, the PCA is used in order to reduce the number of inputs and also to reduce the complexity of the input data.

First, to analyze if there is some influence in performing PCA in the complete dataset or just in the partial dataset, the two approaches are performed, as explained in Fig. 4.18 in the previous chapter. Fast Fourier Transform (FFT) curves are used to perform the calculations and the first 10 PCs are used for both approaches. In order to evaluate each approach without the influence of the samples from a particular region - such as the border samples - they are not contained in another set of the other approach, the same FFT curves are used for both approaches in the training set, as well in the validation set and in the testing set. The ANN topology is 10 inputs, 8 and 8 neurons for the first and second hidden layers and 4 outputs. The logistic function and

L_2 cost function are used as well a step size of 0.1 and a momentum term of 0.5. Table 5.13 shown the results after 500 times runs.

Table 5.13 – Summary results for PCA-total and PCA-partial using FFT curves (500 times runs).

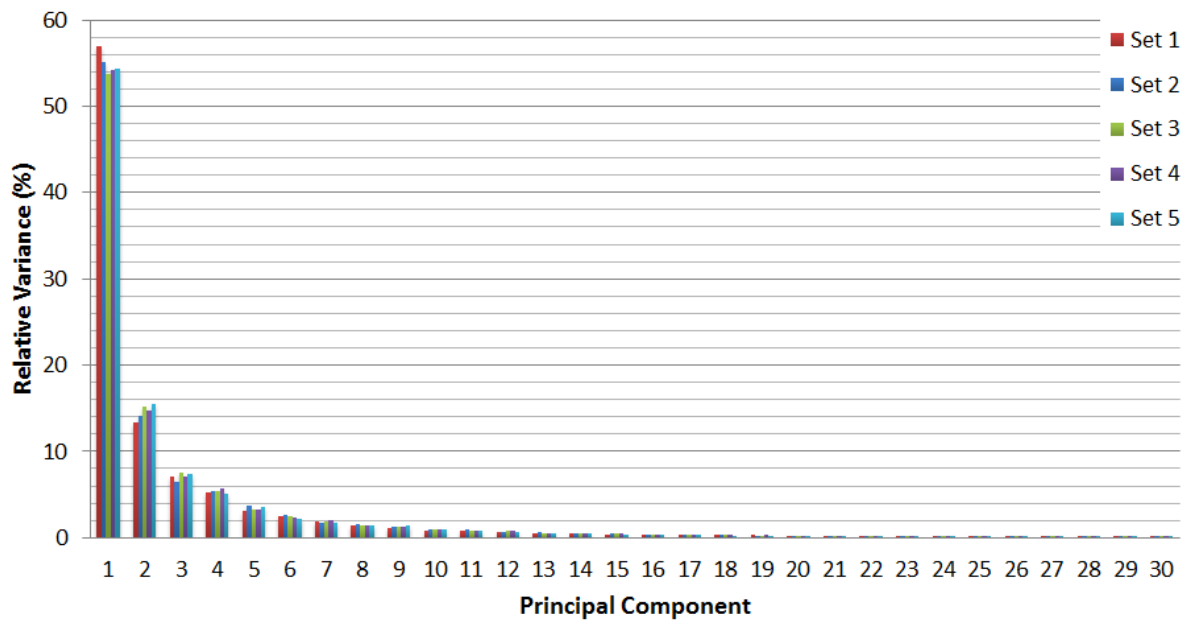
Data set		Approach 1 PCA-total	Approach 2 PCA-partial
Training:	mean accuracy (mean deviation)	94.0% (0.015)	92.0% (0.018)
	minimum accuracy	87.0%	80.0%
	maximum accuracy	99.0%	98.0%
Validation:	mean accuracy (mean deviation)	41.1% (0.076)	41.1% (0.072)
	minimum accuracy	23.5%	17.6%
	maximum accuracy	76.5%	76.5%
Testing:	mean accuracy (mean deviation)	55.2% (0.061)	55.2% (0.057)
	minimum accuracy	24.1%	31.0%
	maximum accuracy	72.4%	79.3%

Source: Author's production.

The results show that, apparently, there is no leakage of information from the training set to the other sets when the PCA is performed in the whole dataset prior to the split step. Since the mean accuracies for all the datasets are the same for both approaches. The small differences between the minimum and the maximum accuracies achieved may be due to the initialization region of the weights and bias of a given iteration.

The accelerance of all 132 FRFs are split into two datasets called training set and testing set using 5-fold cross-validation. The testing set contains approximately 20% of the samples. After that, the training set is again split into two groups, training set, and validation set, where 15% of the samples go to the validation set. After that, the datasets for each fold are introduced to PCA. The relative variance for each fold is shown in Fig. 5.15.

Figure 5.15 – Relative variance (%) for the 5-fold for the first 30 PCs .



Source: Author's production.

Table 5.14 shows the total variance from the 5-folds (sets) using the first 20, 25 and 30 PCs.

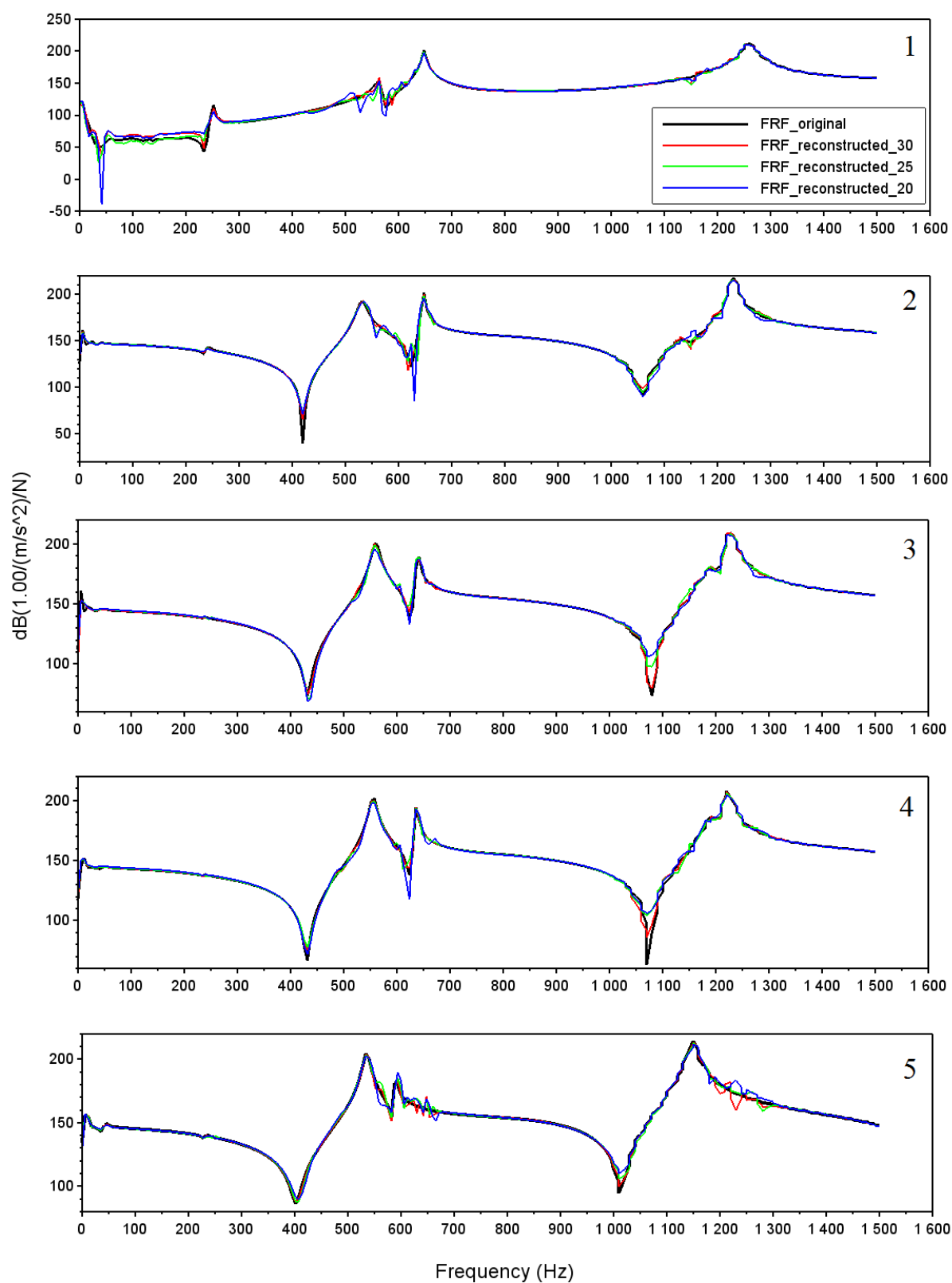
Table 5.14 – Total variance for each 5-folds with 20, 25 and 30 PCs.

Set Number	Total variance (%)	Total variance (%)	Total variance (%)
	20 PCs	25 PCs	30 PCs
Set 1	97.935	98.767	99.236
Set 2	97.706	98.663	99.183
Set 3	97.754	98.678	99.220
Set 4	97.708	98.703	99.232
Set 5	97.922	98.731	99.213

Source: Author's production.

The original and reconstructed FRFs with 20, 25 and 30 PCs are shown in Fig. 5.16 for the 5-folds.

Figure 5.16 – FRFs reconstructed using 20, 25 and 30 PCs for the 5-folds (sets).



Source: Author's production.

5.2.3 Artificial neural networks and pattern recognition: Case I

The first case studied is when the samples introduced in the ANN are all from the four state conditions classes: healthy, damaged level 1, damaged level 2 and damaged level 3 with 2 neurons in the output layer: healthy and damaged.

PSO is used to find the approximate topology of the ANN, in which the program finds the best results, maximizing the accuracy of the validation set. The PSO parameters used are shown in Tab. 5.15.

Table 5.15 – PSO algorithm parameters.

Parameter	Value
C_1	1.2
C_2	1.2
w	0.5
Iterations	5
Particles number	100
Variables number	3
Lower bound values	[1;1;1]
Upper bound values	[2;34;34] for 30 PCs [2;25;22] for 20 and 25 PCs

Source: Author's production.

The best PSO-result for 30 inputs is $X_{top}=[1.68;22.42;16.76]$ with 75.0% of accuracy, for 25 inputs is $X_{top}=[1.77;1.37;6.74]$ with 81.2% of accuracy and for 20 inputs is $X_{top}=[1.89;13.58;17.66]$ with 75.0% of accuracy.

Table 5.16 shows several ANNs simulations with different parameters varying the topology, the activation function, the cost function, the fixed size or variable step size (Backtracking line search), the noise, the regularization technique, the momentum term and if the training data is shuffle or not. The term shuffle refers that for each iteration X% of the training set is used for the training.

Table 5.17 shows the results for the 1st set, *i.e.*, the 1st-fold, for all simulations presented in Tab. 5.16 after 150 times runs. As the accuracy from the training set is high (around 95%) there is no underfitting indication. Table 5.18 shows the four best accuracy results achieved using the other sets (2nd, 3rd, 4th and 5th). It can be noted that set 4 presents the worst results and set 5 the best results.

Table 5.16 – ANNs simulations with different parameters - Case I.

ANN	Topology	Activation function	Cost function	Step size	γ	Regularization technique	Noise	Shuffle samples
ANN_1	30-[22,17]-2	tanh	quadratic	BK	0.3	L_2 (0.001)	no	no
ANN_2	30-[22,17]-2	logistic	2-norm	BK	0.3	no	no	no
ANN_3	30-[22,17]-2	Leaky-ReLU	quadratic	BK	0.3	L_2 (0.001)	no	no
ANN_4	30-[22,17]-2	tanh	quadratic	BK	0.3	no	5 %	55 %
ANN_5	30-[22,17]-2	logistic	2-norm	0.05	0.3	no	no	no
ANN_6	30-[22,17]-2	logistic + softmax	cross-entropy	BK	0.3	no	no	no
ANN_7	30-[22,17]-2	logistic + softmax	cross-entropy	BK	0.3	no	no	55 %
ANN_8	30-[22,17]-2	logistic + softmax	cross-entropy	BK	0.3	no	10 %	55 %
ANN_9	30-[22,17]-2	logistic + softmax	cross-entropy	0.05	0.3	no	10 %	55 %
ANN_{10}	30-[22,17]-2	logistic + softmax	cross-entropy	0.1	0.3	no	10 %	55 %
ANN_{11}	30-[22,17]-2	logistic + softmax	cross-entropy	BK	0.3	no	10 %	55 %
ANN_{12}	30-[22,17]-2	logistic + softmax	2-norm	BK	0.3	no	10 %	55 %
ANN_{13}	25-[1,7]-2	logistic + softmax	cross-entropy	BK	0.3	no	no %	55 %
ANN_{14}	25-[1,7]-2	tanh + softmax	quadratic	BK	0.01	L_2 (0.001)	no	55 %
ANN_{15}	25-[1,7]-2	tanh + softmax	quadratic	BK	0.01	L_2 (0.001)	5 %	55 %
ANN_{16}	25-[1,7]-2	tanh	quadratic	BK	0.3	L_2 (0.001)	no	no
ANN_{17}	25-[30,10]-2	tanh + softmax	quadratic	BK	0.3	no	no	55 %
ANN_{18}	25-[30,10]-2	tanh + softmax	quadratic	BK	0.3	L_1 (0.001)	no	55 %
ANN_{19}	25-[30,10]-2	tanh + softmax	quadratic	BK	0.01	L_1 (0.001)	no	55 %
ANN_{20}	25-[30,10]-2	tanh + softmax	quadratic	BK	0.01	L_1 (0.0001)	no	55 %
ANN_{21}	25-[30,10]-2	tanh + softmax	quadratic	BK	0.01	L_1 (0.01)	no	55 %
ANN_{22}	25-[30,10]-2	tanh + softmax	quadratic	BK	0.01	L_1 (0.1)	no	55 %
ANN_{23}	25-[30,10]-2	tanh + softmax	quadratic	BK	0.01	L_2 (0.0001)	no	55 %
ANN_{24}	25-[30,10]-2	tanh + softmax	quadratic	BK	0.01	L_2 (0.001)	no	55 %
ANN_{25}	25-[30,10]-2	tanh + softmax	quadratic	BK	0.01	L_2 (0.01)	no	55 %
ANN_{26}	25-[30,10]-2	tanh + softmax	quadratic	BK	0.01	L_2 (0.01)	5 %	77 %
ANN_{27}	25-[30,10]-2	tanh + softmax	quadratic	BK	0.01	L_2 (0.01)	no	77 %
ANN_{28}	25-[30,10]-2	logistic + softmax	cross-entropy	BK	0.3	no	no	55 %
ANN_{29}	25-[30,10]-2	tanh + softmax	cross-entropy	BK	0.01	L_2 (0.001)	no	55 %

BK=backtracking line search; γ =momentum term.

Source: Author's production.

Table 5.17 – ANNs simulations with different parameters - results (Set 1) - 150 times run - Case I.

ANN	Training set		Validation set		Testing set	
	Mean - % (mean deviation)	Maximum	Mean - % (mean deviation)	Maximum	Mean - % (mean deviation)	Maximum
ANN_1	92.13 (0.062)	98.88	50.00 (0.041)	62.50	44.44 (0.057)	62.96
ANN_2	75.28 (0.029)	96.63	37.50 (0.054)	62.50	37.04 (0.037)	55.55
ANN_3	94.39 (0.081)	98.88	43.75 (0.067)	62.50	37.04 (0.060)	59.26
ANN_4	96.00 (0.087)	100.00	56.25 (0.047)	68.75	57.41 (0.055)	70.37
ANN_5	89.89 (0.026)	94.38	50.00 (0.025)	50.00	44.44 (0.033)	59.26
ANN_6	93.26 (0.097)	97.75	56.25 (0.072)	62.50	62.96 (0.073)	70.37
ANN_7	96.00 (0.063)	100.00	68.75 (0.051)	75.00	59.26 (0.056)	70.37
ANN_8	96.00 (0.060)	100.00	68.75 (0.054)	75.00	62.96 (0.044)	74.07
ANN_9	92.00 (0.011)	96.00	68.75 (0.028)	75.00	62.96 (0.026)	66.67
ANN_{10}	92.00 (0.012)	96.00	68.75 (0.031)	75.00	59.26 (0.023)	66.67
ANN_{11}	96.00 (0.103)	100.00	68.75 (0.055)	75.00	62.96 (0.048)	74.07
ANN_{12}	92.00 (0.100)	100.00	65.62 (0.056)	75.00	55.56 (0.036)	70.37
ANN_{13}	86.00 (0.126)	96.00	62.50 (0.101)	75.00	59.26 (0.080)	74.07
ANN_{14}	68.00 (0.115)	96.00	62.50 (0.074)	75.00	55.56 (0.050)	74.07
ANN_{15}	68.00 (0.113)	96.00	62.50 (0.072)	75.00	55.56 (0.048)	70.37
ANN_{16}	83.15 (0.206)	95.50	50.00 (0.135)	68.75	55.56 (0.131)	70.37
ANN_{17}	84.00 (0.076)	100.00	62.50 (0.065)	81.25	59.26 (0.050)	74.07
ANN_{18}	84.00 (0.081)	100.00	62.50 (0.061)	75.00	59.26 (0.053)	74.07
ANN_{19}	90.00 (0.078)	100.00	62.50 (0.066)	75.00	62.96 (0.048)	77.78
ANN_{20}	84.00 (0.077)	100.00	62.50 (0.057)	75.00	59.26 (0.044)	74.07
ANN_{21}	90.00 (0.089)	100.00	62.50 (0.062)	75.00	59.26 (0.054)	74.07
ANN_{22}	80.00 (0.112)	98.00	62.50 (0.061)	75.00	59.25 (0.054)	77.78
ANN_{23}	84.00 (0.071)	100.00	62.50 (0.061)	75.00	62.96 (0.041)	77.78
ANN_{24}	86.00 (0.074)	100.00	65.62 (0.061)	75.00	62.96 (0.053)	74.07
ANN_{25}	90.00 (0.080)	100.00	68.75 (0.060)	75.00	66.67 (0.065)	74.07
ANN_{26}	85.71 (0.076)	100.00	56.25 (0.047)	75.00	62.96 (0.050)	74.07
ANN_{27}	85.71 (0.058)	100.00	56.25 (0.047)	75.00	59.26 (0.040)	70.37
ANN_{28}	94.00 (0.111)	100.00	68.75 (0.054)	75.00	62.96 (0.063)	74.07
ANN_{29}	100.00 (0.146)	100.00	62.50 (0.100)	75.00	62.96 (0.103)	74.07

Source: Author's production.

Table 5.18 – ANNs simulations with different parameters - accuracy results (Sets 2, 3, 4 and 5) - 150 times run - Case I.

Set	Dataset	ANN_6		ANN_8		ANN_9		ANN_{24}	
		Mean - % (mean deviation)	Maximum	Mean - % (mean deviation)	Maximum	Mean - % (mean deviation)	Maximum	Mean - % (mean deviation)	Maximum
2	Training	90.00 (0.065)	97.78	100.00 (0.014)	100.00	98.00 (0.006)	100.00	100.00 (0.034)	100.00
	Validation	60.00 (0.071)	80.00	53.33 (0.035)	66.67	53.33 (0.025)	60.00	53.33 (0.064)	66.67
	Testing	62.93 (0.047)	74.07	55.56 (0.036)	62.96	59.26 (0.022)	62.96	59.26 (0.049)	66.67
3	Training	92.22 (0.098)	98.89	94.00 (0.111)	100.00	96.00 (0.019)	100.00	88.00 (0.070)	98.00
	Validation	56.25 (0.048)	62.50	50.00 (0.081)	68.75	56.25 (0.060)	68.75	56.25 (0.091)	75.00
	Testing	69.23 (0.073)	84.61	61.54 (0.100)	88.46	73.08 (0.052)	88.46	76.92 (0.147)	92.31
4	Training	87.78 (0.108)	94.44	98.00 (0.067)	100.00	98.00 (0.0012)	98.00	96.00 (0.126)	100.00
	Validation	37.50 (0.052)	75.00	43.75 (0.040)	62.50	43.75 (0.035)	56.25	43.74 (0.070)	62.50
	Testing	53.85 (0.056)	61.54	61.54 (0.050)	69.23	57.70 (0.030)	65.39	73.08 (0.064)	86.61
5	Training	86.67 (0.100)	97.78	92.00 (0.109)	100.00	96.00 (0.022)	98.00	92.00 (0.076)	100.00
	Validation	62.50 (0.063)	81.25	62.50 (0.057)	81.25	68.75 (0.041)	75.00	68.75 (0.100)	81.25
	Testing	65.38 (0.095)	80.77	65.38 (0.105)	76.92	69.23 (0.026)	76.92	71.15 (0.124)	84.61
<i>Total</i>	Training	90.00 (0.022)	98.89	96.00 (0.024)	100.00	96.00 (0.005)	100.0	92.00 (0.045)	100.00
	Validation	56.25 (0.068)	81.25	53.33 (0.054)	81.25	56.25 (0.010)	75.00	56.25 (0.085)	81.25
	Testing	62.96 (0.036)	84.61	61.54 (0.024)	88.46	62.96 (0.008)	88.46	71.15 (0.052)	92.31

Source: Author's production.

The results presented previously show the average accuracy of each dataset (training, validation, and testing) for each set of samples (5-folds), and the maximum accuracy obtained in the total of 150 external iterations. Although the maximum accuracy showed good results, they do not represent the same internal iteration of the ANN, that is, the best values found among all the simulations for a network architecture in the same internal iteration are shown in Tab. 5.19, which corresponds to ANN_{24} . The time spending for training the ANN is around 16 minutes and 44 seconds, which it depends on the topology, the iteration numbers, the learning rate and the size of the dataset. The CPU used to calculate the training time is Intel Core 2 Duo 3.0 GHz. The generalization accuracy of the ANN is 84.61% for set 3 and 88.46% for set 5.

Table 5.19 – ANN two best accuracy results (ANN_{24} - Set 3 and 5) - Case I.

Set number	Training time	Training accuracy (%)	Validation accuracy (%)	Testing accuracy (%)
Set 3	16'44"	100.0	56.25	92.31
Set 5	16'44"	98.00	81.25	84.61

Source: Author's production.

The confusion matrix for the best result is shown in Tab. 5.20 for set 3 and in Tab. 5.21 for set 5, as well as its parameters in Tab. 5.22. For set 3 recall is 100.00% with no false negative alarms. For the MCC parameter, the best value presented is for set 3, since the error between the classifications is smaller. Because of this both the accuracy and the efficiency for the set 3 is also higher.

Table 5.20 – Confusion matrix for ANN_{24} - Set 3: best result - Case I.

		Actual Class		Total
		Damaged	Healthy	
Predict Class	Damaged	15	2	17
	Healthy	0	9	9
Total		15	11	26

Source: Author's production.

Table 5.21 – Confusion matrix for ANN_{24} - Set 5: best result - Case I.

		Actual Class		<i>Total</i>
		Damaged	Healthy	
Predict Class	Damaged	12	2	14
	Healthy	2	10	12
<i>Total</i>		14	12	26

Source: Author's production.

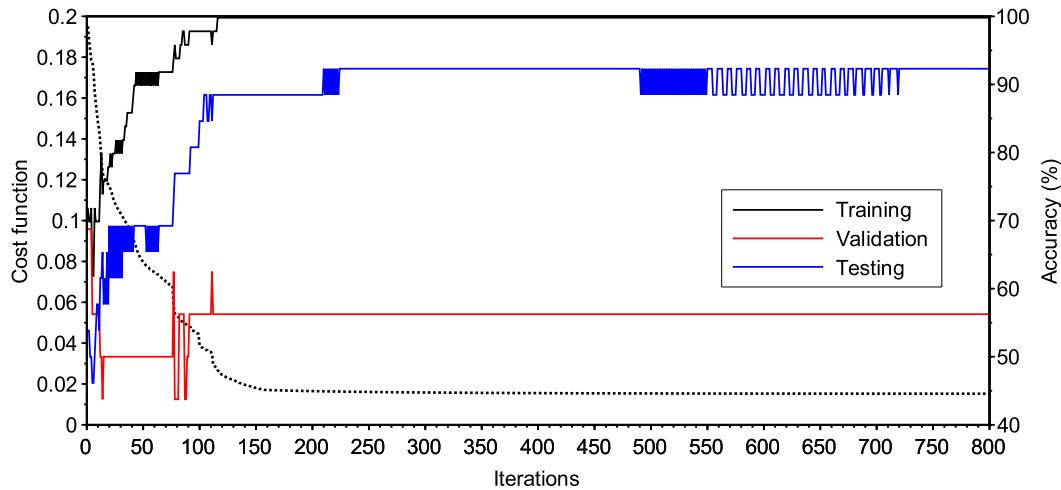
Table 5.22 – Confusion matrix parameters for ANN_{24} - Sets 3 and 5: best result - Case I.

Parameters	Values-Set 3	Values-Set 5
Recall	100.00%	85.71%
Specificity	81.82%	83.33%
Accuracy	92.31%	84.61%
Matthews correlation coefficient (MCC)	0.85	0.69
Efficiency	90.91%	83.52%

Source: Author's production.

The training error curve and the accuracy convergence curves for the three dataset are shown in Fig. 5.17, there is an oscillation in the testing set until iteration 224 and between 490 and 720. So the best is use a early stopping technique at iteration 224.

Figure 5.17 – Training error (dot line) and accuracy (%) convergence curves for ANN_{24} - Set 3: best result - Case I.



Source: Author's production.

5.2.4 Artificial neural networks and pattern recognition: Case II

The second case studied is when the dataset is composed of three state conditions: healthy, damaged level 2 and damaged level 3, with 2 output neurons (healthy and damaged). So as well as, the previous case the PCA is used and the total variance for each 5-folds are shown in Tab. 5.23 for the first 30 PCs.

Table 5.23 – Total variance for each 5-folds 30 PCs - Case II.

Set Number	Total variance (%)
Set 1	99.310
Set 2	99.367
Set 3	99.282
Set 4	99.301
Set 5	99.322

Source: Author's production.

The best PSO-result for 30 inputs is $X_{top}=[1.66;9.29;27.04]$ with 85.7% of accuracy.

Table 5.24 shows several ANNs simulations with different parameters. Table 5.25 shows the results for the 1st set, i.e., the 1st-fold, for all simulations presented in Tab. 5.24 after 150 times run. Table 5.26 shows the three best accuracy results achieved using the other sets (2nd, 3rd, 4th, and 5th).

Table 5.24 – ANNs simulations with different parameters- Case II.

ANN	Topology	Activation function	Cost function	Step size	γ	Regularization technique	Noise	Shuffle samples
ANN_1	30-[9,27]-2	tanh	quadratic	BK	0.3	L_2 (0.001)	no	no
ANN_2	30-[9,27]-2	logistic	cross-entropy	BK	0.3	no	no	no
ANN_3	30-[15,7]-2	tanh	quadratic	BK	0.3	L_2 (0.001)	no	no
ANN_4	30-[15,7]-2	tanh	quadratic	0.05	0.3	L_2 (0.001)	no	no
ANN_5	30-[15,7]-2	tanh	quadratic	0.05	0.3	L_2 (0.001)	5 %	no
ANN_6	30-[15,7]-2	logistic	2-norm	BK	0.3	no	no	no
ANN_7	30-[15,7]-2	Leaky-ReLU	2-norm	BK	0.3	no	no	no
ANN_8	30-[15,7]-2	tanh + softmax	quadratic	BK	0.3	L_2 (0.001)	no	no
ANN_9	30-[15,7]-2	tanh + softmax	cross-entropy	BK	0.3	L_2 (0.001)	no	no

BK=backtracking line search; γ =momentum term.

Source: Author's production.

Table 5.25 – ANNs simulations with different parameters - accuracy results (Set 1) - 150 times run - Case II.

ANN	Training set		Validation set		Testing set	
	Mean - % (mean deviation)	Maximum	Mean - % (mean deviation)	Maximum	Mean - % (mean deviation)	Maximum
ANN_1	98.46 (0.023)	100.00	51.33 (0.052)	71.43	68.58 (0.050)	87.50
ANN_2	92.67 (0.060)	100.00	52.05 (0.044)	64.28	65.17 (0.037)	79.17
ANN_3	99.10 (0.016)	100.00	53.00 (0.049)	71.43	68.75 (0.041)	83.33
ANN_4	99.99 (0.002)	100.00	50.29 (0.040)	64.28	72.00 (0.030)	83.33
ANN_5	100.00 (0.000)	100.00	50.95 (0.035)	64.28	71.75 (0.031)	83.33
ANN_6	88.34 (0.083)	100.00	50.06 (0.043)	64.28	58.61 (0.049)	75.00
ANN_7	83.72 (0.167)	100.00	44.09 (0.107)	71.43	43.40 (0.100)	65.22
ANN_8	96.15 (0.127)	100.00	57.74 (0.097)	78.57	70.83 (0.098)	91.67
ANN_9	98.72 (0.122)	100.00	50.06 (0.087)	71.43	70.83 (0.095)	87.50

Source: Author's production.

Table 5.26 – ANNs simulations with different parameters - accuracy results (Sets 2, 3, 4 and 5) - 150 times run - Case II.

Set	Dataset	ANN_4		ANN_5		ANN_8	
		Mean - % (mean deviation)	Maximum	Mean - % (mean deviation)	Maximum	Mean - % (mean deviation)	Maximum
2	Training	99.98 (0.000)	100.00	100.00 (0.000)	100.00	96.62 (0.115)	100.00
	Validation	59.62 (0.088)	85.57	57.48 (0.075)	92.86	50.00 (0.096)	71.43
	Testing	70.35 (0.031)	82.61	69.56 (0.031)	82.61	69.56 (0.078)	78.26
3	Training	100.00 (0.000)	100.00	100.00 (0.000)	100.00	90.51 (0.135)	100.00
	Validation	70.00 (0.065)	92.86	72.09 (0.062)	92.86	64.29 (0.168)	92.29
	Testing	69.30 (0.036)	82.61	69.83 (0.038)	78.26	69.56 (0.134)	82.61
4	Training	100.00 (0.000)	100.00	99.99 (0.000)	100.00	96.20 (0.146)	100.00
	Validation	60.19 (0.060)	78.57	57.24 (0.060)	78.57	64.29 (0.120)	78.57
	Testing	77.22 (0.041)	91.30	76.32 (0.049)	91.30	78.26 (0.115)	91.30
5	Training	99.93 (0.001)	100.00	99.92 (0.002)	100.00	94.94 (0.163)	100.00
	Validation	65.76 (0.060)	85.71	64.95 (0.053)	78.57	64.30 (0.114)	78.57
	Testing	79.87 (0.040)	85.71	80.46 (0.040)	91.30	78.26 (0.187)	95.65
<i>Total</i>	Training	<i>100.00 (0.000)</i>	<i>100.00</i>	<i>99.99 (0.000)</i>	<i>100.00</i>	<i>96.15 (0.017)</i>	<i>100.0</i>
	Validation	<i>60.19 (0.054)</i>	<i>92.86</i>	<i>57.48 (0.064)</i>	<i>92.86</i>	<i>64.29 (0.051)</i>	<i>92.86</i>
	Testing	<i>72.00 (0.038)</i>	<i>91.30</i>	<i>71.75 (0.038)</i>	<i>91.30</i>	<i>70.83 (0.040)</i>	<i>95.65</i>

Source: Author's production.

Despite the good results in the maximum accuracy between the validation and testing sets, these values do not represent the same internal iteration of the network, so the best result obtained is shown in Tab. 5.27 with 95.65% of the patterns correctly classified. The confusion matrix is shown in Tab. 5.28 as well as the confusion matrix parameters in Tab. 5.29. It can be noted that the classifier missclassifies only one sample from healthy class as a damaged class, that means, a false positive indication. The recall is 100% since no sample is indicated as false negative and the MCC is very good reaching 0.92. The training error curve and the accuracy convergence curves for the three dataset are shown in Fig. 5.18, where it can be observed that the first iterations the ANN are quite unstable and the accuracy of the testing set is oscillates well until it stabilizes after 360 iterations.

Table 5.27 – ANN best accuracy results (ANN_8 - Set 5) - Case II.

Training time	Training accuracy (%)	Validation accuracy (%)	Testing accuracy (%)
5'57"	92.40	78.57	95.65

Source: Author's production.

Table 5.28 – Confusion matrix for ANN_8 - Set 5: best result - Case II.

		Actual Class		
		Damaged	Healthy	Total
Predict Class	Damaged	10	1	11
	Healthy	0	12	12
Total		10	13	23

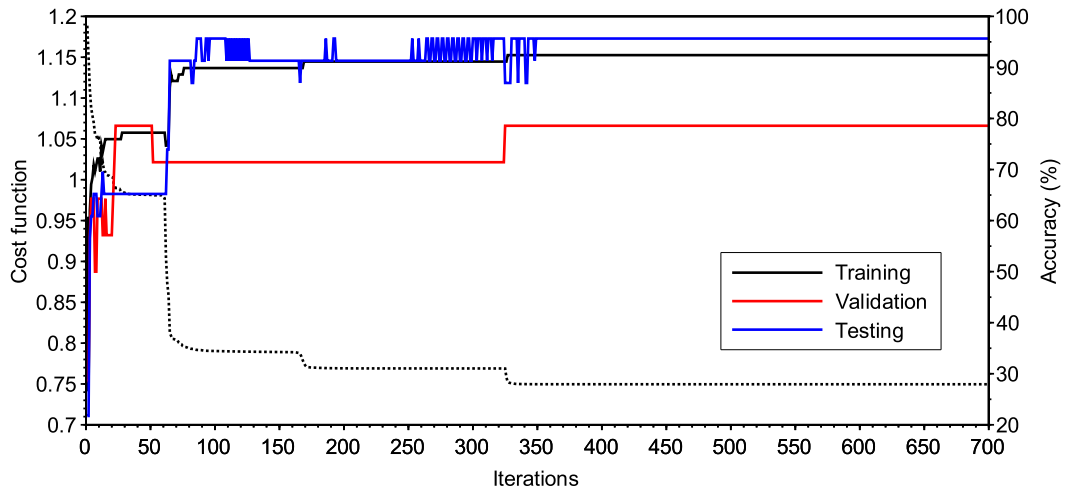
Source: Author's production.

Table 5.29 – Confusion matrix parameters for ANN_8 - Set 5: best result - Case II.

Parameters	Values
Recall	100.00%
Specificity	92.31%
Accuracy	95.65%
Matthews correlation coefficient (MCC)	0.92
Efficiency	96.15%

Source: Author's production.

Figure 5.18 – Training error (dot line) and accuracy (%) convergence curves for ANN_8 - Set 5: best result - Case II.



Source: Author's production.

5.2.5 Artificial neural networks and pattern recognition: Case III

The third case studied is when the dataset is composed of the two-state conditions: healthy and damaged level 1, with 2 output neurons (healthy and damaged). The PCA is used and the total variance for each 5-folds are shown in Tab. 5.30.

Table 5.30 – Total variance for each 5-folds 10 and 20 PCs - Case III.

Set Number	Total variance (%)	Total variance (%)
	10 PCs	20 PCs
Set 1	96.552	99.223
Set 2	95.919	99.205
Set 3	96.031	99.283
Set 4	96.587	99.396
Set 5	96.073	99.291

Source: Author's production.

The best PSO-result for 20 inputs is $X_{top}=[1.08;15.49;9.76]$ with 100.00% of accuracy and $X_{top}=[1.53;2.86;4.31]$ for 10 inputs, with 100.00% of accuracy. Table 5.31 shows two ANNs simulations varying only the topology. Table 5.32 shows the results for the all 5-sets (after 150 times run). Even if the 10-input net shows an average accuracy of 68.10% in the training set, this is not an indicative of underfitting.

Table 5.31 – ANNs simulations with different parameters - Case III.

ANN	Topology	Activation function	Cost function	Step size	γ	Regularization technique	Noise	Shuffle samples
ANN_1	20-[15]-2	tanh	quadratic	BK	0.3	L_2 (0.001)	no	no
ANN_2	10-[3,4]-2	tanh	quadratic	BK	0.3	L_2 (0.001)	no	no

BK=backtracking line search; γ =momentum term.

Source: Author's production.

Table 5.32 – ANNs simulations with different parameters - accuracy results (Sets 1, 2, 3, 4 and 5) - 150 times run - Case III.

Set	Dataset	ANN_1		ANN_2	
		Mean - % (mean deviation)	Maximum	Mean - % (mean deviation)	Maximum
1	Training	88.54 (0.084)	100.00	68.18 (0.164)	100.00
	Validation	62.50 (0.112)	87.50	50.00 (0.131)	100.00
	Testing	57.14 (0.073)	85.71	50.00 (0.148)	85.71
2	Training	88.89 (0.092)	100.00	62.22 (0.159)	97.78
	Validation	50.00 (0.121)	75.00	50.00 (0.148)	87.50
	Testing	76.92 (0.052)	92.31	61.54 (0.155)	92.31
3	Training	86.67 (0.081)	100.00	75.55 (0.130)	97.78
	Validation	62.50 (0.070)	75.00	62.50 (0.118)	75.00
	Testing	69.23 (0.034)	84.61	69.23 (0.064)	84.61
4	Training	88.89 (0.076)	100.00	62.22 (0.134)	100.00
	Validation	62.50 (0.066)	75.00	62.50 (0.128)	87.50
	Testing	61.54 (1.316)	100.00	53.85 (0.153)	84.61
5	Training	88.89 (0.073)	100.00	75.56 (0.107)	100.00
	Validation	75.00 (0.073)	100.00	75.00 (0.118)	100.00
	Testing	53.85 (0.006)	76.92	46.15 (0.101)	76.92
<i>Total</i>	Training	<i>88.89 (0.006)</i>	<i>100.00</i>	<i>68.18 (0.05)</i>	<i>100.00</i>
	Validation	<i>62.50 (0.021)</i>	<i>100.00</i>	<i>62.50 (0.080)</i>	<i>100.00</i>
	Testing	<i>61.54 (0.403)</i>	<i>100.00</i>	<i>53.85 (0.074)</i>	<i>92.31</i>

Source: Author's production.

Table 5.33 shows the best result, as observed the best testing accuracy is 100% for the ANN with 20 inputs in set 4.

Table 5.33 – ANN best accuracy results (ANN_1 - Set 4) - Case III.

Training time	Training accuracy (%)	Validation accuracy (%)	Testing accuracy (%)
1'27"	91.11	62.50	100.00

Source: Author's production.

The confusion matrix (Tab. 5.34) and its parameters (Tab. 5.35) show good results for the classifier, with 100.00% of recall, specificity, accuracy, and efficiency, *i.e.*, there is no false positive and false negative indications. Figure 5.19 shows the training error and the three datasets accuracy curves. During almost all iterations the validation set accuracy to keep constant with 62.50%, it can be observed also that the training set accuracy is less than the testing set accuracy. Also, the testing set accuracy drops after 180 iterations. So, it is recommended to use the early stopping technique.

Table 5.34 – Confusion matrix for ANN_1 - Set 4: best result - Case III.

		Actual Class		Total
		Damaged	Healthy	
Predict Class	Damaged	3	0	3
	Healthy	0	10	10
Total		3	10	13

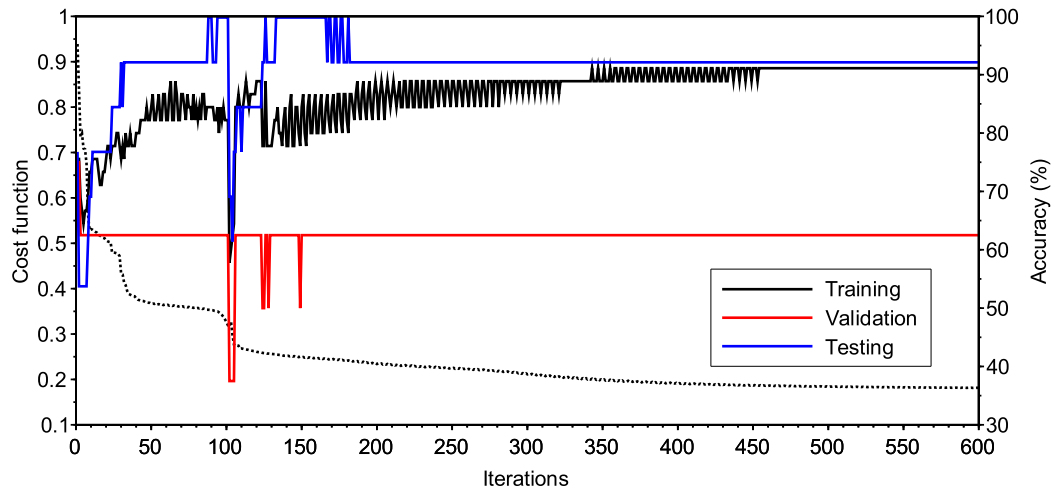
Source: Author's production.

Table 5.35 – Confusion matrix parameters for ANN_1 - Set 4: best result - Case III.

Parameters	Values
Recall	100.00%
Specificity	100.00%
Accuracy	100.00%
Matthews correlation coefficient (MCC)	1.00
Efficiency	100.00%

Source: Author's production.

Figure 5.19 – Training error (dot line) and accuracy (%) convergence curves for ANN_1 - Set 4: best result - Case III.



Source: Author's production.

5.2.6 Artificial neural networks and pattern recognition: Case IV

The fourth case studied is when the dataset is composed of two state-conditions: healthy and damaged level 3, with 2 output neurons (healthy and damaged). The PCA is performed and the total variance for each 5-folds are shown in Tab. 5.36 for the first 25 PCs.

The best PSO-result for 25 inputs is $X_{top}=[1.88;14.41;7.50]$ with 100.0% of accuracy.

Table 5.37 shows three ANNs simulations with different parameters. Table 5.38 shows the results for the 1st, 2nd, 3rd, 4th, and 5th sets after 150 times run. The best result, shown in Tab. 5.46, is for ANN_3 - Set 3 with 100.00% of the patterns classified correctly.

Table 5.36 – Total variance for each 5-folds for 25 PCs - Case IV.

Set Number	Total variance (%)
Set 1	99.208
Set 2	99.249
Set 3	99.198
Set 4	99.220
Set 5	99.323

Source: Author's production.

Table 5.37 – ANNs simulations with different parameters - Case IV.

ANN	Topology	Activation function	Cost function	Step size	γ	Regularization technique	Noise	Shuffle samples
ANN_1	25-[14,7]-2	tanh	quadratic	BK	0.3	L_2 (0.001)	no	no
ANN_2	25-[14,7]-2	tanh	quadratic	0.05	0.3	L_2 (0.001)	no	no
ANN_3	25-[14,7]-2	tanh + softmax	2-norm	BK	0.3	L_2 (0.001)	no	no
ANN_4	25-[14,7]-2	tanh + softmax	cross-entropy	BK	0.3	L_2 (0.001)	no	no

BK=backtracking line search; γ =momentum term.

Source: Author's production.

Table 5.38 – ANNs simulations with different parameters - accuracy results (Sets 1, 2, 3, 4 and 5) - 150 times run - Case IV.

Set	Dataset	ANN_1		ANN_2		ANN_3		ANN_4	
		Mean - % (mean devia- tion)	Maximum	Mean - % (mean devia- tion)	Maximum	Mean - % (mean devia- tion)	Maximum	Mean - % (mean devia- tion)	Maximum
1	Training	100.00 (0.012)	100.00	100.00 (0.000)	100.00	100.00 (0.169)	100.00	100.00 (0.165)	100.00
	Validation	90.00 (0.056)	100.00	100.00 (0.051)	100.00	100.00 (0.163)	100.00	90.00 (0.1648)	100.00
	Testing	82.35 (0.042)	88.23	82.35 (0.031)	88.23	82.35 (0.141)	88.23	82.35 (0.138)	88.23
2	Training	100.00 (0.016)	100.00	100.00 (0.000)	100.00	98.25 (0.100)	100.00	98.25 (0.112)	100.00
	Validation	70.00 (0.067)	90.00	80.00 (0.062)	90.00	80.00 (0.107)	90.00	70.00 (0.120)	90.00
	Testing	82.35 (0.036)	94.12	88.23 (0.032)	94.12	82.35 (0.009)	94.12	82.35 (0.106)	100.00
3	Training	100.00 (0.017)	100.00	100.00 (0.000)	100.00	100.00 (0.206)	100.00	100.00 (0.132)	100.00
	Validation	90.00 (0.062)	90.00	90.00 (0.031)	90.00	90.00 (0.180)	90.00	90.00 (0.124)	90.00
	Testing	88.23 (0.031)	94.12	88.23 (0.031)	94.12	88.23 (0.181)	100.00	88.23 (0.120)	100.00
4	Training	100.00 (0.011)	100.00	100.00 (0.000)	100.00	100.00 (0.180)	100.00	100.00 (0.114)	100.00
	Validation	70.00 (0.069)	80.00	70.00 (0.008)	80.00	70.00 (0.134)	80.00	65.00 (0.092)	80.00
	Testing	76.47 (0.031)	82.35	76.47 (0.029)	82.35	82.35 (0.141)	88.23	76.47 (0.094)	82.35
5	Training	100.00 (0.013)	100.00	100.00 (0.000)	100.00	100.00 (0.222)	100.00	100.00 (0.138)	100.00
	Validation	90.00 (0.031)	100.00	90.00 (0.035)	100.00	90.00 (0.207)	100.00	90.00 (0.140)	100.00
	Testing	81.25 (0.050)	93.75	81.25 (0.042)	93.75	87.50 (0.211)	93.75	87.50 (0.126)	93.75
Total	Training	100.00 (0.000)	100.00	100.00 (0.000)	100.00	100.00 (0.005)	100.00	100.00 (0.006)	100.00
	Validation	90.00 (0.096)	100.00	90.00 (0.088)	100.00	90.00 (0.088)	100.00	90.00 (0.108)	100.00
	Testing	82.35 (0.026)	94.12	82.35 (0.039)	94.12	82.35 (0.026)	100.00	82.35 (0.036)	100.00

Source: Author's production.

Table 5.39 – ANN best accuracy results (ANN_3 - Set 3) - Case IV.

Training time	Training accuracy (%)	Validation accuracy (%)	Testing accuracy (%)
1'52"	100.00	90.00	100.00

Source: Author's production.

The confusion matrix is shown in Tab. 5.40 as well as the confusion matrix parameters in Tab. 5.41, indicating no misclassification. The MCC indicates a very good classifier prediction with 1.00.

Table 5.40 – Confusion matrix for ANN_3 - Set 3: best result - Case IV.

		Actual Class		
		Damaged	Healthy	Total
Predict Class	Damaged	6	0	6
	Healthy	0	11	11
Total		6	11	17

Source: Author's production.

Table 5.41 – Confusion matrix parameters for ANN_3 - Set 3: best result - Case IV.

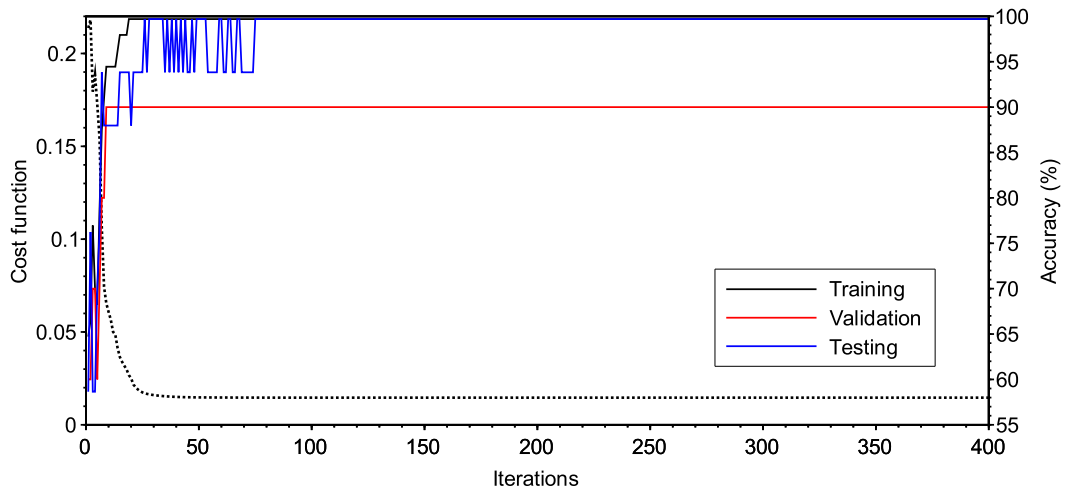
Parameters	Values
Recall	100.00%
Specificity	100.00%
Accuracy	100.00%
Matthews correlation coefficient (MCC)	1.00
Efficiency	100.00%

Source: Author's production.

Figure 5.20 shows the training error and the accuracy curves (for the three datasets) with a little instability until 110 iterations, and a stable learning after that, at first with no evidence of under and overfitting.

In a general overview, the methodology using PCA and ANNs proves to be a very effective tool for delamination damage detection in the different proportions of the damages studied.

Figure 5.20 – Training error (dot line) and accuracy (%) convergence curves for ANN_3 - Set 3: best result - Case IV.



Source: Author's production.

The best results are obtained when using only two classes as the input of the network, as in cases III and IV. In case I and II, there are four and three inputs classes, respectively. Although the output is only two classes (healthy or damaged), the presence of other damages (classes) in the input variables, can cause a larger variance in the samples for the damaged output class. Since the damage class includes damaged level 1, 2 and 3. Consequently, the network presents more difficulties to understand that the different curves for 5 mm, 10 mm and 19 mm are all for the same state condition class (damaged), and it gets worst when the number of the samples are very small.

Despite the results found, the average accuracy in both validation and testing sets is low. This may be due to several factors, such as, very large variance between samples of the same class causing the network to be confused with other classes; samples contained in the sets that do not represent the proper class due to manufacturing or experimental problems; the poor distribution of the samples in the training, validation and testing sets, causing some samples in the testing set not to be contained within the training set domain. As the region of study is very non-convex, the distribution of the samples between the three sets is of fundamental importance, since samples close to the decision boundaries carry more information about the problem. And still addressing non-convexity, depending on the initial choice of the weights and biases the algorithm determines which minimum the algorithm will converge to. Also, the presence of saddlepoints, or regions where the error function is very flat, can cause some iterative algorithms to become stagnated in local minima for many iterations.

It is verified that the use of a fixed learning rate or the use of line search presents

different results for each case and simulation; in one case the use of a fixed learning rate presents better performance, and in another case, the use of line search presents better convergence. The fact comes mainly from the point of initialization of the design variables (weights and biases) and the direction of the search for a minimum. If the algorithm is very close to a minimum the use of a line search method can be more advantageous since the size of the step is variable (and becoming small), if a fixed step is used the algorithm can be going through "above the minimum" and never reach it, thus oscillating the learning process. On the other hand, in a region with very close local minima, a fixed step can lead to reach a point of better minimum than with the use of the line search, since with variable steps becoming smaller and smaller the line search technique is in the attempt to arrive at to another local minimum. However, care should be taken with the choice of step size.

5.2.7 Artificial neural networks and pattern recognition: Case V

The fifth case studied is when the samples introduced in the ANN are all from the four state conditions classes with 4 neurons in the output layer, representing the four states classes. Here the intention besides to identify the existence of the damage is also to evaluate the extension of the damage. So as in the previous case, the PCA is done and the total variance for each 5-folds are shown in Tab. 5.42 for the first 30 PCs.

Table 5.42 – Total variance for each 5-folds 30 PCs - Case V.

Set Number	Total variance (%)
Set 1	99.170
Set 2	99.238
Set 3	99.287
Set 4	99.219
Set 5	99.245

Source: Author's production.

The different ANNs simulations are shown in Tab. 5.43 and after 150 times run the results of them are shown in Tab. 5.44 for the 1st set and in Tab. 5.45 for the other sets. As observed, there is no indication of underfitting as the accuracy of the training set is reaching 100%. The problem of low accuracy in the validation set and the testing set can be an overfitting problem, a wrong topology used as well as bad samples distributions among the datasets.

Table 5.43 – ANNs simulations with different parameters- Case V.

ANN	Topology	Activation function	Cost function	Step size	γ	Regularization technique	Noise	Shuffle samples
ANN_1	30-[20,10]-4	tanh	quadratic	0.05	0.3	no	no	90 %
ANN_2	30-[20,10]-4	tanh	quadratic	0.05	0.3	no	5 %	45 %
ANN_3	30-[20,10]-4	tanh	2-norm	0.05	0.3	no	no	45 %
ANN_4	30-[20,10]-4	logistic	cross-entropy	0.05	0.3	no	no	45 %
ANN_5	30-[20,10]-4	tanh	quadratic	BK	0.3	no	5 %	45 %
ANN_6	30-[20,10]-4	tanh	quadratic	0.001	0.3	no	no	45 %
ANN_7	30-[20,10]-4	leaky-ReLU + softmax	quadratic	0.05	0.3	no	no	90 %
ANN_8	30-[20,10]-4	tanh + softmax	quadratic	BK	0.3	L_2 (0.001)	no	90 %
ANN_9	30-[20,10]-4	tanh + softmax	cross-entropy	BK	0.3	L_2 (0.001)	no	90 %
ANN_{10}	30-[20,10]-4	tanh	quadratic	0.05	0.3	no	no	no

BK=backtracking line search; γ =momentum term.

Source: Author's production.

Table 5.44 – ANNs simulations with different parameters - accuracy results (Set 1) - 150 times run - Case V.

ANN	Training set		Validation set		Testing set	
	Mean - % (mean deviation)	Maximum	Mean - % (mean deviation)	Maximum	Mean - % (mean deviation)	Maximum
ANN_1	100.00 (0.000)	100.00	50.00 (0.063)	68.75	44.44 (0.042)	55.56
ANN_2	100.00 (0.000)	100.00	50.00 (0.066)	68.75	40.74 (0.035)	48.15
ANN_3	100.00 (0.000)	100.00	43.75 (0.060)	62.50	37.04 (0.036)	44.44
ANN_4	100.00 (0.000)	100.00	50.00 (0.064)	68.75	37.04 (0.028)	44.44
ANN_5	97.50 (0.091)	100.00	37.50 (0.065)	62.50	31.48 (0.054)	44.44
ANN_6	55.00 (0.089)	85.00	18.75 (0.048)	37.50	22.22 (0.043)	29.63
ANN_7	94.38 (0.013)	96.63	50.00 (0.067)	68.75	44.44 (0.050)	62.96
ANN_8	70.00 (0.160)	98.75	37.50 (0.088)	62.50	29.63 (0.055)	51.85
ANN_9	86.25 (0.191)	100.00	43.75 (0.108)	75.00	37.04 (0.092)	59.26
ANN_{10}	100.00 (0.000)	100.00	50.00 (0.064)	68.75	48.15 (0.033)	55.56

Source: Author's production.

Table 5.45 – ANNs simulations with different parameters - accuracy results (Sets 2, 3, 4 and 5) - 150 times run - Case V.

Set	Dataset	ANN_1		ANN_7	
		Mean - % (mean deviation)	Maximum	Mean - % (mean deviation)	Maximum
2	Training	100.00 (0.000)	100.00	100.00 (0.013)	100.00
	Validation	53.33 (0.054)	66.67	46.67 (0.066)	66.67
	Testing	44.44 (0.044)	59.26	55.56 (0.054)	70.37
3	Training	100.00 (0.000)	100.00	100.00 (0.000)	100.00
	Validation	50.00 (0.050)	68.75	55.25 (0.067)	68.75
	Testing	46.15 (0.051)	73.08	53.85 (0.057)	69.23
4	Training	100.00 (0.002)	100.00	98.75 (0.014)	100.00
	Validation	56.25 (0.067)	75.00	62.50 (0.077)	81.25
	Testing	50.00 (0.045)	61.54	53.85 (0.056)	69.23
5	Training	100.00 (0.000)	100.00	100.00 (0.000)	100.00
	Validation	40.62 (0.062)	56.25	50.00 (0.064)	68.75
	Testing	38.46 (0.036)	50.00	42.31 (0.045)	57.69
<i>Total</i>	Training	<i>100.00 (0.000)</i>	<i>100.00</i>	<i>100.00 (0.017)</i>	<i>100.00</i>
	Validation	<i>50.00 (0.038)</i>	<i>75.00</i>	<i>50.00 (0.050)</i>	<i>81.25</i>
	Testing	<i>44.44 (0.027)</i>	<i>73.08</i>	<i>53.85 (0.053)</i>	<i>70.37</i>

Source: Author's production.

The best result is shown in Tab. 5.46 with maximum testing accuracy of 73.08% with 62.50% of patterns classified correctly in the validation set. And its multi-class confusion matrix in Tab. 5.47. For healthy class, the classifier hits 34.61% from 46.15%, for damaged level 1 ($D-L_1$) class the classifier does not hit sample, however, in the selected set there is only one sample of this class. For damaged level 2 ($D-L_2$) the classifier hits 19.23% from 19.23% and for damaged level 3 ($D-L_3$) the classifier hits 19.23% from 30.77%.

Table 5.46 – ANN two best accuracy results (ANN_1 - Set 3) - Case V.

Training time	Training accuracy (%)	Validation accuracy (%)	Testing accuracy (%)
27'46"	100.00	62.50	73.08

Source: Author's production.

Table 5.47 – Multi-class confusion matrix for ANN_1 - Set 3 - Case V.

		Actual Class			
		H	$D-L_1$	$D-L_2$	$D-L_3$
Predict Class	H	9 34.61%	1 3.85%	0 0.00%	2 7.70%
	$D-L_1$	1 3.85%	0 0.00%	0 0.00%	0 0.00%
	$D-L_2$	1 3.85%	0 0.00%	5 19.23%	1 3.85%
	$D-L_3$	1 3.85%	0 0.00%	0 0.00%	5 19.23%

Source: Author's production.

The accuracy results obtained for the presence and extension of the damage are smaller when compared to the previous one. A possible cause is that in addition to the number of samples, as in the previous cases, in the current problem, the network needs

to classify four classes, which make the problem more complex, since the decision boundary changes. According to Duda *et al.* (1973), the degree of difficulty of the classification problem depends on the variability in the feature values for objects in the equal class respective to the distinctness between features values of objects in different classes. Therefore, in all simulations cases (experimental and numerical), the results can be a consequence to the fact that the damage imposed to the beam in the four scenarios are very similar, making it difficult to classify these data correctly.

An important fact is a limitation of the gradient descent technique with the necessity to choose a suitable value for the learning rate as well as for the momentum term. A poor choice can contribute to the gradient descent taking many steps to reach the minimum, making the process inefficient. Another parameter that is defined by trial and error is the penalty parameter when the regularization techniques are used. Small or large values can lead to hinder the ability of the model to learn. Also, the topology has a huge influence in the learning and generalization process, since few neurons can lead to underfitting problems and many neurons can contribute to overfitting issues. Also, the number of hidden layers and neurons are connected to the number of samples needed to have good generalization results. An interesting point to note is that the use of the leaky ReLU activation function leads to a better convergence for the V case than the previous cases I to IV.

About the use of techniques to attenuate the overfitting, the use of noise in some cases improved the performance of the validation and testing sets and in others, it worsened. The use of L_2 regularization proved to be more effective when compared to L_1 regularization for the testing set. Shuffle the training set always increase its training accuracy when compared to simulations without shuffling.

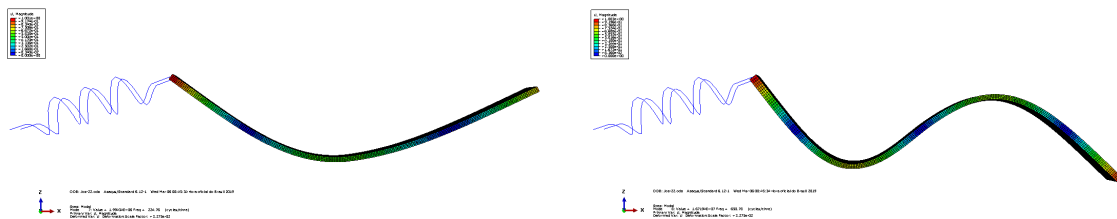
Other simulations of neural networks were performed using only 20 inputs to decrease the number of design variables. After all, the accuracy result is lower than those obtained with 30 inputs. It was also studied the possibility of using the FRF curves in real part values, however, for the same sets of training, validation and testing the results are worse than the same ANN considering accelerance values. In addition to achieving the same variance, it is necessary to use five more inputs causing the network to increase the number of weights and bias, hampering the convergence of the problem.

5.3 Damage Detection in Glass-Fiber/Epoxy Composite Beams: Numerical

5.3.1 Dynamic and modal analysis

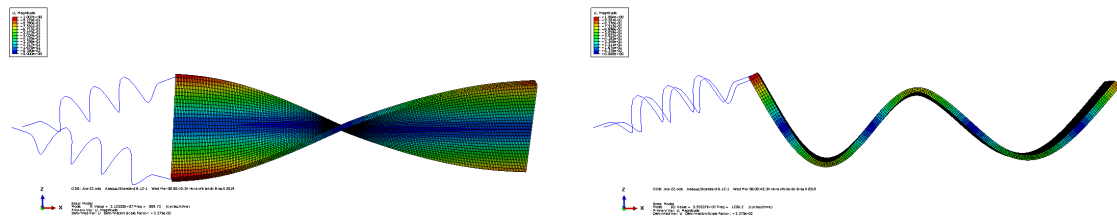
The first four modes corresponding to a healthy beam are shown in Fig. 5.21 and Fig. 5.22 and the frequencies are Mode 1: 224.76 Hz; Mode 2: 650.78 Hz; Mode 3: 889.73 Hz and Mode 4: 1226.20 Hz. Modes 1, 2 and 4 are flexural modes and mode 3 is a torsional mode.

Figure 5.21 – Mode shapes 1 and 2.



Source: Author's production.

Figure 5.22 – Mode shapes 3 and 4.

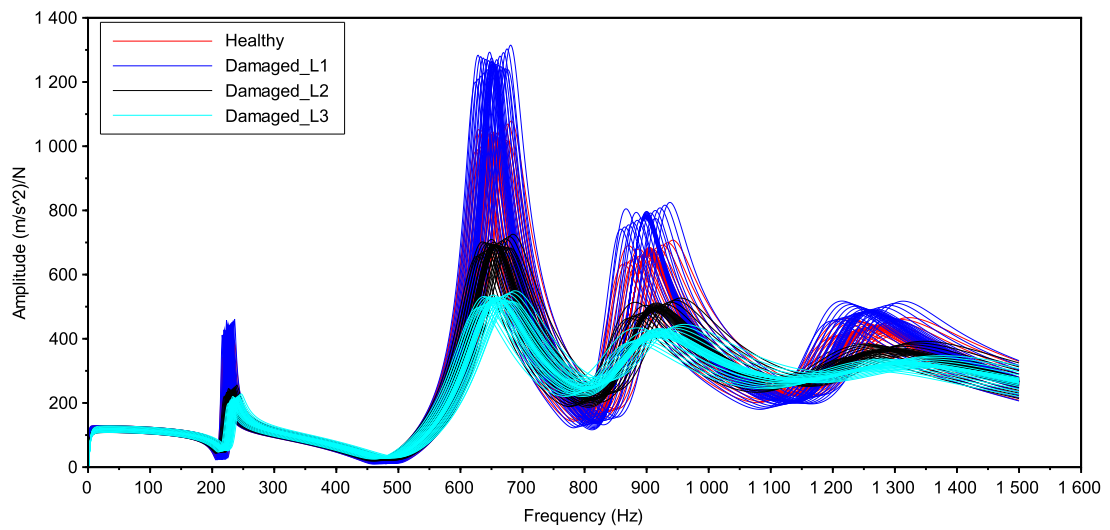


Source: Author's production.

The FRFs for position H_{11} can be seen in Fig. 5.23 and for position H_{21} - FRFs in Fig. 5.24, both using accelerance values.

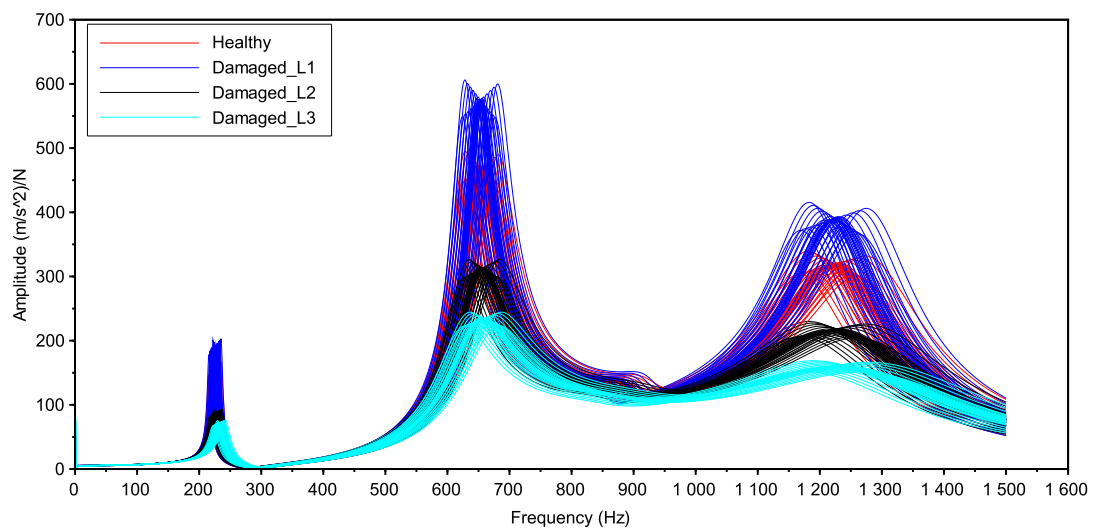
The range between the minimum frequency and maximum frequency for the first four mode shapes from the four states conditions (healthy, damaged level 1, damaged level 2 and damaged level 3) are shown in Fig. 5.25. It may be noted that the interval of each mode for a particular state is large. And when compared to the mode and the states it is noted that many samples are within the same range. So the four scenarios are very similar when comparing the resonant frequencies.

Figure 5.23 – H_{11} - Numerical FRFs curves (accelerance values) - 4 states conditions.



Source: Author's production.

Figure 5.24 – H_{21} - Numerical FRFs curves (accelerance values) - 4 states conditions.

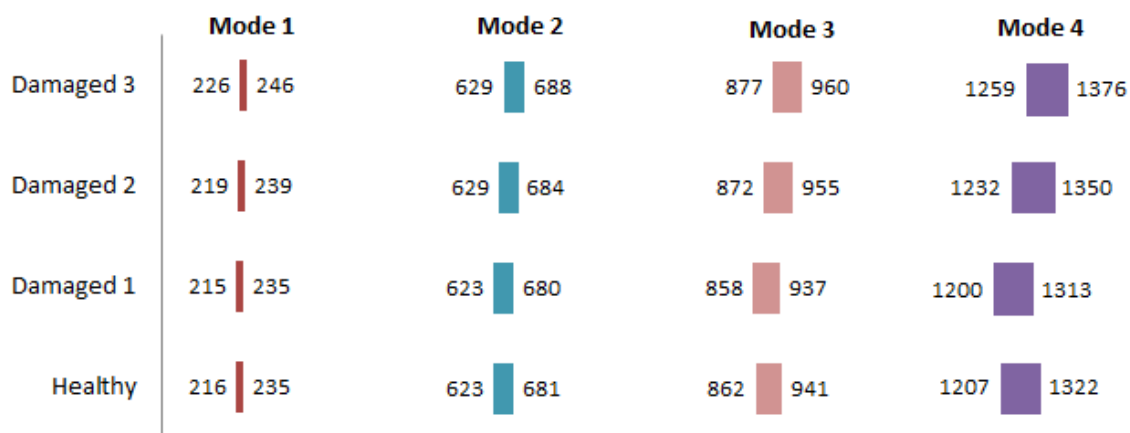


Source: Author's production.

5.3.2 Principal component analysis

The PCA is performed and the total variance for each 5-folds are shown in Tab. 5.48 for the first 10 and 20 PCs using only healthy and damaged level 1. For all the

Figure 5.25 – Frequency range (Hz) for the first four modes shapes from the four damage states (H_{11}).



Source: Author's production.

states conditions the total variance for each 5-folds are shown in Tab. 5.49 for the first 30 PCs.

Table 5.48 – Total variance for each 5-folds 10 and 20 PCs - Numerical (healthy and damaged level 1 sates conditions).

Set Number	Total variance (%)	Total variance (%)
	10 PCs	20 PCs
Set 1	99.817	99.993
Set 2	99.808	99.992
Set 3	99.808	99.993
Set 4	99.816	99.993
Set 5	99.811	99.992

Source: Author's production.

Table 5.49 – Total variance for each 5-folds 10 and 20 PCs - Numerical (4 sates conditions).

Set Number	Total variance (%) 30 PCs
Set 1	99.9975
Set 2	99.9974
Set 3	99.9973
Set 4	99.9977
Set 5	99.9974

Source: Author's production.

5.3.3 Artificial neural networks and pattern recognition: Case I- Numerical

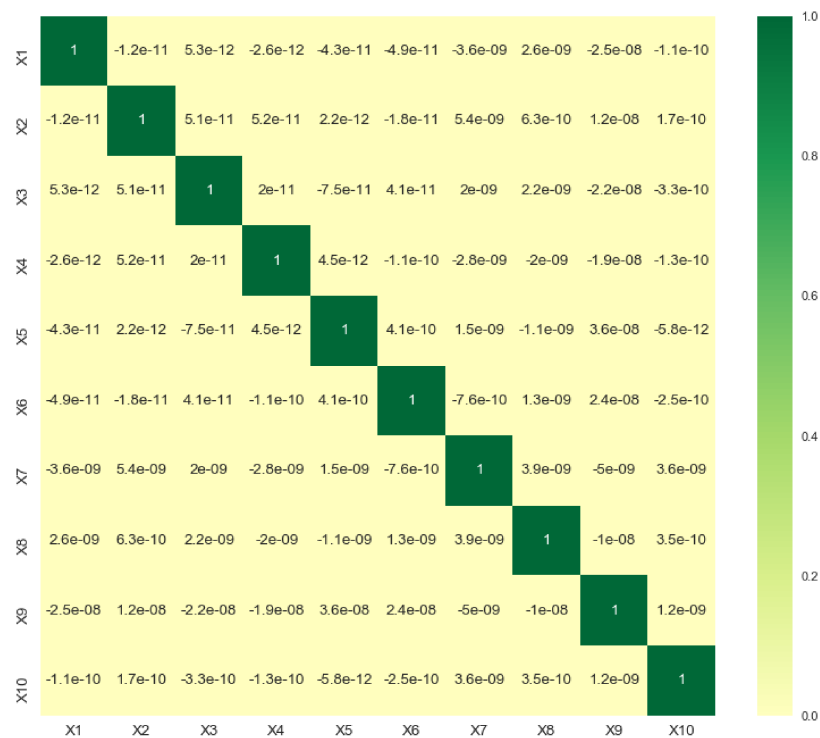
The numerical case I studied is when the dataset is composed of the two-state conditions: healthy and damaged level 1, with 2 output neurons (healthy and damaged). The same studied in Case III using experimental curves.

The 10-inputs variables correlation graph can be seen in Fig. 5.26. To a better visualization of the variables and the surface boundaries the scatter plot can be seen in Appendix A in Fig. A.11. Note that there is no correlation between the variables, as a PCA result. The same is observed when using the data from the experimental procedure (case III) in Fig. A.12 (Appendix A). Also, can be observed in both figures that the decision boundary is not linearly separable, when looking to a 2D plot. The ANN simulations done are the same as performed for the Case III described in Tab. 5.31.

Table 5.50 shows the results for the all 5-sets (after 150 times runs). It can be noted that both simulations present very good results, with almost all accuracies averaging more than 90.00%. The best set is set 4. It can be observed that there is no link between the maximum variance acquired in PCA and the ANN results since the largest variance is found in set 1 (for 10 and 20 PCs) and this one obtained the maximum accuracy of 94.70%.

The worst mean accuracy obtained with the numerical study for the testing set is 89.19% when compared with the experimental study for the same case the worst result is 46.15% for the same ANN. Comparing the best mean accuracy result for the testing set the best value for the numerical study is 100.00% and for the experimental study, the best mean accuracy for the testing set is 76.92%. As seen the two studied have the maximum testing accuracy of 100.00%, however, the numerical case has more iterations represented that value when compared with the experimental case. This response may be due to the increase in the number of samples with a greater

Figure 5.26 – 10 inputs correlation variables.



Source: Author's production.

representatively for each condition. In a numerical study, it can be noted that there are not very large discrepancies between the FRFs of the same state condition case, eliminating the problem of having atypical samples, which do not represent the state condition group. One point to note is that the methodology works very well with samples that have small differences between them. Since healthy FRFs have values very close to FRFs damaged by 5 mm.

Table 5.50 – ANNs simulations with different parameters - accuracy results (Sets 1, 2, 3, 4 and 5) - 150 times run - Numerical.

Set	Dataset	ANN_1		ANN_2	
		Mean - % (mean deviation)	Maximum	Mean - % (mean deviation)	Maximum
1	Training	100.00 (0.003)	100.00	100.00 (0.014)	100.00
	Validation	91.30 (0.027)	100.00	91.30 (0.034)	100.00
	Testing	94.74 (0.015)	100.00	94.74 (0.019)	100.00
2	Training	100.00 (0.006)	100.00	99.21 (0.030)	100.00
	Validation	100.00 (0.020)	100.00	95.65 (0.052)	100.00
	Testing	100.00 (0.014)	100.00	94.74 (0.041)	100.00
3	Training	100.00 (0.006)	100.00	100.00 (0.023)	100.00
	Validation	100.00 (0.011)	100.00	95.65 (0.027)	100.00
	Testing	92.10 (0.016)	97.37	92.10 (0.027)	97.37
4	Training	100.00 (0.000)	100.00	100.00 (0.024)	100.00
	Validation	100.00 (0.021)	100.00	100.00 (0.034)	100.00
	Testing	100.00 (0.000)	100.00	100.00 (0.025)	100.00
5	Training	100.00 (0.023)	100.00	97.67 (0.049)	100.00
	Validation	100.00 (0.027)	100.00	90.91 (0.063)	100.00
	Testing	97.30 (0.000)	97.30	89.19 (0.073)	97.30
<i>Total</i>	Training	<i>100.00 (0.000)</i>	<i>100.00</i>	<i>100.00 (0.014)</i>	<i>100.00</i>
	Validation	<i>100.00 (0.028)</i>	<i>100.00</i>	<i>95.65 (0.029)</i>	<i>100.00</i>
	Testing	<i>97.30 (0.027)</i>	<i>100.00</i>	<i>94.74 (0.028)</i>	<i>100.00</i>

Source: Author's production.

5.3.4 Artificial neural networks and pattern recognition: Case II-Numerical

The numerical case II studied is when the dataset is composed of the four-state conditions: healthy, damaged level 1, damaged level 2, and damaged level 3 with 4 output neurons (corresponding for each state condition). The same studied in Case V using experimental curves.

In order to compare numerical case II with experimental case V the same ANN architecture used in case V is applied here. The results for the 5-folds after 150 times run are shown in Tab. 5.51.

Table 5.51 – ANN simulation - the best accuracy results (Sets 1, 2, 3, 4 and 5) - 150 times run - Case II-N.

Set	Dataset	ANN_1	
		Mean - % (mean deviation)	Maximum
1	Training	100.00 (0.000)	100.00
	Validation	94.87 (0.0016)	97.44
	Testing	98.48 (0.000)	98.48
2	Training	100.00 (0.000)	100.00
	Validation	97.44 (0.011)	100.00
	Testing	95.45 (0.007)	98.48
3	Training	100.00 (0.000)	100.00
	Validation	100.00 (0.002)	100.00
	Testing	95.45 (0.011)	96.97
4	Training	100.00 (0.000)	100.00
	Validation	100.00 (0.005)	100.00
	Testing	95.45 (0.011)	98.48
5	Training	100.00 (0.000)	100.00
	Validation	95.00 (0.022)	100.00
	Testing	98.48 (0.000)	100.00
<i>Total</i>	Training	<i>100.00 (0.000)</i>	<i>100.00</i>
	Validation	<i>97.44 (0.040)</i>	<i>100.00</i>
	Testing	<i>95.45 (0.014)</i>	<i>100.00</i>

Source: Author's production.

The multi-class confusion matrix for the 150 times run each 5-folds is shown in Tab. 5.52. The results are very good with 96.22% of healthy samples correctly classified, 92.22% of damaged level 1 samples correctly classified, 98.15% of damaged level 2 samples correctly classified and all damaged level 3 samples correctly classified. The maximum misclassification error is 2.19% where samples from damaged level 1 are misclassified as healthy samples.

Comparing experimental case V with the numerical case II it can be noted that

Table 5.52 – Numerical Multi-class confusion matrix - Case V.

		Actual Class			
		H	D- L_1	D- L_2	D- L_3
Predict Class	H	13712 27.70%	1085 2.19%	14 0.03%	0 0.00%
	D- L_1	329 0.66%	12865 25.99%	0 0.00%	0 0.00%
	D- L_2	209 0.42%	0 0.00%	10600 21.41%	0 0.00%
	D- L_3	0 0.00%	0 0.00%	186 0.37%	10500 21.21%

Source: Author's production.

with numerical curves the methodology works very well. As explained before, the reasons for the best results are achieved when using numerical curves are, first the increase in the number of samples for all classes, and second in numerical curves there are not uncertainties that come from the manufacturing process neither from the dynamics-tests, so each pattern class has its typical features that can be easily classify by the ANN.

Chapter 6

Conclusions

6.1 Conclusions

In this work, a strategy for fault diagnosis in composite structures based on dynamic tests, principal component analysis (PCA) and artificial neural networks (ANNs) is investigated. With the main purpose of evaluating the potentialities and limitations of the methodology in composite materials, different composites are studied, by varying the type of fibers, different damage types and sizes. Through which this study it is possible to evolve the methodology for real-time monitoring of composite materials structures.

First composites made of carbon-fiber/epoxy are evaluated in two orientations, aiming to assess the behavior of the method, specially when the FRF curves are very close between healthy and damaged state. Also, different types of FRFs curves (accelerance values and real part values) are analyzed. As noted, there is no right answer on which type of curve to use, since it will depend on the complexity of the problem and the decision boundaries. Such that a preliminary analysis of the features and how to extract them is fundamental when working with pattern recognition. Thus, the methodology proves to be very accurate. Subsequently, glass-fiber/epoxy plates are fabricated using the modified-VARTM method with four different state conditions: healthy (H), damaged level 1 ($D-L_1$), with 5 mm delamination size, damaged level 2 ($D-L_2$), with 10 mm delamination size and damaged level 3 ($D-L_3$), with 19 mm delamination size. The vibration-based tests are performed for FRF acquisition. The data dimension is reduced through the PCA, and the samples set introduced into different neural networks by varying several parameters. A multilayer neural network is developed in Julia language using automatic differentiation to sensitivity analysis. This approach rewrites the real numbers of the weight and biases in dual numbers and proceeds to derivative calculations. To minimize the error function in the supervised learning the steepest descent method is used. Also, a particle swarm optimization (PSO) algorithm is used

to find the neural network topology in order to maximize a function based on the validation set accuracy. The best results are extracted and analyzed. Techniques to avoid overfitting are used as k-fold cross-validation, training with noise, early stopping and regularization techniques. Five cases are studied, the first four address the existence of the damage and the last one including the damage extension, the studying cases are described as following

- Case I: $H + D-L_1 + D-L_2 + D-L_3$: 2 classes output (healthy and damaged);
- Case II: $H + D-L_2 + D-L_3$: 2 classes output (healthy and damaged);
- Case III: $H + D-L_1$: 2 classes output (healthy and damaged);
- Case IV: $H + D-L_3$: 2 classes output (healthy and damaged);
- Case V: $H + D-L_1 + D-L_2 + D-L_3$: 4 classes output.

The last step of the study uses a Finite Element Model (FEM) of a glass-fiber/epoxy beam to analyze the performance of the methodology and to compare with the experimental study. The vibration-based test are simulated in ABAQUS 6.12 and the numerical FRFs are used instead of the experimental.

The numerical study, the methodology shows very good performance and an excellent tool for Structural Health Monitoring (SHM). In the experimental cases, the results obtained using carbon-fiber plates indicated the methodology as a good tool for fault diagnosis too. For the glass fiber beams, in general for damage existence evaluation the methodology obtains good results, achieving classifications rates greater than 88%. For damage extension, the methodology shows the worst performance with maximum accuracy in the testing set of 73.08%. This can be explained by the fact that the separation hyperplane, *i.e.*, the boundary surface is more hardly adjusted when there are more classes and a small number of samples. In other words, the complex problems require large networks and large networks require more samples, reducing the Bias and the variance. Taking this into account the use of the PCA technique has become essential to transform the original data into a smaller set. However, it should be noted that excessive reductions of the original dataset may lead to the removal of important features.

The low accuracy in the validation and testing sets is due to the fact that the performance of the learning process depends on the distributions of the training samples and on the nature of the function to be learned. Since there is no way a network can learn the variations in the function that are not reflected in the training samples. If there are some samples in the validation or in the testing set that are not reflecting in the training set the network will not be able to classify the samples correctly. This is why future samples should be from the same probability distribution. Besides the

variability between the FRFs of the samples from the same class is very large as well as the difference between healthy and D- L_1 FRFs be very similar. Another point that must be taken into account is the chance of some sample happen to be very atypical. Underfitting problems is not observed, at least for the majority of the simulation. The big difference between the accuracy of the same set between the simulations studied is due to the different initializations of the design variables (weights and bias) in certain geometric planes more prone to reach the best local minimum. The manufacturing of the plates was also not performed under the best possible manufacturing conditions since the difference of the FRFs curves between the same delamination damages is clear, which is not seen in the FEM simulation. This fact and also the larger number of samples explain the high performance when using the curves from the FEM simulation.

The use of the PSO as a tool for the optimization problem – to find the best topology – is very helpful, since its one of the many parameters to chose by trial and error. Although, it does not give the exact result for the number of layers and neurons, since it works with continuous variables, the results are approximate. In this way, a refinement in the optimization problem should be done to get better results.

The use of neural networks is becoming more widespread and many studies on how to improve network performance have been discussed and proposed. The best way to start weights and bias, use of line search techniques or how to use an ideal step size, use the term momentum, different activation functions to prevent the gradient from vanishing or the training process to die, regularization techniques, introduction of noise in the data, use of early stopping and other techniques not mentioned. However not all of these techniques will present the best performances in all problems, since each problem has its particularity and complexity, and each technique has its advantages and disadvantages. With this, it is believed that the critical step for reach a very good generalization in the network in pattern classification problems is the choice of the distinguishing features that depend on the characteristics of the problem domain.

Generally, the methodology shows good conditions and results to be applicable in composites fault diagnosis as a tool for structural health monitoring. Since the use of composite materials is increasing all over the recent years and with a very promising future. Including the use of machine learning and deep learning, resourced increasingly exploited in the community.

6.2 Suggestions for Future Works

To improve the results obtained in this master thesis, some aspects are suggested as future researches:

- Use data from time-domain response to evaluate the performance of the ANN in

the classification problem;

- Apply unsupervised learning because they do not require damaged samples and thus reduce the problem in varying manufactured samples;
- Apply another technique to feature extraction and compare with PCA;
- Apply a deep learning based on health monitoring systems concepts, where the feature extraction phase is done in more deep learning neural networks, with multiple layers of non-linear transformations, and then there is not the necessity to work with the features before being introduced in the deep learning neural networks;
- Study other proposals of neural networks in order to locate and evaluate the type of damage in composite materials.

Bibliography

ALIANCYS. *Filament Winding*. 2019. <<https://aliancys.com/en/products/processing/filament-winding/>>. Accessed: 2019-01-23.

AMAFABIA, D. M. *et al.* A review of structural health monitoring techniques as applied to composite structures. *SDHM Structural Durability and Health Monitoring*, Bournemouth University, Fern Barrow, Poole, Dorset, BH12 5BB, UK, v. 11, n. 2, p. 91–147, 2018.

ANDERSON, T. L. *Fracture mechanics: fundamentals and applications*. [S.l.]: CRC press, 2017.

ARORA, J. *Introduction to optimum design*. [S.l.]: Elsevier, 2004.

ASTM. *Designation: D2584—18 Standard Test Method for Ignition Loss of Cured Reinforced Resins*. 2019. <https://compass.astm.org/EDIT/html_annot.cgi?D2584+18f>. Accessed: 2019-01-23.

AVITABILE, P. Experimental modal analysis. *Sound and vibration*, v. 35, n. 1, p. 20–31, 2001.

BANDARA, R. P.; CHAN, T. H.; THAMBIRATNAM, D. P. Structural damage detection method using frequency response functions. *Structural Health Monitoring*, Sage Publications Sage UK: London, England, v. 13, n. 4, p. 418–429, 2014.

BASHEER, I. A.; HAJMEER, M. Artificial neural networks: fundamentals, computing, design, and application. *Journal of microbiological methods*, Elsevier, v. 43, n. 1, p. 3–31, 2000.

BENZONI, G.; BONESSIO, N.; LOMIENTO, G. Structural health monitoring of bridges with seismic response modification devices. *Report No. SSRP-13/02*, Department of Structural Engineering University of California, San Diego La Jolla, California, v. 1, n. -, p. 1–197, 2013.

BEZANSON, J. *et al.* Julia: A fresh approach to numerical computing. *SIAM review*, SIAM, v. 59, n. 1, p. 65–98, 2017.

BISHOP, C. M. *Neural networks for pattern recognition*. [S.l.]: Oxford university press, 1995.

BISHOP, C. M. Training with noise is equivalent to tikhonov regularization. *Neural computation*, MIT Press, v. 7, n. 1, p. 108–116, 1995.

BISHOP, C. M. Periodic variables. *Pattern recognition and machine learning*, Springer Science+ Business Media, LLC New York, v. 1, 2006.

BOSE, N. K.; LIANG, P. Neural network fundamentals with graphs, algorithms, and applications (mcgraw-hill series in electrical computer engineering). McGraw-Hill USA, 1996.

BRAGA, A. d. P.; CARVALHO, A.; LUDERMIR, T. B. *Redes neurais artificiais: teoria e aplicações*. [S.l.]: Livros Técnicos e Científicos, 2000.

BROWNLEE, J. *Use Weight Regularization to Reduce Overfitting of Deep Learning Models*. 2018. <<https://machinelearningmastery.com/weight-regularization-to-reduce-overfitting-of-deep-learning-models/>>. Accessed: 2019-01-23.

BUILD. *A history of machine learning*. 2017. <<https://cloud.withgoogle.com/build/data-analytics/explore-history-machine-learning/>>. Accessed: 2019-01-10.

CAMPBELL, F. C. *Structural composite materials*. [S.l.]: ASM international, 2010.

CARLINI, N.; WAGNER, D. Towards evaluating the robustness of neural networks. In: IEEE. *Security and Privacy (SP), 2017 IEEE Symposium on*. [S.l.], 2017. p. 39–57.

CHUNG, D. D. *Composite materials: science and applications*. [S.l.]: Springer Science & Business Media, 2010.

CLESIO, F. *MÉTRICAS DE AVALIAÇÃO DE MODELOS DE CLASSIFICAÇÃO/PREDIÇÃO*. 2014. <<https://mineracaodedados.wordpress.com/2014/11/16/metricas-de-avaliacao-de-modelos-de-classificacaopredicao.html>>. Accessed: 2018-04-20.

CRAIG, R. R.; KURDILA, A. J. *Fundamentals of structural dynamics*. [S.l.]: John Wiley & Sons, 2006.

DACKERMANN, U. *Vibration-based damage identification methods for civil engineering structures using artificial neural networks*. Tese (Doutorado) — Faculty of Engineering and Information Technology, University of Technology Sydney, 2009.

DERVILIS, N. *A machine learning approach to structural health monitoring with a view towards wind turbines*. Tese (Doutorado) — University of Sheffield, 2013.

DUDA, R. O. *et al. Pattern classification*. [S.l.]: Wiley New York, 1973. v. 2.

EASTHAM, M. 2968. on the definition of dual numbers. *The Mathematical Gazette*, Cambridge University Press, v. 45, n. 353, p. 232–233, 1961.

EWINS, D. *Modal Testing: Theory, Practice, and Application*. [S.l.]: Research Studies Press, 2000.

FARRAR, C. R.; DOEBLING, S. W.; NIX, D. A. Vibration-based structural damage identification. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, The Royal Society, v. 359, n. 1778, p. 131–149, 2001.

FARRAR, C. R.; WORDEN, K. An introduction to structural health monitoring. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, The Royal Society, v. 365, n. 1851, p. 303–315, 2007.

FAZITA, M. N. *et al.* Green composites made of bamboo fabric and poly (lactic) acid for packaging applications—a review. *Materials*, Multidisciplinary Digital Publishing Institute, v. 9, n. 6, p. 435, 2016.

FIKE, J.; ALONSO, J. The development of hyper-dual numbers for exact second-derivative calculations. In: *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. [S.l.: s.n.], 2011. p. 886.

FORTMANN-ROE, S. *Understanding the Bias-Variance Tradeoff*. 2012. <<http://scott.fortmann-roe.com/docs/BiasVariance.html>>. Accessed: 2018-04-16.

FU, Z.-F.; HE, J. *Modal analysis*. [S.l.]: Butterworth-Heinemann, 2001.

GUL, M.; CATBAS, F. N. Statistical pattern recognition for structural health monitoring using time series modeling: Theory and experimental verifications. *Mechanical Systems and Signal Processing*, Elsevier, v. 23, n. 7, p. 2192–2204, 2009.

HACLALIOGLU, I. H. *A Review of the Global Composites Market and Turkish Composites Market*. 2018. <<https://www.reinforcer.com/en/category/detail/A-Review-of-the-Global-Composites-Market-and-Turkish-Composites-Market/61/350/0>>. Accessed: 2019-01-10.

HANDBOOK-MIL-HDBK, M. 17-3f: Composite materials handbook, volume 3-polymer matrix composites materials usage, design, and analysis. *US Department of Defense*, 2002.

HASSAN, R. *et al.* A comparison of particle swarm optimization and the genetic algorithm. *American Institute of Aeronautics and Astronautics*, 2004.

HAY, A. The derivation of global estimates from a confusion matrix. *International Journal of Remote Sensing*, Taylor & Francis, v. 9, n. 8, p. 1395–1398, 1988.

HAYKIN, S. *Neural networks a comprehensive foundation*: Prentice hall international. Inc., Englewood Cliffs, 1999.

HAYKIN, S. *Redes neurais: princípios e prática*. [S.l.]: Bookman Editora, 2007.

HEIDA, J. H.; PLATENKAMP, D. J. In-service inspection guidelines for composite aerospace structures. In: *18th World Conference on Nondestructive Testing*. [S.l.: s.n.], 2012. p. 16–20.

HILD, F.; PÉRIÉ, J.-N.; ROUX, S. Evaluating damage with digital image correlation: C. applications to composite materials. *Handbook of Damage Mechanics: Nano to Macro Scale for Materials and Structures*, Springer, p. 1301–1322, 2015.

INSINNA, V. *Poor maintenance contributed to a devastating C-130 crash. Here's how the Air Force will make sure it doesn't happen again*. 2018. <<https://www.defensenews.com>>. Accessed: 2019-01-10.

JABRI, M.; FLOWER, B. Weight perturbation: An optimal architecture and learning technique for analog vlsi feedforward and recurrent multilayer networks. *IEEE Transactions on Neural Networks*, IEEE, v. 3, n. 1, p. 154–157, 1992.

JUNIOR, F. C. C. *Manufacturing technology for aerospace structural materials*. [S.l.]: Elsevier, 2011.

KAPUR, R. *The vanishing gradient problem*. 2016. <<https://ayearofai.com/rohan-4-the-vanishing-gradient-problem-ec68f76ffb9b>>. Accessed: 2018-11-23.

KARPATHY, A. *CS231n Convolutional Neural Networks for Visual Recognition*. 2017. <<http://cs231n.github.io/neural-networks-2/>>. Accessed: 2018-12-10.

KAW, A. K. *Mechanics of composite materials*. [S.l.]: CRC press, 2005.

KEDEM, G. Automatic differentiation of computer programs. *ACM Transactions on Mathematical Software (TOMS)*, ACM, v. 6, n. 2, p. 150–165, 1980.

KOLLÁR, L. P.; SPRINGER, G. S. *Mechanics of composite structures*. [S.l.]: Cambridge university press, 2003.

KUENTZER, N. *et al.* Correlation of void distribution to vartm manufacturing techniques. *Composites Part A: applied science and manufacturing*, Elsevier, v. 38, n. 3, p. 802–813, 2007.

KUNCHEVA, L. I. *Combining pattern classifiers: methods and algorithms*. [S.l.]: John Wiley & Sons, 2004.

LECUN, Y. Efficient learning and second-order methods. *A tutorial at NIPS*, v. 93, p. 61, 1993.

LECUN, Y.; SIMARD, P. Y.; PEARLMUTTER, B. Automatic learning rate maximization by on-line estimation of the hessian's eigenvectors. In: *Advances in neural information processing systems*. [S.l.: s.n.], 1993. p. 156–163.

LEUCK, H.; NAGEL, H.-H. Automatic differentiation facilitates of-integration into steering-angle-based road vehicle tracking. In: *IEEE. Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on*. [S.l.], 1999. v. 2, p. 360–365.

LEVY-NETO, F.; PARDINI, L.; ESTRUTURAIS, C. C. *Tecnologia*. [S.l.]: São Paulo, Edgard Blücher, 2016.

LI, J. *et al.* Damage identification in civil engineering structures utilizing pca-compressed residual frequency response functions and neural network ensembles. *Structural Control and Health Monitoring*, Wiley Online Library, v. 18, n. 2, p. 207–226, 2011.

LOONEY, C. G. Advances in feedforward neural networks: demystifying knowledge acquiring black boxes. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 8, n. 2, p. 211–226, 1996.

- LOPES, H. R. *et al.* A numerical-experimental method for damage location based on rotation fields spatial differentiation. *Computers & Structures*, Elsevier, v. 89, n. 19-20, p. 1754–1770, 2011.
- MALLICK, P. K. *Fiber-reinforced composites materials, manufacturing, and design*. [S.l.]: CRC press, 2007.
- MANZYUK, O. *et al.* Confusion of tagged perturbations in forward automatic differentiation of higher-order functions. *arXiv preprint arXiv:1211.4892*, 2012.
- MARR, B. *A Short History of Machine Learning – Every Manager Should Read*. 2016. <<https://www.forbes.com/sites/bernardmarr/2016/02/19/a-short-history-of-machine-learning-every-manager-should-read/#734df92915e7>>. Accessed: 2019-01-10.
- MATLAB. *Introduction: System Modeling*. 2017. <ctms.engin.umich.edu/CTMS/index.php?example=Introduction§ion=SystemModeling>. Accessed: 2018-11-23.
- MEDEIROS, R. d. *Development of a criterion for predicting residual strength of composite structures damaged by impact loading*. Tese (Doutorado) — Universidade de São Paulo, 2016.
- MENDONÇA, P. d. T. R. de. *Materiais compostos e estruturas-sanduíche: projeto e análise*. [S.l.]: Manole, 2005.
- MONTALVAO, D.; MAIA, N. M. M.; RIBEIRO, A. M. R. A review of vibration-based structural health monitoring with special emphasis on composite materials. *Shock and vibration digest*, Bournemouth University, Fern Barrow, Poole, Dorset, BH12 5BB, UK, v. 38, n. 4, p. 295–324, 2006.
- NEGRI, D. *Previsão no domínio da frequência do comportamento dinâmico de peças de parede fina utilizando redução de ordem de modelo e mudança estrutural*. [S.l.]: Universidade do Estado de Santa Catarina, 2018.
- NG, A. *Principal Component Analysis Algorithm*. 2018. <<https://www.coursera.org/lecture/machine-learning/principal-component-analysis-algorithm-ZYIPa>>. Accessed: 2019-01-10.
- NICOLAIS, L.; MEO, M.; MILELLA, E. *Composite materials: a vision for the future*. [S.l.]: Springer Science & Business Media, 2011.
- NIELSEN, M. *CHAPTER 3 Improving the way neural networks learn*. 2018. <http://neuralnetworksanddeeplearning.com/chap3.html#overfitting_and_regularization>. Accessed: 2018-10-23.
- NTSB. *In-flight Separation of Right Wing Flying Boat, Inc. (doing business as Chalk's Ocean Airways) Flight 101*. 2005. <<https://www.nts.gov/investigations/AccidentReports/Reports/AAR0704.pdf>>. Accessed: 2019-01-10.
- OOIJEVAAR, T. H. Vibration based structural health monitoring of composite skin-stiffener structures. 2014.

PEREIRA, M. S.; BEZERRA, E. M. *Utilização de redes neurais para predição do comportamento em cisalhamento de compostos aeronáuticos*. [S.l.]: Encontro de iniciação científica e pós-graduação do ITA, 2007.

PREETHAM, V. V. *Mathematical foundation for Activation Functions in Artificial Neural Networks*. 2016. <<https://medium.com/autonomous-agents/mathematical-foundation-for-activation-functions-in-artificial-neural-networks>>. Accessed: 2018-11-23.

PUSCASU, G.; CODRES, B. Nonlinear system identification and control based on modular neural networks. *International Journal of Neural Systems*, World Scientific, v. 21, n. 04, p. 319–334, 2011.

RANA, S.; FANGUEIRO, R. *Fibrous and textile materials for composite applications*. [S.l.]: Springer, 2016.

RAO, S. S. *Engineering optimization: theory and practice*. [S.l.]: John Wiley & Sons, 2009.

RAO, S. S.; YAP, F. F. *Mechanical vibrations*. [S.l.]: Prentice Hall Upper Saddle River, 2011. v. 4.

REDDY, J. N. *Mechanics of laminated composite plates and shells: theory and analysis*. [S.l.]: CRC press, 2004.

REVELS, J.; LUBIN, M.; PAPAMARKOU, T. Forward-mode automatic differentiation in julia. *arXiv preprint arXiv:1607.07892*, 2016.

ROBERTSHAW, T. *Introduction to Machine Learning with Naive Bayes*. 2015. <<https://tomrobertshaw.net/2015/12/introduction-to-machine-learning-with-naive-bayes/>>. Accessed: 2018-10-18.

SCHWARZ, B. J.; RICHARDSON, M. H. Experimental modal analysis. *CSI Reliability week*, Orlando FL, v. 35, n. 1, p. 1–12, 1999.

SHA, W.; EDWARDS, K. The use of artificial neural networks in materials science based research. *Materials & design*, Elsevier, v. 28, n. 6, p. 1747–1752, 2007.

SHARMA, S. *Activation Functions: Neural Networks*. 2017. <<https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>>. Accessed: 2018-11-23.

SILBERSCHMIDT, V. V. *Dynamic deformation, damage and fracture in composite materials and structures*. [S.l.]: Woodhead Publishing, 2016.

SILVA, I. B. V. da; ADEODATO, P. J. Pca and gaussian noise in mlp neural network training improve generalization in problems with small and unbalanced data sets. In: IEEE. *Neural networks (IJCNN), the 2011 international joint conference on*. [S.l.], 2011. p. 2664–2669.

SILVA, I. d.; SPATTI, D. H.; FLAUZINO, R. A. Redes neurais artificiais para engenharia e ciências aplicadas. *São Paulo: Artliber*, p. 33–111, 2010.

SIMARD, P. Y. *et al.* Best practices for convolutional neural networks applied to visual document analysis. In: *ICDAR*. [S.l.: s.n.], 2003. v. 3, p. 958–962.

SIMULIA. *Shell elements*. 2013. <<http://130.149.89.49:2080/v6.12/books/usb/default.htm?startat=pt06ch29s06abo27.html>>. Accessed: 2019-02-20.

SMITH, S. *The global composite materials market is expected to reach an estimated \$38.0 billion by 2023 and it is forecast to grow at a CAGR of 4.1% from 2018 to 2023*. 2018. <<https://www.prnewswire.com>>. Accessed: 2019-01-10.

SPECKMANN, H.; HENRICH, R. Structural health monitoring (shm)—overview on technologies under development. In: *Proc. of the World Conference on NDT, Montreal-Canada*. [S.l.: s.n.], 2004.

TALREJA, R.; SINGH, C. V. *Damage and failure of composite materials*. [S.l.]: Cambridge University Press, 2012.

TALREJA, R.; VARNA, J. *Modeling Damage, Fatigue and Failure of Composite Materials*. [S.l.]: Elsevier, 2015.

VIDHYA, T. A. *Practical Guide to Principal Component Analysis (PCA) in R Python*. 2016. <<https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/#comment-129382>>. Accessed: 2018-12-10.

WEBB, A. R. *Statistical pattern recognition*. [S.l.]: John Wiley & Sons, 2003.

WORDEN, K.; MANSON, G. The application of machine learning to structural health monitoring. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, The Royal Society London, v. 365, n. 1851, p. 515–537, 2006.

YUAN, F.-G. *Structural Health Monitoring (SHM) in Aerospace Structures*. [S.l.]: Woodhead Publishing, 2016.

ZANG, C.; IMREGUN, M. Combined neural network and reduced frf techniques for slight damage detection using measured response data. *Archive of Applied Mechanics*, Springer, v. 71, n. 8, p. 525–536, 2001.

ZANG, C.; IMREGUN, M. Structural damage detection using artificial neural networks and measured frf data reduced via principal component projection. *Journal of sound and vibration*, Elsevier, v. 242, n. 5, p. 813–827, 2001.

ZHANG, Z.; CASTELLÓ, A. Principal components analysis in clinical studies. *Annals of translational medicine*, AME Publications, v. 5, n. 17, 2017.

ZHANG, Z.; FRIEDRICH, K. Artificial neural networks applied to polymer composites: a review. *Composites Science and technology*, Elsevier, v. 63, n. 14, p. 2029–2044, 2003.

ZHAO, R. *et al.* Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, Elsevier, v. 115, p. 213–237, 2019.

ZHU, W. *et al.* Sensitivity, specificity, accuracy, associated confidence interval and roc analysis with practical sas implementations. *NESUG proceedings: health care and life sciences, Baltimore, Maryland*, v. 19, p. 67, 2010.

Appendix A

Graphics

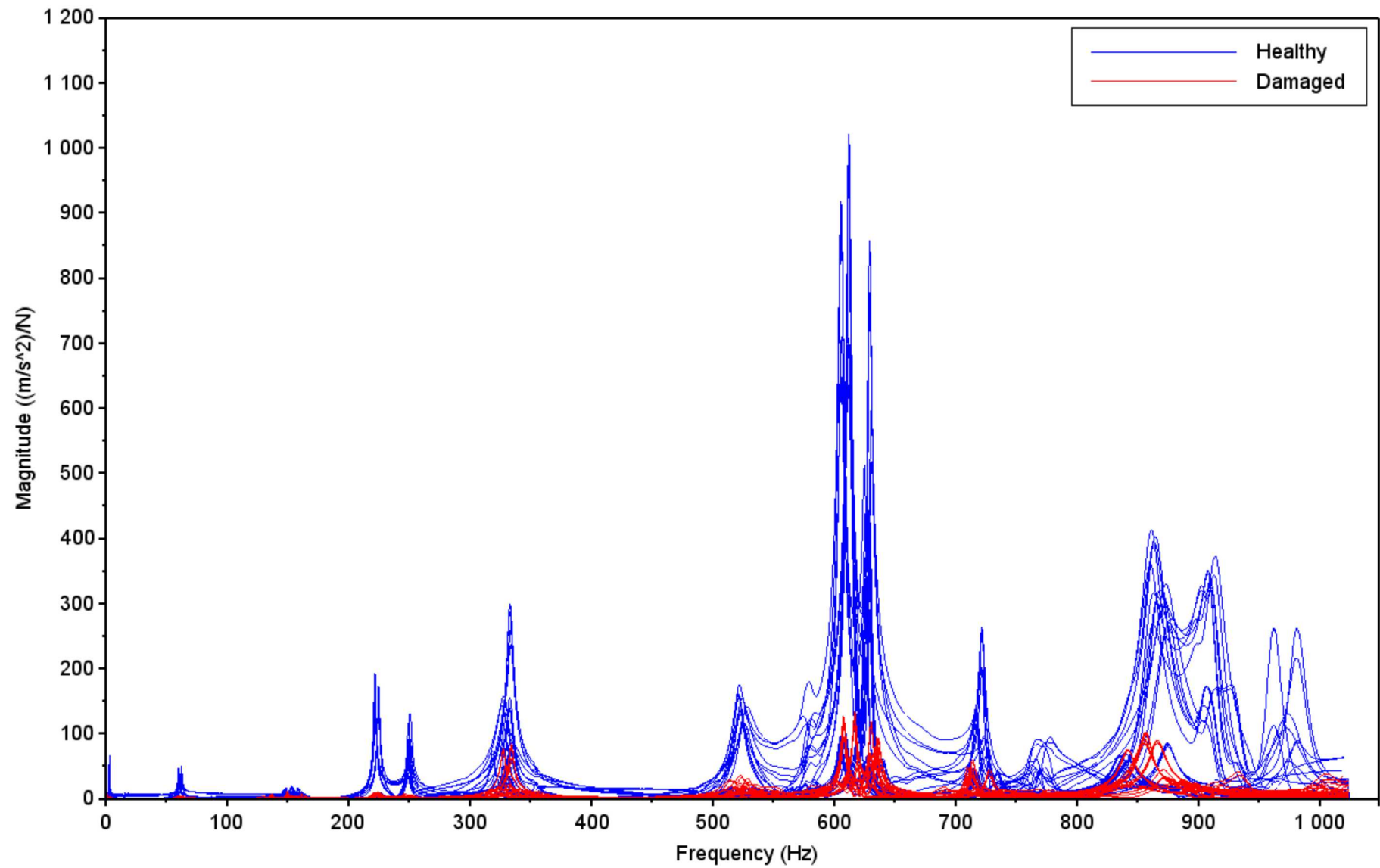


Figure A.1 – Total FRFs from healthy and damaged plates for $[0]_8$ (magnitude part).

Source: Author's production.

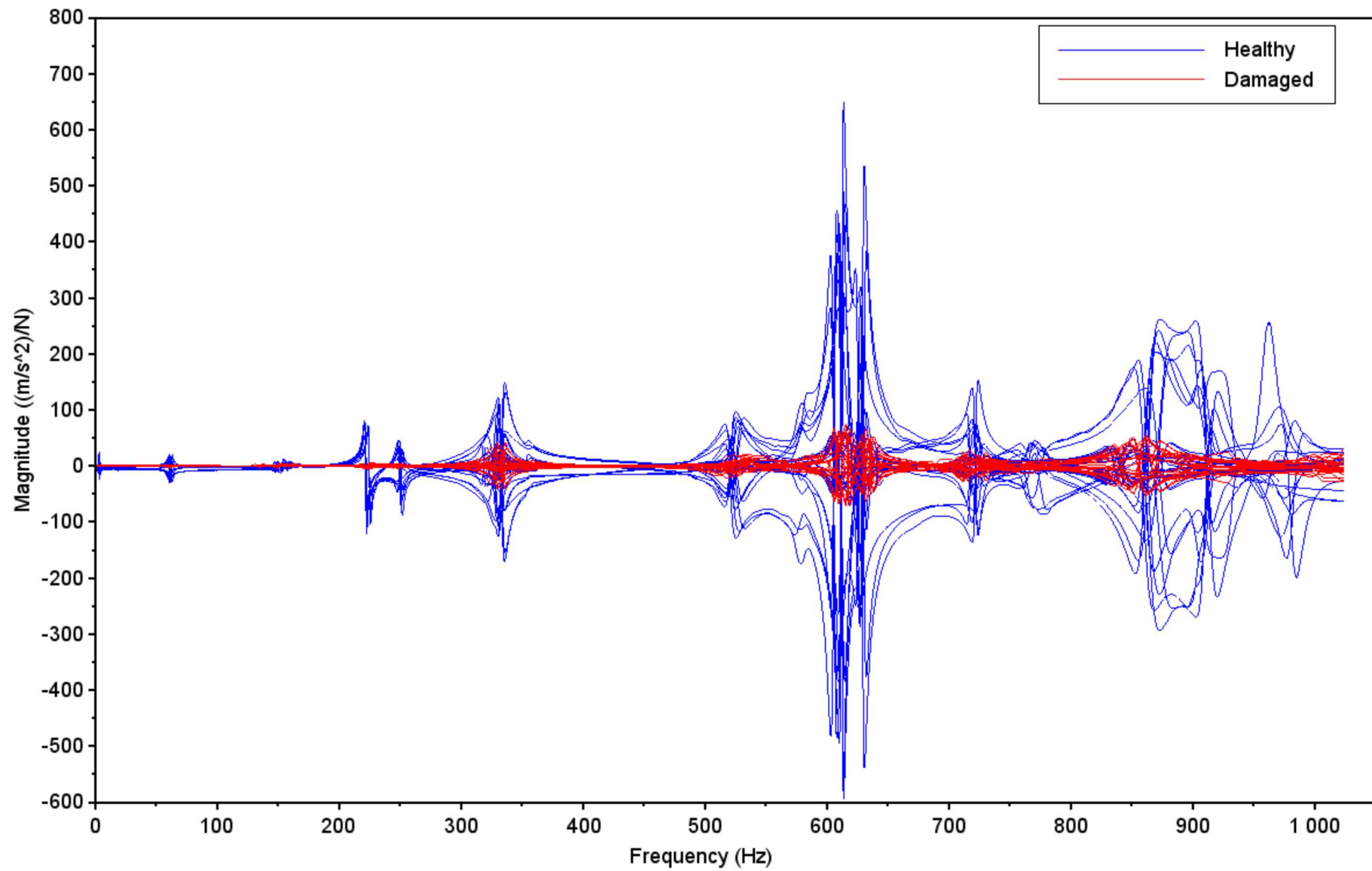


Figure A.2 – Total FRFs from healthy and damaged plates for $[0]_8$ (real part).

Source: Author's production.

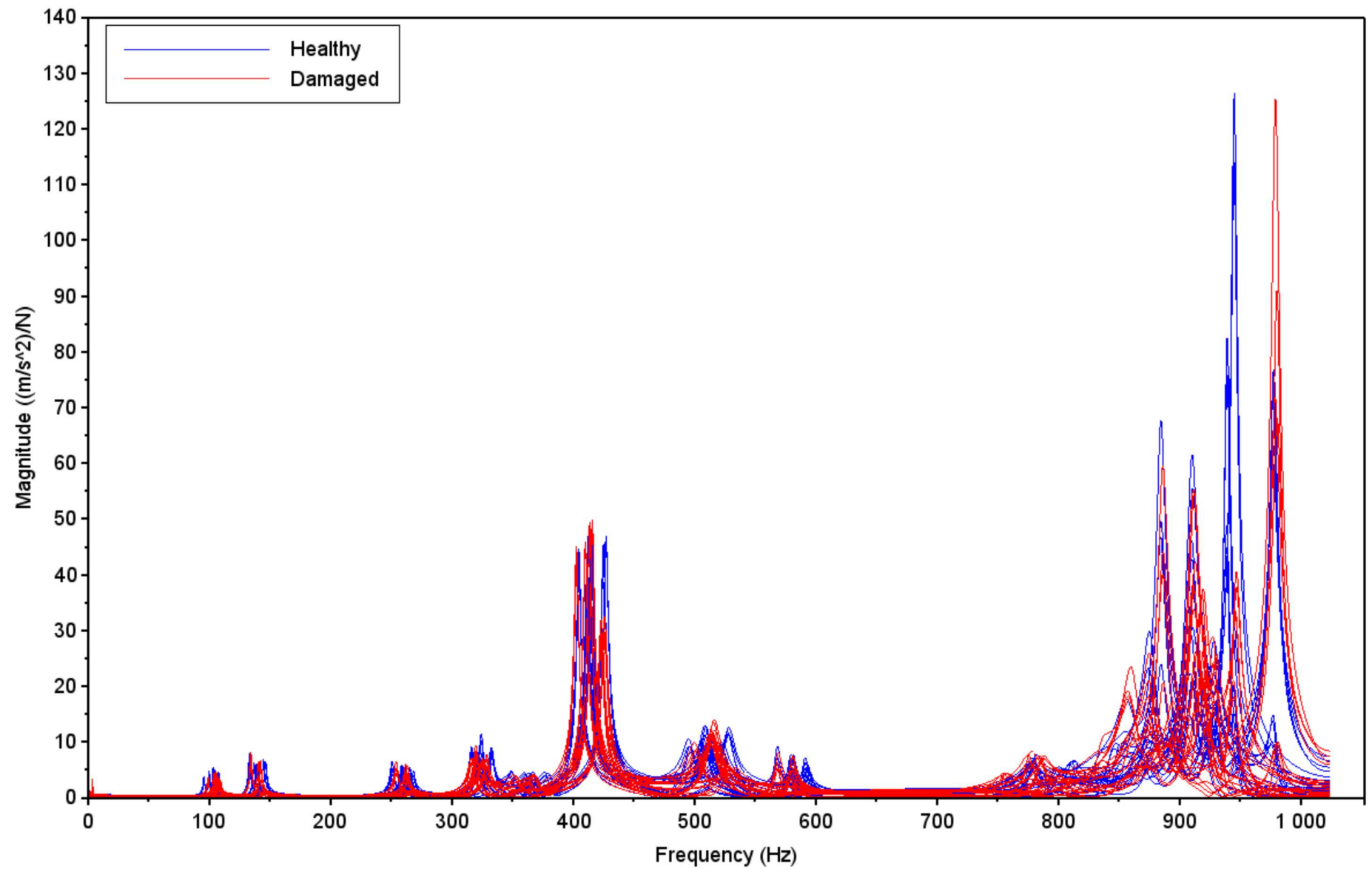


Figure A.3 – Total FRFs from healthy and damaged plates for $[0/15/-15/0/15/-15]_S$ (magnitude part).

Source: Author's production.

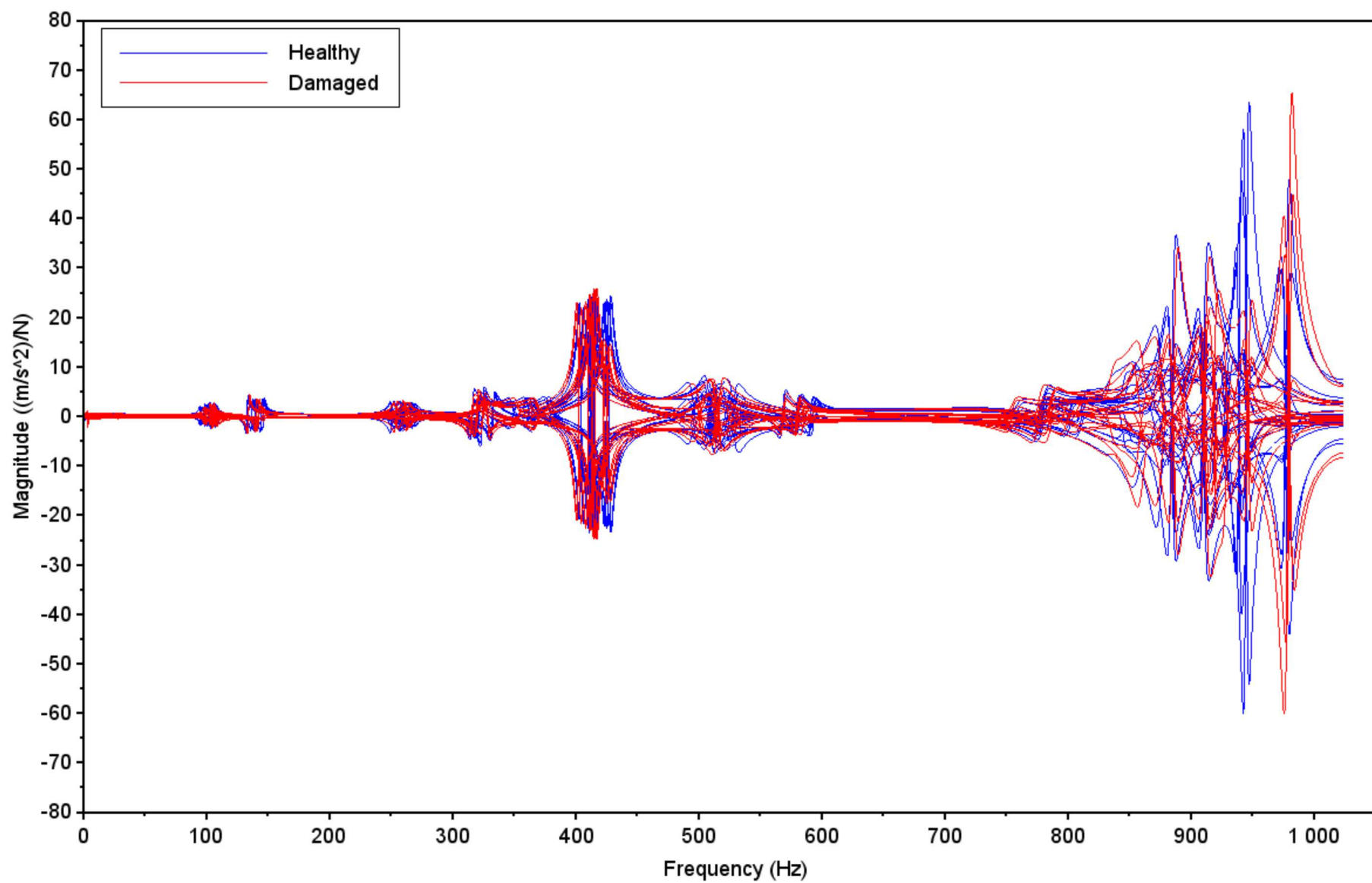


Figure A.4 – Total FRFs from healthy and damaged plates for $[0/15/-15/0/15/-15]_S$ (real part).

Source: Author's production.

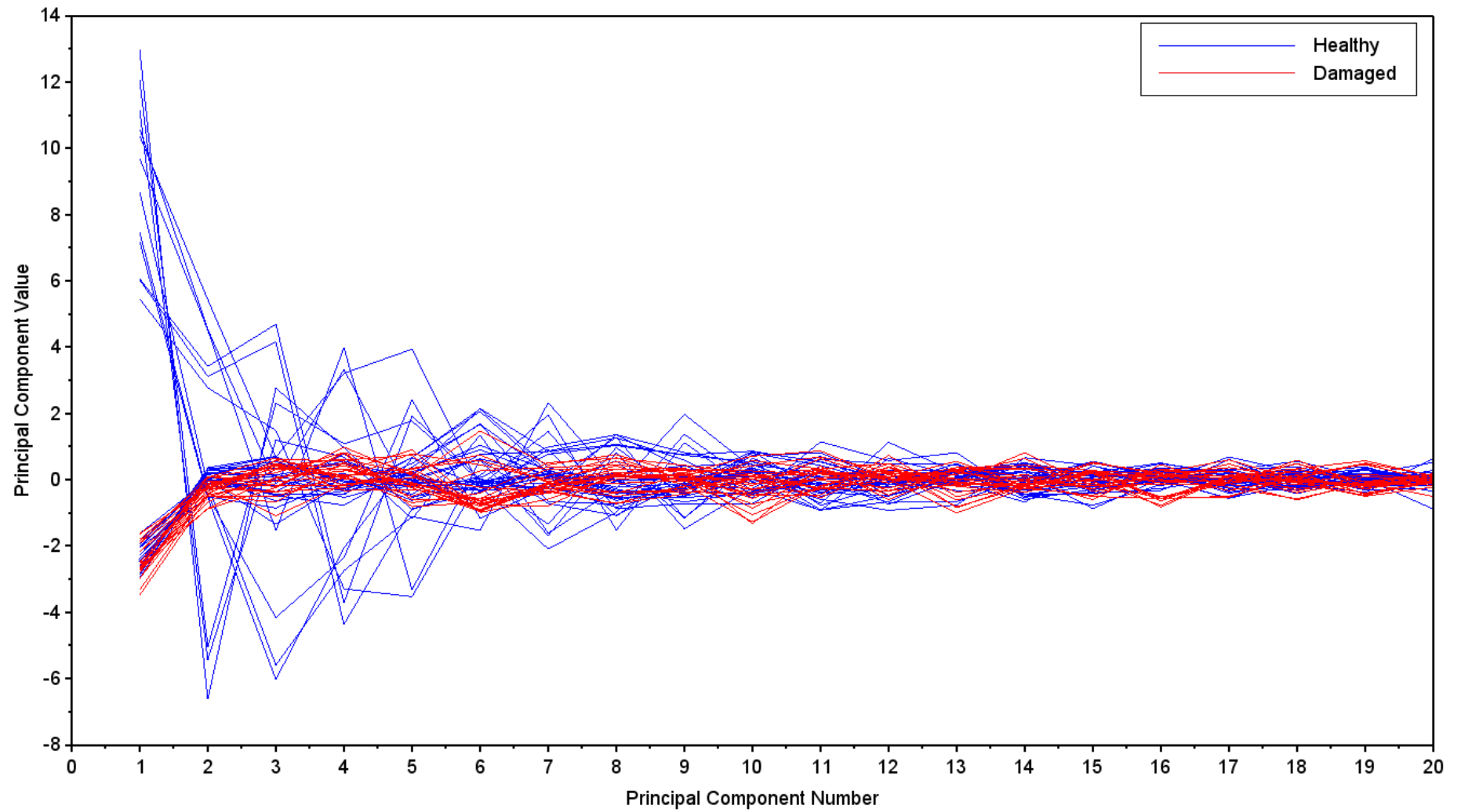


Figure A.5 – Total PCs curves from healthy and damaged plates for $[0]_8$ (magnitude part).

Source: Author's production.

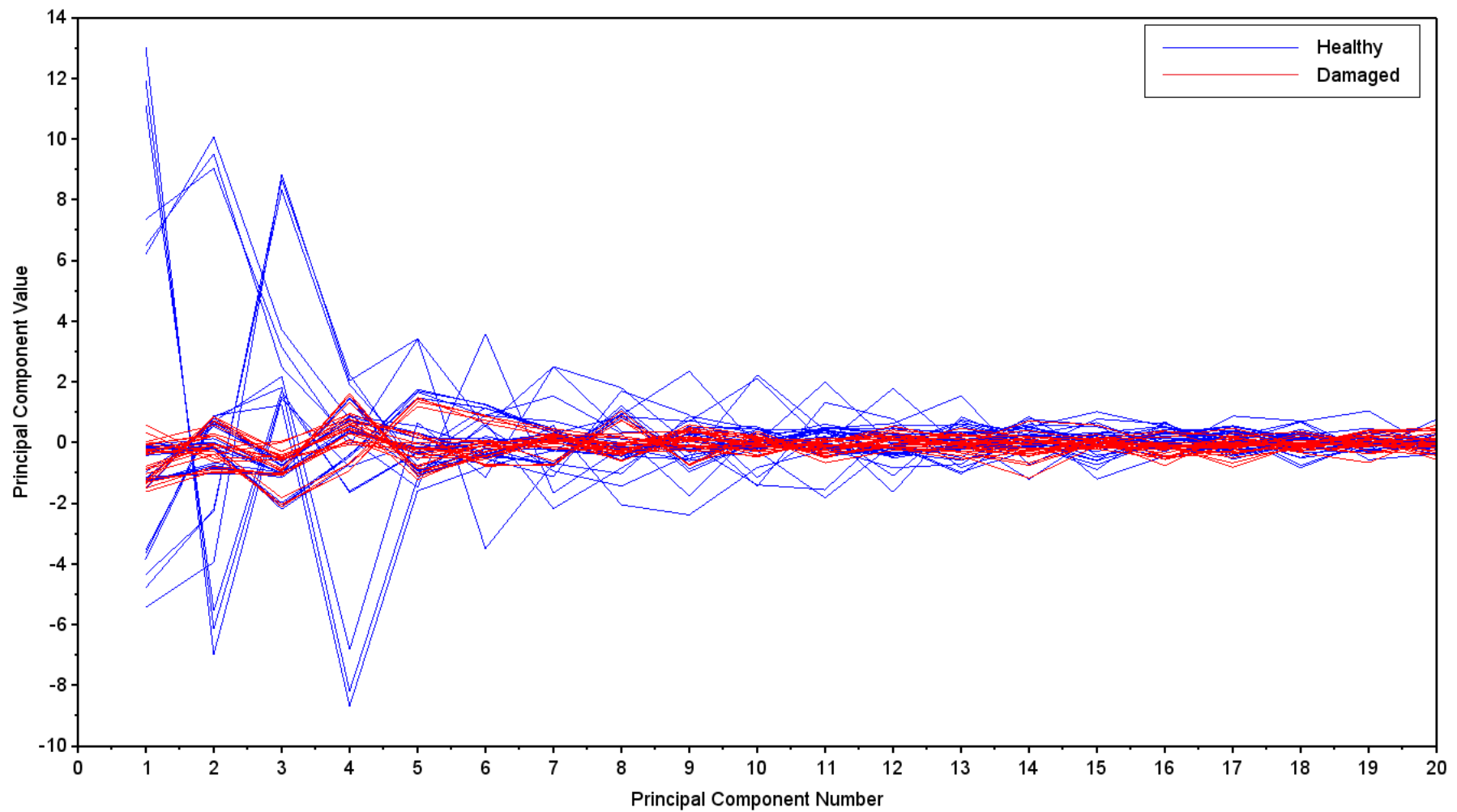


Figure A.6 – Total PCs curves from healthy and damaged plates for $[0]_8$ (real part).

Source: Author's production.

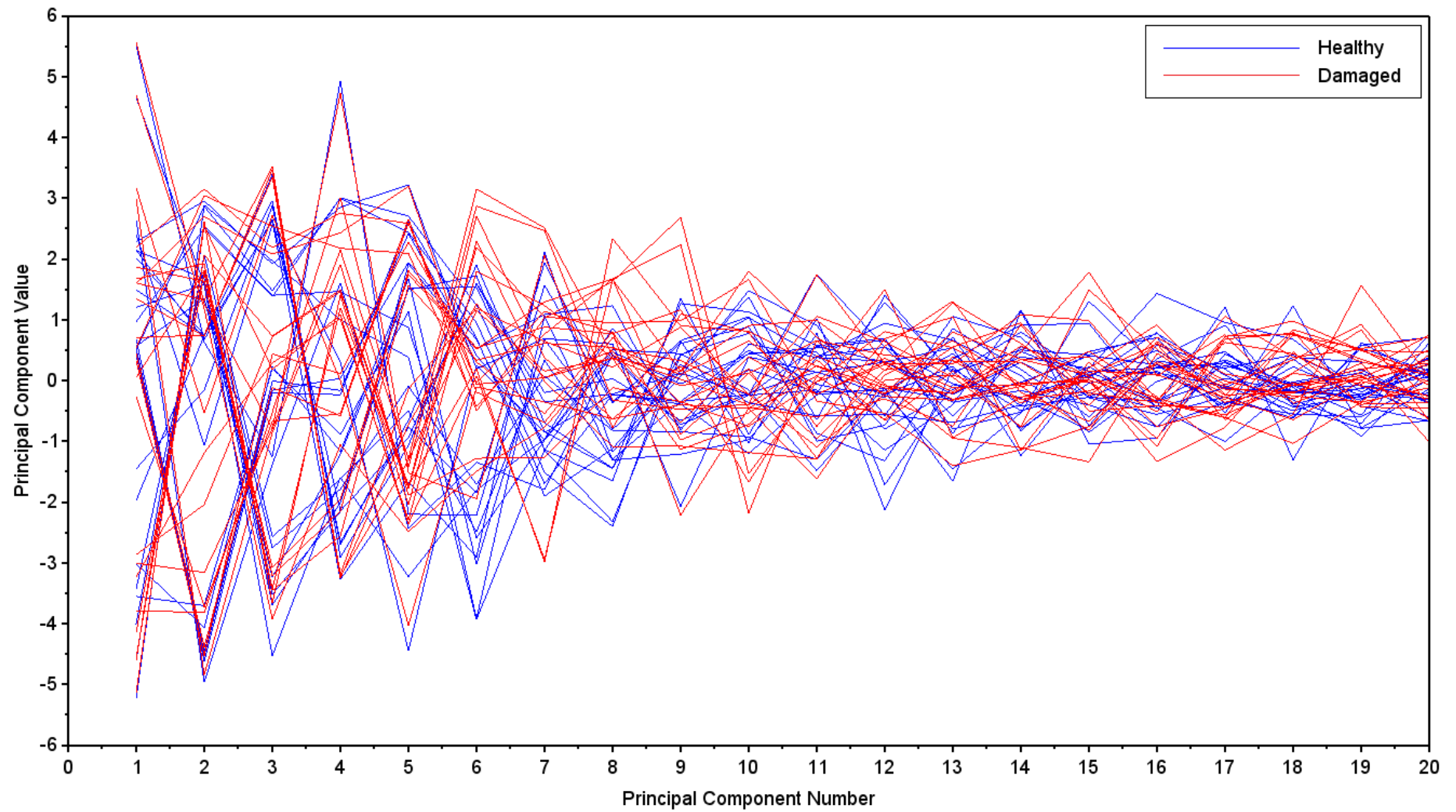


Figure A.7 – Total PCs curves from healthy and damaged plates for $[0/15/-15/0/15/-15]_S$ (magnitude part).

Source: Author's production.

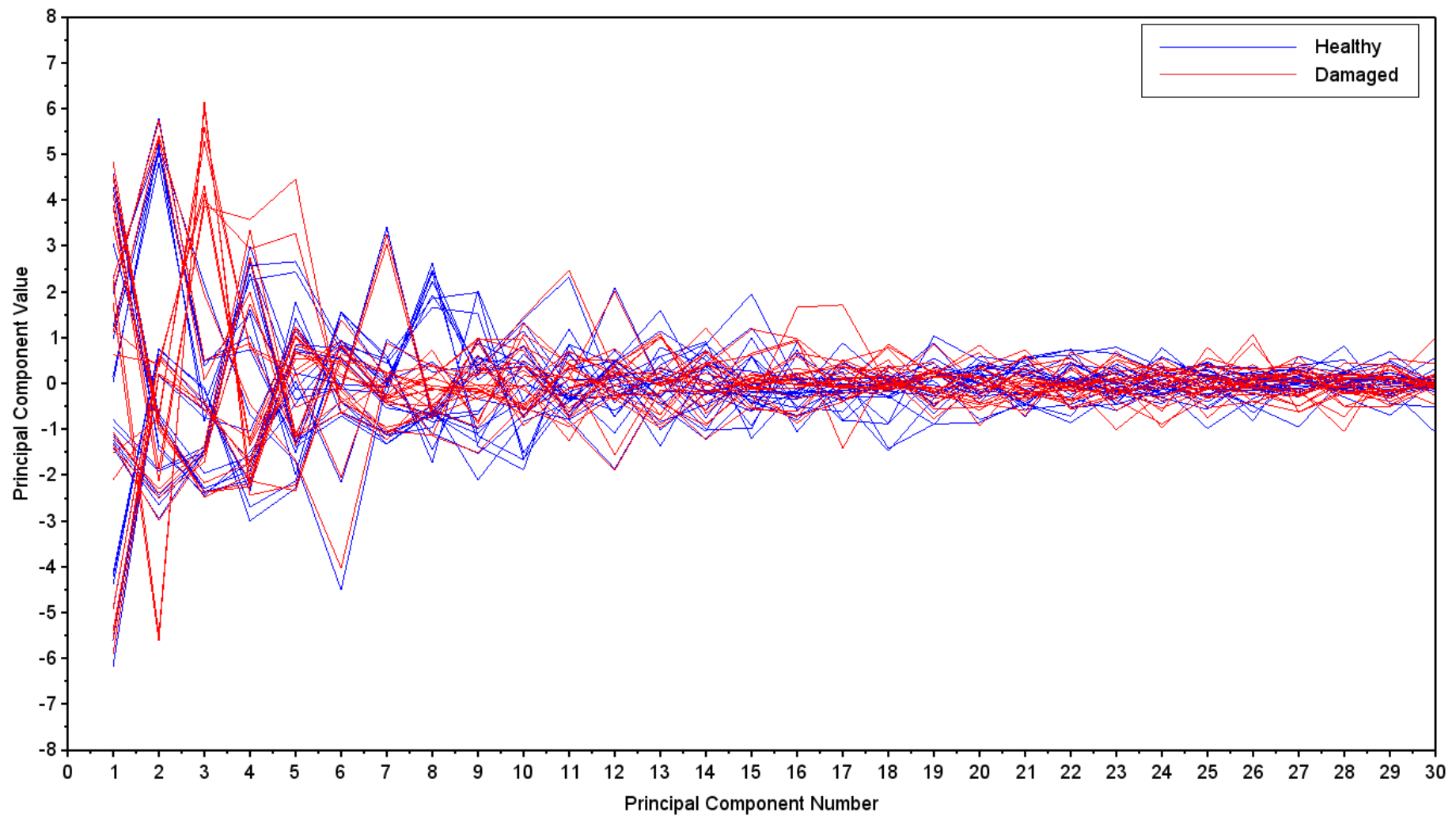


Figure A.8 – Total PCs curves from healthy and damaged plates for $[0/15/-15/0/15/-15]_S$ (real part).

Source: Author's production.

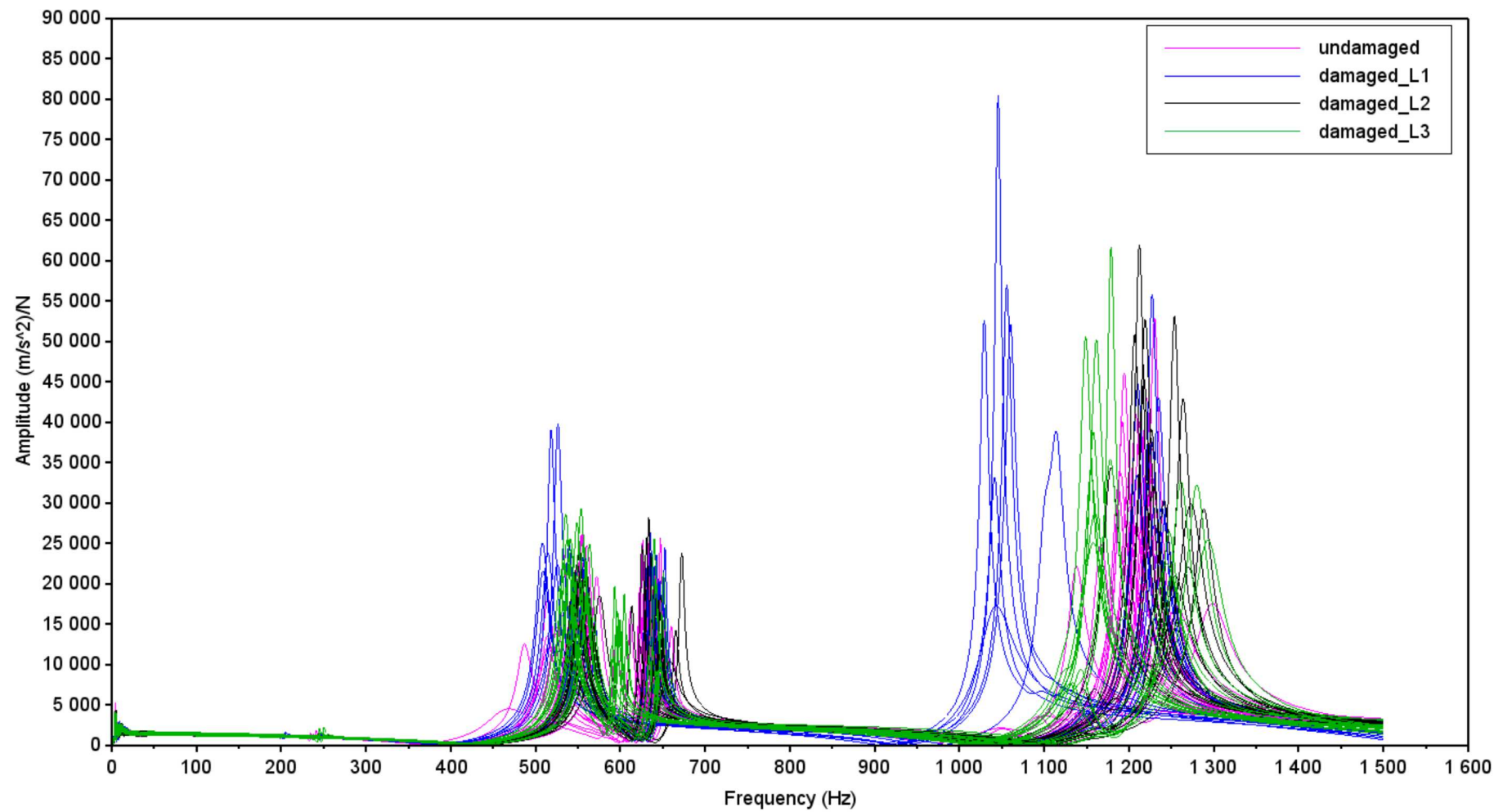


Figure A.9 – Experimental FRFs from all damaged cases: position H_{11} .

Source: Author's production.

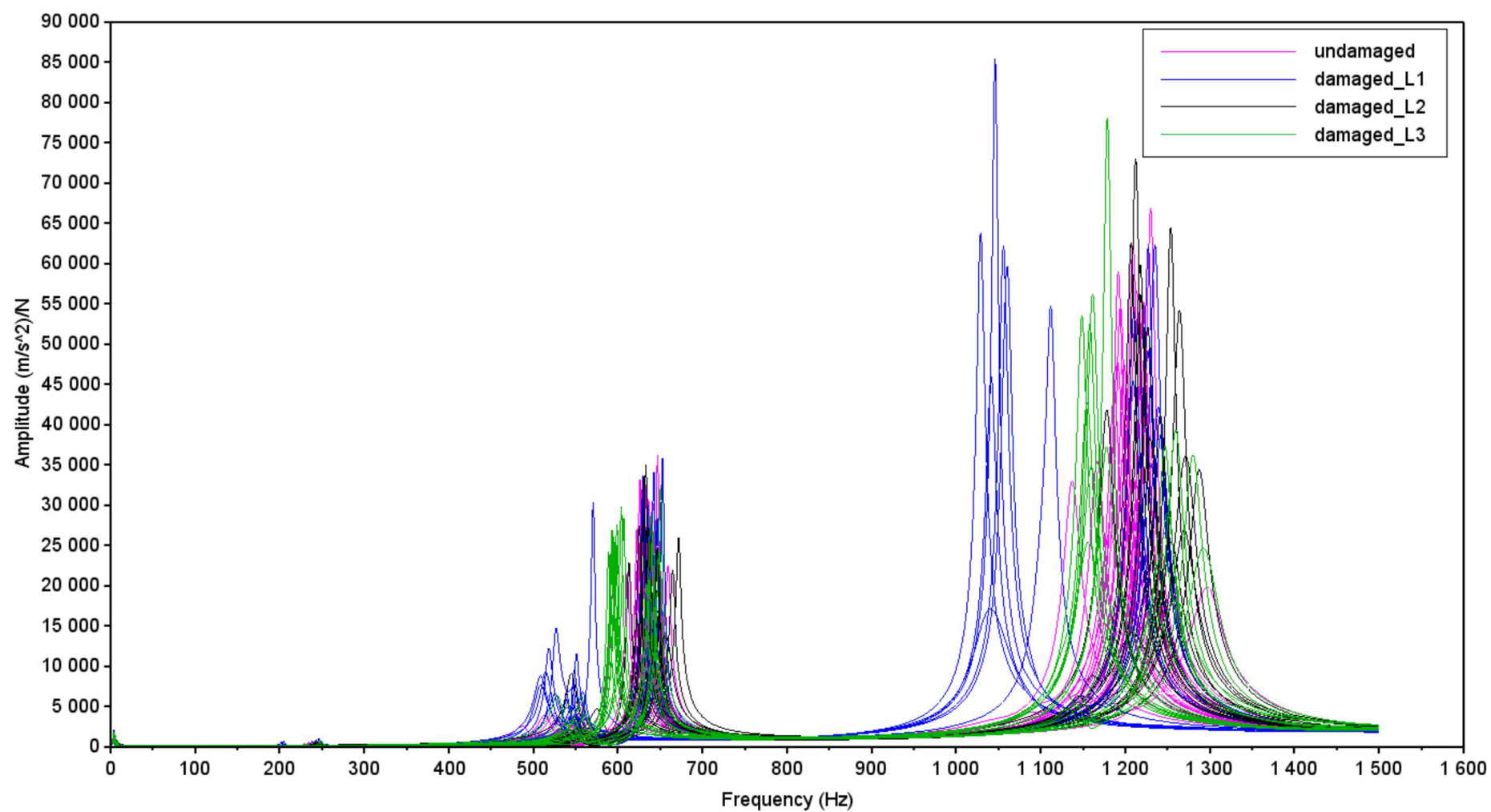


Figure A.10 – Experimental FRFs from all damaged cases: position H_{21} .

Source: Author's production.

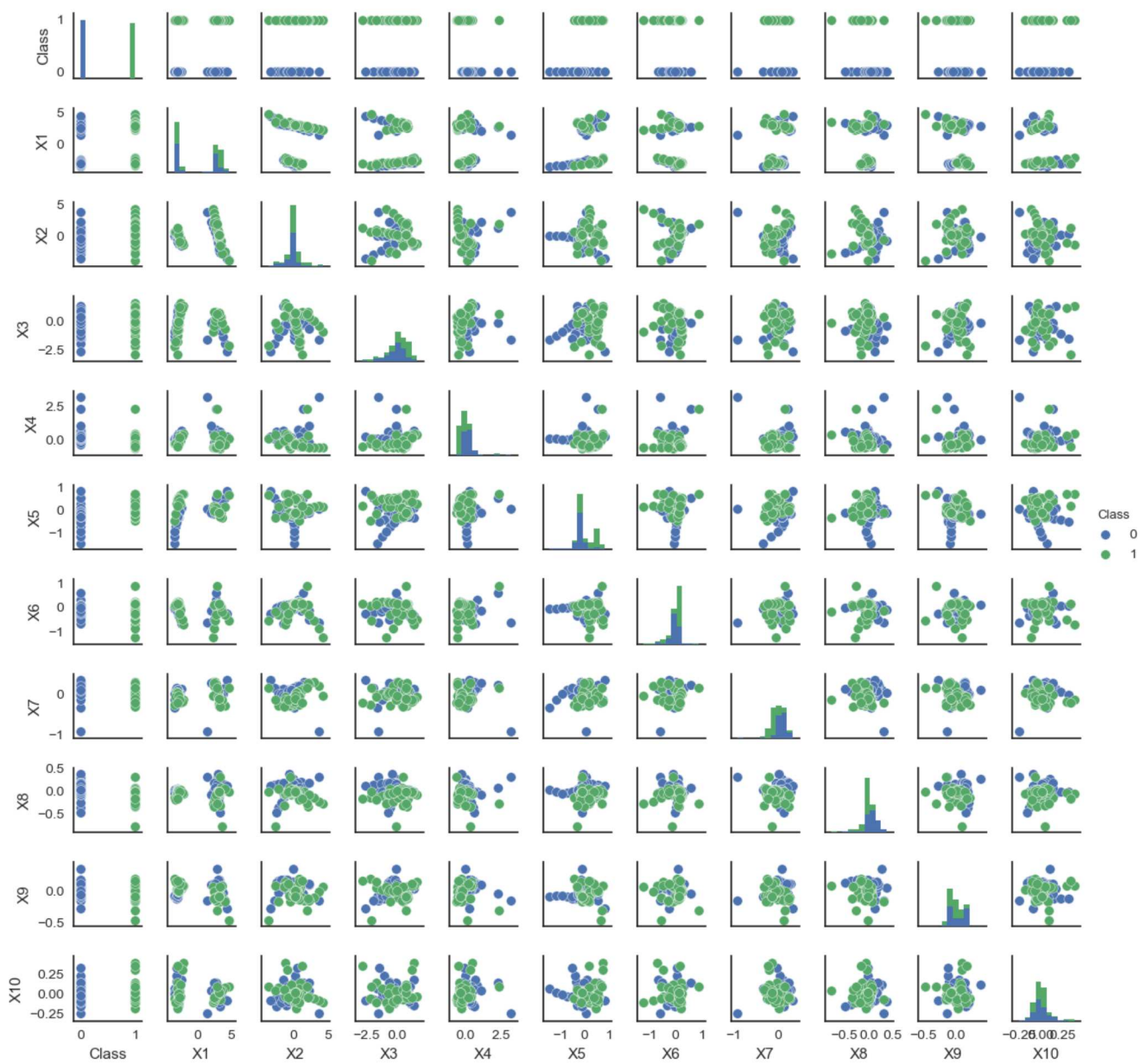


Figure A.11 – Scatter plot (0–healthy and 1–damaged level 1 classes) - numerical.

Source: Author's production.

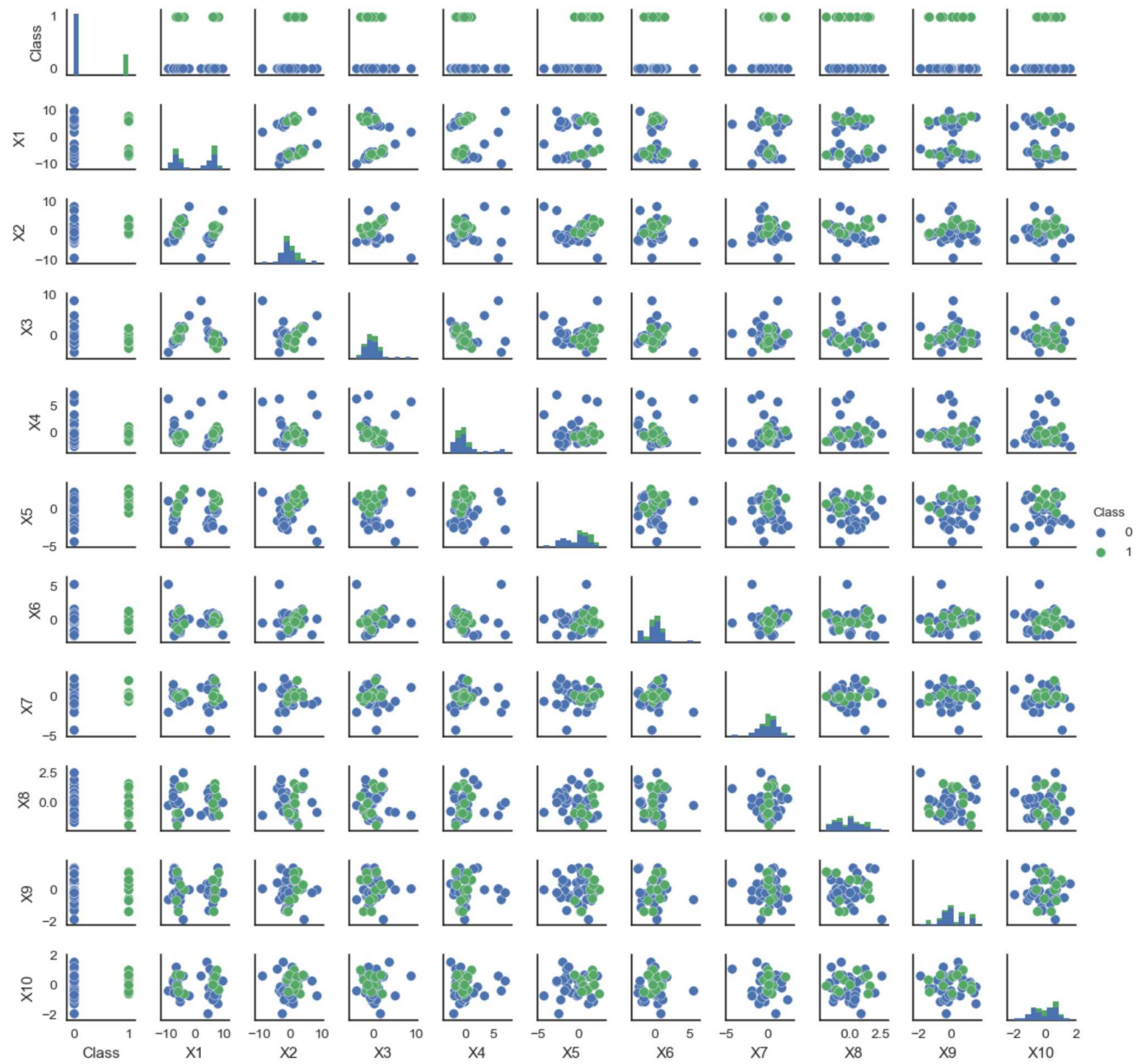


Figure A.12 – Scatter plot (0–healthy and 1–damaged level 1 classes) - experimental (case III).

Source: Author's production.

Appendix B

Fault Diagnosis Methodology Algorithms

```

1  # PROBLEM DEFINITION PSO
2  include("rotinas_RN.jl")
3  include("rotinas_otimizacao.jl")
4  include("otimizacao_RN.jl")
5  using Distributions
6
7  Rede = (LC,p) -> MAIN(LC,p,dados_treinoN,resultados_treinoN,dados_validacao,resultados_validacao,
8                      dados_teste,resultados_teste,[],[],1E-8,0.3,1.0,1000,1,1,0.10,false,0.0)
9
10 function f(x,p)
11     # A primeira posio indica o numero de camadas
12     ncamadas = Int(round(x[1]))
13     nent = size(dados_treinoN,2)
14     nsaida = size(resultados_treinoN,2)
15     # A primeira posio o nmero de entradas
16     LC = Int64[]
17     push!(LC,nent)
18     # Adicionamos camadas de acordo com o seu nmero e seu valor
19     for i=2:1+ncamadas
20         push!(LC,Int(round(x[i])))
21     end
22     # A ultima posio o nmero de saidas
23     push!(LC,nsaida) # apenas 4 neuronio
24     # Agora podemos chamar a rede neural e retornar a informao que interessa
25     pesos, bias, objetivo, vali, melhor_vali, acTeste = Rede(LC,p)
26     # Como maximizao, ento invertermos o sinal
27     return -vali
28 end
29
30 #PSO
31 include("dif_automatizada.jl")

```



```

32 using FD
33 include("definicao_problema.jl")
34
35 function PSO()
36     #define as restrições laterais e o número de variáveis
37     xl= [1,1,1] #x_low
38     xu= [2,34,34] #x_high
39     nvar= 3 #número de variáveis
40     #define os parâmetros do PSO
41     C1= 1.2
42     C2= 1.2 #peso do grupo (global_best)
43     w= 0.5 #fator de inércia
44     NI= 5 #número de iterações
45     NP= 100 #número de partículas
46     #matriz de partículas iniciais
47     X= zeros(nvar,NP) #cria a matriz com as partículas iniciais, linhas: nvar, colunas:NP
48     for i=1:nvar
49         for j=1:NP
50             X[i,j]= xl[i]+rand()*(xu[i]-xl[i])
51         end
52     end
53     #para cada partícula, calcula a função OBJETIVO (número de acurácia)
54     val=zeros(NP,1)
55     for p=1:NP
56         val[p]=f(X[:,p],p)
57     end
58     XPB = copy(X) #X_personal_best
59     valPB = copy(val) #valor da função objetivo para aquela xpb
60     #cálculo do global_best
61     valor_global_best=255E255
62     XGB=zeros(size(nvar,1))
63     for p=1:NP
64         if val[p] < valor_global_best
65             valor_global_best=val[p]
66             XGB=X[:,p]
67         end
68     end
69     #cálculo da velocidade das partículas
70     V=zeros(nvar,NP)
71     for iteracao=1:NI
72         #cálculo da nova velocidade de cada partícula
73         for p=1:NP
74             V[:,p]=w*rand()*V[:,p] + C1*rand()*(XPB[:,p]-X[:,p]) + C2*rand()*(XGB-X[:,p])
75         end
76         #atualiza a posição da partícula
77         for p=1:NP
78             X[:,p]=X[:,p]+V[:,p]

```

```

79         end #for
80         #verifica e aplica as restricoes x_low e x_high
81         for i=1:nvar
82             for j=1:NP
83                  $X[i,j] = \max(xl[i], \min(X[i,j], xu[i]))$ 
84             end #for j
85         end #for i
86         #avalia se a particula melhorou
87         for p=1:NP
88             #novo valor para a funcao objetivo
89             val[p]=f(X[:,p],p)
90             #verifica a melhora da particula
91             if val[p] < valPB[p]
92                 valPB[p] = val[p]
93                 XPB[:,p] = X[:,p]
94             end #if
95             #avalia se o global best melhorou
96             if val[p] < valor_global_best
97                 valor_global_best = val[p]
98                 XGB=X[:,p]
99             end #if
100         end#loop_p
101     end#loop_iteracao
102 end#function

```

```

1  #Otimizacao_RN
2  include("rotinas_RN.jl")
3  include("inicializacao_pesos.jl")
4  include("rotinas_otimizacao.jl")
5
6  #carrega os dados
7  dados_treinoN=readdlm("dados_4r_treino1_PCA35.txt")
8  resultados_treinoN=readdlm("resultados4_treino1_tanh.txt")
9  #VALIDACAO
10 dados_validacao=readdlm("dados_4r_vali1_PCA35.txt")
11 resultados_validacao=readdlm("resultados4_validacao1_tanh.txt")
12 #TESTE
13 dados_teste=readdlm("dados_4r_teste1_PCA35.txt")
14 resultados_teste=readdlm("resultados4_teste1_tanh.txt")
15 pesosx=[]
16 biasx=[]
17 # Topologia da rede
18 LC = [35,20,10,4]
19
20 function MAIN(LC,dados_treinoN,resultados_treinoN,dados_validacao,resultados_validacao, dados_teste,
21             resultados_teste, pesos=[],bias=[], tol=1E-8, momento=0.7, faixa=5.0, Niter = 300,
22             externo=1, tipo_objetivo=2, tolaccuracy::Float64=0.10,flag_ruido::Bool=true,

```

```

23         k_ruido::Float64=0.05)
24
25     # Vetor com os valores. As primeiras posies so de entrada
26     valores = zeros(sum(LC))
27     # O vetor de bias tambm fcil de dimensionar, pois
28     numero_bias = sum(LC[2:end])
29     # Se os bias no foram informados, definimos um vetor aleatrio
30     if bias==[]
31         bias = [ -faixa + 2*faixa*rand() for i=1:numero_bias]
32     else
33         # Verifica se a dimenso do bias que foi informado consistente com a topologia da rede
34         if size(bias,1)!=numero_bias
35             error("A dimensao do bias no consistente com os dados do problema")
36         end
37     end
38     # O vetor de pesos depende das conectividades. Se considerarmos conectividade total teremos
39     numero_pesos = 0
40     for i=2:length(LC)
41         numero_pesos = numero_pesos + LC[i]*LC[i-1]
42     end
43     # Se os pesos no foram informados, definimos um vetor aleatrio
44     topologia = []
45     if pesos==[]
46         pesos, topologia, numero_pesos = Inicializa_Pesos(LC)
47     else
48         # Verifica se a dimenso do pesos que foi informado consistente com a topologia da rede
49         if size(pesos,1)!=numero_pesos
50             error("A dimensao do vetor de pesos no consistente com os dados do problema")
51         end
52     end
53     # Converte as entradas para que elas tenham media nula
54     dados_treinoN= Media_Nula(dados_treinoN)
55     dados_validacao = Media_Nula(dados_validacao)
56     dados_teste = Media_Nula(dados_teste)
57
58     ##### LOOP DE OTIMIZAO #####
59     # Incrementos
60     const delta_pesos = 0.0
61     const delta_bias = 0.0
62
63     for iter=1:Niter
64         # SHUFFLE TODOS OS TREINOS A CADA ITERACAO
65         NPACOTE1, nc = size(dados_treinoN)
66         NPACOTE=80
67         # Gera uma lista aleatoria com a dimenso NPACOTE
68         lista=randperm(NPACOTE)[1:NPACOTE]
69         # Usa a lista para definir os dados de teste e os resultados de teste

```

```

70 dados_treino=dados_treinoN[lista,:];
71 resultados_treino=resultados_treinoN[lista,:];
72 if flag_ruido
73     # Para cada valor de entrada, somamos  $k \cdot N(0,1)$ 
74     aleat = randn(NPACOTE,nc)
75     # Soma  $k_{\text{ruido}} \cdot \text{aleat}$ 
76     dados_treino += k_ruido*aleat
77 end
78
79     # Calcula a derivada do objetivo em relao aos pesos e bias
80     DW, DB = Derivadas_Pesos_Bias(pesos,bias,LC,dados_treino,resultados_treino,tipo_objetivo)
81     # Avalia a norma dos gradientes para verificar a condio necessaria para mnimo
82     norma_DW = norm(DW)
83     norma_DB = norm(DB)
84     if (norma_DW<tol) && (norma_DB<tol)
85         println("\n Tolerancia foi obtida .. saindo do loop de otimizao ")
86         break
87     end
88     # Normaliza as derivadas
89     DW = DW / norma_DW
90     DB = DB / norma_DB
91     # Define as direes de busca
92     dW = -DW
93     dB = -DB
94
95     # Line Search usando o Armijo Backtracing #MUDAR LINESEARCH
96     #alfa = Line_Search_Armijo(DW,DB,dW,dB,pesos,bias,LC,dados_treino,resultados_treino,
97         # tipo_objetivo)
98     alfa=0.05
99
100     # Incremento
101     contrib_pesos = momento*delta_pesos
102     contrib_bias = momento*delta_bias
103     # Incrementos totais momento simples
104     delta_pesos = contrib_pesos + alfa*dW
105     delta_bias = contrib_bias + alfa*dB
106     # Atualiza a posio
107     pesos = pesos + delta_pesos
108     bias = bias + delta_bias
109     # Calcula o novo objetivo, para verificao
110     objetivo = Avalia_Treinos(pesos,bias,LC,dados_treino,resultados_treino,tipo_objetivo)
111     objetivoVali= Avalia_Amostras(pesos,bias,LC,dados_validacao,resultados_validacao,tipo_objetivo)
112     j=0
113     for vv=1:size(dados_treino,1)
114         erro1, saida1 = Treino_individual(pesos,bias,LC,dados_treino[vv,:],resultados_treino[vv,:],
115             tipo_objetivo)
116         if abs(erro1)< tolaccuracy

```

```

117         j=j+1
118     end
119 end #for
120 acT=j/size(dados_treino,1)
121 i=0
122 for v=1:size(dados_validacao,1)
123     erro, saida = Treino_individual(pesos,bias,LC,dados_validacao[v,:],resultados_validacao[v:],
124                                     tipo_objetivo)
125     if abs(erro) < tolaccuracy
126         i=i+1
127     end
128 end #for
129 vali=i/size(dados_validacao,1)
130 k=0
131 for x=1:size(dados_teste,1)
132     errox, saidax = Treino_individual(pesos,bias,LC,dados_teste[x,:],resultados_teste[x:],tipo_objetivo)
133     if abs(errox) < tolaccuracy
134         k=k+1
135     end
136 end #for
137 acTeste=k/size(dados_teste,1)
138 end # iter
139
140 # Avalia o valor final do custo da rede
141 objetivo = Avalia_Amostras(pesos,bias,LC,dados_treinoN, resultados_treinoN,tipo_objetivo)
142 objetivoVali = Avalia_Amostras(pesos,bias,LC,dados_validacao,resultados_validacao,tipo_objetivo)
143
144 # Agora vamos carregar um conjunto de dados de teste e realizar uma verificacao sobre a rede otimizada
145 if length(dados_validacao)>0 && length(resultados_validacao)>0
146     # Para cada linha de verificacao, e com os pesos e bias otimizados
147     # calculamos um padro de saida.
148     j=0
149     for vv=1:size(dados_treinoN,1)
150         erro1, saida1 = Treino_individual(pesos,bias,LC,dados_treinoN[vv,:],resultados_treinoN[vv:],
151                                           tipo_objetivo)
152         if abs(erro1) < tolaccuracy
153             j=j+1
154         end
155     end #for
156     acT=j/size(dados_treinoN,1)
157     i=0
158     for v=1:size(dados_validacao,1)
159         erro, saida = Treino_individual(pesos,bias,LC,dados_validacao[v,:],resultados_validacao[v:],
160                                         tipo_objetivo)
161         if abs(erro) < tolaccuracy
162             i=i+1
163         end

```

```

164     end #for
165     vali=i/size(dados_validacao,1)
166     k=0
167     for x=1:size(dados_teste,1)
168         errox, saidax = Treino_individual(pesos,bias,LC,dados_teste[x,:],resultados_teste[x,:],
169                                         tipo_objetivo)
170         if abs(errox) < tolaccuracy
171             k=k+1
172         end
173     end #for
174     acTeste=k/size(dados_teste,1)
175 end # if
176 return pesos, bias, objetivo, vali, melhor_vali, acTeste, acT
177 end

```

```

1  #inicializacao_pesos
2  function Inicializa_Pesos(LC::Array{Int64})
3      # Numero de pesos
4      const numero_pesos::Int64 = 0
5      for i=2:length(LC)
6          numero_pesos = numero_pesos + LC[i]*LC[i-1]
7      end
8      # Vetor de pesos
9      pesos = zeros(numero_pesos)
10     # Para cada neurônio temos que determinar quais são as posições (pesos) de entrada, para então verificar o seu
11     # número. Com isto, geramos uma distribuição normal com média nula e desvio igual a 1/raiz(numero_entradas)
12     # Cria uma lista de topologias, que tem as posições de cada neurônio na rede na forma neurônio =
13     # [camada pos_inicial pos_final]
14     topologia = []
15     # Faixa de valores de entrada da camada
16     const inicio::Int64 = 1
17     const fim::Int64 = LC[1]
18     # offset da posição dos pesos
19     const offset::Int64 = 0
20     # Loop por cada camada
21     const contador_bias::Int64 = 1
22     const contador_saida::Int64 = LC[1]+1
23     @inbounds for camada=1:length(LC)-1
24         # Para cada neurônio, extraí os pesos e bias e calcula a saída
25         @inbounds for neurônio=1:LC[camada+1]
26             # Posição inicial e final dos pesos deste neurônio no vetor de pesos
27             const pos_ini::Int64 = offset + LC[camada]*(neurônio-1)+1
28             const pos_fin::Int64 = offset + LC[camada]*(neurônio-1)+LC[camada]
29             # Numero de entradas
30             numero_entradas = pos_fin - pos_ini + 1
31             #distribuição de tal forma que tenha média nula e desvio 1/sqrt(numero_entradas)
32             distrib = Normal(0.0, 1.0/sqrt(numero_entradas))

```

```

33         # Gera valores nesta faixa
34         valores = rand(distrib,numero_entradas)
35         # Copia para o vetor de pesos
36         pesos[pos_ini:pos_fin] = valores
37         push!(topologia,[camada pos_ini pos_fin])
38     end #Neuronio
39     # Corrige o offset dos pesos
40     offset = offset + LC[camada]*LC[camada+1]
41     # Atualiza estes offsets das entradas
42     inicio = fim+1
43     fim = fim+LC[camada+1]
44 end #camada
45 return pesos, topologia, numero_pesos
46 end

```

```

1  #Rotinas_RN
2  # Carrega as rotinas de diferenciacao automatica
3  include("dif_automatica.jl")
4  using FD
5
6  function Verifica_Consistencia_Treinos(LC::Array{Int64}, entradas::Array, saidas::Array ,
7                                     dados_verificacao::Array, resultados_verificacao::Array )
8      # Primeiro, verificamos se as dimenses esto consistentes
9      numero_entradas = LC[1]
10     numero_saidas = LC[end]
11
12     # Verifica padres de treino
13     numero_treinos_entrada = size(entradas,1)
14     numero_entradas_entrada = size(entradas,2)
15     numero_treinos_saida = size(saidas,1)
16     numero_saidas_saida = size(saidas,2)
17
18     if numero_treinos_entrada != numero_treinos_saida
19         error("\n Numero de treinos (linhas) na entrada $(numero_treinos_entrada) deve ser igual ao "
20             "nmero de treinos de saida $(numero_treinos_saida) ")
21     end
22     if numero_entradas_entrada != numero_entradas
23         error("\n Numero de entradas (colunas) na entrada deve ser igual a primeira posio de "
24             "LC $(numero_entradas) ")
25     end
26     if numero_saidas_saida != numero_saidas
27         error("\n Numero de saidas (colunas) na saida deve ser igual a ltima posio de "
28             "LC $(numero_saidas) ")
29     end
30     if length(dados_verificacao)>0 && length(resultados_verificacao)>0
31         # Verifica se o nmero de verificacoes consistente
32         if size(dados_verificacao,1)!=size(resultados_verificacao,1)

```

```

33         error("\n Dimenses no conincidem na verificacao ")
34     end
35     if size(dados_verificacao,2)!=size(entradas,2)
36         error("\n Dimenso de dados de entrada deve ser igual o de verificacao")
37     end
38     if size(resultados_verificacao,2)!=size(saidas,2)
39         error("\n Dimenso de resultados de entrada deve ser igual o de verificacao")
40     end
41 end
42 end
43
44 ## Funao Sigmoid – Aqui vamos usar a funo logstica
45 function Sigmoid{T}(a::T)
46     const um::T = 1
47     um/(um+exp(-a))
48 end
49
50 #Funcao leaky–relu
51 function relu{T}(a::T)
52     saida = 0.01*a
53     if isa(a,Dual)
54         if a.real > 0
55             saida=a
56         end
57     else
58         if a>0
59             saida=a
60         end
61     end
62     return saida
63 end
64
65
66 function Saida_Neuronio(entradas_neuronio::Array, pesos_neuronio::Array, bias_neuronio,
67                         flag_act::Bool=false)
68     if flag_act
69         return Sigmoid( dot(entradas_neuronio,pesos_neuronio) + bias_neuronio)
70         # relu(dot(entradas_neuronio, pesos_neuronio) + bias_neuronio)
71     else
72         return tanh( dot(entradas_neuronio,pesos_neuronio) + bias_neuronio)
73     end
74 end
75
76 # Rotina que Propaga a informao da camada de entrada para a camada de saida
77 # Os valores de entrada da rede devem estar nas primeiras posies de valores.
78 function RN(valores::Array,bias::Array,pesos::Array,LC::Array{Int64},flag_soft=false)
79     # Faixa de valores de entrada da camada

```



```

80  const inicio::Int64 = 1
81  const fim::Int64 = LC[1] #entradas na primeira camada
82  # offset da posio dos pesos
83  const offset::Int64 = 0
84  # Loop por cada camada
85  const contador_bias::Int64 = 1
86  const contador_saida::Int64 = LC[1]+1
87  @inbounds for camada=1:length(LC)-1 #primeira camada nao tem neuronios so input
88      # Entradas desta camada
89      const entrada_neuronio = valores[inicio:fim] #entradas primeira camada, input
90      # Para cada neuronio, extrai os pesos e bias e calcula a saida
91      @inbounds for neuronio=1:LC[camada+1] #camada 2, 3 e 4
92          # Posicao dos pesos deste neuronio no vetor de pesos
93          const pos1::Int64 = offset + LC[camada]*(neuronio-1)+1
94          const pos2::Int64 = offset + LC[camada]*(neuronio-1)+LC[camada]
95          # Extrai os pesos e o bias
96          const pesos_neuronio = pesos[pos1:pos2]
97          const bias_neuronio = bias[contador_bias]
98          # Calcula a saida deste neuronio (escalar)
99          const saida_neuronio = Saida_Neuronio(entrada_neuronio,pesos_neuronio,bias_neuronio)
100         # Grava no vetor de valores
101         valores[contador_saida] = saida_neuronio
102         # Atualiza o contador de saida
103         contador_saida = contador_saida + 1
104         # Atualiza o contador de bias
105         contador_bias = contador_bias + 1
106     end #Neuronio
107     # Corrige o offset dos pesos
108     offset = offset + LC[camada]*LC[camada+1]
109     # Atualiza estes offsets das entradas
110     inicio = fim+1
111     fim = fim+LC[camada+1]
112 end #camada
113
114 if flag_soft
115     # Se formos aplicar o softmax na sada, pegamos os valores da sada e aplicamos diretamente no Softmax
116     y_s = valores[end-(LC[end]-1):end]
117     soma = sum(exp.(y_s))
118     expo = exp.(y_s)
119     finale = expo ./ soma
120     valores[end-(LC[end]-1):end].=finale
121 end
122 return valores
123 end
124
125 # Funo que recebe um padro de entrada e devolve os valores da rede
126 function Avalia_Padiao(pesos::Array,bias::Array,LC::Array{Int64},padrao::Array)

```

```

127     # Descobre o tipo das entradas
128     T = eltype(pesos)
129     # Copia o padro de entrada para as primeiras posies do vetor de valores
130     const valores = zeros(T, sum(LC))
131     valores[1:LC[1]] = padrao
132     # Calcula os valores restantes para a ANN
133     valores = RN(valores, bias, pesos, LC)
134 end
135
136 # Funo que calcula o valor do objetivo para um conjunto de entradas/saidas
137 function Treino_individual(pesos::Array, bias::Array, LC::Array{Int64},
138                             padrao_entrada::Array, padrao_saida::Array,
139                             tipo_objetivo::Int64)
140     # Avalia o padro de entrada e obtm os valores da rede
141     const valores = Avalia_Padrao(pesos, bias, LC, padrao_entrada)
142     # A saida da rede estar nas posies finais (ultima informacao do LC)
143     const saida = valores[end-LC[end]+1:end]
144     valor_treino = 0.0
145     if tipo_objetivo==0
146         # Norma 2 da diferenca
147         valor_treino = norm(saida-padrao_saida)
148     elseif tipo_objetivo==1
149         # Funcao quadratica
150         valor_treino = (1.0/2.0)*(norm(saida-padrao_saida))^2
151     else
152         # Cross Entropy
153         try
154             valor_treino = dot(padrao_saida, log.(saida)) + dot(1.0-padrao_saida, log.(1.0-saida))
155         catch
156             println("\n Erro no CE ", saida, " ", padrao_saida)
157             error("")
158         end
159     end
160     return valor_treino, saida
161 end
162
163 # Rotina que Aplica todos os treinos na rede – Funo objetivo
164 function Avalia_Treinos(pesos::Array, bias::Array, LC::Array{Int64}, entradas::Array, saidas::Array,
165                          tipo_objetivo::Int64)
166     # Descobre o tipo das entradas
167     T = eltype(pesos)
168     # Numero de entradas (treinos)
169     nt, nc = size(entradas)
170     # Vamos gravar cada resultado em uma poiso de um vetor, para ver
171     # como as coisas se comportam
172     const valores = zeros(T, nt)
173     for teste=1:nt

```

```

174     const padrao_entrada = vec(entradas[teste,:])
175     const padrao_saida = vec(saidas[teste,:])
176     valores[teste], blop = Treino_individual(pesos,bias,LC,padrao_entrada,padrao_saida,tipo_objetivo)
177 end
178 # ADICAO da tecnica de regularizacao L1 ou L2 weight decay
179 lambda = 0.001
180 L2_regularization = ((lambda/(2*nt))*(norm(pesos))^2)
181 L1_regularization = ((lambda/(nt))*(sum(abs.(pesos))))
182 saida = 0.0
183 if tipo_objetivo==0
184     saida = (1.0/nt)*norm(valores) + L1_regularization
185 elseif tipo_objetivo==1
186     saida=(1.0/nt)*sum(valores) + L2_regularization
187 else
188     saida = -(1.0/nt)*sum(valores) + L1_regularization
189 end
190 return saida
191 end
192
193 # ROTINA QUE APLICA FUNCAO OBJETIVO PARA OS CONJUNTOS DE VALIDACAO E TESTE,
194 #SEM REGULARIZACAO
195 function Avalia_Amostras(pesos::Array,bias::Array,LC::Array{Int64},entradas::Array,saidas::Array,
196     tipo_objetivo::Int64)
197     # Descobre o tipo das entradas
198     T = eltype(pesos)
199     # Numero de entradas (treinos)
200     nt, nc = size(entradas)
201     # Vamos gravar cada resultado em uma poiso de um vetor, para ver
202     # como as coisas se comportam
203     const valores = zeros(T,nt)
204     for teste=1:nt
205         const padrao_entrada = vec(entradas[teste,:])
206         const padrao_saida = vec(saidas[teste,:])
207         valores[teste], blop = Treino_individual(pesos,bias,LC,padrao_entrada,padrao_saida,tipo_objetivo)
208     end
209     saidaA = 0.0
210     if tipo_objetivo==0
211         saidaA = (1.0/nt)*norm(valores)
212     elseif tipo_objetivo==1
213         saidaA=(1.0/nt)*sum(valores)
214     else
215         saidaA = -(1.0/nt)*sum(valores)
216     end
217     return saidaA
218 end
219
220 # Rotina que varre perturba os duais dos pesos e dos bias e calcula as derivadas. Para isto, todas as operaes

```

```

221 # devem ser definidas no mdulo FD.
222 function Derivadas_Pesos_Bias(pesos::Array,bias::Array,LC::Array{Int64}, dados_treino::Array,
223                                resultados_treino::Array, tipo_objetivo::Int64)
224 # Para calcularmos as derivadas, precisamos converter todos os dados
225     # para Dual (internamente)
226     const pesos_i = convert(Array{Dual},pesos)
227     const bias_i = convert(Array{Dual},bias)
228     const dados_treino_i = convert(Array{Dual},dados_treino)
229     const resultados_treino_i = convert(Array{Dual}, resultados_treino)
230     # Para cada peso, colocamos 1.0 na parte dual e avaliamos o treino completo
231     const DW = zeros(size(pesos,1))
232     Threads.@threads for i=1:size(pesos,1)
233         # Peturba
234         pesos_i[i] = Dual(pesos[i], 1.0)
235         objetivo = Avalia_Treinos(pesos_i,bias_i,LC,dados_treino_i,resultados_treino_i,tipo_objetivo)
236         # Armazena esta derivada
237         DW[i] = objetivo.dual
238         # Desfaz a perturbao
239         pesos_i[i] = Dual(pesos[i], 0.0)
240     end #i
241     # Para cada bias, colocamos 1.0 na parte dual e avaliamos o treino completo
242     const DB = zeros(size(bias,1))
243     Threads.@threads for i=1:size(bias,1)
244         # Peturba
245         bias_i[i] = Dual(bias[i], 1.0)
246         # Calcula o objetivo
247         objetivo = Avalia_Treinos(pesos_i,bias_i,LC,dados_treino_i,resultados_treino_i,tipo_objetivo)
248         # Armazena esta derivada
249         DB[i] = objetivo.dual
250         # Desfaz a perturbao
251         bias_i[i] = Dual(bias[i], 0.0)
252     end #i
253     return DW, DB
254 end
255
256 # Converte as entradas para que elas tenham uma mdia nula (por treino)
257 function Media_Nula(entrada)
258     # Entradas com mdia nula
259     const saida = zeros(entrada)
260     # Dimensoes
261     const nl = size(entrada,1)
262     const nc = size(entrada,2)
263     # Para cada treino (linha na entrada) calcula a mdia e translada os valores
264     @inbounds for linha=1:nl
265         media = mean(entrada[linha,:])
266         @inbounds for j=1:nc
267             saida[linha,j] = entrada[linha,j] - media

```

```

268         end
269     end #linha
270     return saida
271 end

```

```

1  #rotina_otimizacao
2  # Line–Search por Armijo
3  # D → Derivada
4  # d → direo
5
6  function Line_Search_Armijo(DW,DB,dW,dB,W0,B0,LC,entradas,saidas,tipo_objetivo)
7      # Passo inicial
8      const alfa = 5.0
9      # Relaxao do alfa
10     const tau = 0.1
11     # Relaxao da inclinacao inicial
12     const c = 0.0001
13     # Define um valor minimo de passo
14     const minimo = 1E–12
15     # Calcula o valor do custo no ponto atual
16     const f0 = Avalia_Treinos(W0,B0,LC,entradas,saidas,tipo_objetivo)
17     # Monta dois vetores globais de derivada e de direo de busca
18     const D = vcat(DW,DB)
19     const d = vcat(dW,dB)
20     d = d/norm(d)
21     # Fator de comparao do mtodo
22     const direita = –c*dot(D,d)
23     # Loop do Mtodo
24     for i=1:1000
25         W = W0 + alfa*dW
26         B = B0 + alfa*dB
27         fu = Avalia_Treinos(W,B,LC,entradas,saidas,tipo_objetivo)
28         if f0–fu < alfa*direita
29             alfa = alfa * tau
30             if alfa<minimo
31                 break
32             end
33         else
34             break
35         end
36     end #i
37     return alfa
38 end # Armijo

```

Appendix C

Ignition Loss Test in Glass-Fiber/Epoxy Composites

To determine the fiber volumetric fraction of the composite material the ignition loss is performed, according to ASTM D2584 (ASTM, 2019). In total three specimens are used to performed the test. In conforming to the standard the specimens are heating in a muffle at 565 °C for one hour.

Table C.1 – Information of the specimens before and after the ignition loss test.

Information	Sample 1	Sample 2	Sample 3
Initial Mass (g)	3.718	3.758	3.596
Volume (m^3)	$2.8859 \cdot 10^{-6}$	$2.8598 \cdot 10^{-6}$	$2.8886 \cdot 10^{-6}$
Final Mass (g)	2.921	2.933	2.760
Mass Loss (%)	21.436	21.953	23.248
Density (kg/m^3)	1301.9	1314.1	1244.9
Resin Mass (g)	0.797	0.825	0.836

Source: Author's production.

Table C.2 – Mechanical properties.

Properties	Epoxy	Glass Fiber
Youngs modulus - E (GPa)	4.3	72
Shear modulus - G (GPa)	1.6	30
Density (kg/m^3)	1090	2550
Poisson (-)	0.35	0.23

Source: Author's production.

According to the Mix Rule (MENDONÇA, 2005), the density of composite is calculated using the total mass of the composite (M_c) and the total volume of the composite

(V_c), defined as

$$\rho_c = \frac{M_c}{V_c}. \quad (C.1)$$

The fiber volumetric fraction V_f is defined as

$$V_f = \frac{M_{fs}}{\rho_f \times t_c}, \quad (C.2)$$

where M_{fs} is the dry fiber mass per unit area, ρ_f if the fiber density and t_c the layer thickness. According to the manufactured the M_{fs} is 0.2302 kg/m^2 . The matrix volumetric fraction V_m is defined as

$$V_m = \frac{\rho_c - \rho_f \times V_f}{\rho_m}, \quad (C.3)$$

where ρ_m is the matrix density. The longitudinal modulus E_{11} and the transversal modulus E_{22} can be calculated as

$$E_{11} = E_{1m} \times V_m + E_{1f} \times V_f, \quad (C.4)$$

$$\frac{1}{E_{22}} = \frac{V_m}{E_{2m}} + \frac{V_f}{E_{2f}}. \quad (C.5)$$

The shear modulus G_{12} and poisson ν_{12} are defined as

$$\frac{1}{G_{12}} = \frac{V_m}{G_{12m}} + \frac{V_f}{G_{12f}}, \quad (C.6)$$

$$\nu_{12} = \nu_{1m} \times V_m + \nu_{1f} \times V_f. \quad (C.7)$$

After some calculations the mechanics properties of the composite are summarized in Tab. C.3.

Table C.3 – Mechanical properties of composite.

Properties	Value
V_f (%)	39.88
V_m (%)	25.27
E_{11} (GPa)	29.80
E_{22} (GPa)	15.55
G_{12} (GPa)	5.84
ν_{12} (-)	0.18
ρ_c (kg/m^3)	1292.30

Source: Author's production.

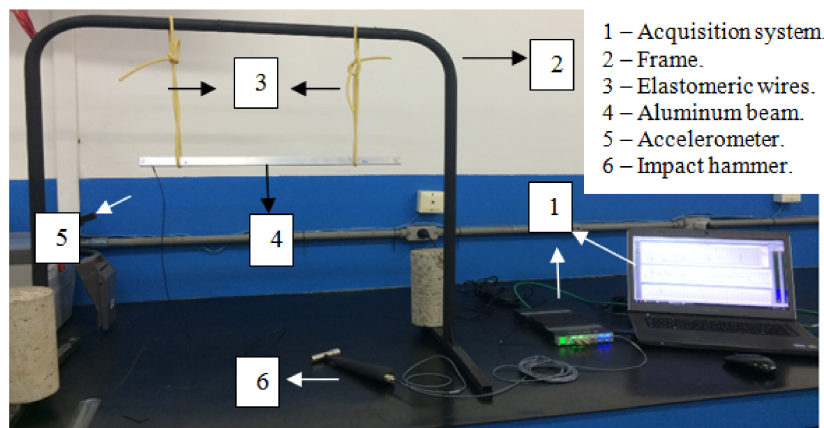
Appendix D

Damage Detection in Metallic Beams

Aluminum beams with a rectangular cross-section of 19.7 mm vs. 4.9 mm and a length of 496 mm, are studied with the damaged patterns: 2 mm crack size, 4 mm crack size, and 8 mm crack size.

Vibration-based analyses are performed using Pulse LabShop software by Brüel & Kjaer, in free-free condition with an impact hammer model 8203-006 (sensitivity 1.12 mV/N) and uniaxial accelerometer model 4397 (sensitivity 10 mV/g), as can be seen in Fig. D.1. The excitation is applied in two different points, such that for each specimen two FRFs are measured, H_{11} for position in 0.08 m and H_{21} for position in 0.42 m. The modes shapes and the positions of the accelerometer, the excitation and the crack are shown in Fig. D.2

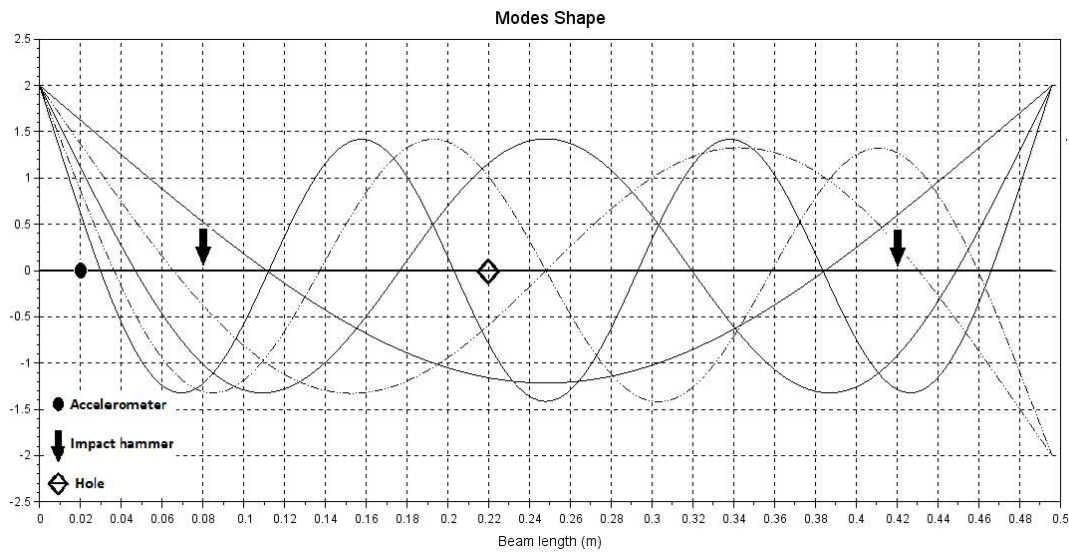
Figure D.1 – Experimental setup for aluminum beams.



Source: Author's production.

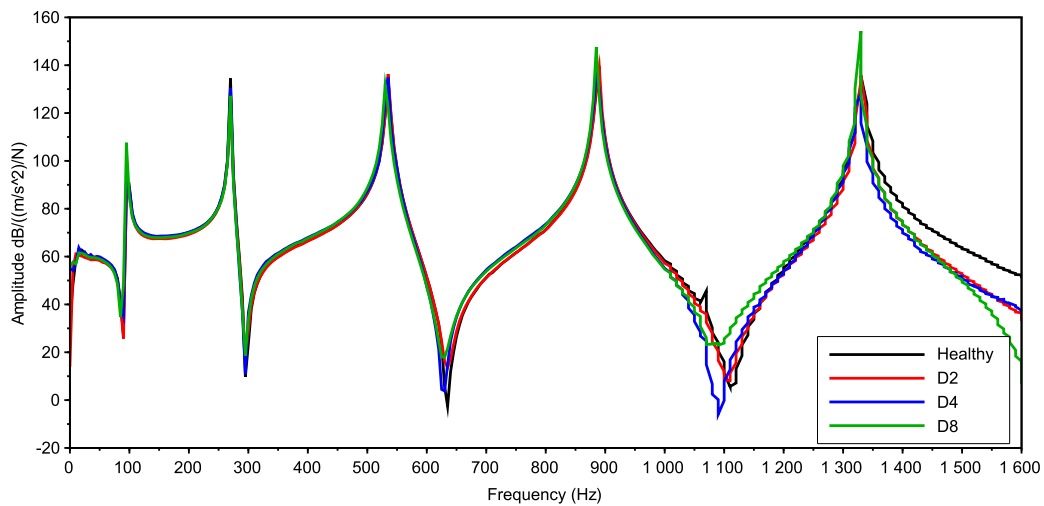
In the total 172 FRFs are taken from healthy (H) and damaged beams with 3201 frequency points from a frequency range of 0-1600 Hz. Figure D.3 shows four FRFs from each case from one specimen. Table D.1 shows the range between the minimum and the maximum natural frequency for each mode and each state condition.

Figure D.2 – Modes shape of the beam.



Source: Author's production.

Figure D.3 – FRFs for healthy, damaged with 2 mm crack size (D_2), damaged with 4 mm crack size (D_4) and damaged with 8 mm crack size (D_8) for the first excitation point in an aluminum beam.



Source: Author's production.

Principal component analysis (PCA) is applied, and the data are reduced to 10 PCs, retained 98.04% of the total variance. The dataset is split into three sets (71% for the training, 10% for the validation and 19% for the testing sets). A fully connected multilayer perceptron neural network using the logistic function as activation function

Table D.1 – Frequency range (Hz) for the first five mode shapes.

Case	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
Healthy	97-98	270-273	534.5-541	888-898	1327.5-1342.5
D_2	96-97.5	269.5-272.5	534.5-540.5	886.5-897	1327-1341
D_4	96.5-97.5	269.5-273	533.5-540	886.5-895.5	1323-1340.5
D_8	95.5-96.5	269.5-272.5	531-538	885-894.5	1324.5-1339

Source: Author's production.

and quadratic function as a cost function is created. The two patterns (healthy and damaged) are presented to the ANN for the learning process. The topology is ten inputs, eight neurons in the first hidden layer, four neurons in the second hidden layer and two neurons in the output layer (healthy or damaged).

The momentum term is 0.8 and a fixed learning rate of 0.3 is applied. Also, 3% of the artificial noise is added to the training set and L_1 regularization technique is applied with a lambda value of 0.001. After 200 times run the summarized results are shown in Tab. D.2. For one simulation, the ANN generalizes 93.55% of the pattern correctly, the results are summarized in Tab. D.3.

Table D.2 – ANNs simulations summary results for aluminum beams (200 times runs).

Data set	Values
Training: mean accuracy (mean deviation)	95.16% (0.014)
Validation: mean accuracy (mean deviation)	82.35% (0.043)
Testing: mean accuracy (mean deviation)	80.65% (0.034)

Source: Author's production.

Table D.3 – ANNs results: aluminum beams.

Training iterations	Training accuracy (%)	Validation accuracy (%)	Testing accuracy (%)
305	95.16	82.35	93.55

Source: Author's production.

The methodology applied in aluminum beams shows good generalization, with no presence of underfitting. The accuracy of 82.35% in the validation dataset can be due to a overfitting problems, or the initialization of the design variables. The fact that the average accuracy, after 200 rounds, is close to 80% (for validation and testing data), shows that the methodology is capable to detect damages.

Appendix E

Scientific Publications

This appendix presented the scientific contributions related to this Master thesis.

VÖLTZ, L. R.; CARDOSO, E.L.; DE MEDEIROS, R. Artificial neural networks applying to detect damage in carbon fiber/epoxy composite. In: 4th Brazilian Conference on Composite Materials (BCCM4), 2018, Rio de Janeiro. Proceedings of the BCCM4, 2018.

VÖLTZ, L. R.; CARDOSO, E.L.; DE MEDEIROS, R. Damage detection on aluminum beams using vibration-based method and artificial neural network. In: 24th ABCM International Congress of Mechanical Engineering (COBEM2017), 2017, Curitiba. Proceedings of the COBEM 2017, 2017.

VÖLTZ, L. R.; CARDOSO, E.L.; DE MEDEIROS, R. Redução de dados das FRFs via análise de componentes principais. In: II Simpósio de Métodos Numéricos em Engenharia, 2017, Curitiba.