Universidade do Minho
Escola de Engenharia

Sinem Tola

Vulnerability Assessment of Existing Bridges to Scour Based on An Indirect Monitoring Approach and Artificial Intelligence Tools

junho de 2026

Vulnerability Assessment of Existing Bridges to Scour Based on An Indirect Monitoring Approach and Artificial Intelligent Tools
Sinem Tola
UMinho | 2026

Universidade do Minho
Escola de Engenharia

Sinem Tola

Vulnerability Assessment of Existing Bridges to Scour Based on An Indirect Monitoring Approach and Artificial Intelligence Tools

Tese de Doutoramento
Programa Doutoral em Engenharia Civil
Trabalho efetuado sob a orientação de

Doutor Joaquim Agostinho Barbosa Tinoco
Professor Assistante José António Campos e Matos
Professor Doutor Eugene O'Brien

junho de 2026

DECLARAÇÃO

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho



Atribuição-NãoComercial-CompartilhaIgual
CC BY-NC-SA

https://creativecommons.org/licenses/by-nc-sa/4.0/

To the people who keep trains moving when the river rises...

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisors, Doctor Joaquim Agostinho Barbosa Tinoco, Assistant Professor with Habilitation José António Campos e Matos, and Professor Eugene O'Brien, for their guidance and support. Special thanks are due to Professor O'Brien and Dr Tinoco for their patience during my transition from design engineering to academic research in the early stages.

I would also like to thank Associate Professor Daniel Cantero (NTNU, Norway) for his generous supervision and for developing and tailoring the TTB-2D version 4.4 simulation tool that was used throughout this thesis.

I would also like to acknowledge Network Rail for supplying structural bridge information and facilitating access to in-service monitoring data collected by Fugro. I am also indebted to the teams at both organisations for their collaboration.

Financial support for this work was provided by the Foundation for Science and Technology (FCT) through national funds under Grant Agreement No. 2021.06162.BD, awarded to Sinem Tola. Additional funding was provided by the European Union's Horizon 2020 research and innovation programme under the Shift2Rail Joint Technology Initiative (contract number 101012456, IN2TRACK3). Support was also received from the Foundation for Science and Technology (FCT)/Ministry of Science, Technology and Higher Education (MCTES) via national funds (PIDDAC) for the Research and Development Unit, the Institute for Sustainability and Innovation in Structural Engineering (ISISE), reference UIDB/04029/2020 (doi.org/10.54499/UIDB/04029/2020), and the Advanced Production and Intelligent Systems Associate Laboratory (ARISE), reference LA/P/0112/2020.

I am grateful to my beloved ones, family, and friends for their unwavering love and support.

STATEMENT OF INTEGRITY

I hereby declare that I have conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

ABSTRACT

Conventional foundation scour assessments are occasional, costly, and challenging to scale. This thesis develops a drive-by monitoring framework for railway bridges that leverages longitudinal track profiles measured by in-service trains to detect and localise scour. The approach integrates physics-informed modelling with machine learning, utilising data collected by *the Rail Infrastructure Alignment Acquisition* (RILA) system, which is mounted on the in-service train.

The proposed scour detection methodology demonstrates the feasibility of drive-by scour detection via a case study built on combined field and simulated track-profile data. The difference in optimised pier stiffness values between the healthy and damaged stages of an existing railway bridge serves as an indicator of scour. To estimate these stiffnesses, a cross-entropy algorithm is utilized. The field measurements provided were from the healthy stage of the case study bridge. Scoured-stage bridge displacement signals are synthetically generated with a vehicle–bridge interaction model and superposed with measured rail irregularities. This superimposition constitutes the theoretical track profile. A finite element bridge model is utilised to compute theoretical static bridge displacements under the moving vehicle. This is achieved by employing moving reference influence lines. The initial pier stiffness in this static bridge model is estimated via the FEMA 356 formula, and scour is represented as a reduction in vertical pier stiffness. The data-driven component of the thesis operationalises the methodology for network-scale screening by building a simulation-supported learning pipeline that detects scour on any railway bridge with RILA track profiles. A physics-based train–track–bridge tool generates displacement signals across representative bridge types, speeds, vehicles, and rail irregularities. Two complementary strategies are introduced: bridge-level models for detection and pier-level models for locating the scour damage. Three different machine learning models are trained, tested, and compared.

The results show that drive-by track-profiles can deliver highly accurate bridge-level scour detection and reliable support-level localisation across sensor orientations and batch sizes. The proposed methodology can support decision-making at the network level under affordable cost conditions.

Keywords: bridge scour; cross-entropy optimisation; drive-by monitoring; machine learning; moving reference influence lines.

RESUMO

As avaliações convencionais de infraescavação das fundações são ocasionais, dispendiosas e difíceis de dimensionar. Esta tese desenvolve uma metodologia de monitorização drive-by para pontes ferroviárias que utiliza perfis longitudinais da via, medidos por comboios em serviço, para detetar e localizar infraescavação. A abordagem integra modelação baseada em dados físicos e aprendizagem automática, utilizando dados recolhidos pelo sistema Rail Infrastructure Alignment Acquisition (RILA), instalado no comboio em serviço.

A metodologia proposta para a deteção de infraescavação demonstra a viabilidade da deteção drive-by através de um caso de estudo baseado em dados de campo e em dados simulados de perfis da via. A diferença nos valores otimizados de rigidez dos pilares entre os estágios saudável e danificado de uma ponte ferroviária existente serve como indicador de infraescavação. Para estimar estas rigidezes, utiliza-se um algoritmo de entropia cruzada. As medições de campo referem-se ao estágio saudável da ponte do caso de estudo. Os sinais de deslocamento da ponte no estágio de infraescavação são gerados sinteticamente com um modelo de interação veículo–ponte e sobrepostos às irregularidades ferroviárias medidas, constituindo o perfil teórico da via. Um modelo de ponte de elementos finitos é utilizado para calcular os deslocamentos estáticos teóricos da ponte sob o veículo em movimento, recorrendo a linhas de influência de referência móveis. A rigidez inicial do pilar neste modelo estático é estimada pela fórmula do FEMA 356, e a infraescavação é representada como uma redução na rigidez vertical do pilar. A componente baseada em dados operacionaliza a metodologia para triagem ao nível da rede, construindo um pipeline de aprendizagem apoiado em simulação que deteta infraescavação em pontes ferroviárias com perfis da via medidos pelo sistema RILA. Uma ferramenta baseada em física de interação comboio–via–ponte gera sinais de deslocamento. São introduzidas duas estratégias complementares de deteção: modelos ao nível da ponte para deteção e modelos ao nível do pilar para localizar os danos por infraescavação. Três modelos de aprendizagem automática diferentes são treinados, testados e comparados.

Os resultados mostram que os perfis da via medidos por drive-by podem fornecer deteção altamente precisa de infraescavação ao nível da ponte e localização fiável ao nível do suporte, em diferentes orientações do sensor e tamanhos de lote. A metodologia proposta pode apoiar a tomada de decisão ao nível da rede sob condições de custos acessíveis.

Palavras-chave: aprendizagem automática; erosão de pontes; linha de influência de referência móvel; monitorização drive-by; otimização de entropia cruzada.

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS iii

STATEMENT OF INTEGRITY iv

TABLE OF CONTENTS vii

LIST OF FIGURES xii

LIST OF TABLES xxi

LIST OF SYMBOLS xxiii

LIST OF ACRONYMS AND ABBREVIATIONS xxxiv

CHAPTER 1: INTRODUCTION 1

1.1 Motivation and Problem Statement 1

1.2 Gaps in the Literature and Research Opportunity 5

1.3 Objectives and Methodology of the Thesis 8

1.3.1 Objectives of Thesis 8

1.3.2 Methodology Overview 8

1.4 Contributions, Publications, and Manuscript-Integration Strategy 11

1.4.1 Original Contributions of the Thesis 11

1.4.2 Publications 12

1.4.3 Manuscript-Integration Policy and Rights 13

1.5 Thesis Structure 14

CHAPTER 2: BACKGROUND AND LITERATURE REVIEW 16

2.1 Scour Mechanisms, Vulnerability, and Identification Approaches 16

2.1.1 Bridge Scour, Consequences and Structural Vulnerability 16

2.1.2 General Overview of Scour Identification Approaches 17

2.1.3 Data Acquisition and Sensing Technologies for Scour Monitoring 20

2.2 Conventional Monitoring-Based and Machine Learning-Based Methods to Identify Scour 22

2.2.1 Methods, Properties, and Main Outcomes of Studies 22

2.2.1.1 Conventional monitoring-based approaches to detect scour 22

2.2.1.2 Machine learning-based research 40

2.2.2 Synthesis of the Results 64

2.2.2.1 Synthesis of conventional monitoring-based studies to detect scour 65

2.2.2.2 Synthesis of machine learning-based studies 66
CHAPTER 3: DRIVE-BY SCOUR DETECTION USING RILA – METHOD AND CASE STUDY 68
3.1 Methodology 68
3.1.1 Moving Reference Influence Lines 70
3.1.1.1 Virtual work theorem and force method 70
3.1.1.2 Betti's theorem and calculation of influence lines 71
3.1.2 Bridge Models for Theoretical Track Profile 72
3.1.2.1 Modelling assumptions and pier stiffness calculation 72
3.1.2.2 Calculation of bridge displacement response 73
3.1.3 Off-Bridge Model 75
3.1.3.1 Model parameters 75
3.1.3.2 Computing displacement response of the track on soil 76
3.1.3.3 Track model results 77
3.2 Data Acquisition System 78
3.3 Cross-Entropy Algorithm to Detect Pier Stiffness 80
3.4 Case Study 81
3.4.1 Data Characteristics and Monitored Bridge 81
3.4.2 Theoretical Bridge Displacement Responses 82
3.4.2.1 Healthy stage simulated displacement response 82
3.4.2.2 Scoured Stage Simulated Displacement Response 83
3.4.3 Track Profile Measurements 84
3.4.4 Obtaining Scoured-Stage Track Profiles 92
3.4.5 Optimisation of Pier Stiffnesses with CE Algorithm 98
3.4.6 Sensitivity Analysis 101
3.4.7 Assessment of Results and Discussion 103
3.5 Conclusion 104
CHAPTER 4: DATA GENERATION FOR ML FRAMEWORK 105
4.1 The Effect of Sensor Orientation on Modelling 105
4.2 Train–Track–Bridge Interaction Model Properties 106
4.2.1 Rail Irregularities 106
4.2.2 Vehicle and Track Properties 107
4.2.3 Properties of Bridge Models 108

4.2.4 Noise Injection 110
4.3 Scour Scenario Design and Simulation Calendars 110
4.4 Final Remarks 115
CHAPTER 5: MACHINE-LEARNING FRAMEWORK 116
5.1 Novelty and Methodology 116
5.2 Task Definitions and ML Implementations 118
5.2.1 Detecting Presence of Scour 118
5.2.1.1 Classification labels, cross-validation protocol and data partitioning 118
5.2.1.2 Input features and batch-based scour detection 121
5.2.1.3 Hyperparameter tuning and evaluation metrics 121
5.2.1.4 ML algorithms and feature scaling 122
5.2.2 Support-level Localisation of Scour 122
5.2.2.1 Classification labels and data partition 123
5.2.2.2 Feature construction, batch-based detection, localisation readout 123
5.2.2.3 Hyperparameter tuning and localisation readout 124
5.2.2.4 ML models, feature scaling, evaluation metrics 124
5.3 Feature Engineering 125
5.3.1 Deriving Input Features 125
5.3.1.1 Application of DWT to track profiles 126
5.3.1.2 DWTs for scour detection ML models 127
5.3.1.3 DWTs for scour locating ML models 136
5.3.1.4 Type A (DWT-based) features 145
5.3.1.5 Type B (ARX-based) features 147
5.3.2 Feature Selection Methods 157
5.3.2.1 Relative-importance–based selection with grid search 158
5.3.2.2 Multi-objective feature selection using NSGA-II 165
5.3.2.3 Assessment and decision for input features 169
5.4 Parameter Normalization 170
5.5 Sliding Windows Method 171
5.5.1 Scour Detection Models: Bridge-Level Sliding Windows 171
5.5.2 Scour Localisation Models: Support-Level Sliding Windows 173
5.6 Handling Class Imbalance 176

5.6.1 The Synthetic Minority Oversampling Technique (SMOTE) 176
5.6.1.1 Implementation for scour detection models 177
5.6.1.2 Implementation for scour localisation models 178
5.6.2 Random Undersampling and Chosen Method 178
5.7 Distributions of Input Features 179
5.7.1 Histograms for Scour Detection Models 179
5.7.1 Histograms for Scour Localisation Models 183
5.8 Machine Learning Models and Preprocessing 187
5.8.1 Hyperparameter Tuning Principles 187
5.8.2 Artificial Neural Network (ANN) Classifier 188
5.8.2.1 Model structure and decision rule 188
5.8.2.2 Network architecture 189
5.8.2.3 Loss function, regularisation and training setup 191
5.8.2.4 Hyperparameters and tuning — scour detection models 192
5.8.2.5 Hyperparameters and tuning — scour localisation models 197
5.8.3 Random Forest (RF) Classifier 202
5.8.3.1 Random Forest formulation and generalisation properties 202
5.8.3.2 Model structure and majority-vote decision rule 204
5.8.3.3 Tree structure and training objective 204
5.8.3.1 Hyperparameters and tuning — scour detection models 206
5.8.3.2 Hyperparameters and tuning - scour localisation models 209
5.8.4 Support Vector Machine classifier 212
5.8.4.1 Decision function and geometric margin 212
5.8.4.2 Hyperparameter tuning for scour detection SVM models 214
5.8.4.3 Hyperparameter tuning for scour localisation SVM models 219
5.9 Concluding remarks 221
CHAPTER 6: MACHINE LEARNING MODEL RESULTS 223
6.1 Scour Detection Model Results 223
6.1.1 Model Metrics 223
6.1.2 Performance Based on Scour Severity 226
6.2 Scour Localisation Model Results 228
6.2.1 Model Metrics 228

6.2.2 Performance Based on Scour Severity 231
6.2.3 True predicted scoured supports 233
6.3 Sensitivity and Feature Importance Analysis 237
6.3.1 Scour Detection Models 237
6.3.1.1 Feature importance analysis 237
6.3.1.2 Markov-order sensitivity analysis 239
6.3.2 Scour Localisation Models 240
6.3.2.1 Feature importance analysis 240
6.3.2.2 Markov-order sensitivity analysis 244
6.4 Discussions and Final Remarks 245
6.4.1 Effect of Batch Size on Performance 245
6.4.2 Differences in Performance Across ML Algorithms 246
6.4.3 Model Interpretability 247
CHAPTER 7: CONCLUSIONS, VALIDITY AND FUTURE WORK 249
7.1 Summary of Findings and Contributions 249
7.2 Positioning Relative to Prior Indirect Approaches 250
7.2.1 Scour Detection: Recent Scour Detection Studies and this Thesis 251
7.2.2 Scour Localisation: Recent Work and This Thesis 252
7.3 Data Source and External Validity 252
7.4 Uncertainty Quantification and Robustness 254
7.5 Limitations and Future Work 255
REFERENCES 260
ANNEXES 289

LIST OF FIGURES

Page

Figure 1.1. Railway bridge collapse due to scour (copied from Rail Magazine, 2023). 2

Figure 1.2. Flowchart of the methodology. 11

Figure 2.1. Scour monitoring devices. Reproduced from Maroni, A et al. (2020). 22

Figure 3.1. Flowchart of scour detection methodology (reprinted from Tola et al., 2025). 69

Figure 3.2. Bridge models: (a) RILA at the front; and (b) RILA at the back (reprinted from Tola et al., 2025). 73

Figure 3.3. Repaired foundation: (a) Plan (Network Rail, 2015c); (b) Elevation (Network Rail, 2015d). 74

Figure 3.4. (a) Real system; (b) Virtual system. 75

Figure 3.5. Support Deflection component, $Z_{x2.}$ 75

Figure 3.6. Track model (reprinted from Tola et al., 2025). 76

Figure 3.7. Displacement response of the track on the soil. 78

Figure 3.8. (a) Displacement response of the track with poor soil; (b) Reduced stiffness zone in the soil. 78

Figure 3.9. RILA unit mounting on a passenger train (reprinted from Tola et al., 2025). 79

Figure 3.10. RILA unit sensors (reprinted from Tola et al., 2025). 79

Figure 3.11. Z_{RILA} curve and Δz distance (reprinted from Tola et al., 2025). 80

Figure 3.12. Rail profiles (reproduced with permission from Network Rail, 2022). 80

Figure 3.13. Absolute track parameters (reproduced with permission from Network Rail, 2022). 80

Figure 3.14. Case study bridge elevation view (reprinted from Tola et al., 2025). 81

Figure 3.15. Google Earth view of Eden viaduct main line railway bridges (Network Rail, 2022). 82

Figure 3.16. Case study bridge foundation repair works (reprinted from Tola et al., 2025). 82

Figure 3.17. Bridge displacements: RILA in front of the train (dashed) and RILA at the back of the train (solid). (reprinted from Tola et al., 2025). 83

Figure 3.18. Bridge scour simulations using the static model (reprinted from Tola et al., 2025). 84

Figure 3.19. Eight RILA measurements: Track 4 (reprinted from Tola et al., 2025). 87

Figure 3.20. Eight RILA measurements: Track 3 (reprinted from Tola et al., 2025). 87

Figure 3.21. Spatial FFT (single-sided amplitude): (a) Track 1 Run 1; (b) Track 4 Run 8. 88

Figure 3.22. Track 3 runs (RILA measurements) Cluster 3a (single run) (reprinted from Tola et al., 2025). 88

Figure 3.23. Track 3 runs (RILA measurements) Cluster 3b (seven runs) (reprinted from Tola et al., 2025). 88
Figure 3.24. Track 4 runs (RILA measurements) Cluster 4a (five runs) (reprinted from Tola et al., 2025). 89
Figure 3.25. Track 4 runs (RILA measurements) Cluster 4b (three runs) (reprinted from Tola et al., 2025). 89
Figure 3.26. RILA rear-mounted runs (measurements) with corrected pier locations: Cluster 3b (reprinted from Tola et al., 2025). 91
Figure 3.27. RILA rear-mounted runs (measurements) with corrected pier locations: Cluster 4b (reprinted from Tola et al., 2025). 91
Figure 3.28. TTB-2D Model Overview (Cantero, 2022). 93
Figure 3.29. TTB-2D Vehicle model (Cantero, 2022). 93
Figure 3.30. Two-level support scheme in TTB-2D v4.4 (Cantero, 2022). 93
Figure 3.31. Steps to obtain scoured stage track profile (reprinted from Tola et al., 2025). 94
Figure 3.32. Rail irregularities of (a) Cluster 3a; (b) Cluster 4a; (c) Cluster 3b; and (d) Cluster 4b (reprinted from Tola et al., 2025). 96
Figure 3.33. Bridge displacements with 20% scour at Pier 6: RILA front-mounted (reprinted from Tola et al., 2025). 96
Figure 3.34. Bridge displacements with 20% scour at Pier 6: RILA rear-mounted (reprinted from Tola et al., 2025). 97
Figure 3.35. Track profiles obtained with Track 3 rail irregularities: RILA front-mounted (reprinted from Tola et al., 2025). 97
Figure 3.36. Track profiles obtained with Track 3 rail irregularities: RILA rear-mounted (reprinted from Tola et al., 2025). 97
Figure 3.37. Track profiles obtained with Track 4 rail irregularities: RILA front-mounted (reprinted from Tola et al., 2025). 98
Figure 3.38. Track profiles obtained with Track 4 rail irregularities: RILA rear-mounted (reprinted from Tola et al., 2025). 98
Figure 3.39. Envelopes of healthy and scoured stage track profiles from Track 3 rail irregularities: RILA front-mounted (reprinted from Tola et al., 2025). 100
Figure 3.40. Envelopes of healthy and scoured stage track profiles from Track 3 rail irregularities: RILA rear-mounted (reprinted from Tola et al., 2025). 100

Figure 3.41. Envelopes of healthy and scoured stage track profiles from Track 4 rail irregularities: RILA front-mounted. 100
Figure 3.42. Envelopes of healthy and scoured stage track profiles from Track 4 rail irregularities: RILA rear-mounted. 101
Figure 3.43. TPDI values from Track 3 rail irregularities: RILA front-mounted (reprinted from Tola et al., 2025). 102
Figure 3.44. TPDI values from Track 3 rail irregularities: RILA rear-mounted (reprinted from Tola et al., 2025). 102
Figure 3.45. TPDI values from Track 4 rail irregularities: RILA front-mounted. 102
Figure 3.46. TPDI values from Track 4 rail irregularities: RILA rear-mounted. 103
Figure 4.1. Rail irregularities of (a) WCM15 bridge; (b) WCM15A bridge. 108
Figure 4.2. Rail irregularities of (a) River Teign bridge; (b) Colonnade bridge. 108
Figure 4.3. Track profiles of Colonnade bridge: RILA front-mounted. 110
Figure 4.4. Simulated scour calendar 2015-2016, south England. 111
Figure 4.5. Simulated scour calendar 2015-2016, north England. 111
Figure 4.6. Teignmouth, UK: annual rainfall averages between 1991 and 2020 (Met Office, 2024). 112
Figure 4.7. Carlisle, UK: annual rainfall averages between 1991 and 2020 (Met Office, 2024). 112
Figure 5.1. Flowchart of the methodology for scour detection, extended from Correia et al. (2025). 118
Figure 5.2. DWT of front-mounted track profiles for the Colonnade bridge under healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4). 129
Figure 5.3. DWT of front-mounted track profiles for the Colonnade bridge under healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2). 129
Figure 5.4. DWT of front-mounted track profiles for the Colonnade bridge under healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1). 129
Figure 5.5. DWT of front-mounted track profiles for the River Teign bridge under healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4). 130
Figure 5.6. DWT of front-mounted track profiles for the River Teign bridge under healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2). 130
Figure 5.7. DWT of front-mounted track profiles for the River Teign bridge under healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1). 130
Figure 5.8. DWT of front-mounted track profiles for the WCM15A bridge under healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4). 131

Figure 5.9. DWT of front-mounted track profiles for the WCM15A bridge under healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2). 131
Figure 5.10. DWT of front-mounted track profiles for the WCM15A bridge under healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1). 131
Figure 5.11. DWT of front-mounted track profiles for the WCM15 bridge under healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4). 131
Figure 5.12. DWT of front-mounted track profiles for the WCM15 bridge under healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2). 132
Figure 5.13. DWT of front-mounted track profiles for the WCM15 bridge under healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1). 132
Figure 5.14. DWT of rear-mounted track profiles for the Colonnade bridge under healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4). 133
Figure 5.15. DWT of rear-mounted track profiles for the Colonnade bridge under healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2). 133
Figure 5.16. DWT of rear-mounted track profiles for the Colonnade bridge under healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1). 133
Figure 5.17. DWT of rear-mounted track profiles for the River Teign bridge under healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4). 134
Figure 5.18. DWT of rear-mounted track profiles for the River Teign bridge under healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2). 134
Figure 5.19. DWT of rear-mounted track profiles for the River Teign bridge under healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1). 134
Figure 5.20. DWT of rear-mounted track profiles for the WCM15A bridge under healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4). 135
Figure 5.21. DWT of rear-mounted track profiles for the WCM15A bridge under healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2). 135
Figure 5.22. DWT of rear-mounted track profiles for the WCM15A bridge under healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1). 135
Figure 5.23. DWT of rear-mounted track profiles for the WCM15 bridge under healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4). 136
Figure 5.24. DWT of rear-mounted track profiles for the WCM15 bridge under healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2). 136

Figure 5.25. DWT of rear-mounted track profiles for the WCM15 bridge under healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1). 136
Figure 5.26. DWT of scour affected zones of RILA front-mounted track profiles, Colonnade bridge, healthy and ≈20% scoured conditions: (a) A_4; (b) D_4. 137
Figure 5.27. DWT of scour affected zones of RILA front-mounted track profiles, Colonnade bridge, healthy and ≈20% scoured conditions: (a) D_3; (b) D_2. 138
Figure 5.28. DWT of scour affected zones of RILA front-mounted track profiles, Colonnade bridge, healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1). 138
Figure 5.29. DWT of scour affected zones of RILA front-mounted track profiles, River Teign bridge, healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4). 138
Figure 5.30. DWT of scour affected zones of RILA front-mounted track profiles, River Teign bridge, healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2). 138
Figure 5.31. DWT of scour affected zones of RILA front-mounted track profiles, River Teign bridge, healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1). 139
Figure 5.32. DWT of scour affected zones of RILA front-mounted track profiles, WCM15A bridge, healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4). 139
Figure 5.33. DWT of scour affected zones of RILA front-mounted track profiles, WCM15A bridge, healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2). 139
Figure 5.34. DWT of scour affected zones of RILA front-mounted track profiles, WCM15A bridge, healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1). 139
Figure 5.35. DWT of scour affected zones of RILA front-mounted track profiles, WCM15 bridge, healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4). 140
Figure 5.36. DWT of scour affected zones of RILA front-mounted track profiles, River Teign bridge, healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2). 140
Figure 5.37. DWT of scour affected zones of RILA front-mounted track profiles, WCM15 bridge, healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1). 140

Figure 5.38. DWT of scour affected zones of RILA rear-mounted track profiles, Colonnade bridge, healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4). 141
Figure 5.39. DWT of scour affected zones of RILA rear-mounted track profiles, Colonnade bridge, healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2). 142
Figure 5.40. DWT of scour affected zones of RILA front-mounted track profiles, Colonnade bridge, healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1). 142
Figure 5.41. DWT of scour affected zones of RILA rear-mounted track profiles, River Teign bridge, healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4). 142
Figure 5.42. DWT of scour affected zones of RILA rear-mounted track profiles, River Teign bridge, healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2). 143
Figure 5.43. DWT of scour affected zones of RILA front-mounted track profiles, River Teign bridge, healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1). 143
Figure 5.44. DWT of scour affected zones of RILA rear-mounted track profiles, WCM15A bridge, healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4). 143
Figure 5.45. DWT of scour affected zones of RILA rear-mounted track profiles, WCM15A bridge, healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2). 143
Figure 5.46. DWT of scour affected zones of RILA front-mounted track profiles, WCM15A bridge, healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1). 144
Figure 5.47. DWT of scour affected zones of RILA rear-mounted track profiles, WCM15 bridge, healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4). 144
Figure 5.48. DWT of scour affected zones of RILA rear-mounted track profiles, WCM15 bridge, healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2). 144
Figure 5.49. DWT of scour affected zones of RILA front-mounted track profiles, WCM15 bridge, healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1). 144

Figure 5.50. Pareto front of NSGA-II feature selection for the RILA front-mounted scour-detection model. 168
Figure 5.51. Pareto front of NSGA-II feature selection for the RILA rear-mounted scour-detection model. 168
Figure 5.52. Pareto front of NSGA-II feature selection for the RILA front-mounted scour-location model. 168
Figure 5.53. Pareto front of NSGA-II feature selection for the RILA rear-mounted scour-location model. 169
Figure 5.54. Histograms of $\log_{10}$-transformed Type-A features from the first crossing in each 7-day window of the scour detection dataset (RILA front-mounted case). 179
Figure 5.55. Histograms of $\log_{10}$-transformed Type-B features from the first crossing in each 7-day window of the scour detection dataset part 1 (RILA front-mounted case). 180
Figure 5.56. Histograms of $\log_{10}$-transformed Type-B features from the first crossing in each 7-day window of the scour detection dataset part 2 (RILA front-mounted case). 180
Figure 5.57. Histograms of $\log_{10}$-transformed aggregated Type-A features for batch size 7 of the scour detection dataset, part 1 (RILA front-mounted case). 181
Figure 5.58. Histograms of $\log_{10}$-transformed aggregated Type-A features for batch size 7 of the scour detection dataset, part 2 (RILA front-mounted case). 181
Figure 5.59. Histograms of $\log_{10}$-transformed aggregated Type-B features for batch size 7 of the scour detection dataset, part 1 (RILA front-mounted case). 181
Figure 5.60. Histograms of $\log_{10}$-transformed aggregated Type-B features, batch size 7 of the scour detection dataset, part 2 (RILA front-mounted case). 182
Figure 5.61. Histograms of $\log_{10}$-transformed aggregated Type-B features, batch size 7 of the scour detection dataset, part 3 (RILA front-mounted case). 182
Figure 5.62. Histograms of $\log_{10}$-transformed aggregated Type-B features, batch size 7 of the scour detection dataset, part 4 (RILA front-mounted case). 182
Figure 5.63. Histograms of $\log_{10}$-transformed aggregated Type-B features, batch size 7 of the scour detection dataset, part 5 (RILA front-mounted case). 183
Figure 5.64. Histograms of $\log_{10}$-transformed Type-A features from the first crossing in each 7-day window of the scour localisation dataset (RILA front-mounted case). 184
Figure 5.65. Histograms of $\log_{10}$-transformed Type-B features from the first crossing in each 7-day window of the scour localisation dataset (RILA front-mounted case). 184

Figure 5.66. Histograms of $\log_{10}$-transformed aggregated Type-A features for batch size 7, part 1 of the scour localisation dataset (RILA front-mounted case). 185
Figure 5.67. Histograms of $\log_{10}$-transformed aggregated Type-A features for batch size 7, part 2 of the scour localisation dataset (RILA front-mounted case). 185
Figure 5.68. Histograms of $\log_{10}$-transformed aggregated Type-B features for batch size 7, part 1 of the scour localisation dataset (RILA front-mounted case). 186
Figure 5.69. Histograms of $\log_{10}$-transformed aggregated Type-B features for batch size 7, part 2 of the scour localisation dataset (RILA front-mounted case). 186
Figure 5.70. Histograms of $\log_{10}$-transformed aggregated Type-B features for batch size 7, part 3 of the scour localisation dataset (RILA front-mounted case). 187
Figure 5.71. Feedforward ANN architecture. 189
Figure 5.72. Random Forest model structure. 206
Figure 5.73. Geometric interpretation of a linear SVM classifier. 213
Figure 6.1. RILA front-mounted: (a) F1 score for 5–20%; (b) F1 score for 0-5%; (c) recall for 5–20%. 224
Figure 6.2. RILA front-mounted: (a) recall for 0-5; (b) precision for 5-20%; (c) precision for 0-5%. 224
Figure 6.3. RILA front-mounted: (a) accuracy; (b) generalisation gap. 225
Figure 6.4. RILA rear-mounted: (a) F1 score for 5–20%; (b) F1 score for 0-5%; (c) recall for 5–20%. 225
Figure 6.5. RILA rear-mounted: (a) recall for 0-5%; (b) precision for 5-20%; (c) precision for 0-5%. 226
Figure 6.6. RILA rear-mounted: (a) accuracy; (b) generalisation gap. 226
Figure 6.7. Scour severity performance of the RF model for the RILA front-mounted configuration across batch sizes: (a) precision; (b) recall; and (c) *F1* score. 227
Figure 6.8. Scour severity performance of the scour detection RF model for the RILA rear-mounted configuration across batch sizes: (a) precision; (b) recall; and (c) *F1* score. 227
Figure 6.9. RILA front-mounted: (a) F1 score for 5–20%; (b) F1 score for 0-5%; (c) recall for 5–20%. 229
Figure 6.10. RILA front-mounted: (a) recall for 0-5%; (b) precision for 5-20%; (c) precision for 0-5%. 229
Figure 6.11. RILA front-mounted: (a) accuracy; (b) generalisation gap. 230
Figure 6.12. RILA rear-mounted: (a) F1 score for 5–20%; (b) F1 score for 0-5%; (c) recall for 5–20%. 230
Figure 6.13. RILA rear-mounted: (a) recall for 0-5%; (b) precision for 5-20%; (c) precision for 0-5%. 231
Figure 6.14. RILA rear-mounted: (a) accuracy; (b) generalisation gap. 231
Figure 6.15. Scour severity performance of the scour localisation RF model for the RILA front-mounted configuration across batch sizes: (a) precision; (b) recall; and (c) *F1* score. 232

Figure 6.16. Scour severity performance of the scour localisation RF model for the RILA rear-mounted configuration across batch sizes: (a) precision; (b) recall; and (c) *F1* score. 232
Figure 6.17. True-positive localisation fractions for the RILA front-mounted configuration, w_s = 7: (a) WCM15 bridge; (b) WCM15A bridge. 234
Figure 6.18. True-positive localisation fractions for the RILA front-mounted configuration, w_s = 7: (a) River Teign bridge; (b) Colonnade bridge. 234
Figure 6.19. True-positive localisation fractions for the RILA rear-mounted configuration, w_s = 7: (a) WCM15 bridge; (b) WCM15A bridge. 234
Figure 6.20. True-positive localisation fractions for the RILA rear-mounted configuration, w_s = 7: (a) River Teign bridge; (b) Colonnade bridge. 235

LIST OF TABLES

Page

Table 2.1. Properties of devices used in direct monitoring studies (adapted from Tola et al., 2023). 24

Table 2.2. Properties of devices used in direct monitoring studies (adapted from Tola et al., 2023). 31

Table 2.3. Properties of recent ML studies for scour prediction (adapted from Tola et al., 2023). 53

Table 2.4. Recent ML research for scour detection and localisation from structural vibrations. 66

Table 3.1. Properties of the bridge models (reprinted from Tola et al., 2025). 73

Table 3.2. Track parameters, adapted from Quirke (2017). 76

Table 3.3. FFT peaks of runs with speeds. 89

Table 3.4. Corrections of each support for cluster 3b in meters (reprinted from Tola et al., 2025). 91

Table 3.5. Corrections of each support for cluster 4b in meters (reprinted from Tola et al., 2025). 92

Table 3.6. TTB-2D (version 4.4) Model properties (reprinted from Tola et al., 2025). 95

Table 3.7. Pier stiffness optimisation results (reprinted from Tola et al., 2025). 99

Table 4.1. Vehicle Properties of Simulations. 109

Table 4.2. Model properties of the bridges. 109

Table 5.1. Global DWT: Energy ratios and MAD values for A_4 and D_4 – RILA front-mounted case. 129

Table 5.2. Global DWT: Energy ratios and MAD values for D_3, D_2 and D_1 – RILA front-mounted case. 130

Table 5.3. Global DWT: Energy ratios and MAD values for A_4 and D_4 – RILA rear-mounted case. 134

Table 5.4. Global DWT: Energy ratios and MAD values for D_3, D_2 and D_1 – RILA rear-mounted case. 135

Table 5.5. Local DWT: Energy ratios and MAD values for A_4 and D_4 – RILA front-mounted case. 140

Table 5.6. Local DWT: Energy ratios and MAD values for D_3, D_2 and D_1 – RILA front-mounted case. 141

Table 5.7. Local DWT: Energy ratios and MAD values for A4 and D4 – RILA rear-mounted case. 145

Table 5.8. Local DWT: Energy ratios and MAD values for D_3, D_2 and D_1 – RILA rear-mounted case. 145

Table 5.9. Impulse-response identification outcomes for scour-localisation ML models. 157

Table 5.10. Feature subset search for RILA front-mounted case-scour detection models. 161

Table 5.11. Feature subset search for RILA rear-mounted case-scour detection models. 162

Table 5.12. Feature subset search for RILA front-mounted case-scour locating models. 164

Table 5.13. Feature subset search for RILA front-mounted case-scour locating models. 164

Table 5.14. Feature Selection Method Results for scour detection models. 169

Table 5.15. Feature Selection Method Results for scour localisation models. 170

Table 5.16. Hyperparameter tuning results of scour detection ANN models. 197

Table 5.17. Hyperparameter tuning results of scour localisation ANN models. 202
Table 5.18. Hyperparameter tuning results of scour detection RF models. 209
Table 5.19. Hyperparameter tuning results of scour localisation RF models. 212
Table 5.20. Hyperparameter tuning results of scour detection SVM models. 219
Table 5.21. Hyperparameter tuning results of scour localisation SVM models. 221
Table 6.1. Scour detection ML model metrics- RILA front-mounted. 224
Table 6.2. Scour detection ML model metrics- RILA rear-mounted. 225
Table 6.3. Scour localisation ML model metrics – RILA front-mounted. 229
Table 6.4. Scour localisation ML model metrics – RILA rear-mounted. 230
Table 6.5. Feature importance analysis results for scour detection models. 238
Table 6.6. Feature importance analysis results for scour localisation models. 244

LIST OF SYMBOLS

$\mathbf{A_{AB}}$, $\mathbf{A_{BC,}}$ $\mathbf{A_{CD}}$	Areas of the triangles of moment diagrams
$\mathbf{A_{base}}$	Baseline test accuracy
$\mathbf{A_j(i)}$	i^{th} approximation coefficient at DWT level j
$\mathbf{A_{j,l}}$	Test accuracy after randomly shuffling predictor j at repeat ℓ
$\mathbf{A_k}$	Spectral magnitude at frequency index k
$\mathbf{A(q^{-1})}$	Autoregressive polynomial of the ARX model
$\mathbf{A_r}$	Rail cross-sectional area
$\mathbf{A_v}$	FRA Class-6 PSD amplitude parameter
$\mathbf{B_F}$	Foundation width
$\mathbf{C}$	Box Constraint hyperparameter in the SVM primal objective
$\mathbf{B(q^{-1})}$	Input polynomial of the ARX model
$\mathbf{D_j(i)}$	i^{th} detail coefficient at DWT level j
$\mathbf{D_{16}}$	16th-percentile grain size
$\mathbf{D_{50}}$	Mean particle diameter
$\mathbf{D_{84}}$	84th-percentile (coarse fraction)
$\mathbf{D_{95}}$	95th-percentile (very coarse tail)
$\mathbf{D_f}$	Embedment depth of the foundation
$\mathbf{D_{ax}}$	Axial/non-bending part of the displacement vector
$\mathbf{D_{ben}}$	Bending part of the displacement vector
$\mathbf{D_{rl}}$	Euclidean distance between Z_r and Z_ℓ
$\mathbf{D_p/d_{50}}$	Relative grain size
$\mathbf{D_F}$	Displacements caused by forces of the system F
$\mathbf{D_Q}$	Displacements caused by forces of the system Q
$\mathbf{D}$	Displacement vector of the released points under the real load
$\mathbf{E}$	Young's Modulus
$\mathbf{E_N}$	Set of directed connections between neurons in the ANN
$\mathbf{E_{ARX}}$	Energy of the ARX-based impulse response
$\mathbf{E_{cap}}$	Energy threshold
$\mathbf{E_{cum,j}(H)}$	Cumulative impulse-response energy up to lag h for simulation j
$\mathbf{E_{Aj}}$	Wavelet energy of approximation coefficients at level j
$\mathbf{E_{Dj}}$	Wavelet energy of detail coefficients at level j
$\mathbf{E_{tot}}$	Total wavelet-domain energy
$\mathbf{E_{tot_imp,j}}$	Total impulse-response energy for simulation j

$\mathbf{EI}$	Flexural rigidity
$\mathbf{E_r}$	Elastic modulus of rail
$\mathbf{E_o(\theta)}$	Unregularised binary cross-entropy loss of the ANN as a function of θ
$\mathbf{E(\theta)}$	Regularised ANN loss as a function of θ
$\mathbf{F}$	Forces of the system F
$\mathbf{F_d}$	Densimetric Froude Number
$\mathbf{Fe}$	Excess Abutment Froude number
$\mathbf{FR_j}$	Fraction of perturbations that flip the predicted label when perturbing predictor j
$\mathbf{FN}$	Count of scoured predicted as healthy (false negatives) on J
$\mathbf{FP}$	Count of healthy predicted as scoured (false positives) on J
$\mathbf{F_r}$	Standard (flow) Froude number
$\mathbf{F_s}$	Spatial sampling rate
$\mathbf{F_w}$	Number of sliding windows
$\mathbf{F_R}$	Redundant forces/moments to be re-introduced to close the releases
$\boldsymbol{G_B}$	Soil shear modulus
$\mathbf{G(q^{-1})}$	Transfer function of the ARX model
$\mathbf{G(l)}$	Gini impurity of node l
$\mathbf{G_{split}}$	Weighted sum of child-node Gini impurities
$\mathbf{H}$	Markov horizon, i.e., number of retained impulse response coefficients
$\mathbf{H_{adapt}}$	Adaptive Markov horizon defined per bridge
$\mathbf{H_j}$	Smallest horizon for simulation j capturing at least 95% of total energy
$\mathbf{H_k}$	L-BFGS approximation of the inverse Hessian
$\mathbf{H_s}$	Hyperparameter space
$\mathbf{H^*}$	median of H_j across simulations
$\mathbf{I}$	Second moment of inertia
$\mathbf{I(\cdot)}$	Indicator function
$\mathbf{I_j}$	Mean accuracy drop under permutation for predictor j
$\mathbf{I_r}$	Rail second moment of inertia
$\mathbf{I_z}$	Index set of crossings in sliding window z
$\mathbf{J}$	Pooled outer-test index set
$\mathbf{J_D}$	maximum decomposition level of DWT
$\mathbf{J_r}$	Test index set of outer fold r
$\mathbf{J^{tr}}$	Pooled outer-training index set
$\mathbf{J^{tr}_r}$	Outer-training index set of fold r
$\mathbf{K}$	Number of nearest neighbours used in the KNN imputation

$\mathbf{K_A}$	Common number of retained autoregressive coefficients across successful ARX fits
$\mathbf{K_B}$	Common number of retained input coefficients across successful ARX fits
$\mathbf{K_1}$	Shape correction coefficient
$\mathbf{K_2}$	Pier skew
Krt(z)	Kurtosis of the coefficient z
KTC	Tuned-correlated kernel matrix
Le	Number of sensitivity evaluations per input feature in the sensitivity analysis
$\mathbf{L_F}$	Foundation length
$\mathbf{L_h}$	Filter length
L(k)	Candidate set of values for the k^{th} hyperparameter in grid search
$\mathbf{L_{min}^{A}}$	Minimum value of the approximation coefficient vector
M	Bending moment
$\mathbf{M_a}$	Sensitivity measure assigned to feature a in the sensitivity analysis
$\mathbf{MAD_{AJD}}$	Mean absolute difference between scoured and healthy approximation coefficients
$\mathbf{MAD_{Dj}}$	Mean absolute difference between scoured and healthy detail coefficients at level j
$\mathbf{MDAD_S}$	Median absolute deviation
$\mathbf{MC_j}$	Mean absolute change in the SVM decision score when perturbing predictor j
$\mathbf{M_n}$	Total number of parameters in the sensitivity analysis
$\mathbf{M_u}$	Bending moment per unit virtual force
N	Total number of pooled test instances in ML model
$\mathbf{N_{ax}}$	Axial force
$\mathbf{N_b}$	Total number of batches across all bridges
$\mathbf{N_{healthy}}$	Number of healthy-class samples in a training set
$\mathbf{N_j}$	Index set of K nearest-neighbour simulations used for KNN imputation of simulation j
$\mathbf{N_L}$, $\mathbf{N_R}$	Numbers of samples in the left and right child nodes after a split
$\mathbf{N_l}$	Number of training samples reaching node l
$\mathbf{N_{sim}}$	Number of simulated train crossings per bridge
$\mathbf{N_r}$	Index set of K nearest neighbours used for imputing row r
$\mathbf{N_s}$	Number of spatial samples on the bridge
$\mathbf{N_{scoured}}$	Number of scoured-class samples in a training set
$\mathbf{N_{syn}}$	Number of synthetic samples generated by SMOTE
$\mathbf{N_{tr}}$	Size of pooled outer-training index set
$\mathbf{N_u}$	Axial force per unit virtual force
$\mathbf{N_w(s)}$	Number of spatial samples in the support-centred window around support s
$\mathbf{P_1(k_t)}$	Single-sided amplitude spectrum

PE	RF generalisation error
Q	Forces of the system Q
Q_{ij}	Entry of the SVM kernel matrix Q
R	Correlation Coefficient
R^2	Coefficient of determination
RI_j	Relative importance of predictor j
R_L, R_R	Left/right support reactions
S	Index set of simulations with successful ARX fits
S_i	Set of tree indices for which observation (Xi,Yi) is out-of-bag
S_N	Number of hyperparameter configurations evaluated
ScrC	Stacked support-level scour vector
Scr_{new}	Scour percentage assigned to a synthetic crossing generated by SMOTE
SCR	Scour-percentage vector
$SCR_{batch}(z)$	Batch scour-percentage vector for window
$S(\Omega)$	One-sided power spectral density
Skw(z)	Skewness of the coefficient vector z
$SNM_{batch}(z,p)$	Support name attached to batch
$S_{tr}(s)$	Collection of training metrics obtained for random-search trial s
$S_{two}(\Omega)$	Two-sided power spectral density
S_v	Variance-based sensitivity measure for feature
$S_{val}(s)$	Collection of validation metrics obtained for random-search trial s
T	Torsion force
T_n	Dimensionless time
T_R	Number of trees in the random forest
TN	Count of healthy predicted as healthy (true negatives) on J
TP	Count of scoured predicted as scoured (true positives) on J
T_s	Spatial sampling step used in the ARX model
T_U	Torsion force per unit virtual force
U_w	Convolution matrix in window w
V	Shear force
V_N	Set of neurons in the ANN graph
V_U	Shear force per unit virtual force
$W^{(IN)}$	Weight matrix of layer I_N in ANN
X	Feature matrix in the ML models
$X_{balanced}$	Balanced feature matrix after appending the synthetic samples

$\mathbf{X}_{\mathbf{batch}}(\mathbf{z})$	Batch feature vector for window z
$\mathbf{X}_{\mathbf{min}}$	Minority-class feature matrix used as input to SMOTE
$\mathbf{X}_{\mathbf{syn}}$	Feature matrix of synthetic minority samples generated by SMOTE
$\mathbf{X}_{\mathbf{test}}$	Feature matrix of the outer-test set in a given outer fold
$\mathbf{X}_{\mathbf{train}}$	Feature matrix of the outer-training set in a given outer fold
$\mathbf{Y}$	Label (output) vector in the ML models
$\mathbf{Y}_{\mathbf{balanced}}$	Balanced feature matrix after appending the synthetic samples
$\mathbf{Y}_{\mathbf{batch}}(\mathbf{z})$	Batch label for window z
$\widehat{Y}_{\mathbf{batch}}(z)$	Predicted batch label
$\widehat{Y}_{i}^{\mathbf{OOB}}$	OOB prediction for observation i
$\mathbf{Y}[\mathbf{k}_t]$	Discrete Fourier transform of u[n] at bin k
$\mathbf{Y}_{\mathbf{min}}$	Minority-class label vector used as input to SMOTE
$\mathbf{Y}_{\mathbf{syn}}$	Label vector of synthetic minority samples generated by SMOTE
$\mathbf{Y}_{\mathbf{test}}$	Label vector of the outer-test set in a given outer fold
$\mathbf{Y}_{\mathbf{train}}$	Label vector of the outer-training set in a given outer fold
$\mathbf{Z}_r$	Standardised descriptor vector for row r
$\mathbf{a}$	Distance from the left support of the span to the axle location
$\mathbf{a}_i$	i^{th} autoregressive coefficient in autoregressive polynomial of the ARX model
$\mathbf{a}_r$	Effective shear area
$\mathbf{a}_s$, $\mathbf{b}_s$	Seasonal logarithmic model coefficients for y(t)
$\mathbf{acc\bar{v}al(\theta)}$	Mean validation accuracy across inner folds
$\widetilde{a}_j$	Imputed autoregressive coefficient vector for simulation j
$\widehat{a}_{j,k}$	Perturbated k^{th} a-coefficient of simulation j
$\mathbf{b}$	Distance from the axle location to the right support of the span
$\mathbf{b}_i$	i^{th} autoregressive coefficient in input polynomial of the ARX model
$\widetilde{b}_j$	Imputed input coefficient vector for simulation j
$\tilde{\mathbf{b}}(\mathbf{k})$	Delay-aligned input coefficient in the Markov recursion
$\widehat{b}_{j,k}$	Perturbated k^{th} b-coefficient of simulation j
$\mathbf{b}^{(IN)}$	Bias vector of layer I_N in the ANN
$\mathbf{c}_{ba}$	Ballast damping
$\mathbf{c}_i$	True class label of instance i
$\hat{\mathbf{c}}_i$	predicted class label of instance i
$\hat{\mathbf{c}}_i^{tr}$	Predicted class label for training instance i
$\mathbf{c}_p$	Rail pad damping
$\mathbf{c}_{sg}$	Subgrade damping

$\mathbf{d}$	Distance between measurement point and nearest axle
$\mathbf{d_{50}}$	Grain Size of The Bed Material
$\mathbf{d_{50}/h}$	Relative Submergence
$\mathbf{d_{50}/l}$	Relative Sediment Size
$\mathbf{d_f}$	The height of effective sidewall contact
$\mathbf{d_H}$	Hamming distance
$\mathbf{d_j}$	Descriptor vector for simulation j
$\mathbf{d_r}$	Descriptor vector for row r
$\mathbf{e = (k,j)}$	Directed connection from neuron k to neuron j
$\mathbf{e(n)}$	Residual (prediction error) of the ARX model at spatial index n
$\mathbf{f}$	Flexibility matrix of the released (determinate) structure
$\mathbf{f_{ANN}(x)}$	Scalar decision function of the ANN classifier
$\mathbf{f_{ax}}$	Axial/non-bending contribution to the flexibility matrix
$\mathbf{f_{ben}}$	Bending contribution to the flexibility matrix
$\mathbf{f_j(h)}$	Fraction of total impulse-response energy captured up to lag h for simulation j
$\mathbf{g(k)}$	Markov parameter at lag k
$\mathbf{g_j}$	Markov vector for simulation j
$\tilde{\mathbf{g}}_{\mathbf{ARX}}\mathbf{(k)}$	k-th impulse-response coefficient obtained from the ARX model in a window
$\mathbf{g_{est}(k)}$	Value at coordinate k after enforcing $[g_{low}(k), g_{high}(k)]$ bounds
$\tilde{\mathbf{g}}_{\mathbf{IR}}\mathbf{(k)}$	k-th impulse-response coefficient obtained from the regularised nonparametric IR
$\tilde{\mathbf{g}}_{\mathbf{FIR}}\mathbf{(k)}$	k-th impulse-response coefficient obtained from the ridge-regularised FIR
$\mathbf{g_{med}}$	Component-wise median impulse-response vector over all successful rows
$\bar{\mathbf{g}}\mathbf{(N_r)}$	Mean impulse-response vector over neighbours in N_r
$\mathbf{g_{low}(k)}$	Lower MAD-based bound at coordinate k
$\mathbf{g_{high}(k)}$	Upper MAD-based bound at coordinate k
$\mathbf{g_k}$	Gradient of the regularised loss at iteration k
$\tilde{\mathbf{g}}_{\mathbf{r}}$	Imputed impulse-response vector for row r
$\mathbf{g_r^*(k)}$	Final perturbed value at coordinate k for imputed row r
$\mathbf{h_j^{(IN)}}$	Post-activation output of neuron j in layer l_N
$\mathbf{h/l}$	Relative Flow Depth
$\mathbf{h(x,\theta_k)}$	Prediction of the k^{th} decision tree in the RF for input feature vector
$\mathbf{j}$	Number of released constraint
$\mathbf{k}$	Frequency index
$\mathbf{k_{ba}}$	Ballast stiffness
$\mathbf{k_{ext}}$	Number of outer folds in the nested cross-validation

$\mathbf{k_{healthy}}$, $\mathbf{k_0}$	Bridge pier stiffness in healthy stage
$\mathbf{k_{int}}$	Number of inner folds in the nested cross-validation
$\mathbf{k_l}$, $\mathbf{k_r}$	Left/right vertical support (spring) stiffness
$\mathbf{k_p}$	Rail pad stiffness
$\mathbf{k_R}$	Equivalent vertical stiffness per sleeper
$\mathbf{k_{sco}}$	Bridge pier stiffness in scoured stage
$\mathbf{k_{sg}}$	Subgrade stiffness
$\mathbf{k_t}$	Spatial frequency
$\mathbf{l_F}$	Output layer in ANN
$\mathbf{l_{hinge}(x_i,y_i;w,b)}$	Hinge loss of training point (x_i, y_i) for a given (w, b)
$\mathbf{l_{low}(\alpha)}$	Index set of dual variables that can move downwards in SMO's KKT-based working-
$\mathbf{l_N}$	Layer index in the ANN
$\mathbf{l_{up}(\alpha)}$	Index set of dual variables that can move upwards in SMO's KKT-based working-set
$\mathbf{m}$	Number of stored (s_i, y_i) pairs used by L-BFGS
$\mathbf{m_{ba}}$	Ballast mass
$\mathbf{min_k(z)}$	Minimum of feature k over window z
$\mathbf{max_k(z)}$	Maximum of feature k over window z
$\mathbf{m(x,y)}$	Limiting (infinite-forest) margin obtained when $T \to \infty$
$\mathbf{m_s}$	Sleeper mass (half)
$\mathbf{m_{try}}$	Number of predictors sampled at each split in a tree
$\mathbf{m_T(x,y)}$	Finite-forest margin for a forest with T_R trees
$\bar{\mathbf{m}}_t$	Mean OOB margin of tree t
$\bar{\mathbf{m}}_{t,j}$	Mean OOB margin of tree t after permuting predictor j
$\mathbf{n}$	Degree of static indeterminacy
$\mathbf{n_a}$	Autoregressive order of $A(q^{-1})$ in the ARX model
$\mathbf{n_b}$	Input order of $B(q^{-1})$ in the ARX model
$\mathbf{n_{feat}}$	Number of input features in ML model
$\mathbf{n_k}$	Input delay (in samples) in the ARX model
$\mathbf{n_s}$	Spatial location of sample on bridge
$\mathbf{n_{supp}}$	Number of supports of a bridge
$\mathbf{n_t}$	Total number of batch samples in the training set
$\mathbf{n_x}$	Number of spatial samples in x
$\mathbf{n_z}$	Number of wavelet coefficients at a given DWT level (length of A_j or D_j)
$\mathbf{p_{scour}(x)}$	Estimated probability that the observation with features x belongs to class scoured
$\mathbf{p_v}$	Number of vanishing moments of the Daubechies wavelet

$\rho_{\delta\xi}$	Correlation coefficient
$\mathbf{p_H}$	Number of components of the feature vectors used in the Hamming distance
$\mathbf{p_k}$	Search direction in parameter space at iteration
$\mathbf{p_{TT}}$	Transfer-tuning sampling distribution over hyperparameters conditioned on θ_{centre}
$\mathbf{p_w}$	Number of weight parameters in the ANN
$\mathbf{p(\theta)}$	Sampling distribution over hyperparameters used in random search at batch size 7
$\rho_{\delta_w \xi_{w(r)}}$	Correlation coefficient between $\delta_w(n)$ and $\xi_{w(}n)$ in row r
$\mathbf{q^{-1}}$	Backward shift operator ($q^{-1}\delta(n) = \delta(n-1)$)
$\mathbf{r_g(j,p)}$	Global row index of support p at simulation j in the stack matrices
$\mathbf{r(j)}$	Outer fold index of CV
$\mathbf{r_\delta(j)}$	Root-mean-square value of the track profile in simulation j
$\mathbf{r_{\delta_{w(r)}}}$	Root-mean-square value of δw(n) in row r
$\mathbf{r(z)}$	Individual feature vector for sliding window
$\mathbf{s}$	Sleeper spacing
$\mathbf{s(j)}$	Scour-percentage vector for crossing j
$\mathbf{s(j,p)}$	Scour percentage at support p for crossing j
$\mathbf{s_k}$	Change in parameter vector between iterations
$\mathbf{snm(p)}$	Name of support p
$\mathbf{s(x)}$	Scalar pre-activation score for the “scoured” class at the ANN output layer
$\mathbf{t}$	Decision threshold
$\mathbf{t_1}$, $\mathbf{t_2}$	Start/end times of a season segment
$\mathbf{w}$	Index of support-centred window
$\mathbf{w_s}$	Sliding window length
$\mathbf{x}$	Distance along the track
$\mathbf{x_a}$	Input feature a used in the sensitivity analysis
$\mathbf{xC}$	Stacked support-level feature matrix
$\mathbf{x_i}$	Minority-class feature vector
$\mathbf{x_j}$	Original value of feature j
$\tilde{\mathbf{x}}_j$	Z-scored value of feature j
$\mathbf{x_k}$	k^{th} input feature
$\mathbf{x_{new}}$	Synthetic minority feature vector generated between x_i and x_{nn}
$\mathbf{x_{nn}}$	One of the k nearest-neighbour minority feature vectors to x_i
$\mathbf{x(j)}$	Feature vector for crossing j at scour detection ML models
$\mathbf{x(j,p)}$	Feature vector for support p at crossing j at scour location ML models
$\mathbf{y}$	Regression output vector used in least squares

$\mathbf{y_a}$	Vector of sensitivity responses for feature a across L_e evaluations
yC	Stacked support-level batch vector
y(j)	Label for for crossing j at scour detection ML models
y(j,p)	Label for support p at crossing j at scour localisation ML models
$\mathbf{y_k}$	Change in gradient between iterations k and k+1
y(t)	Scour percentage at time t
$\mathbf{y/D_p}$	Relative flow depth
$\hat{y}(x)$	Binary class prediction of the ML model for input x
$\mathbf{z_j^{(IN)}}$	Pre-activation at neuron j in layer l_N
$\mathbf{z_{RL}}$, $\mathbf{z_{RR}}$	Left/right support deflections
$\mathbf{z_x}$	Vertical displacement at location x
$\mathbf{z_{x1}}$	Bending component of z_x
$\mathbf{z_{x2}}$	Support deflection component of z_x
z(x)	Bridge displacement profile
$\mathbf{\Delta D_{ben}}$	Bending increment for the displacement vector
$\mathbf{\Delta f_{ben}}$	Bending add-on to the flexibility matrix for each adjacent span pair
Δk	FFT bin width
Δx	Spatial sampling interval
ΔΩ	Frequency resolution
Θ	Generic random tree-building vector
Θ'	Independent copy of Θ
$\mathbf{\Theta_k}$	Random tree-building vector for the k^{th} tree
Φ	Regressor matrix for ARX
$\mathbf{\Phi_0}$, $\mathbf{\Phi_{Ns/2}}$	Phases at the zero-frequency term and the Nyquist-limit term
$\mathbf{\phi_k}$	Random phase at frequency index k
Φ(x)	Scaling function in the DWT
Φ(α)	Dual objective function of the SVM in terms of the dual coefficients α
ψ(x)	Wavelet function in the DWT
Ψ(θ)	Hyperparameter response function
Ω	Spatial frequency
$\mathbf{\Omega_k}$	Discrete angular spatial frequency at frequency index k
$\mathbf{\Omega_r}$, $\mathbf{\Omega_c}$	FRA low-frequency roll-off and corner parameters
α	Blending weight between local KNN mean and global median in the imputation
$\mathbf{\alpha_i}$	Dual coefficient of training sample i in the SVM dual optimisation problem
γ	RBF kernel parameter

$\gamma_{geom}(w,b)$	Geometric margin of the separating hyperplane on the training set
γ_i	Functional margin of training point (x_i, y_i) in the SVM
β_z	Correction factor for embedment
θ	Generic hyperparameter vector
$\hat{\theta}$	Hyperparameter vector selected as best in a given outer fold
θ_c	Candidate hyperparameter vector in the grid search
θ_{centre}	Centre hyperparameter configuration summarising batch-7 results
$\theta_a, \theta_b, \theta_c, \theta_d$	End-rotation angles at the two hinges of the loaded span
θ_k	ANN parameter vector at iteration k of L-BFGS
θ^*	Fold-specific best hyperparameter configuration selected by the random search
κ_A	Integer offset used in the perturbation phase for the a-coefficients
κ_B	Integer offset used in the perturbation phase for the b-coefficients
λ	Wavelength of longitudinal track measurements
$\lambda(ANN)$	L2 regularisation strength in the ANN training objective
λ_{FIR}	Ridge regularisation parameter in the FIR estimator
λ_r	Irregularity wavelength
λ_s	Random number in [0, 1] for interpolation
$\hat{\lambda}$	Anchor L2 regularisation strength taken from the batch-7 tuning
$\hat{\lambda}(w_s)$	Selected L2 regularisation strength for a given batch size w_s
$\delta(x)$	Track longitudinal profile
$\delta_{noisy}(x)$	Noisy track profile
$\delta_w(n)$	Tapered track profile in support centred window
$\varepsilon_{j,k}^{(a)}$	Perturbation added to the k^{th} a–coefficient of simulation j
$\varepsilon_{j,k}^{(b)}$	Perturbation added to the k^{th} b–coefficient of simulation j
η	Signal noise
η_s	Scaling factor for perturbation
μ_d	Mean of the descriptor vectors over simulations with successful ARX fits
μ_j	Sample mean of feature j
$\mu_k(z)$	Aggregated mean of feature k over window z
μ_r	Rail mass per unit length
μ_{tr}	Vector of per-feature means computed from X_{train} in an outer fold
μ_z	Sample mean of the coefficients in vector z
ν	Poisson's ratio of the soil
$\xi(\Omega_k)$	Complex spectrum of the irregularity at frequency index k
$\xi(x)$	Rail irregularity

$\xi_w(n)$	Tapered rail-irregularity sequence in support-centred window w
ρ	Correlation between the correctness indicators of two randomly chosen trees
σ	Noise standard deviation
σ_d	Standard deviation of the descriptor vectors
σ_g	Sediment gradation
σ_j	Sample standard deviation of feature j
$\sigma^{(IN)}(\cdot)$	Activation function used in hidden layer l_N
$\sigma_k(z)$	Standard deviation of feature k over window z
σ_p	Internal stresses due to external forces
$\sigma_s(j)$	Standard deviation of the track profile in simulation j
$\sigma_{sig}(\cdot)$	Logistic sigmoid activation function at the output layer
σ_{tr}	Vector of per-feature standard deviations computed from X_{train} in an outer fold
σ_z	Sample standard deviation of the coefficients in vector z
$\sigma_{\delta w(r)}$	Standard deviation of $\delta w(n)$ in row r
τ	Threshold in a candidate split of the form $x_k \leq \tau$
φ	Conjugate of the golden ratio

LIST OF ACRONYMS AND ABBREVIATIONS

ABR	AdaBoost Regressor
AdaBoost	Adaptive Boosting
AIC	Akaike Information Criterion
ANN	Artificial Neural Networks
AR	Autoregressive model
ARX	Autoregressive model with eXogenous inputs
BFGS	Broyden–Fletcher–Goldfarb–Shanno
BPNN	Back propagation neural network
CE	Cross-Entropy
CNN	Convolutional Neural Network
CV	Cross-Validation
CWT	Continuous Wavelet Transformation
DE	Differential Evolution
DFT	Discrete Fourier Transform
DOF	Degree of Freedom
DWT	Discrete Wavelet Transform
EMD	Empirical Mode Decomposition
FDOT	Florida Department of Transportation
FE	Finite Element
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
HEC-18	Hydraulic Engineering Circular No. 18
IL	Influence Line
IMU	Inertial Measurement Unit
IR	Impulse Response
KKT	Karush–Kuhn–Tucker conditions
KNN	K nearest neighbours
L-BFGS	Broyden–Fletcher–Goldfarb–Shanno
LightGBM	Light Gradient Boosting Machine
M5	Regression Tree
MAE	Mean Absolute Error
MCC	Matthews Correlation Coefficient
ML	Machine Learning

MR-IL	Moving Reference Influence Line
MSE	Mean Square Error
NSE	Nash–Sutcliffe Efficiency
NSGA-II	Nondominated Sorting Genetic Algorithm
OOB	Out-of-bag
PCA	Principal Component Analysis
PSD	Power spectral density
ReLU	Rectified Linear Unit
RI	Relative Importance
RILA	Rail Infrastructure Alignment Acquisition
RBF	Radial Basis Function
RF	Random Forest
RMSE	Root Mean Squared Error
ROC	Receiver Operating Characteristic
SHAP	SHapley Additive exPlanations
SMO	Sequential Minimal Optimisation
SMOTE	Synthetic Minority Over-sampling Technique
STFT	Short-Time Fourier Transform
SVM	Support Vector Machine
TPDI	Track Profile Damage Index
TTB-2D	Train–Track–Bridge 2-D
VBI	Vehicle-Bridge Interaction
XGBoost	eXtreme Gradient Boosting

CHAPTER 1: INTRODUCTION

This chapter (i) establishes the motivation for the thesis by outlining the scour hazard—illustrated by notable failures—and the limitations of prevailing detection and inspection practices; (ii) frames the displacement-based, train-mounted concept, i.e., drive-by sensing with Machine Learning (ML) as a scalable route to scour detection and localisation; (iii) sets out the thesis objectives and methodology (including models and workflow diagrams); (iv) states the original contributions and the policy for integrating prior publications; and (v) summarises the structure of the thesis.

1.1 Motivation and Problem Statement

Flooding has been identified as the most significant natural disaster in the United States (Perry (2000)). From 1980 to October 2024, the United States witnessed 44 major flood events, each of which resulted in damages exceeding one billion dollars, with an average annual economic impact amounting to $4.5 billion, and at least 738 fatalities and bridge collapses (Center for Disaster Philanthropy, 2024). Given the high frequency of flooding events and the substantial damage they cause, it is reasonable to conclude that scour—the erosion of riverbed or seabed material around foundations during high flows—remains a leading cause of bridge distress and collapse. Its effects are often sudden and catastrophic because scour removes the confining support at piers and abutments, undermining load paths that otherwise appear serviceable under routine inspection. A 1973 country-scale Federal Highway Administration research study, which included 383 catastrophic flood-induced bridge collapses, demonstrated that a quarter of these were related to pier damage, while the rest were influenced by abutment damage (Arneson et al., 2012). In April 1987, the New York State Thruway bridge over Schoharie Creek collapsed when scour around its third pier caused the pier to fail, leading to the collapse of the adjacent spans during a significant flood event. (Suro et al., 2020). After two years, the US-51 bridge spanning the Hatchie River in Tennessee failed due to flood-induced scour that undermined its foundation. Even though most scour countermeasures remained intact following floods with considerable stream flows, the report also states that observed occurrences of shifting, slumping, scour holes, and other damages indicate that relying solely on visual inspections may fail to fully detect hidden weaknesses or deterioration. In March 2001, following substantial precipitation and consequent flooding, the Hintze Ribeiro Bridge in Portugal

suffered a collapse, resulting in the tragic loss of 59 lives (Jornal Da Noticias, 2023). This incident was probably caused by scour. In November 2009, a railway bridge spanning the River Crane near Feltham in West London partially collapsed due to scour undermining its east abutment. Although the structural failure disrupted rail services, no fatalities or injuries were reported (Rail Accident Investigation Branch, 2014). The challenges of detecting bridge scour with traditional methods, such as accessing submerged components under difficult hydraulic conditions, have limited timely identification and led to undetected scour in bridges. These limitations are particularly critical during extreme hydraulic events. A recent example is the winter of 2015–2016, which brought record river flows across Northern Ireland, Britain, and parts of Wales, with many sites exceeding 100- to 200-year return periods in the UK. Extreme discharges and saturated ground created ideal conditions for scour, contributing to bridge collapses, including historical ones (Marsh et al., 2016). In June 2019, while strengthening works of Bishopsford Road Bridge in Mitcham, UK, were continuing to address existing scour risk, flooding, and temporary jack post installations likely intensified local scour, leading to the partial collapse of the northern arch (Malena et al., 2021; London Borough of Merton, 2024). A very recent example occurred in June 2023, when the Yellowstone Twin River Bridges in Stillwater County, Montana, collapsed as a train was crossing, sending ten railcars—some carrying hazardous materials—into the Yellowstone River (Figure 1.1). The collapse, believed to have been triggered by severe river scour caused by record river flows from snowmelt and heavy rainfall, highlights the acute risk that scour poses to bridge safety and infrastructure (Rail Magazine, 2023).



Figure 1.1. Railway bridge collapse due to scour (copied from Rail Magazine, 2023).

These structural damages are merely illustrative examples, underscoring the importance of monitoring throughout the service lives of bridges at the risk of scour. Scour monitoring through underwater inspections conducted by divers remains a common practice due to its directness. However, it carries significant limitations and risks. The process is highly dependent on hydrometeorological conditions—

turbidity and strong flow can hinder reliable observation precisely when the risk of scour is most significant. Moreover, the method exposes personnel to safety hazards and carries the possibility of oversight or incomplete detection of scour features under poor visibility or challenging hydraulic conditions. To mitigate the limitations of manual inspections, recent efforts have focused on sensor-based monitoring that enables continuous and safer scour detection. The sensing equipment to monitor scour could be directly implemented on piles to track scour depth, such as micro-electromechanical systems (Lee et al., 2025), soil-buried devices such as magnetic sliding collars (Nasssif et al., 2022), and electromagnetic sensors (Michalis et al., 2015)) to track changes in soil characteristics that result from scour. This type of research often employs hydraulic scour models to simulate expected scour behaviour. Their outputs are validated, calibrated, and refined through comparison with field and laboratory measurements, enabling improved predictive performance, scenario analysis, and increased understanding of scour processes. The downside of using these sensors lies in their costly and site-specific deployment and maintenance, as well as their vulnerability to damage or loss during floods. Additionally, biofouling and debris can hinder long-term functionality, and achieving network-scale coverage across numerous bridges remains impractical. Riverbed changes and scour progression could also be monitored remotely through advanced devices such as sonars (Hou et al., 2022), radars, and lasers (Raju et al., 2022). While these techniques provide reliable and detailed data, their widespread implementation is generally hindered by high costs and logistical challenges.

An alternative approach to monitoring bridge scour involves tracking changes in the structural behaviour of the bridge, which serves as an early warning system to indicate potential damage. This type of monitoring enhances safety by eliminating the need for personnel to access submerged foundations or flood-prone areas where scour may compromise structural stability. Data acquired through sensors installed on the bridge is processed using numerical methods, which are employed to track changes in the structural response resulting from damage. Some of these methods are wavelet transforms (Van der Auweraer and Peeters (2003)), the mode shape ratio method (Malekjafarian et al. (2020c)), and the Hilbert-Huang transform (Malekjafarian and O'Brien (2017)), all of which are used to transform acceleration signals into useful structural response information. Structural characteristics such as modal frequencies (Xiong et al. 2022, Prendergast et al., 2017), and mode shapes (Malekjafarian et al. (2020a)) can be extracted from the monitored accelerations to track the variations in the bridge's response to scour-damaged stages. Research in this category utilizes on-bridge sensor data from bridge-implemented sensors to develop scour detection approaches using numerical bridge models that consider soil-structure or Vehicle-Bridge Interaction (VBI), rather than relying on hydraulic scour models. A recent direct

monitoring study has numerically detected scour by revealing the changes in the frequency trend of a bridge due to scour (Xiong et al., 2022). The approach has been validated using field measurements from a cable-stayed bridge. O'Brien et al. (2023) have collected the accelerations at support locations of a bridge with multiple piers simulated as scoured. To obtain the wavelet-based operational deflection shape, accelerations in the time domain at each support have been transformed into the spatial-frequency domain after being averaged across 1,000 vehicle crossings. The changes in the wavelet-based Operating Deflection Shape amplitudes due to scour of the bridge have indicated its presence.

Instrumenting bridges with sensors, known as direct monitoring, has emerged as a promising alternative to conventional inspections. This approach offers substantial benefits in improving public safety through early risk detection and in minimising downtime by enabling continuous, automated condition assessment. Nevertheless, it is not feasible for all bridge types and may involve significant initial investment and recurring maintenance costs. In addition, the implementation and maintenance of sensor systems can occasionally cause service disruptions, which limit their practicality for in-service structures (Chandrasekaran, 2019). These trade-offs make it essential to develop scalable and cost-efficient monitoring methodologies that maintain reliability without imposing excessive operational burdens.

Given the limitations of current inspection practices and the practical constraints of direct instrumentation, infrastructure owners require a reliable and scalable means to detect and localise scour during normal bridge operations—without necessitating traffic closures or hazardous access. Specifically, this thesis addresses the problem of designing and validating a monitoring approach that:

i. Operates under in-service conditions (no traffic restrictions) and remains robust under operational and environmental variability (speed, axle loads), within defined measurement-noise thresholds. Performance is verified against field measurements where available, acknowledging that initial model training uses simulations.

ii. Detects scour presence at the bridge-level with sufficient updating frequency to support early warning.

iii. Localises damage to specific supports so that inspections can be targeted efficiently.

iv. Produces per-window model scores for batched inputs and applies a fixed decision threshold to make scour/healthy decisions.

v. Discusses external-validity and synthetic–field transfer considerations (cross-bridge, temporal and vehicle transfer), noting that field data are limited and that further verification is needed.

1.2 Gaps in the Literature and Research Opportunity

As a cost-effective alternative to direct monitoring, drive-by (indirect) monitoring is employed, where sensors are installed on vehicles rather than on the bridge itself. This method enables data collection as a vehicle traverses the network. The network-level costs are hence reduced since fewer sensors and minimal on-site personnel are required, and the need for complex installations or maintenance activities on the bridge is eliminated. Because sensing is performed under normal traffic conditions, it avoids service disruptions and enables long-term monitoring. For the method to be operational, signal interpretation and analytics require adequate accuracy and correctly synchronised timing metadata such as timestamps, speed, and geolocation (Lin et al., 2004; Koo et al., 2013). The collected signals (such as accelerations, longitudinal displacements, and pitch responses) are then analysed to infer the structural behaviour of the bridge and to identify potential damage, such as scour-induced stiffness loss. However, interpreting vehicle-based measurements can be complicated by factors such as track irregularities and bridge deflections, which have been the focus of previous research exploring their correlation and impact on monitoring accuracy.

The theoretical correlation between bridge deflections and track profiles has been explored in previous studies (Matsuoka & Tanaka, 2023; Salcher et al., 2019). The study by Salcher et al. (2019) suggests that the peak dynamic deflection response of the bridge resulting from various random rail irregularities can be modelled as a Gaussian distribution. The research presents dynamic amplification coefficients for bridge acceleration and displacement responses in the presence of random rail irregularities. However, the amplification factor is not particularly helpful in distinguishing between bridge displacements and rail irregularities. In the research of Matsuoka and Tanaka (2023), the maximum difference in track elevations between the first and last carriages of a train crossing is calculated to estimate the maximum bridge displacement. The deflections of five bridges that have been crossed by in-service trains with travel speeds of 100 km/h have been estimated with an error of up to 10%. In addition to the speed limitations imposed by the study, the noise generated by track irregularities and the poor quality of measurements at rail joints represent significant drawbacks.

A significant percentage of the damage caused by scour is characterized by horizontal movement. Bridge scour events after floods have resulted in large horizontal displacements (Kadono et al., 2020). Following a flood in Japan in 2018, the fourth pier of a multi-span railway bridge became significantly inclined, causing the tracks to be displaced 0.75 m downstream (Kadono et al., 2020). According to Elsaid and Seracino (2014), horizontal bridge mode shapes are sensitive to reduced support stiffness

caused by scour. Therefore, significant horizontal mode shapes could be utilized to determine the scour level. However, it is also possible to model scour with reduced vertical stiffness at the bridge support (Fitzgerald et al., 2019b) since the soil beneath the foundation pad is removed during scour. Previous laboratory-validated research has demonstrated that the reduction in vertical pier stiffness due to scour alters the dynamic behaviour of the bridge, particularly the first mode shape, as measured by the vibrations caused by the train's passage (Fitzgerald et al., 2019b). This approach is later validated by Tola et. al. (2025) using displacement data from a combination of field measurements and synthetically generated track profiles. Up to a specific measurement noise level, this method successfully detects scour in an existing railway bridge using train crossings with varying travel speeds and weights. However, multiple scour cases are not covered in this study. Wang et al. (2024) demonstrate that bogie-mounted accelerometers on passing maintenance trains can detect reductions in bridge substructure stiffness at high speeds. The root mean square of vertical bogie acceleration is used as a time-domain statistical energy metric, rather than frequency-domain methods such as modal parameter extraction or wavelet decomposition. The method offers network-wide monitoring with minimal computational overhead; however, further testing under realistic scour scenarios and varied operational conditions is needed. Zhang et al. (2022) propose a drive-by scour detection method using vehicle axle accelerations processed with a wavelet transform, where scour is modelled as a reduction in foundation stiffness. A statistical-wavelet approach, incorporating probability distributions and a pier index, enabled both detection and localisation. The method has been validated through numerical simulations and laboratory-scale experiments, showing reliable detection when at least 50 runs are analysed. However, the approach is less sensitive to low scour levels and requires a dedicated instrumented vehicle, with no full-scale field verification.

The data-driven ML approach facilitates the use of monitoring data without requiring explicit mathematical models to define the complex relationships between input and output vibrations. By identifying patterns in signals, ML can detect and localise scour and make predictions based on historic data. Many studies apply ML algorithms to predict scour depth around piers, employing either single methods like Convolutional Neural Networks, i.e., CNNs (Zhang & Zhao, 2020), or hybrid algorithms combining multiple techniques (Khosravi et al., 2021) to improve accuracy. These ML models are typically trained and validated using field or laboratory data that include flow properties corresponding to measured scour depths, with performance often compared against empirical formulas or other ML models.

Besides scour depth prediction, ML techniques are also applied in drive-by data damage detection. For example, Artificial Neural Networks (ANN) and CNN models trained on laboratory-generated vehicle accelerations can detect cracks in bridge beams (Corbally & Malekjafarian, 2024). While both models

detect damage, CNNs have demonstrated enhanced capabilities in classifying damage type, location, and severity, albeit with a slight tendency to overestimate severity. ML has become increasingly prominent in structural health monitoring because it extracts features, detects hidden relationships, and generalizes to new conditions directly from monitoring data, without the need for explicit physical or mathematical modelling. Recent work by Malekjafarian et al. (2020b) employed a drive-by monitoring method, analysing VBI simulations, where axle accelerations underwent Fast Fourier Transform (FFT) analysis. An ANN combined with a Gaussian process was used to predict healthy responses. This method formed a damage index from prediction errors and demonstrated robustness to road roughness and noise up to 5%. However, it relied solely on simulations and did not address damage localisation. Calderon Hurtado et al. (2023) proposed an unsupervised drive-by framework using an adversarial autoencoder trained only on healthy acceleration spectra. Validated with noisy numerical simulations and laboratory-scale experiments, the adversarial autoencoder outperformed Principal Component Analysis (PCA) and conventional autoencoders in detecting and quantifying damage severity at different locations. However, it requires a dedicated instrumented vehicle, assumes constant speed, and lacks validation at full scale.

A review of the extant literature reveals a consensus that vehicle-borne sensing and ML can reveal changes related to scour. However, the majority of demonstrations are constrained in terms of sensing modality, scale, and operational realism. To enable deployment across a rail network, the following gaps should be addressed:

i. Sensing modality (track-geometry/displacement). Drive-by scour detection using longitudinal track-geometry, i.e., displacement measurements from routine in-service trains remains comparatively under-explored at scale, despite its practical relevance for rail operations.

ii. Operational robustness (variability and batching). Speed, axle loads, vehicle configuration, and track state can mask or mimic scour-related changes. This motivates batch-based inference (sliding-window aggregation) to stabilise decisions under realistic variability.

iii. Actionable outputs (support-level localisation). For inspection planning, operators need “where is it?” readouts at the support-level, not only bridge-level indicators. Support-level labels and localisation-oriented evaluation therefore remain a key practical need.

iv. External validity and deployment guidance. Evidence is still limited on transfer across bridges and sensing configurations (including front- vs rear-mounted sensing), and on what site-specific calibration are needed to make outputs decision-ready.

1.3 Objectives and Methodology of the Thesis

1.3.1 Objectives of Thesis

This thesis develops and validates a drive-by framework that utilizes far-from-axle longitudinal track-profile measurements from routine in-service train passages and machine-learning classifiers to detect scour at the bridge-level and localise it to individual supports. The framework is applied on a case study that combines measured track profiles with synthetically generated scoured profiles. Monitoring data are supplied by a UK experimental campaign commissioned by Network Rail using Fugro's RILA system. Part of this work was carried out within the Shift2Rail IN2TRACK3 project (Demonstrator 70, "Scour monitoring from train-induced vibrations"). IN2TRACK3 concluded before the finalization of this thesis. The specific objectives are:

i. *In-service operation and robustness*. Use routine train passages and operate under realistic variability (speed, axle loads, vehicle configuration, and track condition).

ii. *Scour damage detection*. Detect the presence of scour at bridge-level using daily, sliding-window batches (7, 15, 31, or 61 days).

iii. *Scour localisation*. Locate scour to individual supports for targeted inspection.

iv. *Longitudinal track-profile data*. Exploit track-geometry (longitudinal level) profiles from RILA as the sensing input.

v. *Validation and uncertainty*. Demonstrate performance across bridges and time using external cross-validation (CV) and robustness analyses.

1.3.2 Methodology Overview

The approach comprises two complementary strands—physics-informed detection and machine-learning (ML) detection and localisation of scour, both built from data on longitudinal track profile. The flowchart in Figure 1.2 summarises the methodology of the thesis.

A. Physics-informed scour detection methodology — Chapter 3

i. *Measured track profile.* Far-from-axle displacement-based track-geometry profiles are obtained from an onboard sensing system (RILA). The measurements were collected in a UK experimental campaign led by Network Rail using Fugro's RILA measurement system, with parts conducted under the Shift2Rail IN2TRACK3 programme.

ii. *Theoretical track profile.* The bridge-induced displacement response is computed using Moving Reference Influence Lines (MR-ILs), derived via the theorem of virtual work within a Finite Element (FE) model implemented in MATLAB. The model includes vertical support springs, where scour is represented as a reduction in spring stiffness. Two sensor configurations are assessed: front-mounted (leading) and rear-mounted (trailing) RILA installations. Rail irregularities are superimposed onto the simulated displacements to derive the theoretical longitudinal track level (track profile) for each configuration.

iii. *Pier stiffness optimisation.* The pier stiffnesses under healthy and scoured conditions are found by minimising the sum of squared differences between the measured and MR-IL-based theoretical profiles, through optimisation using the Cross-Entropy (CE) algorithm (Chapter 3.3).

iv. *Scour indicator.* A reduction in pier stiffness between the healthy and damaged conditions is interpreted as evidence of scour.

v. *Case-study setup (Chapter 3).* The methodology is applied to a real bridge, using measured profiles for the healthy stage and synthetically generated scoured profiles (MR-IL responses combined with measured irregularities) to define the ground truth for detection and localisation.

B. Extension to Network Level: ML-based scour detection and localisation — Chapters 4, 5, 6

Synthetic train crossings. Generating a diverse Train–Track–Bridge, i.e., Train–Track–Bridge (TTB)-2D version 4.4 (Cantero, 2022) dataset spanning speeds, loads, vehicle configurations, bridge types/spans, and single/multiple-pier scour scenarios (Chapter 4).

i. Feature engineering.

a) *Type-A (wavelet).* Applying the Discrete Wavelet Transform (DWT) to the longitudinal track profile and, for each wavelet level, derive parameters from the wavelet coefficients (e.g., energy, skewness, kurtosis); use these as features for ML models.

b) *Type-B (system identification):* Estimating an Autoregressive model with eXogenous inputs (ARX) directly from input–output data and deriving Markov parameters that summarise the dynamics; for localisation, run ARX per support within windows to yield per-support Markov vectors.

ii. *Feature selection.* Two strategies are used: (i) embedded relative-importance ranking; and (ii) Nondominated Sorting Genetic Algorithm (NSGA-II multi-objective selection, balancing predictive performance, number of features, and selection stability across folds.

iii. *ML Models and optimisation for detecting scour damage*. Train ANN, Random Forest (RF), and Support Vector Machine (SVM) classifiers; tune hyperparameters with grid search (for ANN) and random search (for RF, SVM) under nested Cross-Validation (CV), with transfer-tuning from batch size 7 to larger batch sizes of 15, 31, and 61.

iv. *ML models and optimisation for locating scour damage.* Training and testing ANN, RF, and SVM classifiers; tuning all hyperparameters with random search under nested CV, with transfer-tuning from batch size 7 to larger batch sizes of 15, 31, and 61.

v. *Localisation read-out.* Aggregating per-support probabilities across outer-fold tests to report which supports are flagged as scoured, with spatial consistency checks.

Evaluation protocol — Chapters 6, 7

i. *External validity*. Running cross-bridge and cross-time external CV and comparing front- vs rear-mounted sensing. Set-up and results are provided in Chapter 6, while synthesis and external-validity discussion are in Chapter 7.

ii. *Metrics and readouts*. Computing accuracy and precision/recall/F1 ((the harmonic mean of precision and recall) for healthy vs damaged classes; confusion analyses; calculating per-support localisation metrics; reporting severity buckets (mild 5–10%, moderate 10–15%, or severe 15–20%); and examining batch-size trade-offs (primary reporting in Chapter 6, with overall conclusions drawn in Chapter 7).

This thesis shows potential for network-scale application by assembling multi-bridge datasets and applying sliding-window inference over successive windows of data. It reports outputs at both bridge-level and support-level as threshold-based binary decisions (healthy vs scoured). Temperature fluctuations can affect the flexural rigidity of concrete bridges; however, their impact at the foundation level is relatively small, so temperature is not a primary concern for the proposed approach. The methodology focuses on the structural consequences of scour observed via displacement measurements, rather than the hydrodynamic processes causing it. Practical system integration (for example operational data pipelines, model management, and deployment workflows) is outside the scope of this thesis and is outlined as future work in Chapter 7.

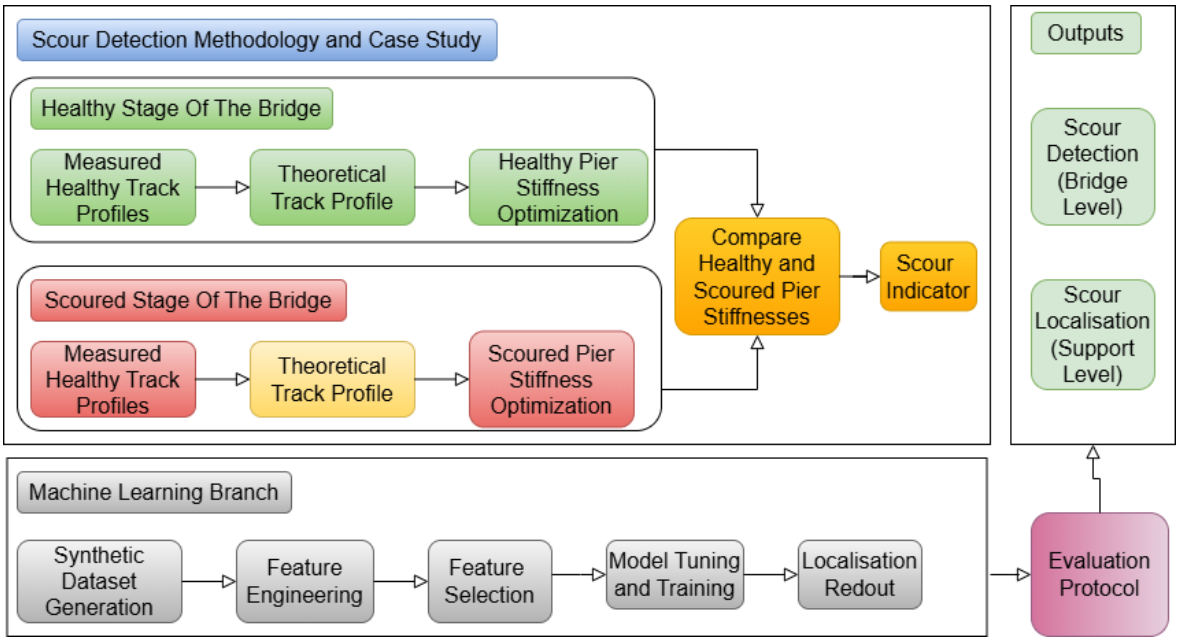


Figure 1.2. Flowchart of the methodology.

1.4 Contributions, Publications, and Manuscript-Integration Strategy

1.4.1 Original Contributions of the Thesis

i. *Displacement-based drive-by scour framework.* Building on prior scour-as-stiffness-loss concepts, this thesis develops a drive-by approach using longitudinal track-profile (displacement) measurements rather than acceleration. The case study combines in-service RILA-measured healthy profiles with synthetically generated scoured profiles to complement the field data.

ii. *Pier-stiffness identification and scour indicator via CE + MR-IL.* Healthy and scoured pier stiffness are estimated through CE optimisation by minimising the mismatch between MR-IL–based theoretical track profiles and measured/synthetic profiles. The resulting stiffness change is interpreted as the scour indicator (physics cross-check of the data-driven pipeline).

iii. *ML pipeline for displacement data with tailored feature families and selection.* The thesis integrates displacement-based track profiles into a supervised classification pipeline by extracting DWT-based descriptors (Type-A) and ARX/impulse-response (Markov) descriptors (Type-B) and applying feature ranking and multi-objective selection (including NSGA-II) to balance predictive performance and feature set compactness.

iv. *Evaluation across algorithms, batching, and localisation readouts.* The thesis benchmarks ANN/RF/SVM for bridge-level detection and extends the framework to support-level localisation, using sliding-window (batch-based) inference over multiple window lengths and assessing effects

such as sensor orientation (front vs rear) and operational variability within the study's experimental design.

1.4.2 Publications

During the development of this PhD thesis, the following papers were published.

Journals

- **Tola, S.**, Tinoco, J., Matos, J. C., and O'Brien, E. (2023). Scour Detection with Monitoring Methods and Machine Learning Algorithms—A Critical Review. Applied Sciences, 13(3), 1661. https://doi.org/10.3390/app13031661.
  Metrics: Clarivate JCR (2023) JIF 2.5; Q1 in *Engineering, Multidisciplinary*. SCImago Q2 (*Engineering—Misc.*; SJR 2024 = 0.521).

  Award: Applied Sciences – Best Paper Award (June 2025).

- **Tola, S.**, O'Brien, E., Cantero, D., Tinoco, J., Matos, J. C., Bose, T., and Berkers, J. (2025). Drive-By Detection Of Scour In A Railway Bridge. Journal of Bridge Engineering. https://doi.org/10.1061/jbenf2/beeng-7422.
  Metrics: Clarivate JCR (2023) JIF 3.5; category: Engineering, Civil (percentile 72.1% reported). SCImago Q1 (Civil and Structural Engineering; SJR 2024 = 1.167).

- Gomes Correia, A., Roshan, M. J., Moldovan, I. D., Tinoco, J., Owusu-Ansah, D., **Tola, S.**, Parente, M., and Azenha, M. (2025). Advancing transportation geotechnics through digital technologies. Soils and Rocks, 48(3), e2025007525. https://doi.org/10.28927/SR.2025.007525.
  Metrics: SCImago Q3 in Geotechnical Engineering and Engineering Geology (SJR 2024 = 0.313).

Project Report

- Tinoco, J., **Tola, S.**, and Matos, J. (2024). Scour monitoring from train-induced vibrations. In A. Ekberg and P. Edlund (Eds.), In2Track3 Project Concluding Report (Section: DEMONSTRATORS). Chalmers University of Technology and Trafikverket, in collaboration with the University of Minho.

Conferences

- **Tola, S.**, Tinoco, J., O'Brien, E., and Matos, J. C. (2025). A machine learning-driven framework for scour detection in railway bridges using onboard sensing. In M. Briffaut and J. M. Torrenti (Eds.), Concrete structures: extend lifetime, limit impacts (Proceedings of the 2025 fib International Symposium, Antibes, France, June 16–18, 2025). AFGC.

- **Tola, S.**, Tinoco, J., O'Brien, E., Matos, J. C., and Cantero, D. (2024). Detecting railway bridge scour using in-service train signals and machine learning tools. In Proceedings of the IABSE Congress 2024, September 25-27, 2024, Costa Rica. IABSE. ISBN: 978-3-85748-205-2.
- **Tola, S.**, Tinoco, J., Matos, J., and O'Brien, E. (2023). Vulnerability assessment of existing bridges to scour, based on an indirect monitoring approach and machine learning tools. *Transportation Research Procedia*, 72, 751–758. https://doi.org/10.1016/j.trpro.2023.01.142 — Conference proceedings (Elsevier Procedia; no JCR IF/quartiles; SCImago: Conferences and Proceedings, Transportation).
- **Tola, S.**, Tinoco, J., Matos, J. C., and O'Brien, E. (2022). Vulnerability assessment of existing bridges to scour: An indirect monitoring approach. In N. Holmes, C. de Paor, and R. P. West (Eds.), Civil Engineering Research in Ireland 2022 and Irish Transport Research Network 2022: Conference Proceedings (pp. 109–113). Civil Engineering Research Association of Ireland. ISBN: 978-0-9573957-5-6.

1.4.3 Manuscript-Integration Policy and Rights

To maintain alignment with peer-reviewed work while presenting new material, limited portions of two journal papers are incorporated.

- *Scope and placement of incorporated material*. Material from the ASCE journal paper is incorporated mainly in Chapter 3; material from the MDPI review paper is incorporated mainly in Chapter 2.
- *ASCE journal content (reuse conditions)*. Reuse is kept below the 25% threshold of the overall thesis. Where figures, tables, or text are reused, complete credit lines are provided, and the Version of Record (VoR) DOI is cited. The Author Accepted Manuscript (AAM) may be deposited online or in an institutional repository with a link to the ASCE Library record, in accordance with ASCE policy. Any third-party items are cleared separately for thesis use.
- *MDPI review content (reuse conditions)*. The review is licensed under CC BY 4.0. Portions of Chapters 2 and 5 are reused with attribution under CC BY 4.0. Where adaptations are made, captions state the changes and reference the original study.
- *Conference papers*. Conference outputs are used as sources when writing Chapter 1 (context, motivation, positioning) but are not integrated verbatim elsewhere in the thesis.

- *General practice*. Wherever published material is reused, clear attribution is provided at the point of use, and any permissions/licence details (as applicable) are documented in Appendix D (Ethics, IP, and Data-Sharing).

1.5 Thesis Structure

Chapter 1 – Introduction. Frames the problem of bridge scour, justifies indirect (drive-by) monitoring with ML, states the aims and research questions, and outlines the contributions and publication record. It then sets the objectives and methodology at a high level with a flowchart and explains how peer-reviewed material is incorporated into the thesis.

Chapter 2 – Background and Literature Review. Synthesizes scour mechanisms and bridge vulnerabilities, reviews conventional assessment and monitoring methods, and surveys drive-by/Vehicle–Bridge Interaction (VBI) concepts and signal processing. Recent indirect-monitoring studies are summarised. ML research for scour prediction—depth estimation and vibration-based detection/localisation—is mapped, including standard feature sets and algorithms. The optimisation tools used later in Chapters 3, 5, 6, and 7 (CE, NSGA-II, grid/random search with nested CV) are introduced. The chapter concludes by situating the present work within the existing literature and highlighting the gaps that the thesis addresses.

Chapter 3 – Drive-By Scour Detection Using RILA — Method and Case Study. Presents the physics-informed strand using far-from-axle displacement/level measurements (RILA). The measurement source and rationale are stated, followed by the bridge–track representation via MR-ILs and the modelling choices for scour, vehicle, track, and bridge. CE optimizes to find pier stiffness by comparing measured/theoretical profiles, and the operational workflow (baselines, cadence, preprocessing, and quality checks) is specified. The framework is then applied to a real bridge, where train crossings and profile alignment are described. Healthy versus scoured profiles are constructed by combining MR-IL responses with measured irregularities, and support-level labels are used for evaluation. A concise sensitivity analysis shows limited change below ~5% scour and an apparent increase from 5% onward. This threshold is adopted later in the ML framework. For localisation, the scour-affected zone length observed from MR-IL displacement responses sets the DWT window at each support.

Chapter 4 – Data Generation for ML Framework. Presents the simulation-based framework for method development and ML training/evaluation: the TTB-2D version 4.4 (Cantero, 2022) interaction model software. The design of scour single-pier and multiple-pier scoured scenarios; sensor-position effects (front vs rear); and robustness/sensitivity analyses via noise injection and stress-testing. Dataset assembly and splits are documented.

Chapter 5 – Machine-Learning Framework. Defines the end-to-end ML pipelines for bridge-level detection and support-level localisation. Task framing, data splits, and sliding-window batches are set out. Feature engineering comprises Type-A (DWT-based measures) and Type-B (ARX-to-Markov) descriptors. Feature selection utilizes embedded importance and NSGA-II, with an explicit balance between predictive performance, feature count, and selection stability. ML Models (ANN, RF, SVM-RBF), scaling, and the batch-based decision scheme are specified. Hyperparameter-tuning principles and spaces are presented, followed by unified pipelines and evaluation metrics for detection and localisation.

Chapter 6– Machine Learning Model Results. Reports stiffness-inference results for the physics-informed strand, then presents damage detection (bridge-level) and localisation (support-level) results, including hyperparameter outcomes, model/feature/sensor comparisons, per-support metrics, and multi-pier analyses. Sensitivity studies, as well as feature importance, are provided, along with comparisons to prior indirect approaches.

Chapter 7 – Conclusions, Validity and Future Work. Synthesises the findings against the thesis aims for the MR–IL drive-by methodology and the ML models for scour detection and localisation. Discusses validity and robustness, with emphasis on the synthetic–field gap, operational variability and modelling assumptions, and decision-threshold choices. States limitations and threats to validity, and outlines future work including targeted field campaigns, transfer studies across bridges/networks, integration of contextual environmental/hydraulic data, and improved deployment-oriented decision rules and model updating.

Annexes. Provide mathematical derivations and algorithms, including the MR-IL-based bridge model MATLAB code, CE algorithm, ARX/Markov Parameters, full hyperparameter grids and final settings, dataset parameters, inventory and variable references, and ethics/IP/data-sharing information, including permissions and usage rights.

CHAPTER 2: BACKGROUND AND LITERATURE REVIEW

This chapter summarises the foundations and recent developments relevant to bridge-scour detection and localisation. Section 2.1 introduces scour mechanisms and bridge vulnerability, frames identification approaches, and outlines sensing and data-acquisition technologies. Section 2.2 reviews the literature in two streams: conventional monitoring (direct bridge-mounted and indirect/drive-by) and machine-learning research. The ML background covers feature extraction and dimensionality reduction, parametric system models (AR/ARX), the classifiers used in this thesis (ANN, RF, SVM), model selection and performance estimation (with stratified and time-ordered CV, and nested CV for tuning), and optimisation methods in ML pipelines (grid/random/Bayesian hyperparameter search, gradient-based vs metaheuristic schemes, the CE optimisation, NSGA-II multi-objective feature selection, and transfer-tuning). The ML studies are then organized into scour-depth prediction and scour detection and localisation from structural vibrations, followed by a synthesis that highlights areas of agreement, limitations, and open issues.

2.1 Scour Mechanisms, Vulnerability, and Identification Approaches

2.1.1 Bridge Scour, Consequences and Structural Vulnerability

Bridge scour refers to the erosion of soil around structural elements that occurs due to the flow of water. The term scour is commonly used to describe the lowering of the riverbed level caused by hydraulic erosion, which can remove supporting material around bridge foundations (Melville and Coleman, 2000). In this thesis, scour is considered through its structural consequence: the decrease of effective soil support beneath the foundation, which is represented as a reduction in support stiffness. May et al. (2002) classify scour into four types: natural, contraction, local, and total scour. Natural scour arises from the inherent variability of river flows and sediment regimes, while contraction scour occurs when channel narrowing increases flow velocity and, consequently, bed shear stresses. Local scour develops due to concentrated turbulence generated by structural elements that obstruct the flow. Total scour is the combined effect of all contributing mechanisms (Pizarro et al., 2020). An alternative, time-based classification distinguishes long-term and short-term scour according to the period over which the erosion develops. In practice, for bridge foundations, local and contraction scour are typically the governing types.

Scour is one of the most common causes of bridge collapses, resulting in substantial loss of life and severe economic disruption. Between 1989 and 2000, the primary causes of 503 bridge collapses were associated with scour at piers and abutments (Wardhana & Hadipriono, 2003). The direct financial costs of scour-related incidents are frequently substantial. As noted by Prendergast and Gavin (2014), during a single flood event in 1993, at least 22 of 28 bridge collapses in the upstream Mississippi and downstream Missouri river valleys were attributed to scour. In New Zealand, annual damages due to scour have been estimated at NZ$36 million (Macky, 1990). For railway infrastructure, 138 flood-related incidents of railway bridges were recorded in the UK between 1846 and 2013, with 17 scour-related incidents identified across the UK and Ireland, including six involving railway bridges and viaducts (Van Leeuwen & Lamb, 2014). Direct effects include material damage, fatalities, and environmentally harmful failures of structural components; indirect effects encompass wider financial losses arising from functional disruption and knock-on impacts on adjacent assets, services, and communities (Faber, 2008). Indirect or intangible losses—such as reputational damage that may deter future investment or reduce tourism revenue—also arise and can be significant, although they are harder to quantify. Therefore, the vulnerability of bridges to scour must be assessed, and appropriate action must be taken. To this end, piers, abutments, foundations, and piles at risk of scour can be protected by taking countermeasures such as backfilling, using sheet piles or sacrificial piles, and repairing damaged elements with ripraps or splitter plates before widespread damage occurs. Even after bridges have been adequately protected against scour, it is important to continue monitoring the scour conditions due to changing environmental conditions.

2.1.2 General Overview of Scour Identification Approaches

Scour detection is a complex task for several reasons. The process occurs underwater and often in turbid, hazardous environments that limit access and visibility. Hydraulic conditions, sediment gradation, and bed mobility vary in space and time, masking or mimicking damage signatures. Sensors can be challenging to install and maintain at foundations, as they are prone to noise, drift, biofouling, and debris. Predictive models are sensitive to site-specific parameters and scaling, while extreme events and operational constraints interrupt measurements and complicate validation. This subsection provides a summary of the various scour detection approaches found in the literature. These include underwater inspections, empirical formulas, hydraulic and representative scour modelling, experimental studies, and probabilistic scour detection methods.

Empirical studies constitute a key approach to scour detection and scour depth estimation (Bihs and Olsen (2011), Nordia et al. (2014), Guan et al. (2019), Oliveto and Hager (2002)). Prior studies are

based on the extraction of scour-length formulas, using either laboratory experimental results or field measurements. The objective of these studies is to quantify the relationship between flow velocity and duration, depth of charge, bed material properties, sediment movement, and substructure geometric properties, and the maximum local scour depth of piers and abutments or stresses in the riverbed. The scour formulas employed include 65-1, 65-2 (Chinese), Melville–Sheppard, MBW, Hydraulic Engineering Circular No. 18 (HEC-18), Dey and Barbhuiya (2005), and Muzammil (2010).

Underwater inspections are classified into two types: visual inspections and equipment-based inspections. In visual inspections, divers undergo a dangerous and challenging search, during which the human eye may fail to detect scour. Ground-penetrating radars, water-penetrating radars, SONAR, and specific laser systems are examples of equipment-based underwater examinations. Penetrating radars mounted on boats or floating units emit electromagnetic waves and receive echoes from the soil. In one study, researchers map the bathymetric profile (water depths and sediment thicknesses) of a riverbed in Northern Ireland and detect scour holes around upstream piers using a water-penetrating radar (Campbell et al., 2021). Another study aims to quantify scour depths in a gravel-bed river in Switzerland and to determine the relationship between the subsurface and the riverbed structure using a ground-penetrating radar (Huber et al., 2018). The researchers detected a four-metre-deep scour hole that bathymetric (water-based) surveys did not capture. SONAR is another technology for collecting underwater images, which helps detect scour holes. Hou et al. (2022) located and measured scour depths by combining SONAR data with a deep convolutional network. Although performance decreased in high turbidity, green-laser technology detected scour around bridge piers economically and safely (Raju et al., 2022).

The scour development process and its relationship to flow, bed material characteristics, and the geometry of the scoured elements have been investigated through hydraulic model-based studies. The research by Bihs and Olsen (2011) is a sample study examining the movement of non-cohesive sediment in sloping areas. Another study by Mutlu (2007) examined the fundamentals of scour development processes and the mathematical modelling of flows around bridge piers/piles, below pipelines, and other structures, such as groins, breakwaters, and seawalls. In the very early stages of scour research, the basic idea behind scour development was believed to be the horseshoe vortex system (Laursen & Toch, 1956). Later, the entire turbulence scheme was shown to be the most important eroding flow process; the horseshoe vortex is only an element of this scheme, and the entire pattern also consists of downflow and acceleration of the discharge around the sides of the pier (Kirkil et al., 2009). Experimental studies have been used to test the effects of different layouts, shapes of piers, and flow parameters on scour (Nordila et al. (2014), Guan et al. (2019), Yağcı et al. (2018), Oben-Nyarko and Ettema (2011)). Scour

in clear water and accumulations around hexagonal arrays of emergent circular cylinders, featuring various solid portions and cylinders with an orientation that resulted in less scour than an individual large pile, were proposed (Yağcı et al., 2018). Another study measured the complex turbulent discharge for different fluid velocities, Reynolds shear stresses, and the density of the horseshoe vortex (Guan et al., 2019). There are also studies on perpendicular wall abutments (Oliveto & Hager, 2002) that investigate how channel width, shape, sediment dimensions, flow depths, and abutment lengths affect scour at piers and abutments.

Furthermore, there have been studies that simulate scour instead of building a hydraulic model that directly represents it. To identify scour, mode shape ratio, apparent profile, eigenfrequencies, and decentralized modal analysis methods are adopted by various studies (Malekjafarian et al., 2018; Prendergast et al., 2016; and Tan et al., 2020). These studies do not aim to measure scour depth, but rather to detect its presence by identifying changes in the dynamic properties of the structure, which serve as indicators of scour. Alternatively, to evaluate the performance of scour-critical bridges, a soil–pile–structure interaction analysis is performed (Klinga & Alipour, 2015). Some researchers have validated their approaches through experiments (Prendergast et al., 2012; Ahmad et al., 2015; Khan et al., 2021), thereby gathering more trustworthy results. Recent studies using scour indexes are detailed in Section 2.2, including their methodologies, model features, outcomes, and limitations.

There are probabilistic approaches that overcome the uncertainties of the deterministic approaches summarised above. Uncertainties in scour identification problems include shortcomings in visual inspections, assumptions, and estimations of parameters (geometric properties, hydraulic parameters, etc.). Since the accuracy of deterministic models is highly dependent on the input, this may lead to over- or underestimation of scour. Probabilistic scour models investigate the probability of structural failure under scour (Kallias and Imam (2016), Muzammil et al. (2022), Tubaldi et al. (2017), Jafari-Asl et al. (2021), Jonkman et al. (2015)). The limit state function is defined as the residual resistance of the structure after accounting for load effects, where the load, resistance, and therefore the limit state function are dependent on random variables (Jonkman et al., 2015). To solve the probabilistic failure problem, various reliability methods exist, including risk-based, numerical, approximation, and semi-probabilistic designs (Jonkman et al., 2015). Changing environmental conditions, such as the effects of climate change, may invalidate initial assumptions and lead to unexpected rainfall regimes, floods, and, consequently, scour issues. The effects of such uncertainty, i.e., gradual change in flow properties, were investigated by Kallias and Imam (2016), and a probabilistic scour failure analysis was performed. The researchers statistically analyse the maximum annual flow of a UK river and aim to predict local pier scour with a Monte Carlo

simulation. Increasing the mean of the expected maximum annual flow shifts the flow and scour-depth distributions to higher values, while increasing variability makes the distributions more spread out. Increasing both mean and variability raises the upper-tail (extreme) scour depths most. The impact of changing flow statistics on failure probability is small for about the first 10–15 years but becomes significant afterwards. It has also been shown that deeper foundations reduce the probability of failure. Another study investigates scour around a complex bridge pier and employs a reliability analysis with an optimisation method, incorporating a semi-probabilistic design (Muzammil et al., 2022). Using this method, these researchers develop a safety factor to reach the desired safety level of the foundation design. Additionally, their sensitivity analysis reveals that pier width and the correction factor are the most effective variables in determining the reliability of the bridge pier against scour. A considerable uncertainty is caused by the memory effects, i.e., the accumulated scouring effects throughout the long service life of bridges (Tubaldi et al., 2017). Using a Markovian approach, the changes in scour depth are aimed to be observed, and the scour depth domain is decomposed into multiple discrete states for various flood events. The research shows that, over time, the probabilistic scour depth distribution evolved and reached a stationary value after the first few years of prediction (Tubaldi et al., 2017). The probabilistic local scour analysis of a bridge pier in a clay and sand mixture sediment is performed using five different reliability methods (Jafari-Asl et al., 2021). These simulation-based methods are line sampling, subset simulation, importance sampling, Monte Carlo simulation, and directional simulation. The most accurate results are obtained through subset simulation.

2.1.3 Data Acquisition and Sensing Technologies for Scour Monitoring

Both ML and conventional monitoring approaches are based on measured or simulated data. However, the way input data is processed creates the main difference between these two approaches. Conventional methods require the creation of models, whereas ML methods eliminate the need to know the relationship between input and output (Ley et al., 2022). Even though it is possible to generate simulated signals, field (full-scale) measurements provide more realistic information. Hence, many different types of sensors have been developed to collect signals. Sensors for monitoring scour were explained under two main clusters (Prendergast et al., 2014): those using devices that measure depth and those sensing the changes in the dynamic properties of the structure. They listed scour hole development devices as follows:

- Single use devices;
- Pulse or radar devices.

- Fiber-Bragg grating (strain) sensors.
- Buried or data-driven equipment.
- Sound wave appliances.
- Electrical conductivity devices.

Sensors such as accelerometers and inclinometers can detect changes in the dynamic properties of a bridge caused by scour. Accelerometers can detect changes in structural frequencies resulting from a reduction in stiffness caused by scour. Inclinometers measure the rotation of structural elements, which is observed in the presence of scour. Bridge scour monitoring devices are shown in Figure 2.1 (Maroni et al. (2020)). With the development of technology, monitoring devices have improved, and alternative systems have been generated. A scour monitoring decision framework was developed, which was based on software (Marr, 2010). This framework helped Minnesota Department of Transportation engineers determine the most suitable fixed scour monitoring equipment for the specific bridge they were working on. It reduced potential issues related to the sensor chosen by the user and provided warnings in the event of abnormal scour activity in the field. One recent experimental monitoring study investigated the influences of scour on bridge frequencies (Boujia et al., 2021). The outcomes of the theoretical model corresponded with the laboratory outcomes. The numerical background of another study was used to decide the location of the sensors and the limitations of the railway bridge monitoring (Purnomo et al., 2021).

Van der Auweraer & Peeters (2003) list the key components of data collection systems as sensing devices, data collection topologies, wireless connection, power supply, synchronization of the data obtained from a set of sensors, environmental effects, and processing systems. Communication aspects include bandwidth and latency limits, link reliability under temporarily disconnects and electromagnetic interference, network/protocol management, and data security. Storage and logging electronics also require attention to on-board capacity and write speed, data integrity and resilience to power loss, durability of the storage medium in harsh environments, and practical data handling (formats, metadata, and offload/backup). The data acquisition system must be adequately implemented in accordance with the structure and service conditions. For large-scale bridges, locating, implementing, and maintaining sensors should be well-planned, as some parts of the structure may be inaccessible, such as the underside of the foundation pad. Although indirect monitoring techniques do not disrupt service conditions, direct monitoring systems require service conditions to be carefully considered. There may be time restrictions during the day or parts of the structure that cannot be closed for service to accommodate data acquisition. After

obtaining data through sensors, wireless connections are used to transfer signals. Wireless sensing units have been enhanced when combined with low-cost mobile devices, and these units continue to be developed for improved software and hardware (Lynch, 2007).

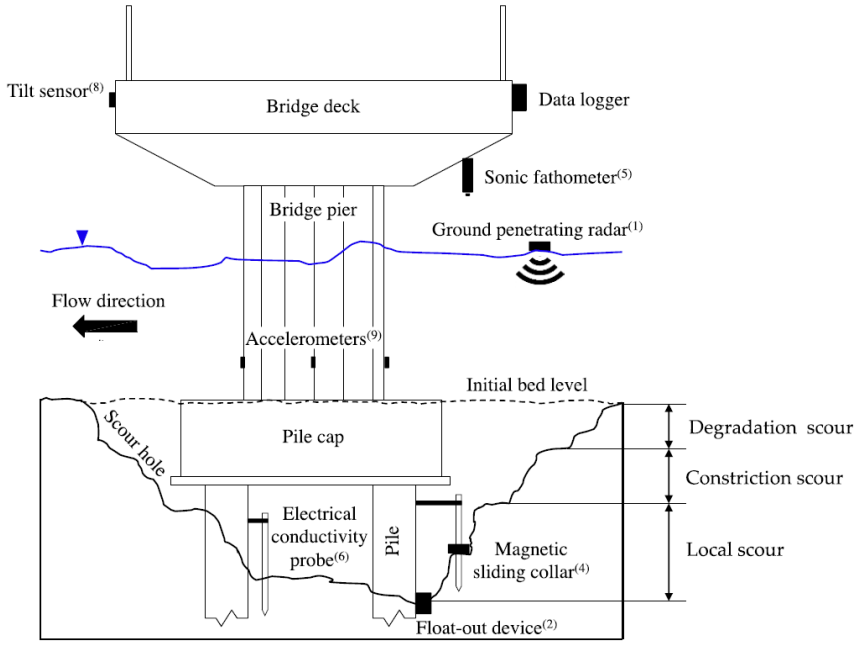


Figure 2.1. Scour monitoring devices. Reproduced from Maroni, A et al. (2020).

2.2 Conventional Monitoring-Based and Machine Learning-Based Methods to Identify Scour

This section provides the details of recent conventional and ML-based scour monitoring research.

2.2.1 Methods, Properties, and Main Outcomes of Studies

2.2.1.1 Conventional monitoring-based approaches to detect scour

Background and Recent Research

This section discusses the recent conventional scour monitoring research. The monitoring techniques employed, numerical approaches utilised, sensors implemented, and dynamic structural properties considered, along with the presence of experimental/field data in the studies, are comprehensively outlined in Table 2.1. This section is divided into two subsections, each of which is dedicated to explaining the details of direct and indirect monitoring studies provided in Table 2.1.

Scour monitoring could be classified under direct and indirect methods (Fitzgerald et al., 2019b). In direct monitoring, structural parts of the bridge are equipped with sensors. In indirect monitoring, the vehicle passing over the structure is instrumented with measurement devices, i.e., the sensors are in

motion. Indirect or so-called drive-by monitoring utilizes the responses of the sensors on the vehicles to infer the condition of the bridge (Fitzgerald et al., 2019b). These responses (measurements) could be accelerations, deflections, or rotations. There are also some challenges associated with direct monitoring, as well as its significant advantages, including ensuring improvement in public safety, early risk detection, and minimising downtime (Chandrasekaran, 2019). There are some disadvantages. Direct monitoring is not suitable for all applications; the entire scheme is costly, and it has some undesired implications, such as high maintenance costs. This was where indirect monitoring became more advantageous, as it was found to be more economical at network level, independent of the structural system's features being assessed, and provided maintenance-friendly measurements. The indirect monitoring technique enables scour identification without interrupting traffic on the railway or highway operations over the bridge. In direct monitoring, signals are obtained from structured form input data, whereas signals from single runs or multiple vehicle batches are products of indirect monitoring. To extract the dynamic properties of the bridge, data were processed with numerical approaches. The distribution of flexural rigidity was calculated retrospectively throughout the length using displacements of a beam found from a VBI FE model (Martinez et al., 2020). They also sought to estimate the level of damage to the bridge. For detecting damage, it is also possible to eliminate the dependence on reference signals from a healthy bridge. Another study focused on monitoring and analysing the difference between right and left accelerations of the train (i.e., roll component) that occurred in the presence of deformation, to demonstrate the imbalance in the bridge behaviour (Bernardini et al., 2020).

The studies listed in Table 2.1 use various sensors. Accelerometers were located at the mid-spans and piers of the bridges. Signals from the superstructure of the bridge were collected using velocity sensors to reveal the dynamic features of scour (Zhang et al., 2019). Their experimental setup consisted of inclinometers, a camera, a level transmitter, and a wireless transmitter. As the flow continued, the velocity sensors detected vibrations, and the inclinometer indicated the angle of tilt during the scour process. Besides accelerometers and velocity sensors, piezoelectric rods, electromagnetic sensors, micro-electro-mechanical systems (MEMS), vibrational power generators, optical sensors (image tracking), and permittivity changes in the soil were utilized for scour detection. When the water flowed, the buried piezoelectric rods were subjected to hydrodynamic effects, resulting in vibrations that created voltage in the rods (Funderburk et al., 2019). By analysing the voltage signal in the frequency domain, the rod's fundamental natural frequency can be identified and then used to back-calculate the exposed length using a cantilever-beam relationship. Because the fundamental frequency is inversely proportional to the square of the exposed length, progressive sediment removal increases the exposed length and is reflected as a

Table 2.1. Properties of devices used in direct monitoring studies (adapted from Tola et al., 2023).

Monitoring Type	Study reference	Numerical Method and Sensor Technology	Presence of Experimental Cases / Field Tests
Direct	Malekjafarian et al. (2020c)	Mode Shape Ratio	None
	Fitzgerald et al. (2019a)	Vibration energy harvesting device	Yes
	Lin and Chang (2017)	Hilbert Huang Transform	Yes
	Liu et al. (2022)	Fiber Optic Sensors	Yes
	Prendergast et al. (2017)	Eigen frequency	None
	Malekjafarian et al. (2020a) Malekjafarian et al. (2018)	Frequency Domain Decomposition	None
	Khan et al. (2021)	Decentralized modal analysis	Yes
	Hachem et al. (2023)	Singular value decomposition	None
	Funderburk et al. (2019) Azhari et al. (2014) Azhari and Loh (2017)	Frequency analysis of piezoelectric rod sensors	Yes
	Farooq et al. (2025)	Photoelectric sensors	Yes
	Crotti and Manzoni (2025)	Fiber Bragg Grating-based Bed Level Seeking System sedimeter with active thermometry	Yes
	Chen et al. (2018)	Unmanned Aerial Vehicle using smart rocks	Yes
	Prendergast and Gavin (2014) Michalis et al. (2015)	Smart probes instrumented with electromagnetic sensors	Yes
	Hashimoto et al. (2020)	Micro energy harvesters	None
	Perugini et al. (2024)	Ultrasonic water level and solar camera velocimetry	Yes
	Elsaid and Seracino (2014)	Horizontally displaced mode shapes and changes in dynamic flexibility	Yes
	Zhang et al. (2019)	Unmanned Aerial Vehicle-based smart rock	Yes
Indirect	O'Brien et al. (2018) Lynch (2007) O'Brien et al. (2023) Zhang et al. (2022) Zhang et al. (2023a)	Wavelet transformation	None
	Prendergast et al. (2016)	Eigen frequency	None
	Cheng et al. (2022)	Mobile vibration frequency analysis	
	Tan et al. (2021)	Closed-form mode shape derivation	Yes
	Feng and Feng (2016)	Mode shape curvature index	Yes
	Fang et al. (2024)	Hilbert–Huang Transform and PSD sensitivity analysis	None
	Li et al. (2020)	Interaction force sensitivity analysis	None
	Wang et al. (2024)	DKF estimation of interaction forces	None
	McGeown et al. (2024)	Analysing the pitch of the crossing vehicle	Yes

reduction in frequency. The local scour depth is then obtained as the difference between the initial exposed length and the instantaneous exposed length estimated from the measured frequency.

Unconstrained distributed fibre optic sensors are a recent technology used for investigating scour. According to the research by Liu et al. (2022), fibres deform freely in the presence of water flow. Deformation is captured using an ultra-weak fibre Bragg grating array inscribed in the optical fibre, which records strain-induced shifts in the grating's central wavelength. Since temperature changes at the pier at an elevation of around 10 meters below water are slow and small, the central wavelength of the fibre is only affected by the altered strains, i.e., deformations of the fibre. Differences in central wavelength changes give scour length. Some researchers have utilized innovative technology, specifically smart rocks, for scour monitoring. (Chen et al., 2018; Zhang et al., 2019). Based on the theory of the magnetic field, the developed algorithms were able to locate the rocks that had been rolled down into scour holes. The proposed system for scour monitoring, as seen in other studies of Prendergast and Gavin (2014), and Michalis et al. (2015), features a probe with an integrated electromagnetic sensor that identifies the different dielectric permittivity values of the soil surrounding the bridge foundation. Using the frequency of the oscillators implemented in the sensors, measuring the capacitance, which only changes with the dielectric property of the soil around them, was proven possible (Marr, 2010). The monitoring system developed by Hashimoto et al. (2020) utilized MEMS harvesters to detect pier scour, which harnessed energy from environmental sources to generate electricity. Further, the authors explain that the electrostatic MEMS harvester produces an electrical output (current/power) that depends on the vibration frequency response of the oscillating electrode, and that the vibration frequency can be detected using a small, coin-sized wireless device.

In structural assessment, a key step is to extract damage-sensitive features from monitoring data through signal processing. There are various processing methods. The Fourier transform is one of the most common techniques in frequency domain analysis. Fourier integrals containing the Fourier transform represent non-periodic (arbitrarily changing) excitations (Chopra, 2007). The direct Fourier transform equals the multiplication of the Fourier transform and the complex frequency response function. To obtain a response in the time domain, the inverse Fourier transform is applied. The Fourier transform is most effectively applied to stationary and transient signals, as it yields the average frequency (Cerna & Harvey, 2000). Like the Fourier transform, the Hilbert spectrum represents amplitude in the time–frequency domain. However, it differs in that it can capture and analyse the characteristics of non-stationary signals (Azhari & Loh, 2017). Feldman (2011) highlights one of the Hilbert transform's most significant features: its ability to detect nonlinear frequency response effects. Since applying the Hilbert transform to the

frequency response of a linear structure regenerates the original frequency response, any distortion could serve as an indicator of non-linearity. Whereas decentralized modal analysis infers modal content locally from sensor-specific segments of the response (Gao (2005), Nagayama et al. (2009), Sim et al. (2010)), eigenvalue analysis determines the global modes by solving the system's mass–stiffness eigenproblem, yielding mode shapes, amplitudes, and natural frequencies. Since Eigenvalue analysis is computationally demanding in systems with a large number of degrees of freedom, more efficient algorithms have been developed over time (Chopra, 2007). Solutions for the eigenvalue problem were categorized under vector iteration, transformation, and polynomial iteration methods.

The Frequency Domain Decomposition method identifies very close and repeated modes (Brincker et al., 2001). Khan et al. (2021) computed the mode shape amplitudes by applying the Frequency Domain Decomposition method to on-bridge acceleration signals. Malekjafarian et al. (2020a) stated that Frequency Domain Decomposition can extrapolate dynamic parameters without any initial information about excitation. After calculating the continuous wavelet transform matrix from the raw data, the singular values and singular vectors are obtained through the singular value decomposition process, which is used to determine damping ratios, frequencies, and mode shapes (Le & Paultre, 2013). Wavelet transformations are defined as visual tools for assessing signals more easily, and the computation of these transformations is a prerequisite when a signal needs to be modified selectively or its pattern needs to be identified (Grossmann et al., 1990). A wavelet function has zero mean and changes with a dimensionless time parameter. It is located in both the time and frequency domains, while the convolution of a discrete sequence with the translated and scaled status of the wavelet transformation was defined as a continuous wavelet transformation (Torrence & Compo, 1998). The selection of the wavelet transformation is important. Compared to the Short-Time Fourier Transform (STFT), the Continuous Wavelet Transform (CWT) provides a more adaptable time-frequency analysis for non-stationary signals. STFT uses a fixed-size time window, which limits its resolution uniformly across frequencies. In contrast, CWT varies its window size with frequency, thereby capturing transient features across different scales in structural signals (Rioul & Vetterli, 1991). The sensitivity of scour severity to the changes in wavelet energy extracted from vehicle-mounted, i.e., drive-by acceleration signals using CWT was demonstrated by Zhang et al. (2020). While the approach demonstrated good scour localisation performance, varying travel speeds and multiple scour scenarios were not considered. The DWT, developed for discrete-time signals, offers significant computational and implementational efficiency advantages over CWT, via non-redundant data analysis (Chui, 1997; Zhang et al., 2025). Signal values are computed using a decomposition algorithm that applies paired low-pass and high-pass analysis filters, allowing the original signal to be analysed and then

reconstructed from its wavelet coefficients (Zhang et al., 2025). To detect structural damage in acceleration data from both a simulated model and a real concrete building, Hou et al. (2000) used the Daubechies DB4 wavelet. DWT coefficients were shown to accurately capture the temporal and spatial dimensions of instant or accumulated types of damage. Lucero and Taha (2005) combined ANNs and Wavelet Multi-Resolution Analysis (WMRA) to detect structural damage to a prestressed concrete bridge. The basic concept of WMRA is to utilise a range of filters to implement the DWT. Wavelet norm indexes, calculated from WMRA coefficients derived from the DWT decomposition of accelerations, were used to quantify the difference between healthy and potentially damaged states; however, verification by real-world measurements is necessary.

All the studies listed in Table 2.1 have numerical models except the one by Azhari et al. (2014). The studies using acquired bridge accelerations to identify scour employed different frequency analysis methods to interpret acceleration signals, thereby obtaining mode shapes or frequencies, and ultimately establishing a scour index to identify scoured cases. In the research by Malekjafarian et al. (2020c), acceleration signals from two points on the bridge were used to gather the amplitudes of mode shapes. The mode shape ratio was the ratio of these amplitudes and served as an indicator of scour. Energy harvesting devices, as used in the study by Fitzgerald et al. (2019a), produce electricity from the strain changes in piezoelectric materials. These were used to identify scour-based frequency shifts of the bridge. Prendergast et al. (2016, 2017) investigated the influence of scour on eigen frequencies of the bridge. To obtain mode shapes from acceleration signals of the structure, Malekjafarian et al. (2020a) and Malekjafarian et al. (2018) used the Frequency Domain Decomposition method. Without information on the input motion, it was possible to evaluate the dynamic parameters. Decentralized modal analysis enabled Khan et al. (2021) to derive the modal amplitudes of sensors at different locations. To detect bridge pile scour, Elsaid and Seracino (2014) utilized the change in the shape of horizontally displaced modes and flexibility deformations of a frame-type bridge. Continuous wavelet transformation was utilized to process acceleration signals in the research of O'Brien et al. (2018, 2021), and Fitzgerald et al. (2019b). By comparing responses to the features of a base wavelet, this method can detect transient changes in frequency signals (Teolis & Benedetto, 1998). To process acceleration signals, Tan et al. (2021) proposed a closed-form model derivation. Based on instantaneous amplitudes calculated for a target bridge frequency, mode-shape squares were obtained, and a damage index was computed.

Direct Monitoring-Based Research

This section includes the main properties and outcomes of only the direct monitoring-based studies listed in Table 2.1. In Table 2.2, devices, sensing mechanisms, methods of sensor signal processing, and the target property of the direct monitoring studies are listed. Additionally, scour validation tests and experiments on real bridges are tabulated here. The assessments, outcomes of the studies, performance, and efficiencies are provided at the end of this section.

The research listed in Table 2.2 mainly focused on the monitoring device used. Azhari et al. (2014), Azhari and Loh (2017), Funderburk et al. (2019), Lin and Chang (2017), Liu et al. (2022), Perugini et al. (2024), Farooq et al. (2025), and Crotti and Manzoni (2025) aimed to measure or estimate scour depth with the sensors they used. Michalis et al. (2015) monitored sediment deposition processes. On the other hand, Liu et al. (2022) investigated the rigid body motions of the piles and piers using velocity sensors. Chen et al. (2018) and Zhang et al. (2019) tried to locate and track the movement of smart rocks they deployed along the scour hole. In both studies, the smart rocks were concrete units containing a permanent magnet, so that their movement with sediment erosion could be inferred indirectly. After deployment, each rock's position became unknown as it could roll or settle into the developing scour hole. A Unmanned Aerial Vehicle equipped with a magnetometer was then used to survey the magnetic-field distribution around the pier (the ambient geomagnetic field superimposed with the rock-induced field). An inverse localisation algorithm was applied to estimate the rock coordinates from the measured field pattern, enabling tracking of rock movement over time. Liu et al. (2022) arranged the fibre Bragg grating sensor in a spiral around the pier to shorten the spacing between sensing points and thus achieve higher spatial resolution than a straight layout. The spatial resolution can be adjusted by changing the spacing between adjacent rings, and the monitoring range along the pier can be extended by increasing the number of fibre rings. The fibre is fixed to the pier at two points on each ring and otherwise left free to swing under flowing water. The resulting deformation is captured as shifts in the central wavelength of the fibre Bragg grating sensor. Perugini et al. (2024) conducted two pilot case studies deploying a monitoring strategy focused on hydraulic parameters to estimate scour depth. Farooq et al. (2025) evaluated photoelectric sensors for direct wireless monitoring of scour in laboratory flume experiments and in a field test, prototype installed in a small creek. As the sediment level dropped due to scour, the sensors switched from a "buried" (low voltage) to an "exposed" (high voltage, water contact) state, enabling mapping of the temporal and spatial evolution of scour and sediment refilling events in both laboratory and field deployments. Crotti and Manzoni (2025) developed and assessed a fiber bragg grating-based bed level seeking system

sedimeter, which measures riverbed level by exploiting differences in heat dissipation between saturated soil and flowing water.

Besides methods that directly monitor scour depth, there are also direct monitoring studies employing vibration-based scour detection using signals acquired from sensors installed on the bridge superstructure. These approaches identify scour by analysing changes in the bridge's dynamic structural response, comparing behaviour between healthy conditions and scour-affected states. To simulate these signals, Fitzgerald et al. (2019a), Malekjafarian et al. (2020a), Malekjafarian et al. (2018), Prendergast et al. (2012, 2017), Hashimoto et al. (2020), Elsaid and Seracino (2014), and Hachem et al. (2023) represented the bridge superstructure (deck and beams) with Euler-Bernoulli beams. Springs were assumed for the piers and the foundations beneath them. Different spring stiffnesses corresponded to different stages of scour. In their studies, Khan et al. (2021) and Malekjafarian et al. (2020c) also included vehicles in their FE models, i.e., VBI models. Vehicles were modelled as rigid masses and were connected with springs and dampers. Prendergast et al. (2012), Fitzgerald et al. (2019a), Elsaid and Seracino (2014), and Khan et al. (2021) also verified their numerical approaches with experiments.

The scour indicators in the research of Fitzgerald et al. (2019a), Malekjafarian et al. (2020c), and Hashimoto et al. (2020) were the frequency shifts of the bridge structure. While Le and Paultre (2013) used differences in the first mode shape amplitudes at pier locations, Malekjafarian et al. (2018) and Elsaid and Seracino (2014) chose mode shape changes. Elsaid and Seracino (2014) specifically focused on the horizontally displaced mode shapes of a frame-type bridge. Khan et al. (2021) set fitting curves to the mode shapes and used root mean square (RMS) differences between healthy and damaged mode shapes to indicate scour. Hachem et al. (2023) identified scour through variations in natural vibration frequencies using a singular value decomposition-based inverse algorithm. The horizontal in-plane and out-of-plane modes of the piers were sensitive to scour-induced foundation stiffness loss, whereas the vertical bending modes of the deck remained insensitive. Funderburk et al. (2019), Azhari et al. (2014), and Azhari and Loh (2017) detected scour through changes in the fundamental frequencies of the polymer rods. Michalis et al. (2015) utilized the increasing permittivity of the soil to identify scour, as the sensors' permittivity suddenly increased as scour developed. Higher apparent permittivity was related to the reduced soil density of the re-deposited sediment around the sensor.

The outcomes of the direct monitoring studies were summarised here. Fitzgerald et al. (2019a) numerically proved that the frequencies decreased due to a reduction in foundation stiffness in two piers (mimicking 24.5% and 44.9% scour cases). In their experimental setup, cantilever-based piezoelectric

energy harvesting devices were utilized, and three different scour scenarios were employed. One device attached to the pier successfully detected the frequency shift at that pier. When the device was attached to a healthy pier, it could detect the scour of another damaged pier. To improve the method's performance, they suggested an initial modal analysis of the bridge using accelerometers, as this would enable monitoring of the obtained modal frequencies in the frequency domain of the device's voltage. Malekjafarian et al. (2018) the normalized mode shape of a two-span integral-type concrete bridge changed by approximately 50%, with a 20% frequency difference for a scour depth of 5 m. They noted that their approach might alleviate the known frequency-associated problems because the method utilizes the same sensors used for gathering the frequency. According to Prendergast et al. (2017), abutments and piers exhibited local vibration modes, and the corresponding frequencies were sensitive only to scour related to the adjacent element. In their experiment, sensors were represented by virtual node points and located on two abutments and one pier. The method successfully identified the local frequencies and the location of the scour. However, the need for experimental validation was indicated.

Malekjafarian et al. (2020c) generated research for a two-span integral-type bridge. For 5 m of scour depth, due to central pier scour, the mode shape ratios of two points on the pier and deck changed by around 50% over the centre of the foundation. The study's main limitations were its sensitivity to vehicle-bridge interaction responses and random errors in ambient loading. Applying the developed approach to abutments by using the first mode was not possible. The approach was extended to a three-span bridge; however, a steady trend could not be established. Prendergast et al. (2017) noted that as scour severity increased, the mean-normalized mode shape values at piers decreased. For example, pier 1 changed from 0.62 to 0.35 when 30% scour was simulated at pier 3. The experimental setup of the study included accelerometers located at two piers and the midspan of the bridge model, resulting in scour depths of 24.5% and 44.9%. Compared to the healthy case, the mean-normalized mode shape increased at the scoured pier, while it decreased at the other piers. Their approach did not need knowledge of vehicle excitation forces, material, and geometrical information of the bridge. However, when multiple piers experienced scour simultaneously, the approach was not successful.

Maroni et al. (2020) showed that the permittivity and porosity of the soil increased when scour was present. The sensors were installed along four-meter-long scour probes in the experimental setup. One probe measured the total scour at a single pier, while the other measured constriction scour at the midpoint of the channel. A 30 cm scour length was measured by the probe in the middle of the channel after a peak flood. When the empirical scour formulas and the experimental scour depths were compared, an overestimation of the formulas was revealed. If the scour hole was not uniform or located at a particular

Table 2.2. Properties of devices used in direct monitoring studies (adapted from Tola et al., 2023).

Study Reference	Device type	Sensing mechanism	Signal processing method	The target property of signal processing	Scour validation tests	Laboratory or field tests	Target property
Liu et al. (2022)	Unconstrained distributed fibre optic sensors	Ultra weak fibre Bragg grating	Empirical formula	Central wavelengths	Detecting signals from fibres in sand and in water	The standard deviation value is higher than zero for several minutes	Scour depth and location
Lin and Chang (2017)	Velocity sensors, inclinometer, wireless transmitter, and camera	2 Velocity sensors	Hilbert transform and empirical mode decomposition.	Individual instant frequencies	Single-pier laboratory scour test	Caisson-type and pile-group foundation scour tests	Rigid body motion
Funderburk et al. (2019)	Rod sensor	Piezoelectric Polymer Film	Wavelet packet transform and Hilbert transform	Instant the natural frequency of the rod	Flume test	Test with different pier cross-sections	Scour depth
Azhari et al. (2014)			Fast Fourier transform		Clamped to a laboratory bench planted in sand	None	
Azhari and Loh (2017)					Flume test	Tested on one pier	
Chen et al. (2018)	1 Direction-Unknown and 1 Direction known smart rocks	Ambient magnetic field	Theory of magnetic field	Distribution of the magnetic field induced by smart rocks	Field validation tests	Tests on the upstream side of a pier	Localise or track the movement of the smart rock
Michalis et al. (2015)	Electromagnetic sensors	Changes in the dielectric permittivity of the soil	The reflection feature of electromagnetic waves	The porosity of the soil	'Static' scour simulations Real-time open channel flume tests	Not provided	Scour depth variation
Zhang et al. (2019)	Unmanned Aerial Vehicle-based innovative rock positioning system	3-axis magnetometer and global positioning system on an Unmanned Aerial Vehicle	Algorithm to locate smart rocks using measured magnetic intensities	Magnetometer measuring magnetic fields pre- and post-smart	Not provided	I-44W Roubidoux Creek Bridge Pier	Depth of scour, i.e., vertical movement of the rock

				rock deployment has been deployed			
Farooq et al. (2025)	Photoelectric sensor array (through-beam and diffusive-reflective)	Optical light transmission/ reflectance	Binary/analogue voltage change at each sensor location	Sensor voltage states (buried/exposed)	Exposure/burial of sensors at different depth levels around the pier (lab flume and field)	Lab flume: simulated pier in sediment under variable flow. Field: creek bed with seasonal flow	Scour depth
Perugini et al. (2024)	Low-cost hydraulic sensors and solar cameras	Water level: ultrasonic/ pressure. Velocity: image velocimetry (Particle Image Velocimetry from video)	Processed hydraulic data using empirical and advanced scour formulas	Surface flow velocity, water level, and derived hydraulic metrics	Compared low-cost sensors to radar and probes Field data analysed for consistency	Field: two bridges in Scotland, continuous monitoring with hydraulic sensors, and validation by radar	Estimated scour depth (Indirect, from hydraulic variables)
Crotti and Manzoni (2025)	Fiber Bragg Gratings-based sensor system (Bed Level Seeking System device)	Fiber optic, active thermometry using heat dissipation differences	Step-response (exponential) analysis, clustering algorithms	Temperature response curves, measured at multiple fibre points	Compared to the echo sounder (reference device)	Field: long-term installation on a real bridge pier in Po River, Italy	Riverbed level (used to indicate scour depth)

point, they suggested that sensor response and the effects of suspended sediments on the values of dielectric permittivity should be examined in a laboratory. In the research of Hashimoto et al. (2020), a 30% reduction in the values of the first three natural frequencies was observed compared to the scoured case. The renewable energy sources monitored by the devices (wind, heat, electromagnetic field, and light) were a key aspect of this study. However, the approach was only validated for particular structural types.

Elsaid and Seracino (2014) demonstrated that vertical mode shapes were not sensitive to scour, and increasing pile heights resulted in a decrease in flexural stiffness and horizontally displaced mode frequencies for four scour cases compared to the healthy case. The experimental model consisted of 11 uniformly distributed, unidirectional Integrated Electronics Piezoelectric accelerometers. The authors concluded that changes in mode-shape curvature and flexibility-based deflections are suitable indicators for scour detection. They recommend these two measures because, in the laboratory tests, they remained more stable and interpretable when the experimentally extracted mode shapes contained unevenness. However, the method was not reliable enough to locate the scour in practice. Khan et al. (2021) found that scour decreased the stiffness of the foundation at the central and right-side piers by 25% and 45%, respectively, and reduced modal amplitudes both locally and globally. The average change in the central pier was 7.2% and 16.8%, respectively. They used seven re-deployable accelerometers: four at the mid-span and three at the piers in their laboratory test setup. They found reductions in the mode amplitudes at pier locations and an increase in the amplitudes of non-scoured piers. As foundation stiffness decreased, the difference in the root mean square between healthy and scoured modes increased linearly. Their study did not require previous information about the structural behaviour and used only two sensors that were sequentially moved along a bridge to detect mode shapes. However, the study was limited to one-dimensional models and simplified experiments.

Using FE modal analysis combined with a singular value decomposition (SVD)-based inverse algorithm, Hachem et al. (2023) estimated the location and severity of scour by analysing shifts in natural frequencies between healthy and scoured structural states. The authors reported that relative changes in frequencies serve as effective indicators of scour, enabling accurate identification of foundation stiffness degradation. Their numerical study validated this non-invasive vibration-based approach as an effective tool for early scour detection and real-time monitoring before failure. Liu et al. (2022) successfully located and measured scour by adopting the ultra-weak fiber Bragg grating technique that was able to sense changes in the wavelength of 0.05–0.06 nm. This change is related to the strain changes of the fibres, as determined using an empirical formula. The effects of water buoyancy and loosening of the sand on

the signals were easily excluded. A value of 0.002 for the standard deviation indicated scour, and the accuracy was ± 2.5 cm. The instability index developed by Lin and Chang (2017) enables the rapid evaluation of bridge safety conditions. A caisson-type experiment with 10 sensors was performed. These sensors were located at the centre of every deck panel and the cap beam of a pier, as well as 14 velocity sensors for the pile group. Pier 1 was always in a state of instability. Before the pile started to incline too much, the instability index changed significantly relative to its original state. Their study could serve as an early warning system, but it was limited to the specific types of foundations tested, namely caissons.

Funderburk et al. (2019) and Azhari et al. (2014) calculated the scour height, i.e., the exposed length of the rod, via a time history analysis in the frequency domain, induced by the hydrodynamic effect. Sensors were tested in the sand surrounding each pier in a laboratory. They observed that as the exposed length increased, the natural frequencies diminished. The measurements could last throughout the scour duration, and the sensors were cost-effective. To improve the accuracy, Funderburk et al. (2019) emphasized the need for full-scale tests, and Azhari et al. (2014) pointed out the requirements for more realistic simulations in a flume. Later research by Azhari and Seracino (2017) compared measured and observed scour depths, showing that the sensor results were accurate. At the same time, they revealed difficulties in installing sensors in real-life applications, suggesting that the debris effect could pose a challenge for the proposed system. Magnetic field-based algorithms were developed by Chen et al. (2018) and Zhang et al. (2019) for locating and tracking smart rocks moving around scour holes. Chen et al. (2018) tested two smart-rock designs at a bridge pier: a direction-unknown rock, where the embedded magnet can adopt any orientation after deployment, and a direction-known rock, where the magnet is intended to align with the projected direction of the ambient magnetic field in the horizontal plane to simplify localisation. By repeatedly estimating the rock coordinates, they tracked its vertical movement and, in turn, the development of peak scour depth. They report improved localisation by explicitly accounting for the ambient magnetic field and by performing field calibration/validation. However, the direction-known rock showed larger errors in the field because its alignment was unstable when the ambient field varied due to environmental effects. They also observed that nearby steel components distorted the local geomagnetic field and recommended selecting measurement points close to the pier where the field is stronger to reduce error.

In the study by Zhang et al. (2019), real-time monitoring was employed to achieve more accurate results. Before implementing the smart rock, the magnetic field and coordinates were measured by an Unmanned Aerial Vehicle. The Unmanned Aerial Vehicle could detect the location with an error of less than 36 cm, as verified by total station measurements. In the study by Michalis et al. (2015), different

permittivity values were observed for pre- and post-scour conditions. To test the application, six instrumented probes were installed in the bed segment and fixed on the flume floor. As scour developed, the sensors' permittivity increased. The higher apparent permittivity is believed to be related to the reduced soil density of the re-deposited sediment around the sensor. It was an economical, realistic, and real-time monitoring method, but it was not efficient in saline water. Perugini et al. (2024) employed a combination of low-cost sensors, including solar cameras, to capture river surface velocity through image velocimetry, as well as river level sensors and traditional radar for validation. Their approach aimed to estimate scour depth indirectly by applying advanced formulas to hydraulic data. The results showed that low-cost sensors provided accuracy comparable to that of expensive radar systems, with easier installation and lower costs, demonstrating potential for scalable bridge scour risk assessment.

Farooq et al. (2025) tested two types of photoelectric sensors —through-beam and diffuse-reflective— in sediment-water environments at multiple depths on a simulated pier. Both sensors effectively detected scour progression and sediment refill by identifying transitions between buried and exposed states. The field study confirmed sensor robustness under variable environmental conditions, highlighting the technology's low power needs, low cost, and strong potential for remote scour monitoring. The bed level seeking system, developed by Crotti and Manzoni (2025), utilizes temperature differences between saturated soil and flowing water, which are detected through fiber-optic sensors heated intermittently. Intermittent heating applies a controlled thermal step so that differences in heat dissipation (mainly convection in flowing water versus conduction in saturated soil) produce distinct temperature-rise responses of the sensors. Two algorithms were proposed for riverbed detection: one based on physical step-response parameters and another using data clustering. Field data collected over a two-year period showed strong agreement with traditional echo sounder measurements. The study demonstrated the system's reliability in harsh environments and its readiness for engineering applications.

Overall, direct monitoring-based scour detection studies have made substantial progress in developing a diverse range of sensor technologies, including smart rocks, piezoelectric rods, optical fibers, cameras, and electromagnetic sensors, all aimed at providing direct physical measurements of scour depth or related properties. These sensor systems generally offer advantages over traditional empirical scour formulas by delivering more accurate and cost-effective assessments. However, practical challenges such as sensor deployment difficulties after flooding, maintenance demands, exposure to harsh environmental conditions, and occasional measurement inaccuracies limit their widespread use in the field. Complementing these, vibration-based monitoring approaches analyse changes in the dynamic structural behaviour of bridges to identify scour through frequency shifts and mode shape variations, presenting promising

non-invasive alternatives for early detection of scour. While numerical and laboratory validations show encouraging results, broad and rigorous field validation remains necessary to confirm their reliability and operational readiness. Overall, the combined body of research indicates advancing capabilities in scour monitoring approaches but also underscores the critical need for enhanced sensor robustness, practical deployment strategies, and extensive real-world testing before these methods can be fully adopted for routine infrastructure safety management.

Indirect Monitoring-Based Research

This section includes only indirect-monitoring-based studies listed in Table 2.1. In the research where the scour detection methodology was developed based on vehicle/bridge dynamic interaction models, vehicles were generally modelled as half or quarter cars, representing one or two axles of a vehicle. The quarter car was particularly unrealistic, but its simplicity had value in that it kept the focus on the bridge structure and did not require assumptions on axle spacing and other vehicle properties. It should be noted that the half-car was also simplistic, but it did serve to introduce some inter-axle effects that have a significant influence.

The studies addressed in this section processed the acceleration signals resulting from multiple train passes. O'Brien et al. (2018) employed a dynamic model that accounted for vehicle-bridge interactions, a quarter-car model of the train, and two simply supported bridge spans with a shared pier. A zero-mean base wavelet, a "Mexican Hat", was chosen and a set of wavelets was derived from this base by modifying it with position and scale parameters. Coefficients are the results of a continuous wavelet transformation, and they relate the result to a specific region in the signal—they indicate the presence of specific frequencies that are present in the signal at a given time point. Scour was detected by comparing the wavelet coefficients of the healthy and scoured cases. The dynamic model of Tan et al. (2021) consisted of a simple vehicle and a simply supported Euler-Bernoulli beam representing the bridge. The acceleration of the vehicle was obtained using equations of motion, and a Hilbert transform was applied to the accelerations to find bridge mode shapes that were already known to be sensitive to the damping of the structure. Tan et al. (2021) investigated the effect of damping on the mode shape and degree and rearranged the formula that calculates mode shape squares using the amplitude of the Hilbert transform. The difference between the scoured and healthy case mode shapes was the scour indicator. The researchers also verified their algorithm numerically with two laboratory experiments.

Fitzgerald (2019b) and O'Brien et al. (2021) also used wavelet coefficients. The dynamic model of Fitzgerald (2019b) first analysed the vehicle and then computed the local and global mass and stiffness

matrices of the beam using the finite element method. The forces were distributed according to the degrees of freedom using shape functions, which considered the vehicle's location that changed with every time step. The chosen base was the Morlet wavelet. The scoured case was simulated in the model by reducing the stiffness of the bridge pier. For each of the scour cases, 200 vehicle runs were generated in the dynamic model. Continuous wavelet transformation was applied to accelerations of both healthy and scoured cases, and the differences of the coefficients indicated scour. Prendergast et al. (2016) generated a vehicle–bridge–soil interaction model. The horizontal accelerations due to passing vehicles were recorded from the top of the pier. They extracted mode shapes from acceleration measurements through an eigenvalue analysis of a highway bridge. Three different levels of soil stiffness were considered to represent dense, medium-dense, and loose sand. For all types of soil, lateral frequencies were calculated for both zero and 10m scour depths.

The results of the study by Fitzgerald et al. (2019b) showed that bogie accelerations decreased due to a 30% scour case. Identifying a scoured pier was possible by using the difference between the wavelet coefficients of the scoured case and the healthy case. Vehicle properties such as mass and speed were considered stable. Fitzgerald (2019b) detected and located scour by showing the difference between the wavelet coefficients of the scoured case and the healthy case. An external party generated vehicle signals numerically. Batches of 20, 50, 100, and 200 vehicle crossings were tested to understand the influence of the number of vehicles. For batches with a low number of vehicles, false-positive scour indications were seen to increase. O'Brien et al. (2021) measured instantaneous accelerations at every support, and the mean value was obtained and converted into the frequency–spatial domain using 1000 vehicle passages. Operating deflection shapes were evaluated for scour cases of 25% and 45%. Tan et al. (2021) extracted mode shapes from drive-by data, using the Hilbert transform. In their experimental setup, one of eight accelerometers was mounted on the vehicle, and the remaining seven were fixed to the bridge to gather modal parameters. Foundation scour could be identified only for the fifth bridge mode shape. They noted that high vehicle speed decreased the accuracy of the mode shape. Near the damage location, the damage index reached a maximum value. Nevertheless, an edge effect limited the ability to find the exact location with this approach. The effects of operational and environmental parameters were not considered. Finally, taking the arithmetic average of multiple runs was suggested. Prendergast et al. (2016) investigated the response of lateral acceleration at the top level of a pier, and a 40% frequency change was obtained due to scour. The results were only validated for the two-axle truck case.

Fang et al. (2024) analysed drive-by accelerations obtained from a single sensor on a test vehicle using the Hilbert–Huang Transform and successfully located damage at single and multiple bridge

locations. The method was further enhanced by applying power spectral density (PSD) sensitivity analysis: after the damage locations were identified via time-frequency features in the acceleration signals, the PSD sensitivity method was used to quantify the damage degree at those locations by iteratively comparing the response power spectra before and after simulated damage, enabling precise severity assessment. Nevertheless, the study was subject to certain limitations, including vehicle mass, to prevent modal confusion and speed limitations, and to ensure that the signal's sensitivity to damage was not diminished. The research was not verified by experimental studies. Li et al. (2020) estimated the state of the dynamic system using a Dual Kalman filter, along with two sets of laboratory-produced drive-by signals acquired from wireless sensors installed on the vehicle. The interaction forces between the vehicle and the bridge were obtained with the Dual Kalman filter, and an interaction force-based sensitivity analysis was conducted to reveal the damage. However, for this damage-detection approach to be effective, the noise level should not exceed 5%. In addition to damage detection, to overcome pseudo-frequency and model aliasing problems, Zhang et al. (2023a) introduced a characteristic wavelet coefficient by applying a continuous wavelet transform and complete ensemble empirical mode decomposition with adaptive noise to contact-point accelerations. The study detected 20% girder damage in the presence of white noise with a signal-to-noise ratio of 20; however, this finding had not been experimentally verified. Further, the study lacked field measurements.

Feng and Feng (2016) analysed drive-by displacement signals to compute their power spectral density functions. The first-order mode shape was obtained through the power spectral density functions. The first-order mode component governed the displacement response of a bridge to a moving vehicle. Since the first mode shape was shown to be insensitive to the damage location, mode shape curvatures were introduced to enhance the sensitivity. A damage index was defined as the difference between the amplitudes of healthy and damaged bridge mode shape curvature peak(s). The study had been numerically and laboratory-validated but did not include real-life applications. The numerical and experimental study by Zhang et al. (2022) identified and located scour through observing an increase in wavelet energy concurrent with scour progression. The numerical method combined time-frequency analysis of acceleration data with wavelet transform to capture scour-induced vibration changes of the bridge structure during train passage. Experimental validation was performed using controlled laboratory tests with scaled models to measure acceleration signals, confirming the method's ability to detect scour severity. The researchers acknowledge several limitations, including the absence of field tests, the approach's sensitivity to varying environmental conditions, and its dependence on the number of train batches analysed for

robustness. The study provides a valuable framework for scour monitoring but encourages further validation through large-scale field testing.

Cheng et al. (2022) conducted an analysis revealing differences in bridge frequencies before and after flooding in the vertical direction of the mid-span of the deck and in the flow direction of seven bridges, using a car-based vibration testing system. The method involved automated mobile vibration measurement and sophisticated signal analysis techniques to evaluate structural changes indicative of scour. The study identified frequency shifts correlating with scour damage, providing a non-invasive monitoring approach. Limitations noted include the number of measurement points, which could affect spatial resolution; sensitivity to road surface roughness, which may introduce noise. The influence of vehicle travel speed on data quality; and challenges posed by bridge contraction joints that affect vibration signals. Field validations were performed across multiple real bridges, enhancing the practical applicability of the findings. However, the study acknowledged that extended monitoring over longer periods and diverse environmental conditions could further strengthen the reliability of the results. Wang et al. (2024) developed a drive-by inspection methodology for rapid assessment of high-speed railway bridge substructures using bogie accelerometers on passing marshaling trains. In this research, the "marshaling train" refers to a comprehensive inspection train used in China, modelled as a consist of eight coupled vehicles. The seventh vehicle is selected as the instrumented test vehicle for the bogie-acceleration measurements. The technique estimated wheelset–bogie interaction forces using a dual Kalman filter and defined a damage indicator as the work associated with the change in interaction force. The work is computed as the integral of the healthy–damaged force difference over the pier region. The method demonstrated robustness to variations in pier damage scenarios, vehicle parameters, and track irregularities, and was capable of reliably localizing and quantifying damage at high train speeds. The root mean square of vertical bogie acceleration served as a time-domain statistical metric, offering a computationally efficient alternative to traditional frequency-based analyses, such as modal parameter extraction or wavelet decomposition. While promising for network-wide monitoring, the authors emphasized the need for additional validation under realistic scour and operational environments. McGeown et al. (2024) introduced a drive-by concept that utilizes the pitch, i.e., body rotation, of a crossing vehicle as a damage-sensitive indicator for scour. A 2D VBI model (three 10-DOF carriages over a four-span bridge) explored scour as a loss of pier stiffness, showing clear pitch changes on spans adjacent to the affected pier. The concept was then validated on a scaled 5.2 m four-span laboratory bridge using a tractor–trailer outfitted with wireless triaxial accelerometers (200 Hz). Longitudinal acceleration was low-pass filtered and converted to rotation to isolate quasi-static pitch. Laboratory results again showed consistent separation between healthy and scoured states,

with robustness to a ±5% variation in vehicle mass. The approach requires no bridge-mounted or riverbed sensors and no ML classifier, relying instead on feature-level divergence against a healthy baseline. Reported limitations and directions include developing 3D models, incorporating track-profile effects and higher speeds, addressing environmental/operational variability through baseline grouping, and conducting field trials with instrumented trains. The authors also highlight the potential for using vehicle roll to sense lateral pier movement and for establishing regression mappings from pitch features to scour characterisation.

Overall, the body of work demonstrated that indirect, drive-by approaches could detect scour by processing batches of vehicle-induced responses, using time–frequency methods (e.g., continuous wavelet transforms and Hilbert–Huang analysis), mode-shape/operating-deflection-shape extraction, frequency-shift tracking, and force- or spectrum-based indicators (including PSD-sensitivity metrics). Because continuous wavelet transforms yielded information in both spatial and frequency domains, localisation and richer modal interpretation were achievable, and operating-deflection-shape-type procedures generally proved more reliable for pinpointing scour than conventional modal analyses. However, small batch sizes increased the likelihood of false positives. Spatial resolution, edge effects, road/track roughness, vehicle speed and mass, and other operational/environmental variabilities influenced accuracy. Additionally, many studies still required broader field validation. Compared with direct monitoring, these methods typically used fewer sensors, had lower power demands, avoided underwater inspections, and operated under regular service without service interruptions—features that made them cost-effective. Taken together, the evidence suggests that indirect monitoring is a highly promising strategy for bridge-scour investigations. Future work should expand field campaigns, test a broader range of vehicles and operating conditions (including realistic scour scenarios), and undertake more detailed, noise-robust numerical and experimental analyses.

2.2.1.2 Machine learning-based research

Background and Recent Research

Scour monitoring encounters challenges of incomplete signals, environmental noise, and evolving operational conditions. ML offers a potential solution for handling missing features from faulty sensors (Polikar et al., 2010), managing high-dimensional data (Zhao et al., 2008), and capturing nonlinear relationships between observed signals and scour severity. This section first outlines feature-extraction and dimensionality-reduction methods, parametric system models (such as AR and ARX models), feature selection methods, optimisation strategies, and the principal algorithms frequently used in ML-based scour

prediction and damage detection. The section then reviews recent research on bridge scour that employs ML algorithms under two subsections: the first addresses studies predicting scour depth, and the second focuses on scour detection and localisation using monitoring signals. For completeness, adjacent categories that do not directly detect or locate scour are excluded from the review—namely, risk classification from inspection-report attributes (e.g., Wang et al. (2023)), flood detection with pier-mounted accelerometers (e.g., Deng et al. (2024)), and settlement estimation from onboard train accelerations using simulation-trained regressors (e.g., Abdu et al. (2023)).

Feature Extraction, Dimensionality Reduction, Parametric System Models. Beyond time–frequency or time–space analysis, scour-sensitive feature extraction requires recognizing response patterns, capturing variable relationships, and modelling system dynamics. PCA supports this by transforming correlated variables into a reduced set of uncorrelated components that preserve the dominant variance, thereby highlighting underlying patterns (Jolliffe & Cadima, 2016). PCA does not assume a specific distribution and enables effective dimensionality reduction with minimal loss of critical information (Jolliffe & Cadima, 2016). Its limitation lies in only capturing linear relationships. Kernel Principal Component Analysis overcomes this by implicitly mapping the data into a higher-dimensional feature space, where non-linear structures become linearly separable, before reducing dimensionality (Precup & Coates, 2016; Murray, 2016). For example, Santos et al. (2016) employed Kernel Principal Component Analysis to map autoregressive (AR)-derived features from acceleration data into a higher-dimensional space, thereby enhancing sensitivity to structural damage. Extracted damage-sensitive features are then used to train ML algorithms that classify damage. AR and Autoregressive model with eXogenous inputs (ARX) models capture temporal dynamics by linking current measurements to past values and inputs. To find the coefficients of the AR model, various algorithms can be employed, such as the Least-Squares Method, Yule-Walker, and PCA (Santos et al., 2016). Model coefficients could also be used as an intermediate step to estimate more complex models.

The ARX model offers the most efficient parameter estimation among polynomial approaches, as it involves directly solving linear regression equations (Dahari, 2009). ARX model estimations are unique in that this method consistently achieves the global minimum of the loss function. Anastasia et al. (2023) employed AR modelling of strain data from drive-by inspections to detect, quantify, and locate damage to railway bridges. An ML clustering algorithm and PCA are used to eliminate environmental and operational effects before tracking AR coefficients. Validated on both a simulated beam and the Mascarat Viaduct in Spain, strain-based features prove more stable and efficient than acceleration-based ones. Nevertheless, the study does not account for all environmental factors or test complex damage scenarios. Daneshvar

et al. (2021) applied a combined AR and ARX model to acceleration data of a cable-stayed bridge to extract damage-sensitive features. An initial AR model is fitted to the data, and its residuals are treated as "exogenous inputs" for a second-stage ARX model, producing residual sequences that are highly sensitive to structural changes. The dimensionality of these residuals is then reduced and analysed for outliers. However, the method relies heavily on large healthy datasets for training and has been validated on only one bridge.

Feature Selection Methods. The performance and generalisation capability of ML models are highly dependent on the selection of input features. Feature selection is employed to reduce data dimensionality, enhance accuracy, and improve computational efficiency by removing irrelevant and redundant features from the dataset (Dhal & Azad, 2022). Methods for feature selection are commonly categorised into three main types: filter, wrapper, and embedded approaches. Filter methods select features based on statistical relevance independently of any learning algorithm. Embedded methods, in turn, incorporate feature selection within the model training process itself. One key example is the use of relative importance (RI)-based techniques. These include feature importance scores derived from decision tree ensembles, such as random forest (RF) and extreme gradient boosting (XGBoost), which rank features according to their contribution to the predictive model (Dhal & Azad, 2022; Wang et al., 2024). Recently, approaches like SHapley Additive exPlanations (SHAP) have gained attention as model-agnostic embedded techniques, providing interpretable importance values for features. However, these may be computationally demanding for large datasets (Wang et al., 2024).

Wrapper approaches search for optimal feature subsets by directly evaluating model performance, often employing techniques such as genetic algorithms or other evolutionary strategies to efficiently search the vast feature space (Dhal & Azad, 2022). Corbally et al. (2025) presented a knowledge transfer-enabled wrapper-based feature selection framework for drive-by bridge anomaly detection, employing the Binary Slime Mould Algorithm (BSMA) to identify the most damage-sensitive features from vehicle-mounted accelerations. The optimal feature subset is selected based on data from a calibrated finite element (FE) model and has been validated using measurements from the physical bridge in a healthy state. The combined simulated and real-bridge measurements are then transferred to real bridge monitoring, enabling effective damage detection through a one-class Support Vector Machine (SVM). While the approach achieves high detection accuracy, distinguishing between different levels of damage severity in experimental conditions remains challenging.

A widely used way to trade off input compactness (fewer, more robust predictors) against predictive performance is to pose the task as a multiobjective optimisation problem and solve it with the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al., 2002). NSGA-II combines three mechanisms: fast non-dominated sorting to rank solutions into Pareto fronts, an elitist replacement that merges parents and offspring and then reduces the population size to N, and crowding-distance diversity preservation to maintain a well-covered front. The algorithm's complexity is $O(MN^2)$ for M objectives and population N (Deb et al., 2002). O denotes the order of growth of the algorithm's computational cost, namely time complexity, as the problem size increases. NSGA-II yields stable and diverse feature subsets without requiring hand-tuning of niche/sharing parameters, and allows for straightforward constraint handling (e.g., hard caps on subset size) via tournament selection with a crowding comparator. In structural health monitoring, Peng et al. (2018) employed NSGA-II with a binary encoding over 64 candidate sensor locations to minimise the number of sensors while maximizing a separability index computed from wavelet-band energies and PCA features across 128 impact classes. Experiments supported the Pareto designs, with an average recognition rate of 59% using three sensors and 77% with four, and NSGA-II achieved a shorter elapsed time (faster convergence) than the multi-objective optimisation monkey algorithm used for comparison. This baseline algorithm is a nature-inspired multiobjective heuristic based on the "monkey algorithm", which searches by alternating local improvement ("climb"), broader exploratory moves ("watch-jump"), and global repositioning steps ("somersault") to avoid local optima.

An impact-localisation study by Li et al. (2015) framed sensor count vs class-separability as the two objectives and solved it with NSGA-II, yielding Pareto-optimal layouts under tight sensor budgets. For vibration-based damage identification, Raich & Liszkai (2012) applied a multi-objective GA from the same family to minimise sensors while maximising frequency response function-based sensitivity to element-level damage, producing Pareto-optimal sensor/excitation layouts that improved subsequent damage identification. Extending beyond placement, Yao et al. (2024) co-optimised piezoelectric energy-harvester geometry for self-powered sensing as a dual-objective problem—maximising harvested energy and unsupervised damage-detection accuracy from the harvester's voltage. They used Kriging surrogates to reduce simulation cost, reported informative Pareto fronts with experimental validation, and achieved up to 13% higher detection accuracy than acceleration-based sensing for the best designs.

Hyperparameter optimisation. Before training the main algorithm, most ML models require the selection of a set of hyperparameters, whose determination markedly affects the efficiency and generalisation of the resultant model (Claesen & De Moor, 2015). The calibration of these hyperparameters is a key factor in achieving accuracy and robustness. Three widely used strategies are Grid Search, Random

Search, and Bayesian Optimisation (BO). Grid Search systematically evaluates every combination within a predefined search space; it is transparent, non-stochastic, and highly parallelisable, and—given sufficient computational resources—can identify the best configuration within that grid (Yu and Zhu (2020)). Formally, it enumerates the Cartesian product of candidate values per hyperparameter, so the number of trials grows exponentially with dimensionality—a phenomenon commonly referred to as the curse of dimensionality—and uniform grids can miss important directions, particularly when only a few hyperparameters are relevant and the problem has low effective dimensionality. Random Search samples configurations independently from predefined distributions; it is especially effective when influential hyperparameters occupy small or irregular regions of the space or when good values are unevenly distributed (Bergstra & Bengio, 2012). It typically outperforms Grid Search because hyperparameter response surfaces often have low effective dimensionality. Therefore, random sampling explores more distinct values along the important axes, is trivially parallel, and allows for "anytime" (you can add/stop trials) and easy combination with early-stopping schedulers. In practice, both are used successfully in structural/bridge Structural Health Monitoring. For example, Lan et al. (2023) used Grid and Random Search to tune Adaptive Boosting (AdaBoost) base learners for damage detection from raw drive-by accelerations. Piraei et al. (2025) tuned all model hyperparameters using a grid search to automate the selection process and obtain the optimal settings, while Huynh et al. (2025) employed a grid search with k-fold CV to enhance robustness and mitigate overfitting.

Bayesian optimisation is a sequential, model-based method. It fits a probabilistic surrogate to past trials and predicts both mean performance and uncertainty. The following configuration is chosen by maximizing an acquisition function, such as the Probability of Improvement, Expected Improvement, or an Upper or Lower Confidence Bound. The surrogate is often a Gaussian process, though RFs and the Tree-structured Parzen Estimators are also used. It typically finds strong settings with far fewer expensive evaluations. In Genetic Programming-based Bayesian optimisation, the surrogate is trained on N observations, where N is the number of hyperparameter configurations already evaluated, so the training cost scales as $O(N^3)$. Pure Bayesian optimisation is inherently sequential, so practitioners use batch or parallel BO, multi-fidelity methods such as early stopping, and hybrids like Hyperband or BOHB. In a related drive-by study, Hajializadeh (2022) used Bayesian optimisation (BO) to tune CNN hyperparameters on STFT spectrogram inputs (Mendes, n.d.). In related applications, Rageh and Linzell (2020) trained ANNs on damage-sensitive features from monitored strain data and used Bayesian regularisation to balance data-fit and weight penalties. At lower damage intensities, false positives and negatives occur, which indicated scope for improvement. Hajializadeh (2022) employed Bayesian optimisation to tune CNN

hyperparameters—such as learning rate, dropout, and training epochs—on STFT spectrogram inputs, thereby improving simulation-based drive-by damage classification, despite not examining variable speeds or rail irregularities. Wang et al. (2025) tuned RF, Gradient Boosted Decision Trees, XGBoost, AdaBoost, and Light Gradient Boosting Machine (LightGBM) with BO—a Gaussian-process surrogate with Expected Improvement—over defined ranges, minimising Mean Absolute Error (MAE) of the test set; they preferred BO over grid/random because it copes with many hyperparameters and converges faster with fewer trials.

Model selection and performance estimation. Comparison of the candidate ML models and generalization assessment can be conducted using various methods, including CV, hold-out splits, bootstrap estimators, information-criterion measures, and time-ordered evaluations. CV is the standard approach to estimate generalisation error and to conduct model selection without information leakage. Stratified k-fold is widely recommended, and all data-dependent preprocessing—including feature selection and scaling—must be fitted inside each training fold. When hyperparameters are tuned, nested CV is preferred to obtain an unbiased performance estimate. For time-series, folds must respect temporal order via forward-chaining or rolling windows (Boulesteix, 2006, Moscovic and Rosset, 2022). Time-series variants must respect temporal order. To mitigate tuning bias and estimate generalization error, ML studies employ CV—typically stratified k-fold with repeats, leave-one-out for small samples, forward chaining for time-series data, and nested CV when hyperparameters are tuned. CV resamples the available data to estimate out-of-sample error and to tune hyperparameters while limiting overfitting.

In k-fold CV, the data are split into k disjoint folds; training is performed on $k-1$ folds and validation on the held-out fold, cycling until each fold serves as validation. Stratified k-fold CV preserves class ratios and is recommended for imbalanced classification and repeated k-fold averages over several random partitions to reduce variance slightly. Leave-one-out CV uses one case per validation fold; it is nearly unbiased but has high variance and can be costly. A single hold-out split is simple but gives a higher-variance estimate. For time series, CV must preserve the temporal order, where training is performed on earlier blocks and validation is conducted on later blocks using forward chaining or rolling windows (Bhagat & Bakariya, 2025). When hyperparameters are tuned, use nested cross-validation: the inner loop selects the settings and the outer loop gives an unbiased performance estimate (Berrar, 2018; Bhagat & Bakariya, 2025). Good practice includes stratification when relevant, keeping *all* preprocessing/feature selection inside each training fold to avoid leakage, and reporting mean performance with variability across folds. These principles were put into practice in recent studies. Huynh et al. (2025) employed a grid search with K-fold CV to select CNN/Long Short-Term Memory Network hyperparameters and then

evaluated the selected model on unseen data to guard against overfitting. Piraei et al. (2025) determined the K parameter for K-Nearest Neighbours by CV as part of their model selection workflow.

Optimisation methods in ML pipelines. Beyond these search frameworks, two broad optimisation families are gradient-based and heuristic/metaheuristic methods (Zanakis & Evans, 1981). In gradient-based schemes, the search direction (the gradient) is defined a priori; such methods can become trapped in local optima, struggle with nonlinear, hidden, or time-varying constraints, and often require detailed constraint specification (Aldwaik & Adeli, 2014; Topping, 1998). To address these limitations in complex, multimodal, or partly discrete spaces (common in ML pipelines), heuristic/metaheuristic approaches—such as genetic algorithms, particle swarm optimisation, and simulated annealing—are frequently adopted to tune hyperparameters efficiently. Heuristic methods gained rapid prominence because they solve real-world problems quickly and effectively (Zanakis & Evans, 1981). Well-known techniques include: (i) genetic algorithms, which are simple to implement and widely used, but do not guarantee a global optimum (Sivanandam and Deepa, 2008; Hare et al., 2013). (ii) ant colony algorithms, which have been applied to NP-hard problems (Dorigo et al., 2006), construction time and cost optimisation (Ng and Zhang, 2008), task scheduling (Christodoulou, 2005), and design problems (Kaveh and Talatahari, 2010). (iii) particle swarm optimisation, in which each particle's position is updated iteratively using a velocity vector influenced by social behaviour, allowing the swarm to return to promising regions discovered earlier (Venter and Sobieszczanski-Sobieski, 2003).

Within the heuristic family, the CE optimisation is a multifunctional tool that solves complex optimisation and estimation problems (Botev et al., 2013). CE is derived from the Kullback-Leibler distance, i.e., the distance between two probability distributions (Moss, 2020). Like the Genetic Algorithm (Holland et al., 2021), CE achieves robustness by considering a population or distribution of possible solutions at each stage of the process. It is more suitable for continuous variables than for discrete variables. As it works with a population of solutions at each stage, it is highly robust and insensitive to issues such as local optima. CE optimisation is highly effective in various studies, including Gonzalez et al. (2013) who fitted theoretical displacements to laboratory measurements to estimate stiffness distributions (damage detection); Geyer et al. (2019) who estimated the probability of structural failure by determining the sampling density that governs the efficiency of importance sampling; Rodrigues et al. (2022) who found the optimum beam and column cross-sectional dimensions of a 2D concrete frame under seismic loading; and Guan et al. (2017) who identified damage in a multi-story shear structure by minimising the difference between modal frequencies extracted from measurements and a numerical model. Tang et al. (2018) used the CE method to identify track profiles on a high-speed railway by analysing vehicle accelerations.

Keenahan et al. (2020) used the CE method, in conjunction with direct integration, to determine the road profile. Ren et al. (2022) employed the CE method with an inverse Newmark-β approach to derive the apparent profile and vehicle characteristics from simulated drive-by accelerations resulting from train passes over a bridge. While using the CE method, premature convergence is a common occurrence. However, known solutions, such as restarts (Botev & Kroese, 2004) and dynamic variance smoothing (Botev et al., 2013), can be employed to obtain a non-degenerate result.

An ANN model comprises several neurons (or layers), one to three hidden (or implicit) neuron layers, and a final layer of output neurons (Wang et al., 2003). These algorithms do not require extensive statistical training; however, they have some limitations, such as a high volume of network training calculations and the need for pre-processing of independent variables or predictors (Wang, 2003). Still, they can help reduce the uncertainties of the problem. They eliminate the need for a clear description of the physical links between bridge scour and numerous effects, and they provide better results than empirical formulations owing to minor errors and correlation coefficients (Ahmad et al., 2015). Back propagation neural network (BPNN) is defined as a method that regularly updates the weights of the network's connections to minimise the difference between the actual output vector and the desired output vector (Rumelhart et al., 1986). It is a monitoring-learning technique that utilizes gradient descent and multivariate linear regression to recognize changes in the weights of connections within the network (Li et al., 2012). In many studies that adopted BPNN, the accuracy levels of empirical scour formulas were increased (Pal, 2019). Multivariate linear regression aims to determine the relationship between the variable being analysed and its independent variables and is applied for predictive purposes (Maulud & Abdulazeez, 2020). Trial-error method (Sun et al. (2019)) and beetle antennae search (Sun et al. (2019, Sun et al. (2022)) are generally preferred when the hyperparameters of back propagation neural network models are tuned.

SVMs, as proposed by Vladimir Vapnik, map input vectors from a primary feature space to a multi-dimensional feature space via a Kernel function (Charfi et al., 2012). SVMs were derived from statistical learning theory and are a relatively new approach that encourages learning to sort functions to recognize patterns or perform evaluations for functions within a regression problem (Ukil, 2007). Variations of CV, such as leave one out and k-Fold, Xi-Alpha bound, generalized comparative Generalized Kullback–Liebler Distance, approximate span bound, and Radius-margin bounds, are some of the successful performance measures to pick hyperparameters of a SVM model. Vapnik Chervonenkis bound failed to provide and efficiently estimate the parameters (Duan et al., 2003). SVMs can deal with various feature spaces and can categorize the data with no structure or semi-structure (i.e., image and texts) very well. Nevertheless, they require high computation of complicated data, are bad at processing noisy input, and comprehending

the resulting model, the effect, and weights of the variables is not easy as noted by Uddin et al. (2019). RF is an ensemble learning method that builds many decision trees on bootstrapped samples and injects feature-level randomness at each split. This dual randomisation decorrelates trees and reduces variance, yielding strong generalisation for both classification and regression tasks (Breiman, 2001). In addition to predictive accuracy, RF offers practical diagnostics: out-of-bag (OOB) samples—those not included in a tree's bootstrap draw—provide an internal estimate of generalisation error and enable variable-importance assessment without a separate hold-out set (Breiman, 2001). These properties make RF attractive in civil/structural applications with nonlinear, multi-factor inputs. Recent studies report high accuracy, robustness to noisy, high-dimensional data, and practical post-hoc interpretability via permutation importance and SHAP analyses (Dzięcioł & Sas, 2024). In bridge engineering, RF has been applied in vehicle-induced, response-based damage identification frameworks and shown to outperform SVM and BPNN baselines. Moreover, particle swarm optimisation has been used to tune key RF hyperparameters (number of subtrees and candidate split features), further improving classification performance (Zhang et al. (2023b)).

Compared to conventional algorithms based on gradient descent, extreme learning machines learn more quickly and eliminate some challenges associated with setting a criterion to stop, the rate of learning, the duration of learning, and local minima, according to Sun et al. (2022). Their fundamental benefit is reducing the volume of calculation, which is particularly relevant when working with templates in high-dimensional spaces, as described by Charfi et al. (2012). Differential evolution algorithms, as in Dong and Yang (2021), statistical learning theory, as in Oneto et al. (2017), and particle swarm optimisation, as in Bao et al. (2021), have been proven effective in optimizing the hyperparameters of extreme learning machines. Gradient tree boosting seeks to identify a mapping that can transform the input to the output, minimising the discrepancy between the loss function for the predefined and anticipated values, as discussed by Truong et al. (2020). Although the process is time-consuming, Bayesian hyperparameter optimisation is a method for setting the parameters of a gradient tree boosting model, as well as grid search. graphics processing unit acceleration enables faster selection processes, as demonstrated by Anghel et al. (2018). Another algorithm providing successful applications for scour estimation, as shown by Ghodsi and Khanjani (2020) and Najafzadeh and Azamathulla (2015), is the group method of data handling, a procedure that organizes itself through models created step-by-step, based on their performance assessment on a group of multi-input–single-output data couples, as in Madandoust et al. (2012). Grid search is one of the most common methods for setting parameters in the group method of data handling models, as noted by Xu et al. (2020), Stefenon et al. (2020), and Samsudin et al. (2011). Finally, adaptive-network-

based fuzzy inference system is a composite neuro-fuzzy system, as described by Martinek and Zidek (2014), which combines neural network and fuzzy logic methods. Internal parameters are learned offline, similar to how a neural network operates, and act as a fuzzy logic system when it is operated as described by Vieira et al. (2004). The Grasshopper optimisation algorithm, as proposed by Han and Zhao (2011), and the cultural algorithm, as introduced by Fattahi and Hasanipanah (2021), were utilized to set the parameters of the adaptive-network-based fuzzy inference system models.

Research For Scour Depth Prediction

This section presents a detailed overview of ML research focused on the prediction of scour depth, which constitutes the majority of ML studies on bridge scour. These studies are summarised in Table 2.3. The following aspects should be considered: the characteristics of the input data used for training and validation, the primary and supplementary algorithms employed, the specific scour properties examined, and the key findings obtained.

Some of the most important features of ML studies are the characteristics of the input data, such as its quality, source, and properties. Details of the input data of the studies in Table 2.3 were noted here. Zhang and Zhao (2020) noted only that the training data source included both field and laboratory measurements, without providing further details. They also indicated that the verification data belonged to the Hangzhou Bay Bridge in China. The input data for the model training and validation by Dong et al. (2020) consisted of scour depth measurements of bridge piers. Data were gathered from the following two sources: the field data were collected from three bridges in China—Mingchu, Silo, and Houfeng bridges; while the second source was laboratory experiments—two sets of laboratory tests were obtained from J. Sterling Jones Water Conservancy Laboratory of Turner-Fairbank Highway Research Center and Colorado State University. Additional laboratory tests were also used from related studies. Ultimately, they obtained a total of 12 sets of measurements. Four sets of parameters were used for training the model. These parameters (data labels) included flow velocity, water depth, sediment size, and pier width. The relative scour depth data used by Khosravi et al. (2021) were obtained from flume experiments. At the Technology Institute of India, 295 runs were performed in the hydraulic laboratory for three different abutment shapes. The dimensions of the flume were 20m (longitudinal), 0.7m (depth), and 0.9m (width).

Tien Bui et al. (2020) obtained experimental local scour depth measurements of complex piers from the National Hydraulic Research Institute of Malaysia and Sharif University of Technology in Iran. The laboratory models were scaled versions of existing bridges in Malaysia. The tests were performed under clean water conditions on various complex piers with different geometric parameters, such as plan

dimensions of the pier and the piles, water depth, distance of the pile to the edge of the pile cap, flow direction, the distance between piles, and pile cap elevation, as in Amini and Mohammad (2017). Experimental setups were adjusted to eliminate the effects of flow depth, sediment size, and contraction on piers. The data sources of Ebtehaj et al. (2017) consisted of field measurements from 14 distinct bridges located in Canada, Pakistan, and India. Four different data labels were available: sharp, round, cylindrical, and square-shaped bridge piers' measurements. There were also sub-labels of this scour depth data for narrow, wide, and intermediate-sized piers' measurements. Finally, the sub-labels were grouped under five dimensionless categories of scour depth to flow depth ratio, Froude number, median diameter to flow depth ratio, pier length to flow depth ratio, standard deviation of bed size grain diameter, and pier diameter to flow depth ratio. Later, in another study, Ebtehaj et al. (2018) collected the input data from three different studies. Two of the studies performed the experiments in a four-meter-long flume, controlling the mean sediment size (0.98 mm) and pipe diameter (0.016m). The measurements were taken for various pile plan layouts (1:2, 4:1, etc.). The third study of data resource used a smaller sediment size (0.80 mm) and pipe diameters (0.06m and 0.041m), but a longer flume (46m). In another study, Khosravi et al. (2021) utilized the same data that Ebtehaj et al. (2019) had in 2017.

The data set of Sreedhara et al. (2021) included both scour conditions of clear-water and live-bed. The effects of the independent parameters, including flow depth, velocity of approach, duration of flow, and median sediment size, were investigated on predicting scour. There were also four different data labels of sharp-nosed, round-nosed, rectangular, and circular piers' scour depth measurements. Cheng et al. (2015) used field scour depth measurements collected from 79 sites in 17 states of the USA to predict equilibrium scour depth. They excluded scour in bridges whose substructure at a support is a pier group rather than a single pier, cohesive bed material, as well as moderate and substantial debris effects. The parameters considered were the width and shape factor of the pier, the skew of the pier to the approach flow, the size of the grain in the bed, the gradation of the bed material, and the depth and velocity of the flow. Hoang et al. (2018) used laboratory data, which includes four datasets from the Hydrotech Research Institute of National Taiwan University. Complex pier foundation scour measurements were taken in a sand bed. Parameters considered were soil-covering height, median grain size, standard deviation of riverbed material, ratio of the mean velocity of the sediment to the critical velocity, flow depth, pier width perpendicular to the flow direction, the width of the pile cap, and the longitudinal extension of pile-cap face out from pier face.

Kim et al. (2015) combined two sets of field data—those from the Bridge Scour Data Management System (BSDMS) and Federal Highway Administration documentation by the United States Geological

Survey. Departing from the information on the different directions of vortices (and therefore scour developments) in upstream and downstream areas, their first data label was location. Velocity and flow depth of approach, median grain size, sediment gradation, and scour depth were other data labels. To improve the overall quality of the input, data with missing location or other properties listed above were excluded. Pal (2019) selected 232 upstream scour length measurements from a study by Mueller and Wagner (2005), which collected scour observations in 79 different rivers across several states in the USA. Data included skew of the pier to approach flow, pier shape factor, pier dimensions, flow velocity and depth, grain size, and gradation of bed material. Only raw data were used to obtain better performance. In testing the dataset, the root mean square values of both neural networks within the scope of the research were compared to obtain the ideal values for various parameters defined by the user.

Further examples of input data characteristics are summarised as follows: Wang et al. (2025) built their models on a field-monitoring database from 79 U.S. bridge stations (Mueller & Wagner, 2005), covering 487 piers with nine inputs spanning pier geometry, approach velocity/depth, and graded bed sizes, after normalization. Deng et al. (2024) utilized flume data for three abutment shapes (vertical-wall, 45° wing-wall, and semicircle), comprising 297 clear-water runs (99 per shape), and split them 70/30 for training and testing—the inputs centred on the excess abutment Froude number and key length-scale ratios. Eini et al. (2024) compiled 213 sharp-nose (field only) and 347 round-nose (field + lab) pier cases without scour protection, with inputs including velocity, flow depth, pier width/length, sediment d_{50}, fluid properties, and attack angle. Flayyih et al. (2025) aggregated validated experimental datasets from the literature. They constructed a dimensionless input set (e.g., bedload transport rate, flow angle of attack, depth-to-width, and time scales) to predict maximum scour depth. Huynh et al. (2025) curated 100 laboratory cases for complex bridge piers (column, pile-cap, and pile-group). Then they transformed the variables to nondimensional Π-groups using the Buckingham-Pi theorem, removing outliers before modelling. Kumar et al. (2024) merged three laboratory datasets (Kothyari (1989), Chang (2004), Oliveto and Hager (2005)) into 555 rows with six features. They modelled time-dependent scour using nondimensional inputs (e.g., y/D_p, D_p/d_{50}, F_d, s_g, and scaled time) and split about 58%/42% for training/testing. Finally, Piraei et al. (2025) used 859 U.S. field measurements (Benedict & Caldwell, 2014), Holnbeck (2011); Wilson (1995)), adopting seven nondimensional inputs (pier-nose shape K_1, skew K_2, y/b, V/V_c, σ, D_{50}/b, F_r) with a 70/30 train/test division.

In terms of the accuracy level of their predictions, Zhang and Zhao (2020) and Dong et al. (2020) stated that their predictions were in good agreement with the measurements, with only one dataset falling outside the 50% error line. They stated that their approach was more effective in terms of accuracy when

compared to empirical formulas such as Melville–Sheppard, Melville (1997), HEC-18, 65-1, and 65-2. According to their sensitivity analysis, Dong et al. (2020) stated that the most effective parameter in predicting scour was pier width. They also suggested increasing the quantity of training data to more accurately predict scour depths.

Table 2.3. Properties of recent ML studies for scour prediction (adapted from Tola et al., 2023).

Study Reference	Quantity of data	Training / Validation %	Base algorithm	Assisting Approach/ Algorithm	Compared Algorithms / Existing Formulas	Most significant parameters considered	Target
Zhang and Zhao (2020) Dong et al. (2020)	Not provided 11 sets of field and lab data (scour depth measurement, bathymetric data measured with point laser sensors)	Not provided	Convolutional Neural Network Multiple linear regression method	Not provided The cost function for determining the accuracy of the model	Empirical Formulas: - 65-1, 65-2 of China - Melville-Sheppard - Melville (1997) - HEC-18	Velocity of flow Depth of water Diameter of the sediment Pier width	Local scour depths around piers
Khosravi et al. (2021)	99 examples of relative scour depths of a 0.7 m deep flume	70 % Training 30 % Validation	Kstar model with five hybrid algorithms: - Weighted Instance Handler Wrapper-Kstar	Pearson correlation coefficient (to pick the most relevant input parameters)	Empirical equations of Dey and Barbhuiya (2005) and Muzammil (2010)	Relative Flow Depth Excess Abutment Froude number Relative Sediment Size Relative Submergence	Relative scour depth around abutments
Tien Bui et al. (2020)	122 laboratory datasets of scour depths. An experiment was conducted in a sand bed flume, and measurements were taken using a vertical point gauge.		Reduced Error Pruning Tree-based classifier	Taylor diagram (For fitting and performance optimisation)	- ANNs - SVM - M5P - Reduced Error Pruning Tree algorithms and 2 empirical formulas of the Florida Department of Transportation and HEC-18	Pile cap width Thickness Column width	Local scour depth at complex piers
Ebtehaj et al. (2017)	476 field pier scour depth measurements for four different geometric pier shapes.	80 % Training 20 % Testing	- The Extreme Learning Machine regression method - The self-adaptive version of Differential Evolution	- SVM - ANN	Not provided	Pier dimensions Sediment mean diameter	Scour depth around piers
	321 experimental	75 % Training	Extreme Learning Machines	Different sets of input combinations were used	- SVM	Critical and average flow velocity	

Ebtehaj et al. (2018)	datasets of flumes, scour depths measured with a point gauge	25 % Testing		to find the most effective variables.	- ANN	Flow depth Median diameter of particles Pile diameter Number of piles normal to the flow Distance between adjacent piles in line with the flow	
Ebtehaj et al. (2019)	476 field pier scour depth measurements	80 % Training 20 % Testing	Extreme Learning Machines	Dimensional analysis to detect effective dimensionless parameters	Existing regression-based models Richardson and Davis (2001) Johnson (1992), Shen (1969) Laursen and Toch (1956)	Ratio of pier width to flow depth Ratio of pier length to flow depth	
Sreedhara et al. (2021)	104 sets of experiments to measure scour depths with an electronic total station device	Not provided	- Gradient Tree Boosting - Group Method of Data Handling technique.	Coefficient of Determination as to the performance index	SVM Adaptive-network-based fuzzy inference system Particle Swarm Optimisation-Based SVM	For clear water scour: Sediment size and quantity Velocity Flow time	The scour depth of circular, rectangular, round-nosed, and sharp-nosed piers
Cheng et al. (2015)	237 pier scour depth measurement datasets taken with an echo sounder	Not provided	Evolutionary Radial Basis Function Neural Network model = Radial Basis Function Neural Network and Artificial Bee Colony	Not provided	GE BPNN Regression Tree SVM - HEC18 -Mississippi's method Van Wilson (1995) Laursen and Toch (1956) Froehlich (1988)	Pier shape factor Pier width Skew of the pier to approach the flow Velocity of the flow Depth of flow Grain Size of the Bed Material (d_{50}) Gradation of bed material	Scour depth
Hoang et al. (2018)	170 data samples of clear water scour depths	Not provided	Support Vector Regression-based model	Filter and wrapper feature selection strategies (for performance improvement)	HEC18 Richardson and Davis (2001) Melville and Coleman (2000) Ataie-Ashtiani (2010)	Under three groups: Pier geometry Flow property Material characteristic of riverbed	Local scour around complex piers
	403 sets of upstream and 61 sets	80 % Training	NSGA-II		HEC18 Froehlich (1988)	Pier width Approaching flow depth Median grain size,	

Kim et al. (2015)	of field downstream scour depth measurements	20 % Validation		SVM for increasing the pool of field data	Gene expression programming model	Sediment gradation coefficient Gradation of bed material	Critical scour depth
Pal (2019)	232 field data	66 % Training 34 % Testing	Deep Neural Network	Back-Propagation Neural Network	Froehlich Equation (1988) Froehlich Design HEC-18 HEC-18/Mueller Equation (1996) BPNN	Not provided	Local scour around bridge piers
Liao et al. (2018)	175 experimental datasets for scour depth	Not specified	Sequential quadratic programming optimisation Least Squares Support Vector Machine	Sequential quadratic programming to seek the optimal coefficients	HEC18 - Melville and Coleman (2000) - Ataie-Ashtiani (2010)	Flow direction Pile-cap width Covering soil height Pier length Velocity of sediment movement Flow velocity, Median grain size Flow depth, Riverbed material Standard deviation	Scour depth of a bridge with a complex pier
Kumar et al. (2024)	555 lab samples (Kothyari 1989; Chang 2004; Oliveto & Hager 2005)	58 / 42 (split)	XGBoost, Random Forest Regressor, Extra Trees	Bagging & boosting ensembles	Gradient Boosting Regressor, M5P, weighted instance handler wrapper-Kstar, Bagging Kstar, AdaBoost Regressor, Support Vector Regression, Gaussian Processes Regression, empirical equations (Oliveto–Hager 2002; Johnson, 1992; Ettema, 2011; HEC-18)	$log\ (T_i)$, σ_g, F_d, y/D_p, D_p/d_{50}	Time-dependent pier scour depth (d_s/D_p)
Piraei et al. (2025)	859 field-based samples collected from various studies in the USA (Benedict & Caldwell, 2014; Holnbeck, 2001; Wilson, 1995)	70 / 30 (split)	16 individual ML models and a stacked Case Based Reasoning meta-learner	Grid search hyperparameter tuning; stacking ensemble (2-level meta-model) + SHAP interpretability analysis + reliability analysis	Compared against nine empirical equations (Inglis-Poona II, Laursen & Toch, Laras, Arunachalam, Breusers, Jain-Fischer, Froehlich, HEC-18, Florida Department of Transportation	K_1, Upstream flow depth/pier width (y/b), Fe; D_{50}, σ_g, Velocity ratio (V/V_c), K_2, D_{50}/b ratio	Equilibrium (local) scour depth around bridge piers

Deng et al. (2024)	297 experimental cases (99 per abutment shape) under clear-water, uniform sediment conditions.	70/30 random split into training and testing sets.	M5′ Model Tree, Multivariate Adaptive Regression Splines, Locally Weighted Polynomial Regression, Multigene Genetic Programming.	Correlation analysis + sensitivity analysis to choose input set; four input combinations evaluated; uncertainty analysis and Taylor diagrams for reliability/performance assessment; additional validation on an expanded dataset.	Empirical equations of Dey and Barbhuiya (2005) and Muzzammil (2010)	Dimensionless inputs: F_e, grain-size ratios d_{50}/l and d_{50}/h, flow-depth ratio h/l	Predict relative equilibrium scour depth at vertical-wall, 45° wing-wall, and semicircular bridge abutments.
Eini et al. (2024)	560 records total: 213 sharp-nosed (field only) + 347 round-nosed (field + lab).	70 / 30 split (train/test)	ANN (Multilayer Perceptron)	Metaheuristic hyperparameter tuning with Brown-Bear Optimisation (BOA) and Siberian Tiger Optimisation (STO); SHAP for global interpretability; explicit ANN equation derived.	Benchmarked against SVM, KNN, Genetic Programming, and empirical equations (highlighting Froehlich 1988)	SHAP shows pier width D as the most influential factor. Least influential: L (sharp-nosed) / d_{50} (round-nosed)	Equilibrium local scour depth around non-circular (sharp- and round-nosed) bridge piers with shallow foundations.
Flayyih et al. (2025)	581 observations compiled from published experiments covering hydraulic, sediment, and geometric variables.	Not provided	Multilayer Perceptron, Radial basis function network, Multigene Genetic Programming	Dimensional analysis to define input π-groups; standardized/normalized inputs; variable-importance and sensitivity analyses; broad metric suite (R, R^2, RMSE, MAE, NSE, WI, etc.).	Internal comparison among ANN, Radial basis function network, and Multigene Genetic Programming (no direct benchmarking to specific empirical formulas)	Bedload transport rate (q^* or related group), depth-to-width ratio (h/B), Fe, exposure (flow) time	Relative scour depth
Huynh et al. (2025)	100 laboratory cases for complex bridge piers (square-nose column + pile-cap + pile-group) compiled from Martin-Vide et al.	64% train, 16% validation, 20% test (80%: train/validation as	1D-CNN, Long Short-Term Memory Network	Buckingham-Pi dimensional analysis to build dimensionless inputs; outlier removal (boxplot/IQR); correlation analysis; grid search with K-fold CV for	HEC-18, FDOT (Sheppard–Renna), Coleman (2005)	Strong correlations among some (e.g., $\frac{d_{50}}{b_{pc}}$with $\frac{S_n}{b_p}$).	Maximum local scour depth at complex bridge piers, predicted as

	(1998), Coleman (2005), Beheshti & Ataie-Ashtiani (2010, 2016), Amini et al. (2014)	80/20; 20% as test)		hyperparameters; Adam/MSE training setup			$\frac{S_d}{b_{pc}}$(dimensionless).
Wang et al. (2025)	487 pier records from an open-source field database (79 bridge monitoring stations)	80 / 20 (train/test); no separate validation set (no additional validation due to limited samples)	Ensemble tree models: RF, Gradient Boosted Decision Trees, XGBoost, AdaBoost, LightGBM (best: LightGBM).	Bayesian hyperparameter optimisation; feature-importance analysis; Taylor diagram & violin plots; Conditional Generative Adversarial Network for data augmentation and verification.	Internal comparison among RF/Gradient Boosted Decision Trees/XGBoost/AdaBoost/LightGBM (no empirical-equation baseline reported)	Highest influence: pier depth, pier diameter, current velocity. lowest influence: $D_{16}/D_{50}/D_{84}/D_{95}$; pier length and slope angle.	Local scour depth (SD, m) at bridge piers (underwater), predicted from 9 physical inputs.

Khosravi et al. (2021) investigated scour occurrence for different shapes of the abutment. They noted that, for all shapes of the abutment, the most effective parameter was the Excess Abutment Froude number (Fe). Moreover, for vertical-wall-shaped abutments, the most effective parameter combination contained the Excess Abutment Froude number, relative sediment size (d_{50}/l, where "d_{50}" is the sediment size and "l" is the dimension of the abutment which is perpendicular to the flow), and relative flow depth (h/l). Finally, for semicircular and 45˚wing shape abutments, the combination of Excess Abutment Froude number and relative sediment size was the most effective parameter combination in scour prediction. The Weighted Instance Handler Wrapper (Kstar) for vertical-wall abutments, random committee-Kstar for semicircular walls, and 45˚ wing wall were the best algorithms among the five novel hybrid algorithms studied. Their algorithms outperformed the empirical formulas of Dey and Barbhuiya (2005) and Muzzammil (2010). The hybrid approach of Zhang and Zhao (2020), based on a random subspace meta-classifier, identified the pile cap level as the most sensitive factor in predicting the local scour depths of complex piers. The Reduced Error Pruning Tree-base classifier yielded similar root mean square errors to those of ANNs, SVMs, and the M5P model. The predictions obtained through the reduced Error Pruning Tree, and other machine learning algorithms were significantly better than the scour depths computed using the empirical models of Florida Department of Transportation (FDOT) and HEC-18, as noted by Richardson and Davis (2001). Both Khosravi et al. (2021) and Tien Bui et al. (2020) successfully enhanced the predictive power of standalone algorithms by proposing hybrid algorithms.

Ebtehaj et al. (2017) proposed a Self-Adaptive Evolutionary Extreme Learning Machine to predict scour around bridge piers. They indicated that the ratio of the median diameter of particle size to flow depth, the ratio of pier length to flow depth, and the ratio of pier width to flow depth were the most effective parameters. Self-adaptive evolutionary extreme learning machines outperformed ANNs and SVMs. In 2018, Ebtehaj et al. proposed 25 models to predict scour around coastal and hydraulic pile groups (Ebtehaj et al., 2018). The extreme learning machine model generated had the optimal input parameter combination and provided better results than the ANNs and SVMs considered. They also identified that pier diameter affected the predictions the most. Later in 2019, Ebtehaj et al. (2019) demonstrated that extreme learning machines were one of the most effective heuristic optimisation algorithms for nonlinear systems. The sensitivity analysis comprised 31 models with varying input combinations, as per Ebtehaj et al. (2019). Their approach outperformed the empirical equations of Richardson and Davis (2001), Johnson (1992), Shen (1969), and Laursen and Toch (1956). They recommended that the proposed methodology be improved by utilising other artificial intelligence methods, such as gene expression programming and the group method of data handling.

Sreedhara et al. (2021) developed a model using gradient tree boosting to predict scour depth around piers with various geometries, including rectangular, circular, sharp-nosed, and round nosed. In conclusion, they stated that gradient tree boosting was more accurate and effective than the group method of data handling for all shapes. It was noted that the model worked best with the rectangular form. Cheng et al. (2015) proposed a model using an Evolutionary Radial Basis Function Neural Network. It outperformed not only empirical HEC-18, Mississippi's, Laursen and Toch's (1956), and Froehlich's methods, but also other algorithms being compared (back-propagation neural network, GE, M5 regression tree, and SVM). They suggested that future studies consider the efficiency, duration required for computation, and stability of artificial intelligence methods, as well as techniques that complement each other. In their research based on support vector regression, Hoang et al. (2018) obtained better predictions when support vector regressions were used in conjunction with feature selection algorithms. The variable neighbourhood search algorithm performed best when compared with sequential forward selection and sequential backward selection for parameter selection. The proposed method also made more accurate scour depth predictions compared to HEC18, Melville and Coleman (2000), and Ataie-Ashtiani (2010). They recommended a future study of the support vector regression and other Kernel Functions together.

The nondominated sorting genetic algorithm-based method proposed by Kim et al. (2015) outperformed both the hydraulic engineering circular No. 18 and the Froehlich equations, as well as gene expression programming. The authors highlighted that their approach was as simple as empirical formulas. They noted that the failure rates should be decreased. They also suggested modifying the function and picking field data and other independent parameter combinations more explicitly. Pal (2019) found that ignoring randomly selected neurons (dropout layers) in deep neural networks could not significantly improve the accuracy level. When the prediction values were compared with the measurements, it was seen that both deep neural networks and back-propagation neural networks performed equally well in predicting the scour depth. Additionally, predictions from deep neural networks outperformed four empirical formulas listed in Table 2.3, with root mean square errors that were 2-4 times better. Liao et al. (2018) proposed an algorithm based on sequential quadratic programming optimisation, which increased the accuracy of the existing formulas of HEC18, Melville and Coleman (2000), and Ataie-Ashtiani (2010) by up to 2–4 times.

Recent work continues to favour ensemble and hybrid ML models while noting limits in data breadth and transferability. Wang et al. (2025) tested five tree ensembles and found that LightGBM performed the strongest on the field dataset (test $R^2 \approx 0.936$, RMSE ≈ 0.241 m, WI ≈ 0.966). They also utilized Conditional Generative Adversarial Network for augmentation and cautioned that their conclusions are

based on only 487 samples, suggesting that future exploration of physics-informed networks is warranted. Deng et al. (2024) compared the M5′ model tree, Multivariate Adaptive Regression Splines, Locally Weighted Polynomial Regression, and Multigene Genetic Programming for three abutment shapes. They showed that the optimal model depends on geometry: Multigene Genetic Programming was best for vertical-wall and semicircular abutments (e.g., testing $R^2 \approx 0.989$ and 0.984, respectively), whereas Locally Weighted Polynomial Regression led for 45° wing-wall (testing $R^2 \approx 0.985$). Overall gains over Dey & Barbhuiya (2005) and Muzzammil (2010) were significant in RMSE terms. Eini et al. (2024) coupled an ANN with Brown-Bear and Siberian Tiger optimizers, outperforming SVM, K nearest neighbours (KNN), and Genetic Programming on mixed field/lab data. SHAP emphasized pier width as the dominant driver, and the authors called for validation on additional geometries and flow regimes.

Flayyih et al. (2025) reported that ANN was the most accurate in their compiled experiments ($R^2 \approx 0.94$, RMSE ≈ 0.032), followed by Multigene Genetic Programming ($R^2 \approx 0.91$), and Radial basis function network was the most conservative. Sensitivity analysis highlighted the importance of bedload rate, dimensionless time, and depth-to-width ratio. Huynh et al. (2025) developed 1D-CNN and Long Short-Term Memory Network models for complex piers. They demonstrated that the CNN outperformed both Long Short-Term Memory Network and empirical baselines (with $R^2 \approx 0.85$ and RMSE ≈ 0.1125), recommending the use of larger and more varied datasets. Kumar et al. (2024a) evaluated XGBoost, RF, and Extra Trees for time-dependent scour. XGBoost performed best overall, but all models tended to predict peak scour depths that were lower than the measured peaks. RF was relatively closer at peak values. The authors attribute this, in part, to the way separate experimental runs were merged into a single sequence, so peak values in one run are followed immediately by low initial values from the next, which can bias learning away from extreme peaks. Sensitivity analysis indicated that the scaled time variable was the most influential. Piraei et al. (2025) introduced a stacking approach using CatBoost as a meta-learner over 16 base models, surpassing standalone models and nine empirical equations (test $R^2 \approx 0.87$, NSE ≈ 0.84; RI ≈ 0.972). They identified the shape factor and depth ratio as key features. They also analysed reliability and SHAP contributions.

Overall, most studies allocate the larger share of their datasets to model training, with the remainder reserved for validation. In many cases, this is complemented by a held-out test set to assess generalisation. Careful preprocessing of the training data—normalisation, outlier control, and correlation screening—improves optimisation stability, reduces bias, and accelerates convergence toward accurate predictions. Performance is typically reported using RMSE, MSE, coefficient of determination (R-squared), and cumulative sums of orthogonal distance, with several studies additionally reporting Nash–Sutcliffe efficiency

(NSE), Willmott's index of agreement, or a composite ranking index to benchmark overall agreement. The common objective is to produce models that are more robust than empirical scour formulas and conventional algorithms, with reliability across hydraulic regimes and geometric configurations. To this end, recent work frequently employs hybrid or ensemble learning, combined with meta-heuristic hyperparameter tuning, thereby balancing accuracy with computational efficiency. Consistent with Sivanandam and Deepa (2008), the growing use of heuristic search reflects its practicality and effectiveness for complex engineering problems. Studies that incorporate sensitivity analysis and systematic hyperparameter optimisation, tend to achieve faster and more accurate predictions by revealing influential inputs and mitigating overfitting. Representative approaches include particle swarm optimisation and artificial bee colony, alongside other bio-inspired strategies (e.g., those inspired by the behaviour of the brown bear and Siberian tiger, as well as genetic algorithms). Such methods facilitate identification of the most effective variables or combinations, improving both feature screening and interpretability. Notably, the "most influential" parameter is not universal: findings vary with the underlying data, including differences in sediment grading, pier/abutment geometry, and flow regime. As all studies aim to surpass empirical formulas, comparisons against frequently used reference algorithms remain informative; in general, extreme learning machines, gradient tree boosting, radial basis function neural networks, and Reduced-Error Pruning Trees outperform standard baselines such as ANNs and SVMs, and—when paired with robust validation—these gains persist across multiple error metrics.

Research for Scour detection and localisation from structural vibrations

With ML, drive-by methods have shifted from hand-crafted vibration features to models that learn damage-sensitive patterns directly from vehicle-borne data (de Souza et al., 2024; Hajializadeh, 2022). Briefly, de Souza et al. trained a sparse autoencoder on healthy drive-by accelerations. They flagged damage via reconstruction error—effective across speeds but reliant on a healthy baseline and not accounting for environmental effects, sensor orientation, or multiple vehicle types. Hajializadeh converted simulated indirect accelerations to STFT spectrograms and used CNNs to learn features and classify damage, without considering variable speeds or rail irregularities. While these studies illustrate the broader data-driven shift in drive-by damage identification, the focus of this subsection hereafter is on ML studies that detect and/or locate scour. This subsection provides the data characteristics, details of the ML models that were used to detect scour, and the results of the recent related research that is provided in Table 2.4.

Several recent ML studies specifically targeted scour damage detection. Fernandes et al. (2024) used an unsupervised deep autoencoder on multi-axle railway bridge simulations with multiple sensor placements for binary anomaly detection. Vertical accelerations were taken from car body, front/rear bogies and front wheel on the first, middle and last vehicles. Operational variability (rail roughness profile, track irregularities, speed dispersion, vehicle mass and suspension variability) and 5% measurement noise were included. Detection used a Kullback–Leibler–divergence damage index over batches of passages with Receiver Operating Characteristic (ROC) analysis to quantify required sample size and optimal sensor locations (front bogie of the first vehicle and rear bogie of the last vehicle performed best for 5–10% damage). Gasparotti et al. (2024) applied a supervised CNN to a highway bridge with a single-DOF sprung-mass vehicle, fixed sensor location, and classified scour severity at a single mid-span location with two damage levels of 5% and 10%. 1×2500 acceleration vectors were reshaped to 50×50×16 inputs. Data were split 70/15/15 (train/validation/test), the network was trained with Adam, and ∼99% test accuracy and strong ROC separation were reported on simulations. In both research by Fernandes et al. (2024) and Gasparotti et al. (2024), scour was represented as stiffness reduction (not probability): Fernandes analysed 5%, 10% (and 20%) pier-foundation stiffness loss in a five-vehicle C model, whereas Gasparotti et al. used reduction factors β=0.95 and β=0.90 to emulate minor and significant damage in a beam model with a mid-span foundation spring.

The number of ML studies focused on vibration-based scour detection remained relatively small compared to those addressing other types of damage or predicting the depth of scour around bridge piers. This literature gap left scope to investigate a broader range of algorithms and incorporate additional data types beyond accelerations, with the capability to detect the location of scour. The limitations of previous studies—such as sensitivity to operational and environmental variability (Fernandes et al., 2024; Gasparotti et al., 2024)), dependence on a single sensor placement (Gasparotti et al., 2024), and focus on a single scoured pier (Fernandes et al., 2024), Gasparotti et al. (2024))—could be addressed by extending simulated vehicle crossings to a broader range of structural and operational parameters. Incorporating structural dynamics and bridge–vehicle interaction models into physics-informed ML frameworks, together with wider variability in simulations, could improve generalisation and robustness and could help offset the lack of field data (Souza et al. (2024), Hajializadeh (2022), Fernandes et al. (2024), Gasparotti et al. (2024)).

Another study by Huang et al. (2023) proposed and validated a simulation-informed framework for detecting bridge scour by pairing a Siamese neural network with a 3D bridge–vehicle–wave interaction model that couples VBI and fluid–structure interaction. Field data consisted of deck accelerations

recorded by 42 accelerometers with a scan rate of 100 Hz during controlled two-axle truck crossings at speeds of 20–30 km/h. Simulated data were generated by reducing foundation stiffness by 0%, 5%, 25%, and 45% at specific piers to emulate scour. All time histories were converted into STFT image inputs for the Siamese Neural Network, which learned to match real and simulated feature embeddings for scour location and damage level identification. Validation on a real bridge showed 81% accuracy for location and 78% for damage level overall (≈91% for severe damage; ≈64% for minor), and the Siamese Neural Network outperformed CNN, BP, and SVM baselines. Accuracy decreased with higher vehicle speeds and with sensors farther from the piers, underscoring the sensitivity to speed and placement. Limitations/future work noted by the authors include the method's focus on local damage rather than multi-scale (global + local) features, class imbalance between healthy/minor states, reliance on simulation-to-field transfer, and the need to examine different encoding sizes/datasets and to perform broader, longer-term field trials under varying traffic and environmental conditions.

Minski & Lopez (2023) proposed a drive-by scheme, in which a 1D CNN was used to classify scour-induced damage levels from raw vertical acceleration measurements taken at the front bogie of a simulated five-vehicle train. All data were numerically generated with varying speed, track irregularities, and noise; scour was represented as support-stiffness loss with five labels (baseline, 5%, 10%, 20%, 50%). The CNN was trained end-to-end on the raw signals and reported an accuracy gain of ~20% over search-tree and unsupervised baselines. Confusion-matrix results were computed using 100 test passages per class in a five-label model. The main errors occurred at the low-damage end. The model sometimes confused the baseline (healthy) class with the 5% and 10% damage classes. To focus on these subtle cases, the authors also trained binary models. The baseline vs 5% model achieved about 90% average precision, and the baseline vs 10% model reached about 95% average precision. The study focused on detection/severity (no localisation), provided limited detail on CV or splits, and remained simulation-only, using a very long scenario (bridge ~4 km, train ~1.83 km), which may limit external validity. A probabilistic, temperature-mitigated vibration-based framework was developed by Zheng et al. (2020) for detecting and localising scour at the pier level. On a nine-span Mississippi bridge FE model, the authors simulated modal frequencies and mode shapes (first eight modes) at 13 points across temperature ranges. They prescribed scour at four piers (including expansion-joint effects). PCA reduced stacked modal parameters to 7 principal components. A Gaussian Process regression model was used to learn the mapping from temperature to the undamaged bridge modal feature vector. The training set comprised 1,100 simulated cases and the test set comprised 500 cases, both generated with varying temperatures. At inspection, the Genetic Programming-predicted temperature component was removed from measured modal data,

after which Transitional Markov Chain Monte Carlo estimated pier-specific scour indices with uncertainty. With temperature mitigation, the maximum relative error was approximately 11.3% (noise-free) and ≈approximately 14.7% (5% noise), compared to approximately 19.6–27.1% without mitigation. The study remained simulation-based, assumed similar temperature effects before/after scour, and depended on reliable modal identification and adequate temperature coverage. Field validation and broader environmental factors were identified as future work.

Taken together, vibration-based scour detection/localisation studies using monitoring data remain comparatively few, and most are still simulation-centred with limited field or laboratory validation. Across the reviewed work, scour is typically modelled as stiffness reduction at one support or a mid-span spring; evaluations report strong classifier performance in controlled settings (e.g., CNN/Siamese Neural Network, autoencoders) but sensitivity to operational/environmental variability, sensor placement, vehicle speed, and a frequent focus on a single scoured pier or a fixed sensor location. Where field or laboratory elements are present, findings confirm feasibility yet expose transfer-to-field challenges (e.g., speed/placement effects, class imbalance, baseline dependence). These patterns underscore the value of VBI frameworks, broader variability in simulations, and richer sensing configurations to improve generalisation and robustness. Because ML studies that detect and locate scour from monitoring signals are markedly fewer in number than depth-prediction works, the literature leaves a clear opening—which this thesis targets—to advance the capability, realism, and demonstrable transferability of drive-by scour monitoring.

2.2.2 Synthesis of the Results

Fifty-nine studies were examined and grouped into two areas: Conventional Monitoring-Based Approaches to Detect Scour (Cluster 1: direct bridge-mounted monitoring; Cluster 2: indirect/drive-by monitoring) and Machine Learning-Based Research (Cluster 1: scour-depth prediction; Cluster 2: scour detection and localisation). The synthesis summarises performance, data requirements, and limitations within each cluster. Recent publications were not exhaustively included by design: methodologically similar works using overlapping datasets or approaches were consolidated by selecting a representative paper. Studies outside the scope—for example, risk/exposure mapping rather than detection/localisation, or ML models that did not use monitoring signals—were not discussed in detail. This curation enables a focused comparison of techniques most relevant to monitoring-based scour detection and localisation.

2.2.2.1 Synthesis of conventional monitoring-based studies to detect scour

Direct monitoring. Sensors for direct monitoring have diversified with advances in technology. Devices now include not only inclinometers and velocity sensors, but also micro-cameras, wireless transmitters, Fibre Bragg Grating, electromagnetic/permittivity probes, piezoelectric rod sensors, energy-harvesting devices, and Unmanned Aerial Vehicle-deployed smart rocks. Studies using these alternative sensing mechanisms process signals through bed-level image recognition, instantaneous frequency analysis, magnetic-field mapping, and changes in soil permittivity. These devices can provide direct, often real-time information on scour, reducing the need for underwater inspections. However, implementation and maintenance remain challenging and may require improved device design; debris and saline water can also affect performance. Vibration-based direct monitoring with conventional sensors (e.g., accelerometers, Global Positioning System) detects scour through changes in natural frequencies and mode shapes arising from reductions in foundation stiffness. Such approaches are advantageous because they do not require knowledge of vehicle forces or detailed geometric/mechanical properties of the bridge. Energy-harvesting devices are increasingly used as sensors, detecting frequency changes while supplying their own power; nonetheless, loading sensitivities and potential errors—especially when multiple piers are simultaneously scoured—are important limitations.

Indirect (drive-by) monitoring. Scour can be identified and located by processing vehicle-mounted signals from single or multiple passages. In these studies, mode shapes or frequencies are extracted, or differences between operating-deflection shapes, continuous wavelet transform coefficients, or related time–frequency indicators are used to flag scour between healthy and damaged states. Mode-shape-ratio methods have been widely applied and show pronounced changes under scour; even so, sensitivity to vehicle loading and random ambient variability remains an issue. For example, the mean-normalised mode shape method of Malekjafarian et al. (2020a) succeeded for a single scoured pier. However, it was less effective when multiple piers were scoured concurrently—an area that would benefit from further numerical investigation. Frequency Domain Decomposition has been frequently used to obtain modal properties from acceleration signals, while the Hilbert–Huang Transform has proven effective for decomposing accelerations into mode functions. However, very short data segments may be insufficient for this purpose. Indirect approaches typically require fewer sensors, avoid specialised underwater devices and placement challenges, and do not interrupt service, making instrumented vehicles a cost-effective option. Common limitations include variations in vehicle parameters (mass and speed) and the need for multiple runs to stabilize the indicators.

Table 2.4. Recent ML research for scour detection and localisation from structural vibrations.

Study reference	Data & signals	Model/setup	Method	Features / input	Split/validation	Task & key results
Fernandes et al. (2024)	Simulation only; vertical accelerations (on-vehicle sensors; 1 kHz); 5-vehicle train; 1000 passages/condition; operational variability in speed (≈55–106 km/h), vehicle mass/suspension, and added measurement noise	VBI model of a 20 m, simply-supported bridge with an elastic (pier–foundation) spring; responses re-sampled in space (0.01 m steps; 5830 points/passage)	Unsupervised deep autoencoder; anomaly index via KL-divergence over batches; ROC-based analysis of detectability & sensor placement. (Model trained on healthy data)	Raw vertical acceleration per passage (Space-normalized)	Trained on 200 healthy passages; evaluation via ROC; sample-size study	Binary anomaly detection under strong operational variability; paper reports feasible detection for 5–10% stiffness loss with suitable on-vehicle sensor placement
Gasparotti & Lopez (2024)	Simulation: bridge accelerations under a single-DOF sprung-mass vehicle; ISO-8608 class "H" road profile	Euler–Bernoulli beam (25 m) with a mid-span foundation spring; damping ≈3%; VBI-2D MATLAB framework.	Supervised CNN classifier (Adam optimizer)	Pre-processed acceleration vectors (CNN inputs)	Train/validation protocol described Emphasis on overfitting control	Severity classification at a fixed mid-span location on simulations; high separation reported; real-world validation identified as the next step
Huang et al. (2023)	Field + simulation; 42 deck accelerometers at 100 Hz; two-axle truck crossings at 20/30 km/h	3D bridge–vehicle–wave interaction (coupled VBI + Fluid–Structure Interaction) to generate labelled synthetic data across 10 scenarios and four damage levels (0, 5, 25, 45% stiffness loss at specific piers)	Siamese Neural Network matching real and simulated STFT images to infer location + damage level	STFT images (Hann window, 128-pt FFT; 0–20 s, ±49 Hz)	80/20 training/ validation on pooled (field + simulated) images	Location accuracy 81.11%, damage-level 78.06% overall. serious ≈91.11%, minor ≈64.44%; accuracy drops at 30 km/h and with sensors farther from piers, Siamese Neural Network outperformed CNN/BP/SVM.
Minski & Lopez (2023)	Simulation only; raw vertical acceleration at the front bogie of a 5-vehicle train; varied speed, track irregularities, and noise	VBI railway bridge scenario (long bridge/train domain)	1D-CNN (end-to-end on raw signals)	Raw time histories	Reported confusion-matrix analysis over 100 test passages/class; binary variants improved precision (≈90–95% for small-damage detection)	Detection & severity; simulation-only; limited CV detail
Zheng et al. (2020)	Simulation; modal frequencies + shapes (first 8 modes) at 13 points over temperature ranges; ±5% noise case studied	9-span Mississippi bridge FE model; expansion-joint effects included	PCA to Gaussian Process (temperature: features); at inspection, remove predicted temperature effect; Transitional Markov Chain Monte Carlo for per-pier scour indices	Stacked modal parameters (reduced to 7 principal components)	1,100 train / 500 test cases for Genetic Programming, Transitional Markov Chain Monte Carlo at inspection	Localisation and quantification with temperature mitigation: max relative error ≈11.3% (noise-free) and ≈14.7% (5% noise), vs ≈19.6–27.1% without mitigation

Recommendations. There is a strong need to validate numerically developed methods and algorithms on full-scale bridges before they can be considered fully acceptable. Extending simplified models to two- or three-dimensional representations, together with more field testing and systematic replication, would help overcome structure-specific limitations and clarify the scope of future studies.

2.2.2.2 Synthesis of machine learning-based studies

Machine-learning research on bridge scour falls into two streams: (i) predicting scour depth from hydraulic, geometric, and sediment inputs, and (ii) detecting/localising scour from monitoring signals

(bridge-mounted or drive-by). Adjacent topics that do not directly detect or locate scour (e.g., risk scoring from inspection reports, flood detection, settlement estimation) are excluded to keep focus.

Scour-depth prediction. This cluster dominates the ML-based scour research. Methods ranging from Support Vector Regression, Extreme Learning Machines, Radial Basis Function Neural Network/ERBNN, adaptive-network-based fuzzy inference system, and deep nets to tree ensembles (RF, XGBoost/LightGBM) and stacked models generally surpass traditional empirical equations (HEC-18, Laursen–Toch, Froehlich, Melville (1997), etc.). Reported sensitivity analyses identify influential variables; however, the "most important" features depend on the dataset (sediment grading, pier/abutment geometry, and flow regime). Key constraints persist, including uneven field coverage, heterogeneous curation, and limited transferability beyond the training domain. Recent work addresses these limitations with disciplined data preparation, dimensionless inputs, uncertainty checks, and reliability analysis.

Scour detection and localisation from structural vibrations. This cluster is relatively minor but growing. Supervised and unsupervised models (CNN/Siamese Neural Network, autoencoders, one-class schemes) achieve high separability in simulation and controlled tests. Nevertheless, scour is typically modelled as a stiffness-reduction surrogate at one support or a mid-span spring, and results show sensitivity to operational/environmental variability, sensor placement, vehicle speed, and class imbalance; multi-pier scenarios and long-duration field trials remain scarce. These gaps motivate richer sensing layouts, broader variability in simulations, and VBI models to improve robustness and field transfer.

Taken together, ML has advanced accuracy beyond empirical baselines and shown clear promise for automation with monitoring data. Because monitoring-based detection/localisation studies are comparatively few, the literature leaves a clear opening—pursued in this thesis—to expand capability, realism, and demonstrable transferability of drive-by scour monitoring.

CHAPTER 3: DRIVE-BY SCOUR DETECTION USING RILA — METHOD AND CASE STUDY

This chapter incorporates material from the published paper by Tola et al. (2025) and its extended content. It presents a comprehensive, end-to-end methodology, followed by detailed descriptions of the data acquisition system, on-bridge and off-bridge models, specifically the Virtual Work Theorem and MR-ILs. The chapter concludes with a presentation of the Cross-Entropy (CE) optimisation method used to infer pier stiffness.

The methodology proposed in this chapter uses bridge drive-by signals to detect foundation scour. Previous research by Fitzgerald et al. (2019) has numerically detected scour using the bogie accelerations of a train crossing a bridge multiple times. Scour has been simulated by reducing the vertical stiffness of the pier, and the approach has been verified in the laboratory. Novel contributions of this thesis are as follows:

a) Verification of an approach proposed by some of the authors (Fitzgerald et al., 2019; Prendergast et al., 2016) with a combination of synthetically generated and field measurements.

b) Use of onboard displacement-based data analysis for scour detection.

The acquisition of drive-by is facilitated by an innovative sensor unit that takes measurements from in-service vehicles rather than traditional track geometry cars.

3.1 Methodology

The methodology employed in this chapter is outlined in Figure 3.1. The proposed framework necessitates the implementation of both before and after flood monitoring, encompassing healthy and (potentially) scoured stage drive-by measurements for scour detection. To demonstrate the methodology's capacity to identify scour, a case study has been conducted. However, for practical reasons, only healthy-stage measurements could be obtained. Hence, scoured stage data has been synthetically generated using a model that simulates vehicle-bridge dynamic interaction (Cantero, 2022). Foundation scour is modelled by reducing the vertical pier stiffness. The difference between the pier stiffnesses of the scoured and healthy stages of the bridge indicates scour. The best-fit healthy and scoured pier stiffness values are found with a CE optimisation algorithm that minimises the sum of squared differences between the

measured / synthetically generated track profile elevations and the theoretical track profile elevations that include static bridge displacements computed with the Finite Element (FE) numerical model. The FE model calculates the moving reference static displacement response under the moving vehicle, a concept similar to the apparent profile experienced by the vehicle. The MR-ILs, which is defined as the deflection per unit load at a moving reference point, is derived from the Theorem of Virtual Work. Bridge piers are represented by springs, and foundation scour is represented as a reduction of vertical spring stiffness. The track profile includes the rail irregularities extracted from field measurements and all deflections under the weight of the train. An off-bridge model is also developed to calibrate the train model, i.e., track on ballast. In contrast to previous approaches (Fitzgerald et al., 2019; Prendergast et al., 2016), the data used in the case study of this paper are displacement signals, which are more sensitive to damage detection as they are measured directly. The field data used is acquired by a mobile mapping system called the Rail Infrastructure Alignment Acquisition system (RILA), which is mounted on a train passing a railway bridge in the UK. The RILA unit can be attached to the front or back of the train in minutes and collects data of the track and the railway corridor while the vehicle is in motion without causing service disruptions. This feature facilitates the installation of measurement devices on a wide array of vehicles that are not specialist track recording vehicles. Consequently, it opens up the possibility of more frequent measurements, which may be used to complement or potentially even replace those obtained from track recording vehicles. This facilitates the mounting of a warning system based on frequent measurements, thereby enhancing operational efficiency and safety.

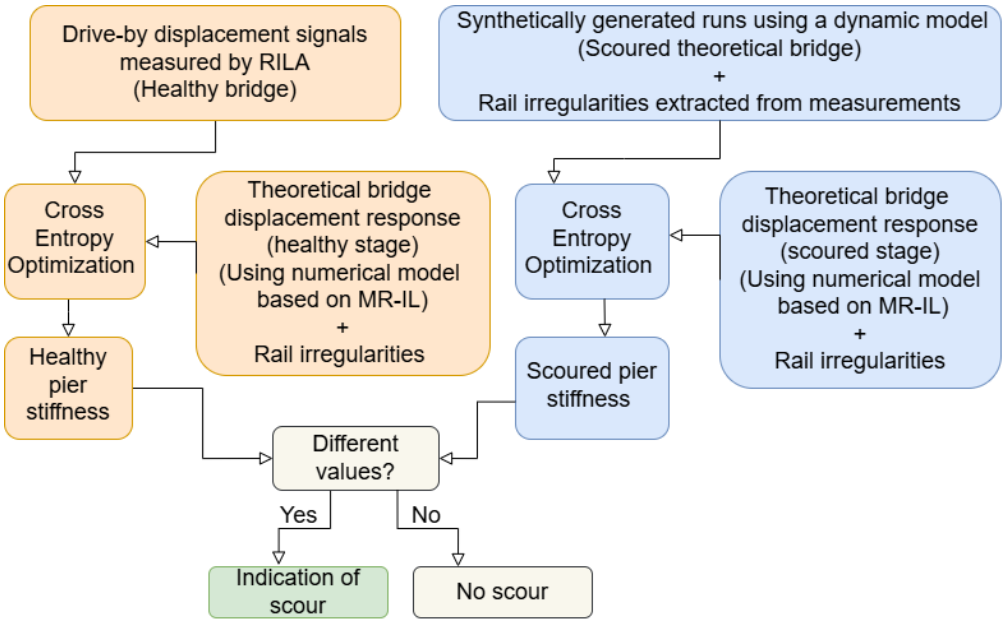


Figure 3.1. Flowchart of scour detection methodology (reprinted from Tola et al., 2025).

3.1 Numerical Models

3.1.1 Moving Reference Influence Lines

Influence lines (ILs) give the value of a response quantity (support reaction, shear, bending moment, etc.) at a specified point as a function of the position of a unit action moving along the structure. They are used both to compute effects of moving loads and to locate the load position that maximises a chosen effect (Ghali et al., 2017).

3.1.1.1 Virtual work theorem and force method

For linear elastic systems under a kinematically admissible virtual deformation, the virtual work done by the external forces F_1, F_2, F_3, ... F_n equals the virtual work done by internal stresses bending moment *(M),* shear force *(V),* axial force *(N),* and torsion (*T*) at a cross-section (Ghali et al., 2017), as expressed in Eq. (3.1):

$$\left(\frac{1}{2}\right)\sum_{\{i=1\}}^{n} F_i D_i = \left(\frac{1}{2}\right)\int \frac{N_{dx}^2 dl}{Ea} + \left(\frac{1}{2}\right)\int \frac{M^2 dl}{EI} + \left(\frac{1}{2}\right)\int \frac{V^2 dl}{G a_r} + \left(\frac{1}{2}\right)\int \frac{T^2 dl}{GJ} \tag{3.1}$$

where D_i denotes the displacements at the location and in the direction of the external forces acting on the system. To compute the response of statically indeterminate structures under applied loads and/or imposed deformations, the force (flexibility) method could be used. The method transforms the original indeterminate structure into an equivalent statically determinate system by removing as many constraints as the degree of indeterminacy. Then it calculates the additional forces required to restore the system's original compatibility conditions. The procedure is as follows (Ghali et al., 2017).

i. Determining the degree of static indeterminacy *n*.

ii. Creating an equivalent determinate system by releasing *n* constraints (e.g., removing selected supports or introducing internal hinges).

iii. Introducing unknown redundant F_{Ri} corresponding to each released constraint *j* and oriented consistently with the release definition.

iv. Computing the displacement at each released coordinate using the unit-load form of virtual work (Eq. 3.1): Applying a unit action at release *j* to generate the internal forces due to a unit virtual load (N_u, M_u, V_u, T_u) and evaluating the virtual-work integral by combining these with the internal force due to real external load (N_{ax}, *M, V, T*) to obtain the displacement (flexibility coefficient) at *i* and then repeating for all releases to assemble the flexibility (displacement) coefficients.

v. Imposing compatibility to solve for R_i: enforce that the net translation/rotation at every release is zero; the resulting set of R_i restores the original geometry and boundary conditions.

3.1.1.2 Betti's theorem and calculation of influence lines

Betti's theorem states that the work of the forces of system *F* through the displacements of a system *Q* is equal to the work of the forces of the system *Q* through the displacements of the system *F* (Ghali et al., 2017), as provided in Eq. (3.2):

$$\sum_{i=1}^{n} F_i D_{iQ} = \sum_{i=n+1}^{m} Q_i D_{iF} \tag{3.2}$$

where D_{1Q}, D_{2Q}, ..., D_{mQ} are displacements caused by forces of the system Q_{n+1}, Q_{n+2}, ... , Q_m.

Bending moment ILs

For a support bending-moment influence line, the rotational restraint at that support is released, and a unit rotation ($w = 1$ rad) is imposed. The elastic curve of this released structure under the unit rotation is the IL for the corresponding support moment. This theory is supported by the arguments regarding reciprocity and virtual work presented by Betti in the work of Ghali et al. (2017). The practical procedure to construct bending-moment influence lines using the force method and virtual work is as follows:

i. Creating the primary system: The restraint associated with the target quantity (e.g., a support rotation for a support moment, a vertical reaction for a support shear) is released to obtain a statically determinate structure. Then, a virtual unit load is applied at that coordinate (unit rotation or unit displacement in the positive direction of the target). The internal forces / moments N_u, M_u, V_u, T_u generated due to the virtual unit load applied at *x*.

ii. Forming ordinates by virtual work: To obtain the IL ordinate at any location *x* in the direction of a unit load applied at x, the unit-load form of the virtual-work relation (Ghali et al., 2017) in Eq. (3.3) is used:

$$z(x) = \int \left[\frac{N_u N_{ax}}{EA} + \frac{M_u M}{EI} + \frac{V_u V}{G a_r} + \frac{T_u T}{GJ}\right] ds \tag{3.3}$$

iii. For a support reaction, the corresponding translation is released, and a unit translation in its positive direction is applied. For a support moment reaction, the rotation is released, and unit rotation is applied in its positive direction. The sign of the IL follows the sign convention used for the unit action.

Scope note. In this work, only vertical displacement influence lines, z_n, are used for a multi-span simply supported bridge with elastic supports; influence lines for internal actions (shear, moment, and torsion) and for fixed-ended members are outside the scope and are therefore omitted.

3.1.2 Bridge Models for Theoretical Track Profile

3.1.2.1 Modelling assumptions and pier stiffness calculation

This section introduces static bridge models used to calculate the bridge displacement component of the theoretical track profile. The bridge models used in the CE optimisation algorithm are employed to calculate the stiffness of the pier. These models calculate the theoretical bridge displacement responses (see Figure 3.1), namely, the static MR-IL displacement response of the bridge under the moving vehicle. The calculation details are provided in Section 3.2.2.2. The model, developed in MATLAB, represents a 7-span simply supported bridge. The bridge is modelled as a Euler-Bernoulli beam, with its properties listed in

Table 3.1. The bridge is traversed by a 6-axle car, specifically a Class 66/0 Co-Co Diesel, with equal axle loads of 181.4 kN. This is the same vehicle on which the RILA measurements were taken. Figure 3.2 shows two bridge models that consider the positioning of the data acquisition system, the RILA, on the train. The direction of the train is the same for both models:

- RILA mounted in front of the train, 2 m ahead of the first axle (Figure 3.2 (a)).
- RILA mounted at the back of the train, 2 m behind the last axle (Figure 3.2 (b)).

The bridge piers are represented by springs with vertical stiffness, k_0. The soil model assumes linear elastic deformations under the given load and does not explicitly consider strains. Eq. (3.4), from the formula FEMA 356 (2000), is used to calculate the theoretical stiffness value of the healthy bridge embedded foundation:

$$k_0 = \beta_z \cdot \left[\frac{G_B}{1-v}\right] \cdot \left[1.55 \cdot \left(\frac{L_F}{B_F}\right)^{0.75} + 0.8\right] \tag{3.4}$$

$$\beta_z = \left[1 + \left(\frac{1}{21}\right) \cdot \left(\frac{D}{B_F}\right) \cdot \left(2 + 2.6 \cdot \frac{B_F}{L_F}\right)\right) \cdot \left(1 + 0.32 \cdot \left(\frac{d_f * (B_F + L_F)}{(B_F \cdot L_F)}\right)^{\left(\frac{2}{3}\right)}\right] \tag{3.5}$$

where β_z in Eq. (3.5) is the correction factor for embedment, d_f is the height of effective sidewall contact, $D_f$is the embedment depth of the foundation, G_B is the soil shear modulus, v is the Poisson's ratio of the soil, L_F is the foundation length (m), and B_F is its width (m). β_z is computed as 1.34, v is assumed to be 0.1, and G_B is computed as 45,455 kN/m^2. For the 11.2 meters × 1.8 meters rectangular

foundation of the 2-track bridge, the healthy pier stiffness, k_0 is computed as 467,966 kN/m. Plan and elevation of the pier foundation are seen in Figure 3.3. To model foundation scour, the calculated pier stiffness value is reduced. For example, to model 10% scour at a pier, vertical spring stiffness is reduced by 10%. The vertical stiffness, k_0, is reduced as a result of the following mechanisms in the presence of pier scour: The foundation dimensions set forth in Eq. (3.4) are applicable as soon as the soil has surrounded said lengths. The loss of surrounding soil caused by scour reduces the area available for load transfer. The remaining soil's capacity to resist vertical loading is significantly diminished, yielding increased vertical displacement under the initial loads (before scour). Scour leads to a reduction in the lateral load-resisting capacity of the soil, as well. Therefore, a reduction in shear modulus is directly proportional to a decrease in the vertical stiffness, k_0, as illustrated in Eq. (3.4). The minimum scour percentage considered in the scope of this research is 5%. According to Eq. (3.4), 5% loss of vertical stiffness corresponds to a loss of area of 6.6 m^2 around the foundation.

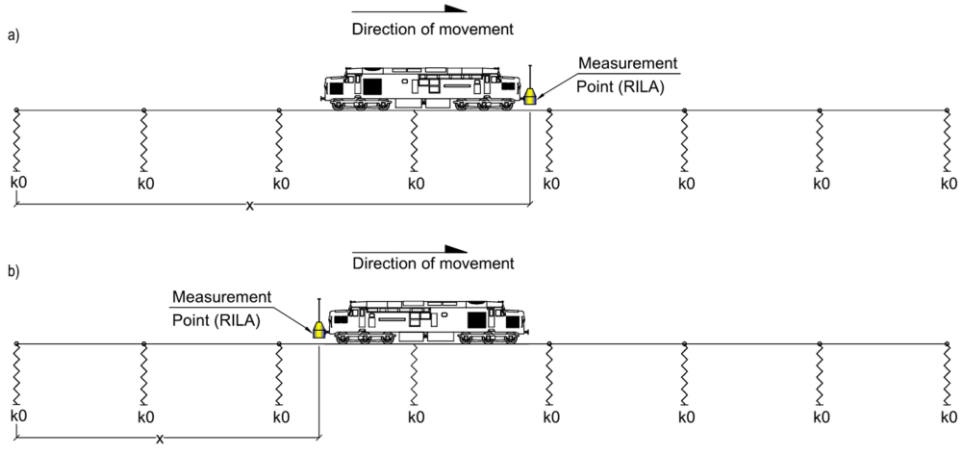


Figure 3.2. Bridge models: (a) RILA at the front; and (b) RILA at the back (reprinted from Tola et al., 2025).

Table 3.1. Properties of the bridge models (reprinted from Tola et al., 2025).

Span lengths (m)	E (kN/m^2)	Cross section area (m^2)	EI (kNm^2)	Pier Spring Stiffness (kN/m)	Respective axle spacings (m)
The first and last spans 13.20 Interior spans: 14.00	$34.30 \cdot 10^6$	8.0	$30.9 \cdot 10^6$	467,966	2.00 / 2.20 / 9.00 / 2.20 / 2.00

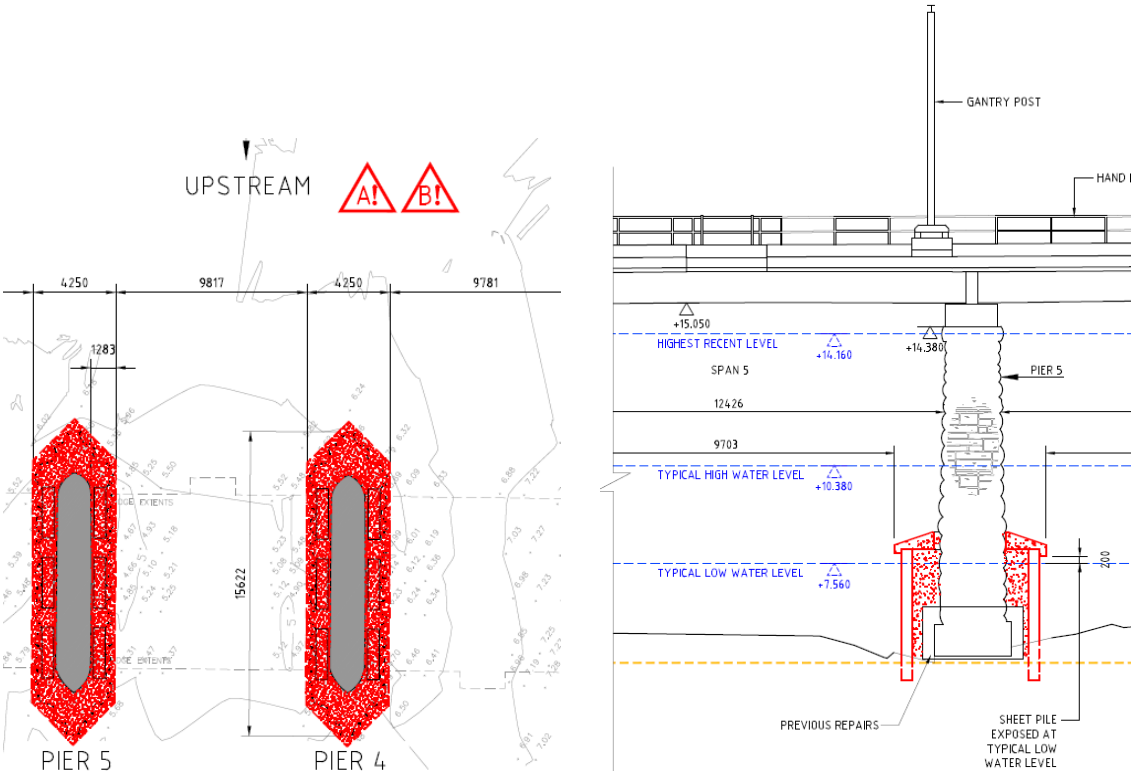
Note: E = Young's Modulus, I = second moment of inertia.

3.1.2.2 Calculation of bridge displacement response

The bridge model, which calculates the MR-ILs, has been developed from a single-axle, single-span case to a multiple-span, simply supported, multiple-axle case that represents the case study bridge conditions. The displacement response at point x is denoted as z_x. This parameter is computed due to the

moving load F at x-d, where d = 2m is the distance between the measurement point and the closest axle, as shown in Figure 3.4 (a) There are two components of z_x, as expressed in Eq. (3.6): the first is due to the bending of the beam, i.e., z_{x1}, and the second is the displacement due to the support deflection, denoted as z_{x2}.

$$z_x = z_{x1} + z_{x2} \tag{3.6}$$



(a) (b)

Figure 3.3. Repaired foundation: (a) Plan (Network Rail, 2015c); (b) Elevation (Network Rail, 2015d).

Calculating z_{x1}. Using the unit-load theorem, the moving-reference influence line (MR-IL) for vertical displacement at the observation point x is formed by applying a unit vertical action at x in the virtual system while the real axle load F traverses the span at $x-d$ (Figure 3.4). The bending contribution is the convolution of the virtual and real bending-moment fields, giving Eq. (3.7):

$$z_{x1} = \int_0^L \frac{M_u(x)M(x)}{EI}\,dx \tag{3.7}$$

where $M_u(x)$ is the bending moment diagram due to the unit action at x (virtual system), and $M(x)$ is the moment under the moving load F at x-d.

Calculating z_{x2}. Deflections of supports that are demonstrated in Figure 3.5 are expressed in Eq. (3.8) and Eq. (3.9):

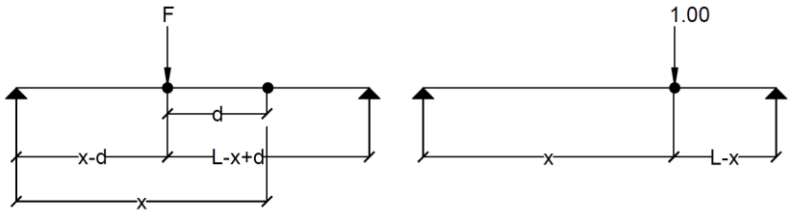
$$z_{RL} = \frac{R_L}{k_L} = \frac{l-x+d}{k_L} \cdot L \tag{3.8}$$

$$z_{RR} = \frac{R_R}{k_R} = \frac{x-d}{k_R} \cdot L \tag{3.9}$$

Assuming that $z_{RR} < z_{RL}$, the support deflection component of displacement at x is as in Eq. (3.10):

$$z_{X2} = z_{RR} + \frac{L-x}{L}(z_{RL} - z_{RR}) \tag{3.10}$$

where the loading point is x-d away from the left support. The details of the calculation of the MR-ILs of the case study bridge, along with the MATLAB code, are provided in Annex A. This appendix also includes the verification of the bridge model with SAP2000.



(a) (b)

Figure 3.4. (a) Real system; (b) Virtual system.

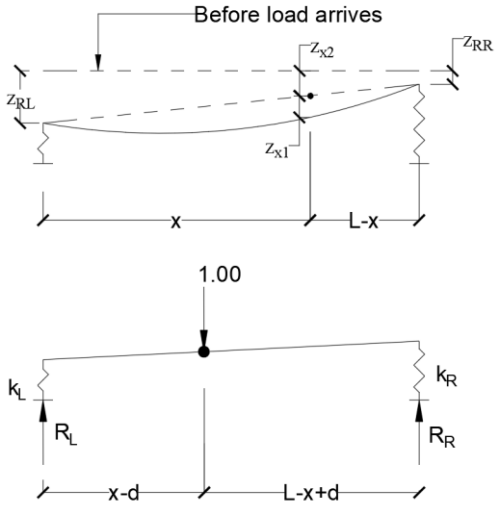


Figure 3.5. Support Deflection component, z_{x2}.

3.1.3 Off-Bridge Model

3.1.3.1 Model parameters

This section introduces an off-bridge model for calibrating the train model, i.e., track on ballast. The off-bridge model, i.e., the continuous 2-rail track on the ground, is modelled as a beam on Winkler springs (Figure 3.6). As before, the displacement at location x consists of two displacement components, one

due to span bending and a second due to support deflection. The bending component is again determined using the Force Method of Analysis (Ghali et al., 2017), while the support deflection component is calculated based on the properties of the supports. The details of the calculation steps are provided in Section 3.2.3.2. Sleepers support the track at a spacing of 0.5 meters, denoted with s. The spring stiffness of each sleeper support is determined by the stiffnesses of the rail pad, ballast, and subgrade (soil) layers. To account for all layers beneath the rail, the equivalent vertical stiffness per sleeper, k_R, is taken as the series combination of the rail-pad (k_p), ballast (k_{ba}), and subgrade (k_{sg}) springs, as in Eq. (3.11):

$$\frac{1}{k_R} = \frac{1}{k_p} + \frac{1}{k_{ba}} + \frac{1}{k_{sg}} \tag{3.11}$$

This results in k_R = 28,131 kN/m. This modelling approach follows Choi (2014). The track properties are adapted from Quirke (2017), as provided in Table 3.2.

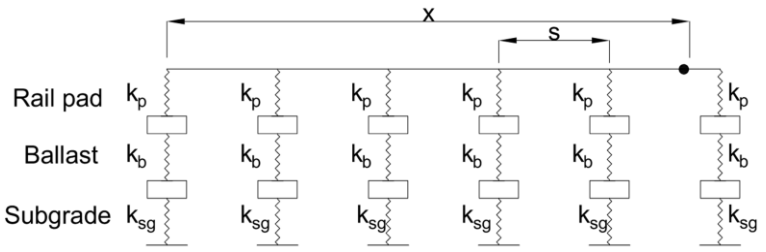


Figure 3.6. Track model (reprinted from Tola et al., 2025).

Table 3.2. Track parameters, adapted from Quirke (2017).

Property	Unit	Value
Elastic modulus of rail	N/m^2	2.059×10^{11}
Rail cross-sectional area	m^2	7.69×10^{-3}
Rail second moment of inertia	m^4	3.217×10^{-5}
Rail mass per unit length	kg/m	60.64
Rail pad stiffness	N/m	6.5×10^{7}
Rail pad damping	Ns/m	7.5×10^{4}
Sleeper mass (half)	kg	125.5
Sleeper spacing	m	0.50
Ballast stiffness	N/m	137.75×10^{6}
Ballast damping	Ns/m	5.88×10^{4}
Ballast mass	kg	531.4
Subgrade stiffness	N/m	77.5×10^{6}
Subgrade damping	Ns/m	3.115×10^{4}

3.1.3.2 Computing displacement response of the track on soil

The track is modelled as 40m long on sleepers. This is a statically indeterminate system. To calculate the MR-ILs of the track for the moving load, the following steps are implemented:

Step 1: Determinate reference system. Releasing n_{sup}-2 internal bending restraints to match the static indeterminacy, i.e., releasing the n_{sup}-2 hinges, where n_{sup} is the number of supports. The resulting system is determinate.

Step 2: Displacement vector of released points (real system). Under the real loading (axle *F* at *x-d*), computing the displacement vector *{D}* of the released points, and z_x^{step2}.

Step 3: Flexibility matrix (virtual system). Applying unit loads one-by-one at the released points to form the flexibility matrix *[f]*, where f_{ij} implies displacement at *i*, due to unit load at *j*.

Step 4: Compatibility to recover redundant. Enforcing closure of the releases through Eq. (3.12):

$$[f]\{F_R\} + \{D\} = 0 \;\Rightarrow\; \{F_R\} = -[f]^{-1}\{D\} \tag{3.12}$$

where $\{F_R\}$ is the vector of redundant hinge moments to be reintroduced.

Step 5: Response at the measurement point. The displacement at *x* combines the contribution from the real system with releases and the contributions of the unit virtual actions scaled by the recovered redundant, as expressed in Eq. (3.13):

$$z_x = z_x^{(\text{step 2})} + \sum_{j=1}^{\mathrm{n}_{sup}-2} (z_x)^{(j)}\, F_{Rj} \tag{3.13}$$

where z_x^j is the displacement at *x* produced by the j^{th} unit redundant in the virtual system, i.e., z_x^{step3}, F_{Rj} is the recovered redundant from Step 4. The support-deflection contribution z_{x2} is computed from vertical spring settlements and linearly interpolated within each span. Further calculation details, MATLAB code, and its verification with SAP2000 are provided in Annex B.

3.1.3.3 Track model results

The simulated moving reference displacement response of the track due to six axles is shown in Figure 3.7. The peaks follow the frequency of the sleeper spacing, i.e., 0.5 meters. A track simulation with a zone of reduced stiffness is also created to simulate different ground conditions. A 5-meter area with poor soil conditions is considered, as shown in Figure 3.8(b). A linear reduction was assumed, and the spring stiffnesses in the poor soil zone were reduced by a maximum of 30%. The maximum MR-IL displacement response increased by 123% (Figure 3.8(a)) compared to the initial stage, as shown in Figure 3.7.

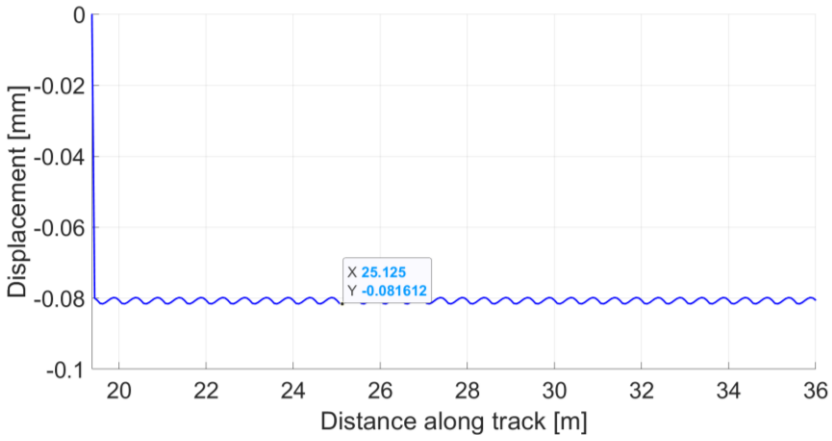
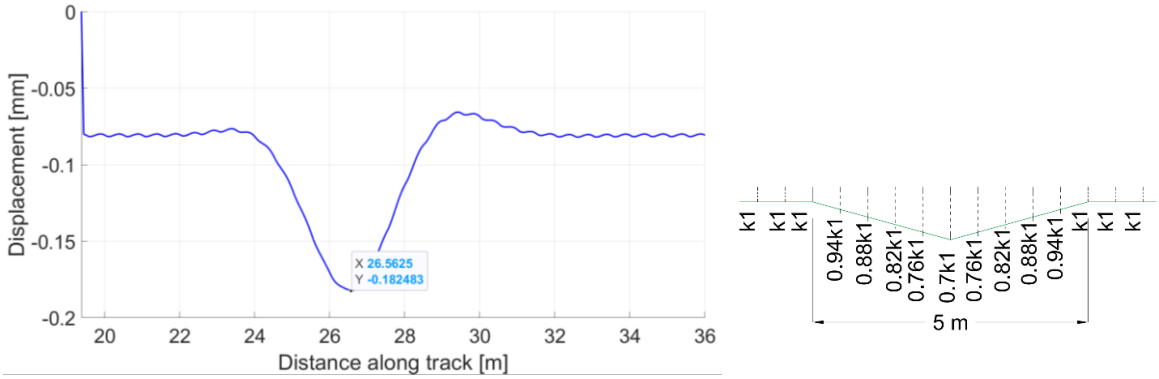


Figure 3.7. Displacement response of the track on the soil.



(a) (b)

Figure 3.8. (a) Displacement response of the track with poor soil; (b) Reduced stiffness zone in the soil.

3.2 Data Acquisition System

Track profile is typically measured using track geometry cars. A significant difference between these track recording vehicles and the data acquisition system of this research (RILA) is the distance between the nearest axle and the sensor location. The measurement taken from a position distant from the axle is designated as 'unloaded' (Wang et al., 2021). Figure 3.9 (Fugro, 2020) demonstrates the non-zero distance between the RILA and the wheel. The RILA unit is composed of the components shown in Figure 3.10 by Network Rail 2022):

- Inertial Measurement Unit (IMU)
- Two rail scanners (laser vision systems)
- 360^0 LiDAR (Light Detection and Ranging) scanner
- Three 165^0 field of vision video cameras

- GNSS (Navigation Satellite System) antenna

The following steps are applied to calculate relative track geometry (Network Rail, 2022):

- The Z_{RILA} curve in Figure 3.11 is measured with the IMU.
- The difference between the IMU measurement level and the rail, i.e., Δz in Figure 3.11, is supplied by the rail scanner (Figure 3.10) measurements.
- The partial outlines are found from the scanners. The rail profile in Figure 3.12 is calculated as, Z_{RILA}-Δz (Figure 3.11).
- The absolute track geometry parameters T_1, T_2, R_1, R_2, P_1, P_2, T_c, R_c in Figure 3.13 are derived from the rail profiles.
- Based on the centreline, the relative track geometry is obtained by filtering the absolute track geometry parameters with a 4th-order Butterworth filter in accordance with EN 13848-1 (Wang et al. 2021, European Committee for Standardization, 2019).



Figure 3.9. RILA unit mounting on a passenger train (reprinted from Tola et al., 2025).

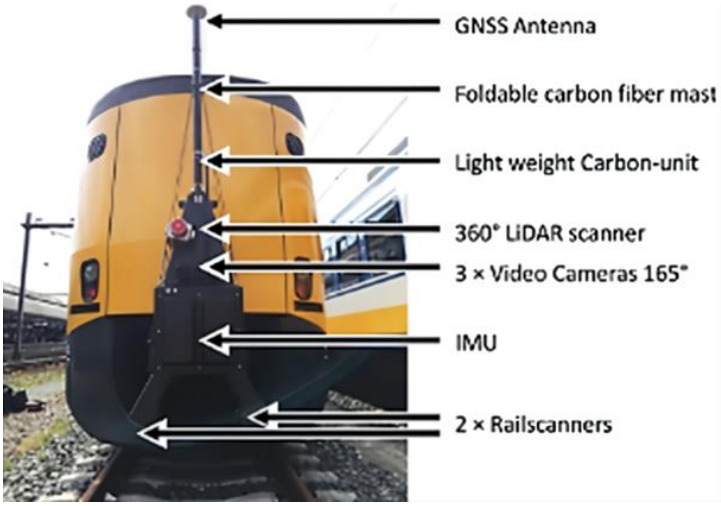


Figure 3.10. RILA unit sensors (reprinted from Tola et al., 2025).

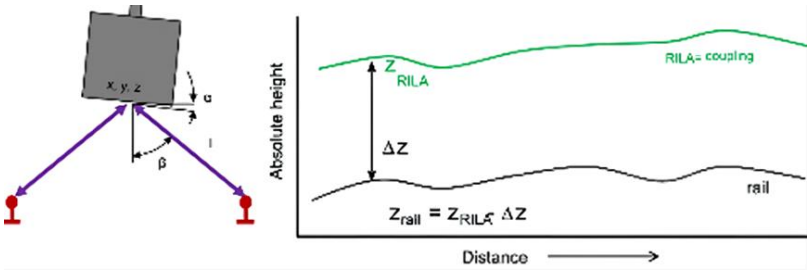


Figure 3.11. Z_{RILA} curve and Δz distance (reprinted from Tola et al., 2025).

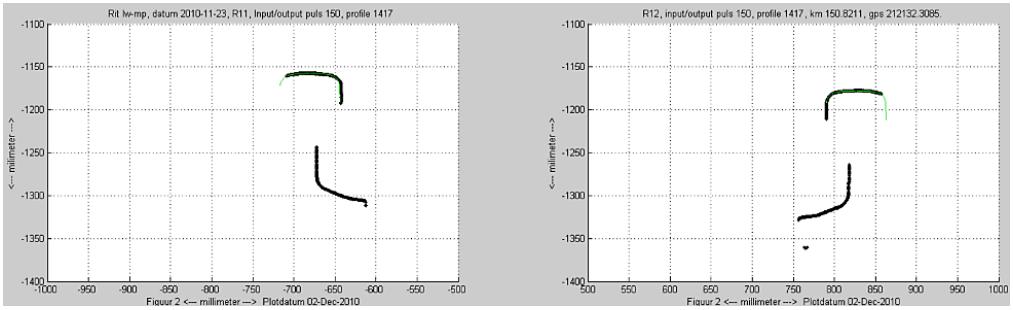


Figure 3.12. Rail profiles (reproduced with permission from Network Rail, 2022).

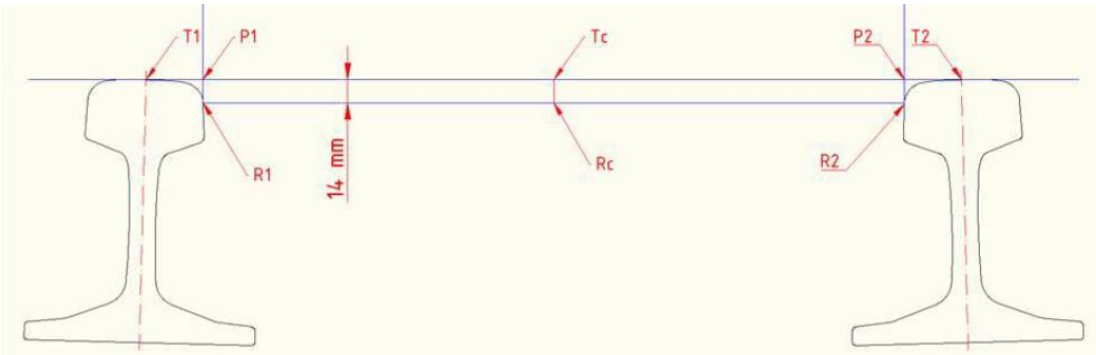


Figure 3.13. Absolute track parameters (reproduced with permission from Network Rail, 2022).

3.3 Cross-Entropy Algorithm to Detect Pier Stiffness

The CE optimisation algorithm by Botev et al. (2013) is used to determine pier stiffness by minimising the sum of squared differences between the measured and theoretical track profiles at every measurement point for all crossings (runs). Consequently, the computational time is elevated due to the density of the measurements. While simpler local optimisation algorithms may be sufficient, CE is adopted due to its strong robustness and because there is no pressure on computation time. The objective function is provided in Eq. (3.14).

$$O(k) = \sum_{j=1}^{J} \sum_{i=1}^{N} \left(\delta_{ij}^{Mea} - \delta_{ij}^{Sta}\right)^2 \tag{3.14}$$

where k is the pier stiffness, J is the number of runs, and N is the number of model nodes, δMea is the measured track profile, and δ^{Sta} is the superposition of the rail irregularities and bridge displacements computed using the numerical model explained in Section 3.1.2. Further details regarding the optimisation process are provided in the case study (see Section 3.4). The optimisation is performed separately for scoured and healthy stages (see Figure 3.1). The difference in inferred pier stiffness between these two stages serves as an indicator of scour.

3.4 Case Study

3.4.1 Data Characteristics and Monitored Bridge

The monitoring data used in this case study pertains to the Eden Viaduct Main Line railway bridge (No WCM1 5) in the UK, as shown in Figure 3.14 (Network Rail (2015b)), and Figure 3.15 (Network Rail (2020)). The bridge carries two rail lines rated for speeds of up to approximately 160 km/h. The rails are continuously welded and rest on concrete sleepers (Network Rail, 2020). The bridge comprises seven spans of simply supported beams with a continuous deck. Six masonry piers and two abutments support the spans. The bridge superstructure comprises eight prestressed concrete girders and a cast-in-situ reinforced concrete deck. Structural drawings of the bridge are provided in Annex A. Between 2007 and 2010, maintenance was carried out to repair scour holes around the foundation of Pier 4 (Network Rail (2015b)). An underwater inspection revealed that the foundation layers were protruding and that the shale around the piers, which was covered by silt, had been washed out (Network Rail (2015b)). From July to October 2015, the bridge underwent repairs due to scour. Concrete backfill and permanent sheet piling were installed to protect the piers and foundation from scour, as shown in Figure 3.16 (Network Rail (2015b)). The train crossings from which RILA measurements were obtained occurred in the period after the bridge was repaired, 2016-2019.



Figure 3.14. Case study bridge elevation view (reprinted from Tola et al., 2025).



Figure 3.15. Google Earth view of Eden viaduct main line railway bridges (Network Rail, 2022).



Figure 3.16. Case study bridge foundation repair works (reprinted from Tola et al., 2025).

3.4.2 Theoretical Bridge Displacement Responses

This section presents the simulated, i.e., theoretical displacement response of the bridge. The healthy and scoured stage analysis results are obtained using the numerical bridge modes described in Section 3.2.2.

3.4.2.1 Healthy stage simulated displacement response

Figure 3.17 displays the bridge's simulated moving-reference displacement responses when RILA is positioned in the front and at the back of the train. These simulations have been generated using the static bridge models outlined in Section 3.2.2. When RILA is in front of the train (dashed curve), a non-periodic component is observed at the beginning – deflection increases as axles enter the bridge. When

RILA is positioned at the back of the train (solid curve), the non-periodic part is observed at the end as the axles leave the bridge. These non-periodic intervals occur in the calculation because the bridge models do not include the off-bridge parts of the track, i.e., the sections before and after the bridge are assumed to be infinitely stiff. This would not occur in real life but has no bearing on the behaviour at the scoured pier. The peaks in both graphs correspond to the locations of the piers. The curves in the figure represent the displacements along the bridge when the measurement point is located 2 meters in front of the first axle. When the first axle and the measurement point are not at the same span, the measured displacement caused by the axles is comprised solely of the support deflection component, with no span bending contribution. This results in the formation of kinks, i.e., discontinuities of slope, at the peaks of the curves. In both curves, the troughs occur when the closest axle to the measurement point is situated in the middle of the span. The maximum displacement obtained is approximately 1.4 millimetres.

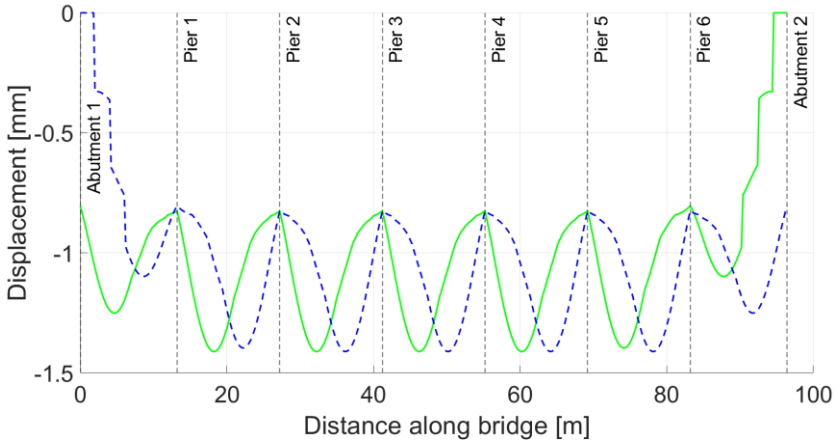


Figure 3.17. Bridge displacements: RILA in front of the train (dashed) and RILA at the back of the train (solid). (reprinted from Tola et al., 2025).

3.4.2.2 Scoured Stage Simulated Displacement Response

Three simulations have been generated with the RILA in front of the vehicle to determine the effect of scour through the static bridge model (Figure 3.18. Pier 3, the fourth support, is simulated to be scoured. The vertical stiffness value of Pier 3, k_4, is changed to $0.9k_0$ and $0.8k_0$, representing 10% and 20% scour, respectively, where k_0 is the stiffness value of the healthy pier. Similar to the dashed curve in Figure 3.17, the non-periodic part at the beginning of Figure 3.18 is caused by the front axles entering the bridge. The peaks correspond to the support locations, while the troughs occur approximately at each mid-span. The maximum displacement increases by 8.7% and 3.5% for the 20% and 10% scour cases, respectively, compared to the no-scour case depicted in the dashed curve in Figure 3.17. Figure 3.18 shows that pier scour significantly modifies the displacement response across roughly half of the two

adjacent spans. One of the findings of a previous study by Malekjafarian et al. (2020a) is useful in establishing a correlation between the corresponding contact area loss and vertical stiffness. The study found that a loss of 2.9 m^2 of soil-foundation contact area due to scour can be expected to result in a 30% loss of vertical stiffness and a reduction in shear modulus. The same contact area loss, namely 2.9 m^2, would result in a vertical stiffness loss of 32.7% according to the calculations conducted using Eq. 3.4.

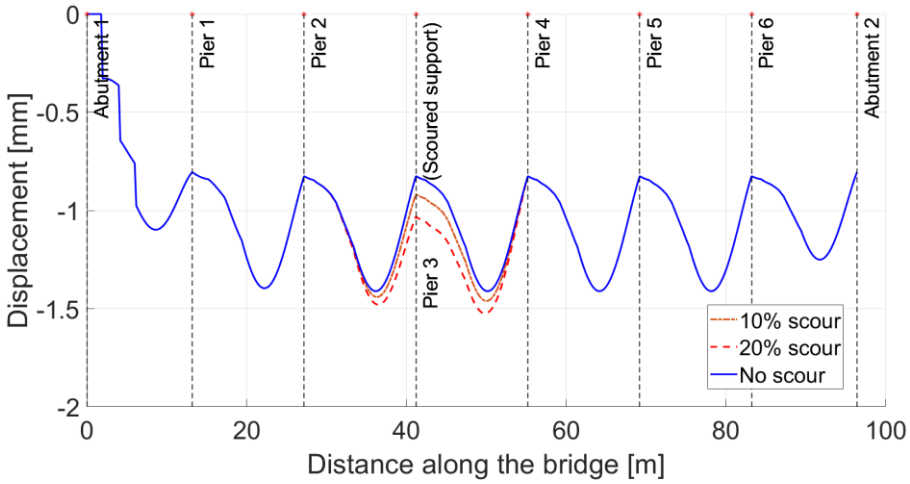


Figure 3.18. Bridge scour simulations using the static model (reprinted from Tola et al., 2025).

3.4.3 Track Profile Measurements

The RILA displacement measurements for the Eden Viaduct Main Line railway bridge comprise 16 runs that belong to the healthy, unscoured bridge condition. The train crossing the bridge is a 6-axle vehicle known as a Class 66 locomotive. The measurements were filtered within the wavelength range of D1, specifically $3m < \lambda \leq 25m$, according to EN 13848-1 (European Committee for Standardization 2019). This range is appropriate for scour detection along the case study bridge, as the displacement simulations indicate that the zone affected by the scour of a pier does not exceed 25 meters. It is important to note that multiple scoured piers are not within the scope of this study. Therefore, longer wavelength effects such as D2 ($25m < \lambda \leq 70m$) and D3 ($70m < \lambda \leq 150m$) are intentionally excluded from consideration. The wavelength filtering ($3m < \lambda \leq 25m$) and the Butterworth filtering to obtain relative track geometry, as described in Section 3.2, contribute to noise reduction in track geometry analysis through complementary mechanisms. The wavelength filtering focuses on relevant wavelengths pertinent to track deviations, whereas the Butterworth filter smooths the absolute geometry by attenuating high-frequency noise.

The bridge has two tracks, designated as Tracks 3 and 4. Figure 3.19 and Figure 3.20 show displacement measurements for Tracks 4 and 3, respectively, before, on, and after the bridge. A train on

track 3 begins travel at 0.45 km and terminates at 0.7 km, while a train on track 4 commences its journey at 2.05 km and concludes at 2.30 km. Accordingly, the direction of travel of the trains depicted in Figure 3.19 and Figure 3.20 is from left to right. Each track has eight runs. The zero-displacement level serves as smoothed reference for evaluating rail displacements. Negative measurements indicate downward displacements. Dots mark the measurements taken at pier locations. Dashed curves indicate measurements taken when RILA is in front of the train, while solid ones indicate measurements taken when RILA is at the back of the train. The measurements align well off-bridge but diverge on-bridge since for off-bridge measurements, the effect is largely unaffected by the train weight, which varies between runs, i.e., the profile is largely fixed, with little deformation in response to load. As the trains cross the bridge, RILA-trailing and RILA-leading runs split into two groups. It can therefore be concluded that the measurements contain information related to the bridge. The IMU of the RILA unit contributes to the calculation of relative track geometry measurements, with the acquired accelerations on bridge, assumed to be integrated twice to estimate elevation. Therefore, the RILA measurements provide the track profile perceived by the train at that specific point. This is the result of the superposition of bridge displacements $z(x)$ and any pre-existing rail irregularities $\xi(x)$, as shown in Eq. (3.15).

$$\delta(x) = z(x) + \xi(x) \tag{3.15}$$

The difference between the simulations presented in Sections 3.4.2 and 3.4.3, i.e., bridge displacements and RILA measurements is due to the pre-existing rail profile. The RILA runs have been categorized into four clusters based on the direction of travel and the orientation of the sensor unit.

- Cluster 3a: Track 3, RILA in front of the train (1 run)
- Cluster 3b: Track 3, RILA at the back of the train (7 runs)
- Cluster 4a: Track 4, RILA in front of the train (5 runs)
- Cluster 4b: Track 4, RILA at the back of the train (3 runs)

Figure 3.22 and Figure 3.23 display Cluster 3a and 3b runs while Figure 3.24 and Figure 3.25 shows Cluster 4a and 4b runs, respectively. As there were no reports of scour during data acquisition, it is expected that the measurements at the pier locations in each cluster would be very similar to each other. This is generally true for the runs when RILA is mounted in front of the train (Figure 3.22 and Figure 3.24). However, there are more significant deviations in the measurements at the piers for the runs when RILA is at the back (Figure 3.23 and Figure 3.25). For instance, Abutment 2 shows a deviation of up to

1 mm in Figure 3.25. The effects of travel velocities and train weights have been examined to identify the reasons for the disparities at pier locations.

A spatial Fast Fourier Transform (FFT) is applied to the runs sampled at $\Delta x = 0.2$m. Spatial FFT graph of two runs from each track is plotted in Figure 3.21. Spatial sampling rate is $F_s = 1/\Delta x = 5\text{m}^{-1}$, and the Nyquist frequency is 2.5 m^{-1}. For each run, the discrete spectrum is expressed as a single-sided amplitude as in Eq. (3.16):

$$P_1[k_t] = \frac{2 \cdot |Y[k_t]|}{N_s} \tag{3.16}$$

where $f[k_t] = k_t \cdot F_s / N_s$ is the spatial frequency in m^{-1}, N_s is the number of spatial samples on the bridge, and $Y[k_t]$ is the discrete FFT of $\delta(x)$ at bin k. Even though measurements were taken at various travel speeds (≈12–30 m/s), the displacement measurements at the fastest and slowest runs were not significantly different from the remainder. Likewise, the FFTs of the fastest and slowest runs do not yield different peak frequencies or amplitudes compared with other crossings. As seen in Table 3.3, across all runs, the dominant peak lay at 0.0724 m^{-1} for Track 3, and 0.0723 m^{-1} for Track 4, implying an average span length $L \approx 1/f \approx 13.8$m and thus recovering the bridge periodicity. Three Track 4 runs report $f = 0.0621$ m^{-1}, which is precisely one FFT bin, i.e., discrete slot below 0.072 m^{-1}, where a bin corresponds to $\Delta k = F_s/N_s = 0.01035$ m^{-1}. This one-bin shift is assessed to be a leakage effect rather than a physical change.

Figure 3.16 shows that approximately half of the adjacent-span measurements are affected when scour is present. In other words, scour at a pier lengthens the spans it supports to $L + L/2$ (approximately 21 m). Hence, the presence of the scour would correspond to a peak near $1/(L + L/2) = 1/21$ m ≈ 0.0476 m^{-1}. However, no such peak is observed in any run. This outcome supports the interpretation that the analysed train crossings correspond to the healthy stage of the bridge. The precise longitudinal locations of the piers on the tracks, in terms of kilometre values, could not be provided to the research team for practical reasons. Consequently, the central locations of the bridge piers have been calculated geometrically using the provided approximate edge coordinates of some piers, along with the longitudinal locations of the gantry posts and the distances between supports as indicated in the bridge structural drawings. The approximate coordinates, along with the discrepancies between the constructed and structural drawing locations, have resulted in approximate pier longitudinal locations. As a result, the discrepancies between the pier measurements of the cluster 3b and 4b runs are presumed to be attributable to inaccuracies in the pier longitudinal locations. To correct locations, the deviations of each pier measurement from the mean value of all measurements are calculated at each pier location.

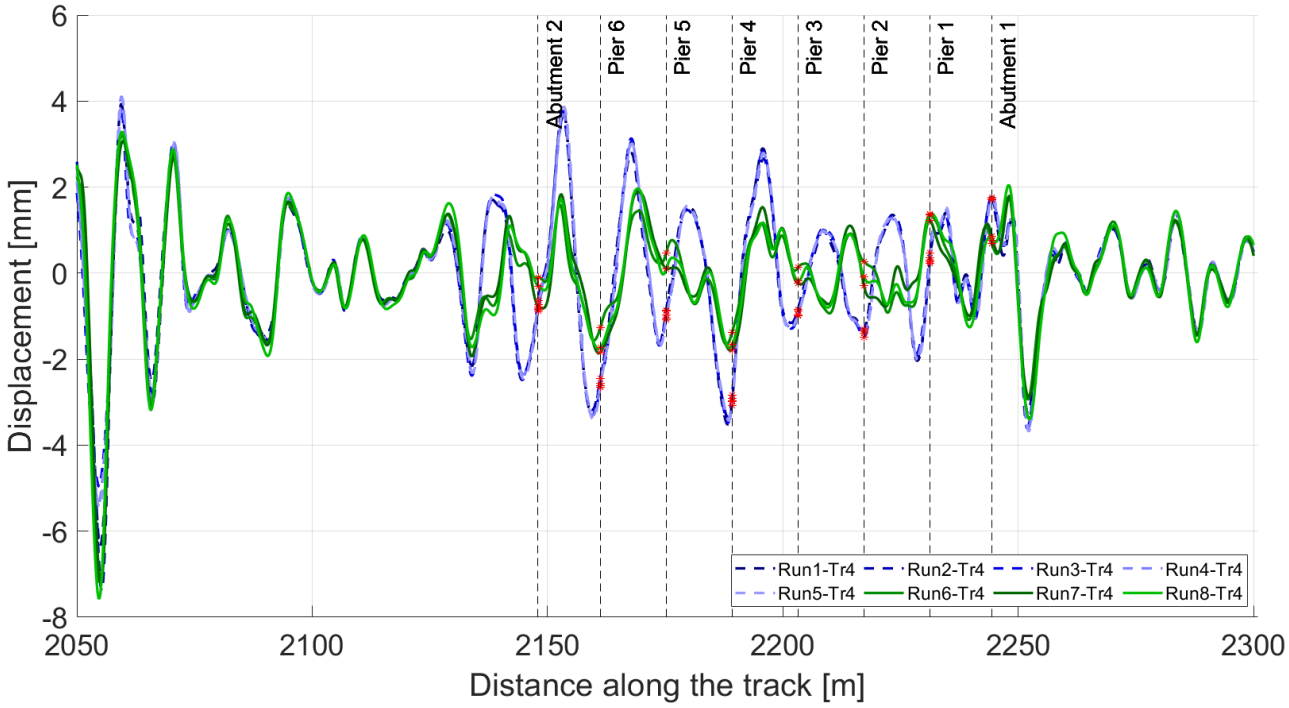


Figure 3.19. Eight RILA measurements: Track 4 (reprinted from Tola et al., 2025).

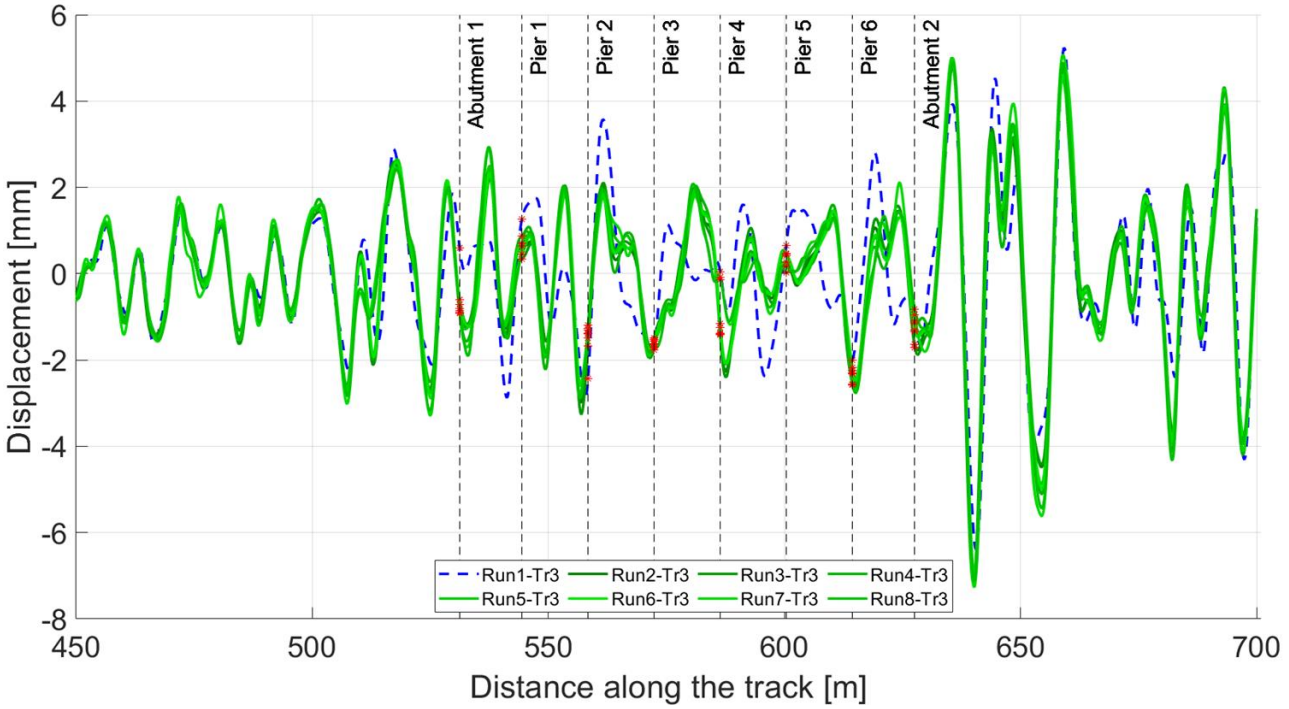
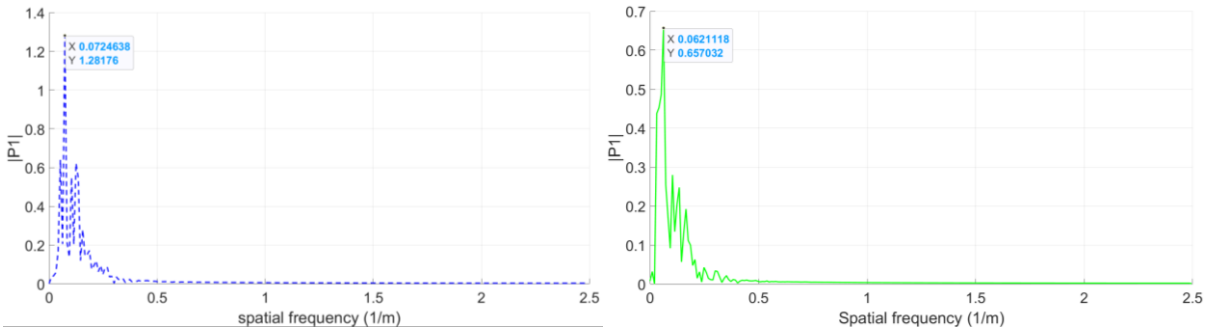


Figure 3.20. Eight RILA measurements: Track 3 (reprinted from Tola et al., 2025).



(a) (b)

Figure 3.21. Spatial FFT (single-sided amplitude): (a) Track 1 Run 1; (b) Track 4 Run 8.

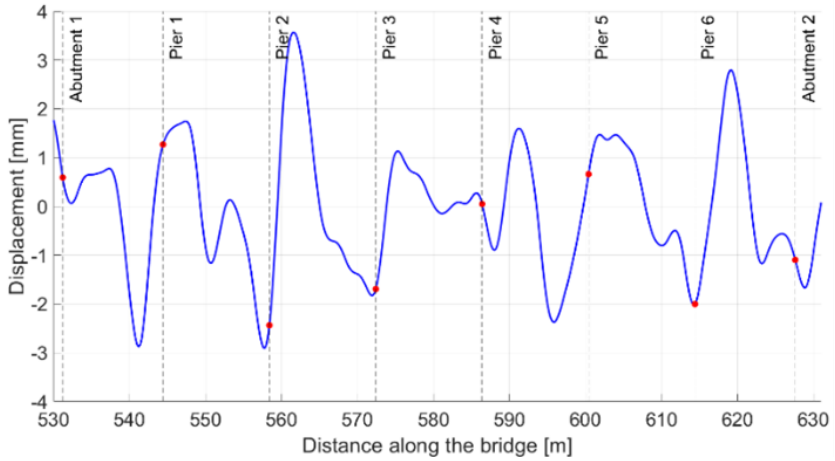


Figure 3.22. Track 3 runs (RILA measurements) Cluster 3a (single run) (reprinted from Tola et al., 2025).

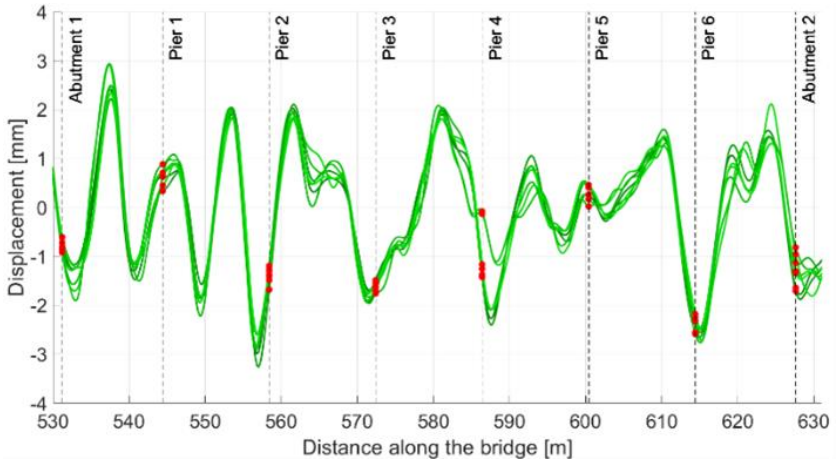


Figure 3.23. Track 3 runs (RILA measurements) Cluster 3b (seven runs) (reprinted from Tola et al., 2025).

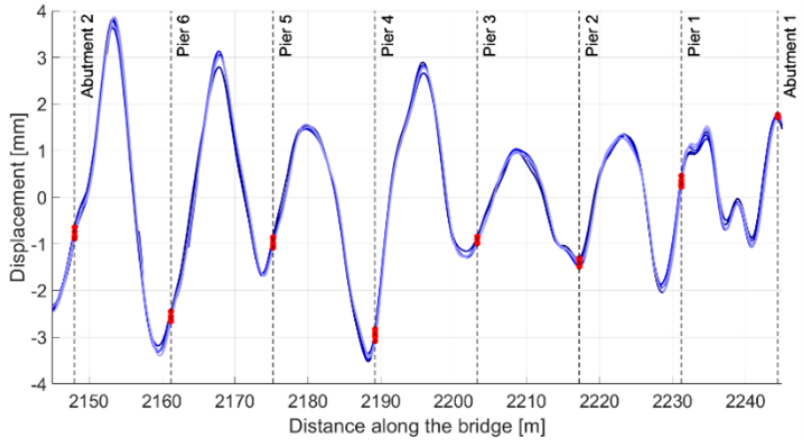


Figure 3.24. Track 4 runs (RILA measurements) Cluster 4a (five runs) (reprinted from Tola et al., 2025).

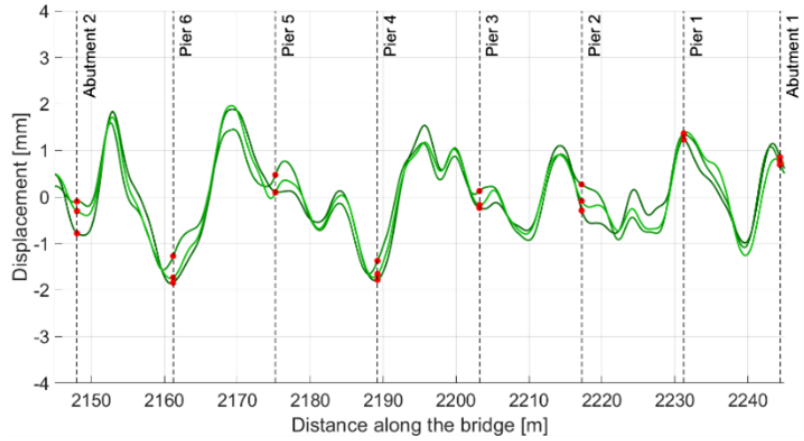


Figure 3.25. Track 4 runs (RILA measurements) Cluster 4b (three runs) (reprinted from Tola et al., 2025).

Table 3.3. FFT peaks of runs with speeds.

Track No	Run No	Run name	Peak f (1/m)	Peak |P1|	Average speed (m/s)
3	1	a20160510_1	0.072464	1.281762	21.43
3	2	a20160511_1	0.072464	0.844406	20.67
3	3	a20160513_1	0.072464	1.005046	20.95
3	4	a20161206_1	0.072464	0.806306	30.81
3	5	a20170403_1	0.072464	0.833722	28.38
3	6	a20170404_1	0.072464	0.800889	26.25
3	7	a20181105_1	0.072464	0.905169	24.33
3	8	a20181106_1	0.072464	0.891017	23.32
4	1	b20160510_1	0.072464	1.749900	12.91
4	2	b20160511_1	0.062112	0.545949	10.18
4	3	b20160513_1	0.062112	0.578491	23.74
4	4	a20161206_1	0.072464	1.699690	19.02
4	5	a20170403_1	0.072464	1.749108	19.16
4	6	a20170404_1	0.072464	1.740279	20.17
4	7	a20170404_2	0.072464	1.786000	19.84
4	8	a20171113_1	0.062112	0.657032	19.07

Specifically, pier locations are adjusted slightly from the calculated values using an objective function, the sum of squared differences between RILA measurements at the piers for each run. The CE method is used to perform optimisation separately for Clusters 3b and 4b. To limit the extent of the corrections, a penalty is incorporated into the objective function (Botev et al., 2013) that penalizes deviations from the original measured values. The total objective function is the sum of the squared differences of the deviations of the pier RILA measurements from the reference (the mean of all pier measurements) plus a penalty, as shown in Eq. (3.17).

$$O(\Delta x) = \left\{ \sum_{j=1}^{J} \sum_{m=1}^{M} [\delta(x_m^j + \Delta x_m^j) - \delta^j{}_{mean}(x_m^j + \Delta x_m^j)]^2 + p * \Delta x_m^{j\,2} \right\} \qquad (3.17)$$

where δ is displacement, M is the number of piers, J is the number of runs, x_m^j is the original pier location, Δx is the matrix of corrections with size $M \times J$, and p is the penalty function coefficient chosen by engineering judgment. The initial value for p is selected to ensure that the penalty term, represented by $p\Delta x_m^{j\,2}$, does not exceed roughly 20% of the magnitude of the final objective function. The value is then gradually decreased until the difference between the runs becomes minimal. Using the principles of CE (Botev et al., 2013), a set of correction values is randomly generated using Monte Carlo simulation and Normal distributions. An elite set of the best values is obtained, and a new mean and standard deviation are calculated using this elite set and used to generate the next generation of trial value sets.

Figure 3.26 and Figure 3.27 show Cluster 3b and 4b runs, with corrected pier measurements. Table 3.4 and Table 3.5 tabulate the magnitudes of corrections of Cluster 3b, and Cluster 4b runs, respectively. Negative corrections indicate the correction in the direction opposite to that of the train travel direction. The location-optimised version has a sum of squared deviations that is 82% lower than the original Cluster 3b measurements. Among all piers, the measurements at pier four in the original Cluster 3b runs are the furthest from each other. After the location corrections, the gap at Pier 4 decreased significantly. Similarly, the displacements of Cluster 4b runs at Abutment 2 and Piers 2, 3, 5 and 6, have been improved. The sum of squared deviations of the location-optimised version has improved by 52%. Although this value is less than the improvement of Cluster 3b, it should be remembered that the original Cluster 4b pier measurements were closer to each other than Cluster 3b runs originally were. It should be noted that if the measurements were taken during the scoured stage rather than the healthy stage, the assumption of location correction would still be reasonable. This is because the deflections per unit load over the piers would be considerably less than those in the spans, even if the piers deflected under the weight of the train. Following the correction of the longitudinal locations of the piers, the remaining discrepancy in measurements within the same track is up to 2 mm. This discrepancy is attributed to the heavier side of

the train being located at the opposite side where RILA is located. The minor mismatches of the runs within the same cluster could be attributed to the different travel speeds (Malekjafarian et al., 2019) and different weights.

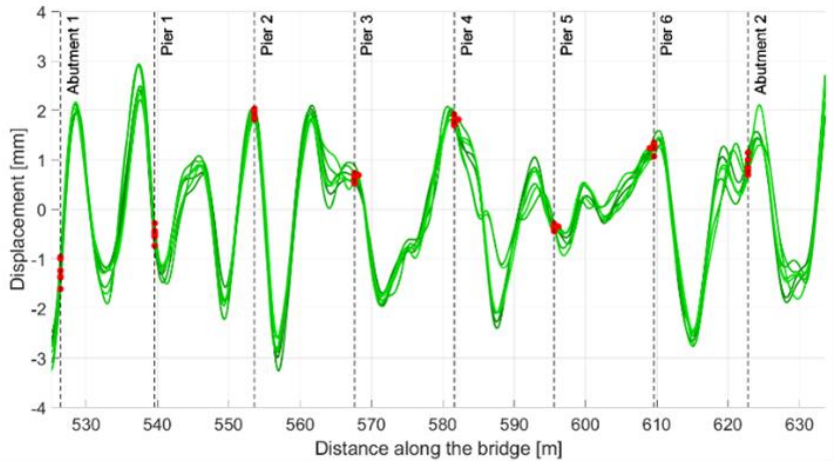


Figure 3.26. RILA rear-mounted runs (measurements) with corrected pier locations: Cluster 3b (re-printed from Tola et al., 2025).

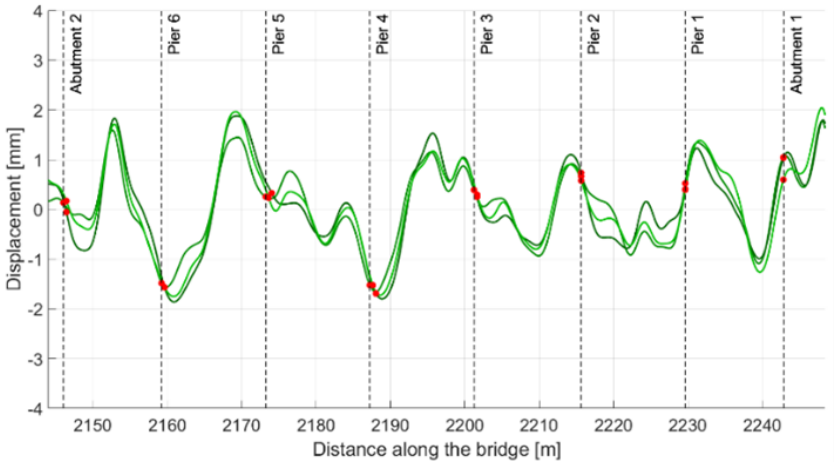


Figure 3.27. RILA rear-mounted runs (measurements) with corrected pier locations: Cluster 4b (re-printed from Tola et al., 2025).

Table 3.4. Corrections of each support for cluster 3b in meters (reprinted from Tola et al., 2025).

Support name / Run name	Run2-Tr3	Run3-Tr3	Run4-Tr3	Run5-Tr3	Run6-Tr3	Run7-Tr3	Run8-Tr3
Abutment 1	-4.80	-4.80	-4.80	-4.80	-4.80	-4.80	-4.80
Pier 1	-4.80	-4.80	-4.80	-4.80	-4.80	-4.80	-4.80
Pier 2	-4.80	-4.80	-4.80	-4.80	-4.80	-4.80	-4.80
Pier 3	-4.80	-4.80	-4.20	-4.80	-4.80	-4.80	-4.80
Pier 4	-4.80	-4.80	-4.20	-4.80	-4.80	-4.80	-4.80
Pier 5	-4.80	-4.80	-4.20	-4.80	-4.80	-4.80	-4.80
Pier 6	-4.80	-4.80	-5.40	-4.80	-4.80	-4.80	-4.80
Abutment 2	-4.80	-4.80	-4.80	-4.80	-4.80	-4.80	-4.80

Table 3.5. Corrections of each support for cluster 4b in meters (reprinted from Tola et al., 2025).

Support name / Run name	Run2-Tr4	Run3-Tr4	Run8-Tr4
Abutment 1	-2	-1.6	-1.6
Pier 1	-2	-1.6	-1.6
Pier 2	-2	-1.2	-1.6
Pier 3	-2	-1.2	-1.6
Pier 4	-2	-1.6	-1.6
Pier 5	-1.6	-1.6	-1.6
Pier 6	-1.6	-1.6	-1.6
Abutment 2	-1.6	-1.6	-1.6

3.4.4 Obtaining Scoured-Stage Track Profiles

For practical reasons, scoured stage track elevation measurements were not available. To obtain the track profile for the scoured stage, the steps in Figure 3.31 are applied. Since the numerical model presented in 3.1.2 does not consider dynamic effects, a 2D model has been used to generate bridge displacements, i.e., z(x). This VBI model accurately reflects the complexities of real-world scenarios, thereby providing a robust and reliable framework for analysis. It was generated using the TTB-2D version 4.4 software. The model overview and vehicle module are illustrated in Figures 3.28 and 3.29, respectively (Cantero, 2022). This software solves the coupled vehicle–track–bridge problem in 2D, modelling (i) the bridge as an Euler–Bernoulli beam finite element system on elastic/viscous supports, (ii) the track as rail–pad–sleeper–ballast layers, and (iii) the train as a multibody system with prima-ry/secondary suspensions. Wheel–rail contact is enforced throughout the crossing; rail irregularities are injected from prescribed power spectral densities. Time integration employs a Newmark-β scheme to generate displacement/acceleration histories, as well as internal forces, for both infrastructure and vehicles.

TTB-2D organizes the problem in vertical "layers" that are coupled through the wheel–rail contact and solved as a single system. The vehicle is a convoy of independently defined 6-DOF cars (comprising a car body, two bogies, and two wheelsets) connected by linear primary/secondary suspensions (springs and dashpots); standard sets are provided, but every mass, inertia, and suspension constant is user-editable. Rail irregularities are injected from PSDs and act at the wheel–rail interface, so the vehicle continuously excites the infrastructure while remaining in contact. The resulting coupled equations are integrated in time using the Newmark-β method. The track is represented by a rail beam supported at sleeper locations by lumped layers, including pad, sleeper mass, ballast, and sub-ballast. The bridge is modelled as an Euler–Bernoulli beam using finite elements, with user-defined support locations and vertical and rotational stiffnesses, allowing for simply supported, continuous, or mixed boundary conditions. A vehicle, track, and bridge are assembled into a single time-varying system through wheel–rail contact

constraints, from which displacement, acceleration, and internal force histories are obtained across all layers.

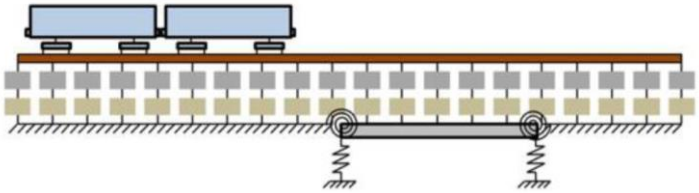


Figure 3.28. TTB-2D Model Overview (Cantero, 2022).

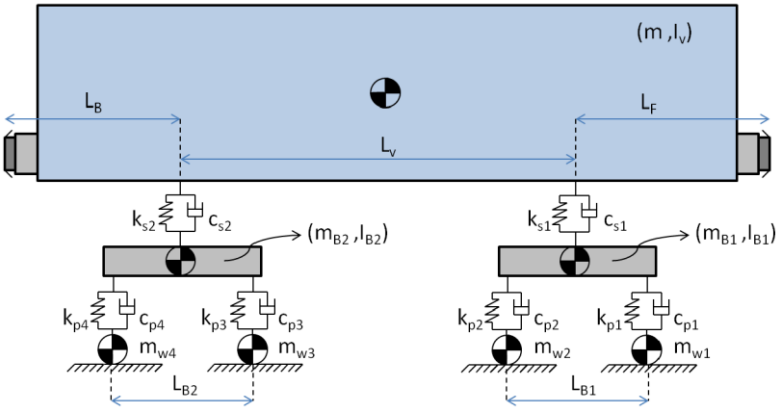


Figure 3.29. TTB-2D Vehicle model (Cantero, 2022).

In version 4.4, the code supports multi-span layouts with optional pile elements at supports. Each span has two boundary points with a distance between them (Figure 3.30). A vertical stiffness of -1 denotes an ideal vertical fix (vertical DOF constrained, no pile unit included), 0 denotes a vertical release, and a positive value models an elastic vertical support and includes a pile unit (lumped mass–spring–dashpot). At intermediate supports, two vertical stiffness values are defined—one for each adjacent span end. In this case study, the bridge was modelled as simply supported by releasing rotation and assigning project-specific positive vertical stiffnesses at all supports. Each span end uses a two-level support: an upper support spring at the beam end and, beneath it, a pier unit (m_{pi}, k_{Bi}, c_{Bi}). The support-level springs were set ten times the corresponding pier stiffnesses. The pier stiffnesses k_{Bi} were computed from Eq. (3.4). The pier unit was implemented as an undamped mass–spring with $m_p = 10^4$ kg and $c_B = 0$.

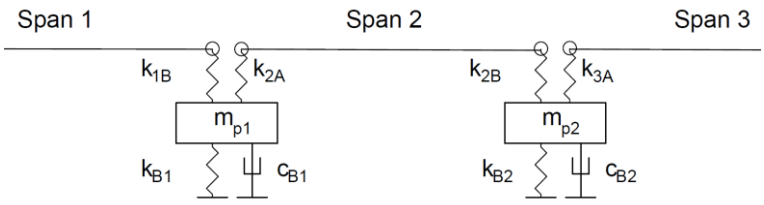


Figure 3.30. Two-level support scheme in TTB-2D v4.4 (Cantero, 2022).

The VBI model parameters were calibrated to align with the speed, vehicle, mechanical, and geometric characteristics of the bridge for which the RILA measurements were taken. The properties presented in Table 3.1 also apply to the VBI model. Table 3.6 provides the remaining bridge, vehicle, and track parameters that are adopted in the VBI model. When generating the healthy bridge displacements at step 1, the known travel velocities of RILA runs were used. Since the train weights of crossings are not known, train body weights for each crossing are randomly generated using Monte Carlo simulation and selected from a normal distribution within a ± 10% range of the Class 66/0 Co-Co Diesel technical data sheet. Monte Carlo simulation is used to study models whose changes occur in a random manner rather than following predetermined deterministic rules (Landa & Binder, 2014). It utilizes sequences of random numbers to generate different system states or paths. While each run with a different random sequence produces distinct outcomes, these results converge statistically within an error margin. This method is especially valuable for investigating complex physical systems where direct analytical solutions are infeasible.

The mechanical properties of the vehicle are assumed to remain constant, as the measurements were obtained from the same vehicle. Later, in step 2, rail irregularities for each run are found by subtracting healthy-stage bridge displacements from the healthy-stage track profile, i.e., from the RILA displacements. The rail irregularities of each track fall into two groups. The mean rail irregularities for the four clusters calculated in step 3 are shown in Figure 3.32.

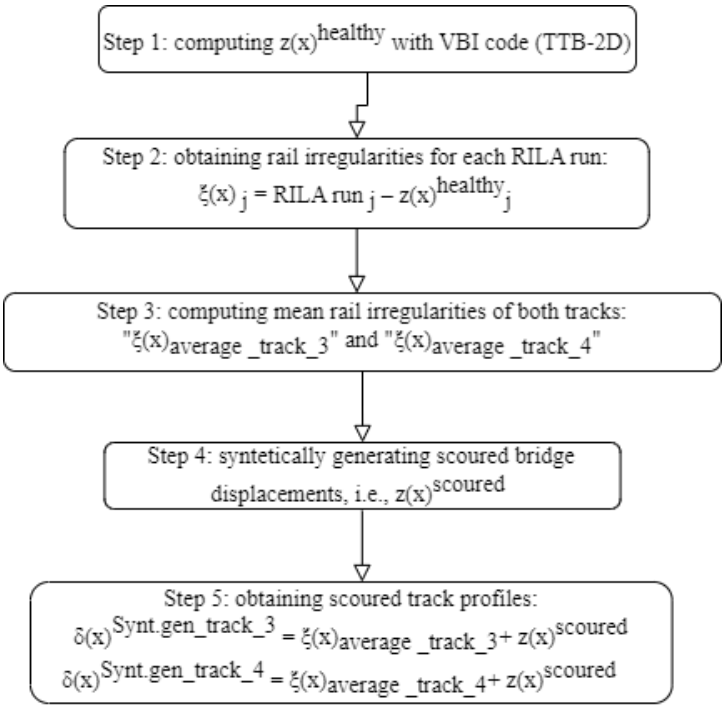


Figure 3.31. Steps to obtain scoured stage track profile (reprinted from Tola et al., 2025).

Table 3.6. TTB-2D (version 4.4) Model properties (reprinted from Tola et al., 2025).

Parameter	Value	Parameter	Value
Mass per unit length (kg/m)	22826	Train body mass (kg)	$72{\cdot}10^3$-$88{\cdot}10^3$
Damping ratio	1.40	Travel speed (m/s)	12 - 30
Rail Young's modulus (kN/m^2)	$2.06 \cdot 10^8$	Main body mass moment of inertia (kN m^2)	21574
Rail Second moment of inertia (two rails) (m^4)	$23.22{\cdot}10^{-5}$	Main body length (m)	13.34
Mass per unit length (two rails) (kN/m)	20.60	Additional distance (m), front and back	4.00
Rail damping (%)	0.10	Number of bogies	2
Pad Vertical stiffness (kN/m)	$6.50{\cdot}10^4$	Each boogie mass (kN)	78.50
Vertical damping (kNs/m)	75.00	Bogies mass moments of inertia (kNm^2)	196
Sleeper spacing (m)	0.40	Each bogie length (m)	4.21
Sleeper mass (kg)	2125	Number of wheels	6
Ballast mass (kg)	591	Each wheel mass (kg)	2498
Ballast stiffness (kN/m)	$137.75{\cdot}10^3$	Primary suspension stiffness (kN/m)	350
Ballast damping (kNs/m)	58.80	Primary suspension viscous damping (kNs/m)	20.00
Sub ballast stiffness (kN/m)	$77.5{\cdot}10^3$	Secondary suspension stiffness (kN/m)	1000
Sub ballast damping (kNs/m)	31.15	Secondary suspension viscous damping (kNs/m)	150

In step 4, scoured-stage bridge displacements are generated with TTB-2D version 4.4 (Cantero, 2022). To decide which support to be simulated as scoured, a CE algorithm is used. CE optimisation minimises the sum of squared differences between the bridge displacements of the static model in Section 3.1.2, and those of the synthetically scoured runs generated with TTB-2D version 4.4. As the water level reaches all bridge piers, each pier is potentially vulnerable to scour. Therefore, synthetically generated simulations consist of nine scenarios. Each of these assumes one pier has been scoured; the final scenario is the healthy case. Specifically, Scenario 1 assumes that Abutment 1 is scoured, Scenario 2 assumes that Pier 1 is scoured and so on, with Scenario 8 assuming that Abutment 2 is scoured. For each scoured scenario, the scour percentage is 20%. Fifty simulations with RILA front- mounted, and fifty with RILA rear-mounted were generated for each scenario. Then, for each scenario, the CE optimisation algorithm in Eq. (3.18) is applied:

$$O^m = \sum_{j=1}^{J} \sum_{i=1}^{N} \left(z_{ij}^{Syn} - z_{ij}^{Sta}\right)^2 \tag{3.18}$$

where m is the number of the support, J is the number of runs, N is the number of model nodes (the spans are divided into 0.2m long elements), z^{Syn} is the bridge displacement generated with TTB-2D version 4.4, and z^{Sta} is the bridge displacement computed using the static bridge model outlined in Section 3.1.2. Of the nine scenarios subjected to optimisation, the one where Pier 6 is scoured yields the low-est value of the objective function, O^m, suggesting that Pier 6 is scoured. Figure 3.33 shows the RILA front-mounted, and Figure 3.34 shows the RILA rear-mounted bridge displacement simulations, i.e., $z(x)^{Sco}$, for the scenario in which Pier 6 is 20% scoured. Fifty simulations with randomly generated travel speeds and train body weights are selected using Monte Carlo simulation. The velocities are randomly chosen from a

normal distribution with a mean of 21 m/s and a standard deviation of 10 m/s, to ensure compatibility with the speed range of the field measurements. The train body weights are chosen from a normal distribution with a mean of 80,000 kg and a standard deviation of 1,500 kg. The mean has been taken from the Class 66/0 Co-Co Diesel technical data sheet.

Rail irregularities are assumed to remain relatively unchanged from the healthy stage until pier scour occurs. To attain scoured-stage track profiles on step 5, the average rail irregularities of the cluster are superposed with the scoured- stage bridge displacements. Then, a random noise following a normal distribution with a mean of 0 and a variance of 0.5% of the maximum value of each cluster is added to the obtained scoured profiles. Figure 3.35 and Figure 3.36 show track profiles obtained with Track 3 rail irregularities for RILA front-mounted and rear-mounted runs, respectively. Figure 3.37 andFigure 3.38 show the track profiles obtained with track 4 rail irregularities for RILA front-mounted and rear-mounted runs, respectively.

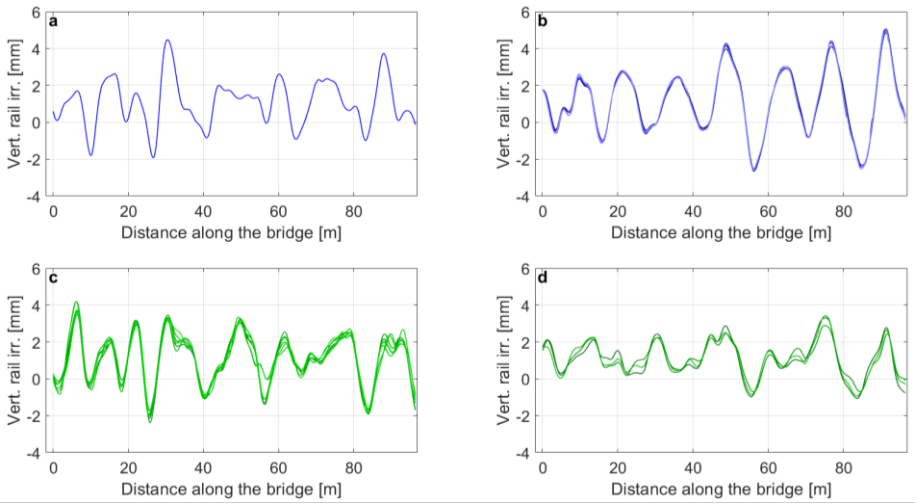


Figure 3.32. Rail irregularities of (a) Cluster 3a; (b) Cluster 4a; (c) Cluster 3b; and (d) Cluster 4b (reprinted from Tola et al., 2025).

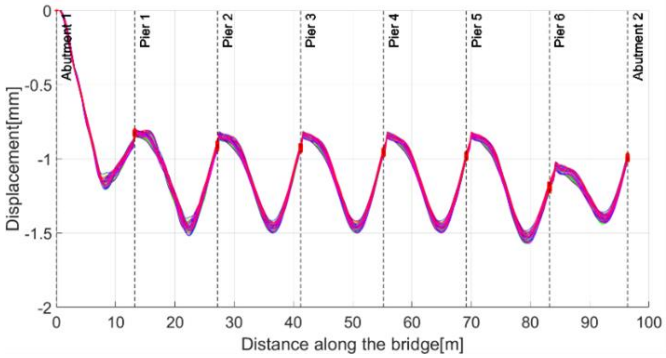


Figure 3.33. Bridge displacements with 20% scour at Pier 6: RILA front-mounted (reprinted from Tola et al., 2025).

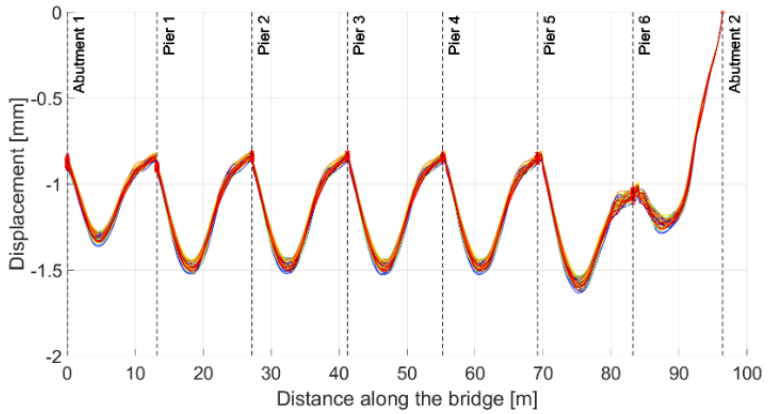


Figure 3.34. Bridge displacements with 20% scour at Pier 6: RILA rear-mounted (reprinted from Tola et al., 2025).

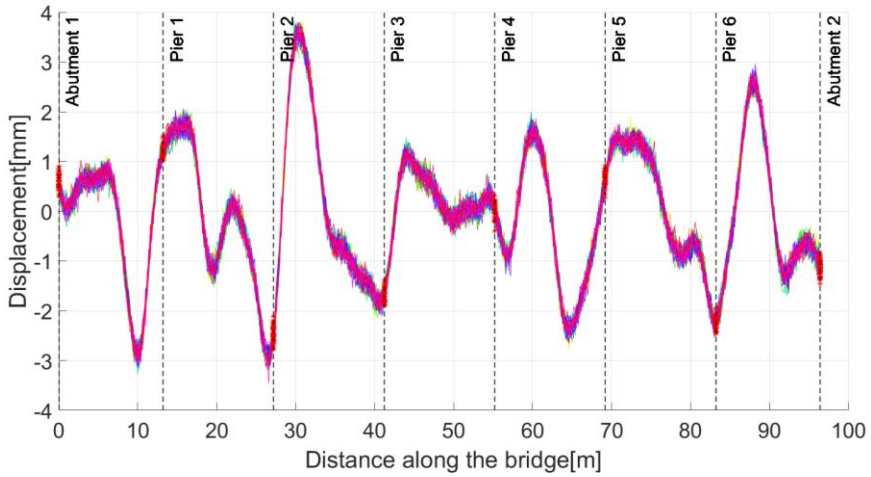


Figure 3.35. Track profiles obtained with Track 3 rail irregularities: RILA front-mounted (reprinted from Tola et al., 2025).

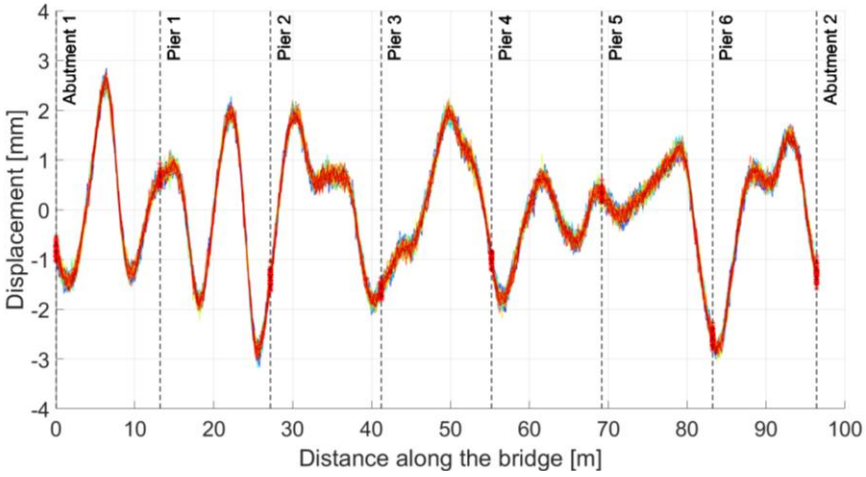


Figure 3.36. Track profiles obtained with Track 3 rail irregularities: RILA rear-mounted (reprinted from Tola et al., 2025).

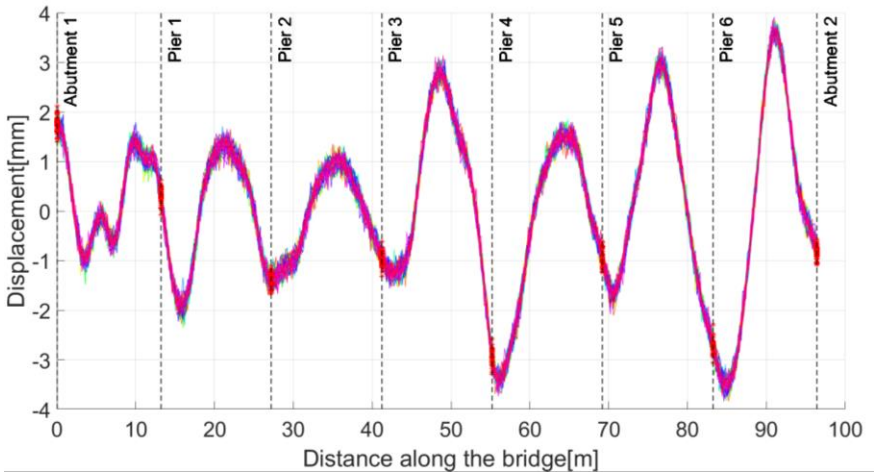


Figure 3.37. Track profiles obtained with Track 4 rail irregularities: RILA front-mounted (reprinted from Tola et al., 2025).

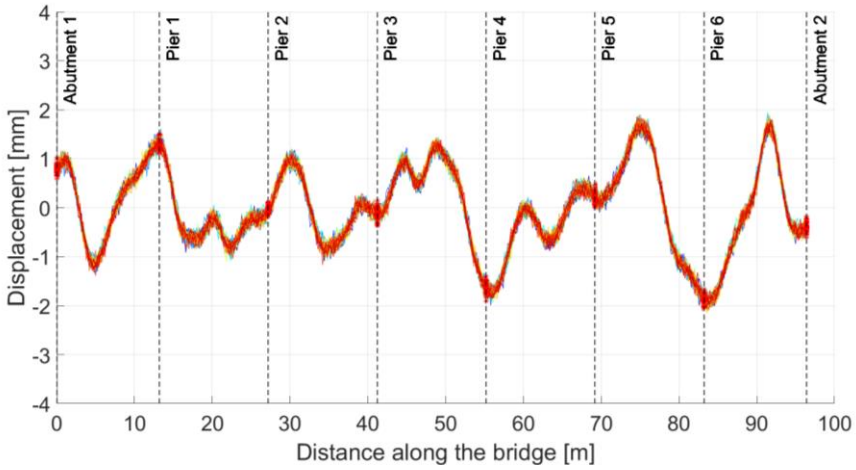


Figure 3.38. Track profiles obtained with Track 4 rail irregularities: RILA rear-mounted (reprinted from Tola et al., 2025).

3.4.5 Optimisation of Pier Stiffnesses with CE Algorithm

For the healthy stage, the algorithm calculates a single pier stiffness value (for all piers) that minimises the sum of squared differences between the healthy track profile and RILA measurements. The objective function is presented in Eq. (3.19).

$$O(k_{healthy}) = \sum_{j=1}^{J} \sum_{i=1}^{N} \left(\delta_{ij}^{Mea} - \delta_{ij}^{Sta}\right)^2 \tag{3.19}$$

where $k_{healthy}$ is the healthy stage pier stiffness, J is the number of runs, and N is the number of model nodes (the spans are divided into 0.2-metre-long elements), δ^{Mea} are RILA displacement measurements, and δ^{Sta} is the superposition of the rail irregularities and bridge displacements computed using the numerical model explained in Section 3.1.2. Four CE optimisations of the healthy stage are performed for the

four clusters of RILA measurements. One optimal pier stiffness value is computed for each cluster. The CE optimisation algorithm adapted from Botev (2013) is provided in Annex C. To optimize the stiffness of the scoured pier, the CE algorithm calculates the optimal scoured pier stiffness that minimises the sum of the squared differences between the scoured track profile obtained using static bridge displacements and the scoured track profile calculated using the TTB-2D model. The objective function of the scoured stage is rewritten as Eq. (3.20):

$$O(k_{sco}) = \sum_{j=1}^{J} \sum_{i=1}^{N} \left(\delta_{ij}^{Syn} - \delta_{ij}^{Sta}\right)^2 \tag{3.20}$$

where k_{sco} is the scoured pier stiffness, δ^{Sta} is the superposition of the rail irregularities and scoured bridge displacements computed using the static model explained in Section 3.1.2. The term, δ^{Syn} represents the scoured track profiles (Figure 3.35 and Figure 3.36 for Track 3; Figure 3.37 and Figure 3.38 for Track 4). Four groups of scoured stage optimisations are performed for four different track profiles (two RILA locations and two rail irregularities). Restarts are performed as often as necessary after the first optimisation to avoid premature convergence.

Table 3.7 presents the resultant pier stiffnesses for both healthy and scoured (Pier 6) optimisations. A significant difference between the inferred healthy and scoured stage pier stiffnesses indicates scour. Figure 3.39 and Figure 3.40 show envelopes, which are $\mu \pm \sigma$ curves for healthy and scoured track profiles for Track 3. The dashed curves in Figure 3.39 and Figure 3.40 represent the envelopes of the RILA front-mounted and rear-mounted healthy measurements, respectively. The scoured envelopes deviate from the healthy envelopes at Pier 6, the graphical indication of scour. Similarly, the scoured track profile envelopes of the Track 4 profile envelopes deviate from the healthy envelopes, as seen in Figure 3.41 and Figure 3.42.

Table 3.7. Pier stiffness optimisation results (reprinted from Tola et al., 2025).

Pier stiffness (kN/m ·10³)	**Healthy stage (RILA measurements)**		**Scoured stage (Synthetically generated track profile)**	
	Track 3	Track 4	Track profile obtained with Track 3 rail irregularities	Track profile obtained with Track 4 rail irregularities
RILA front-mounted	478	473	337	354
RILA rear-mounted	476	475	342	342

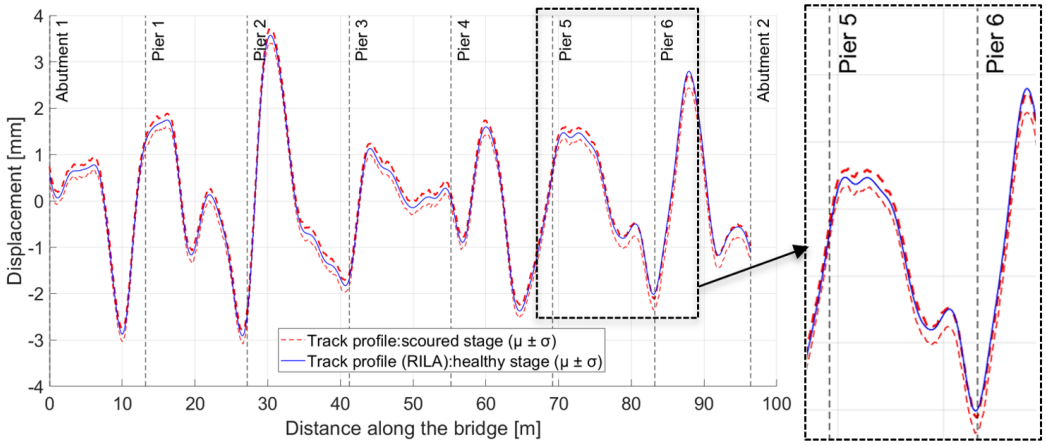


Figure 3.39. Envelopes of healthy and scoured stage track profiles from Track 3 rail irregularities: RILA front-mounted (reprinted from Tola et al., 2025).

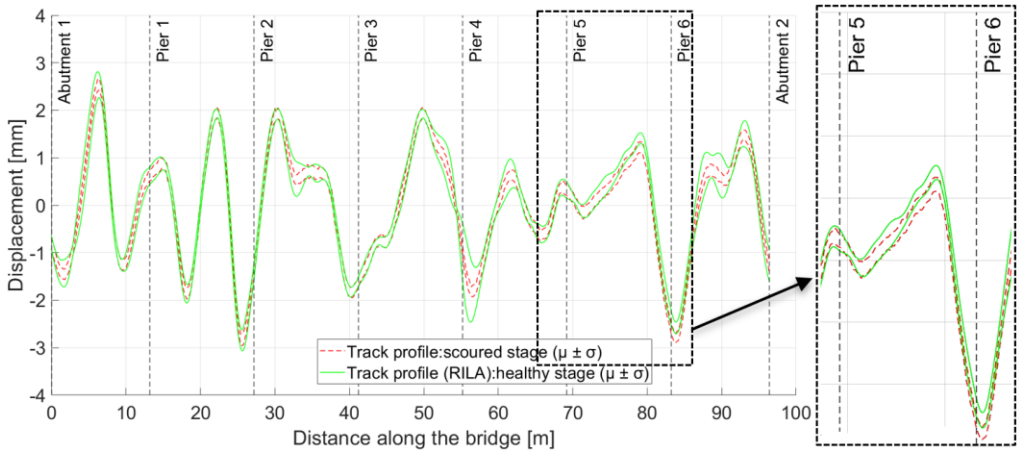


Figure 3.40. Envelopes of healthy and scoured stage track profiles from Track 3 rail irregularities: RILA rear-mounted (reprinted from Tola et al., 2025).

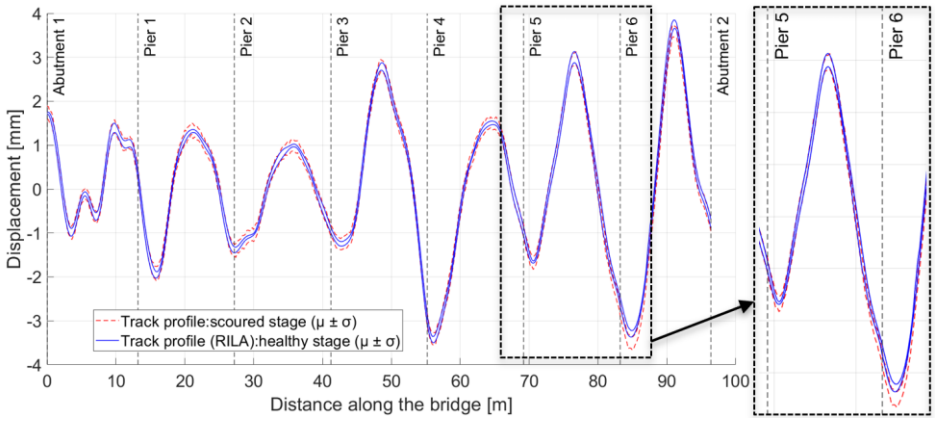


Figure 3.41. Envelopes of healthy and scoured stage track profiles from Track 4 rail irregularities: RILA front-mounted.

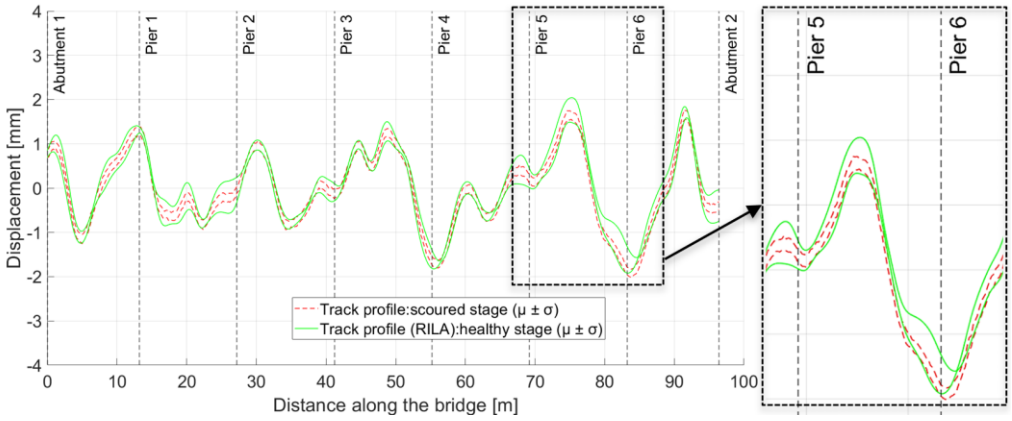


Figure 3.42. Envelopes of healthy and scoured stage track profiles from Track 4 rail irregularities: RILA rear-mounted.

3.4.6 Sensitivity Analysis

A sensitivity analysis has been conducted to evaluate the effect of different scour levels on the approach. In this analysis, track profiles have been observed by coupling each scoured set with a healthy set. When obtaining track profiles, the average values of rail irregularities provided in Figure 3.32 have been superimposed with TTB-2D-simulated bridge displacements. Ten sets of bridge displacement simulations, each consisting of ten simulations, have been used. The study compares five sets of healthy case simulations (5·10 = 50 runs) to five sets of scoured simulations (5·10 = 50 runs), each with varying degrees of scour at Pier 6 (5%, 10%, 15%, 20%, and 25%). The different scour percentages are achieved by decreasing the vertical spring stiffness of Pier 6. To present the results of the sensitivity analysis, a Track Profile Damage Indicator (TPDI) is introduced along the bridge for each scour level. A TPDI value at point *i* is defined as the sum of ten squared differences between healthy and scoured track profiles, as expressed in Eq. (3.21).

$$TPDI_{id} = \sum_{j=1}^{10} \left(\delta_{ij}^{Hea} - \delta_{ij}^{Sco}\right)^2 \tag{3.21}$$

where *i* is the small beam element number along the bridge, *d* is the scour level, δ_{ij}^{Hea} is the j^{th} healthy track profile at point *i*, and δ_{ij}^{Sco} is the j^{th} scoured track profile at point *i*. Figure 3.43 and Figure 3.44 illustrate TPDI values obtained with Track 3 rail irregularities when RILA is front-mounted and rear-mounted, respectively. In both front-mounted and rear-mounted cases for track 3, it has been observed that as the percentage of scour increases, TPDI values increase in the vicinity of the scoured pier. The Track 4 TPDI values yield similar results to Track 3, as seen in Figure 3.45 and Figure 3.46.

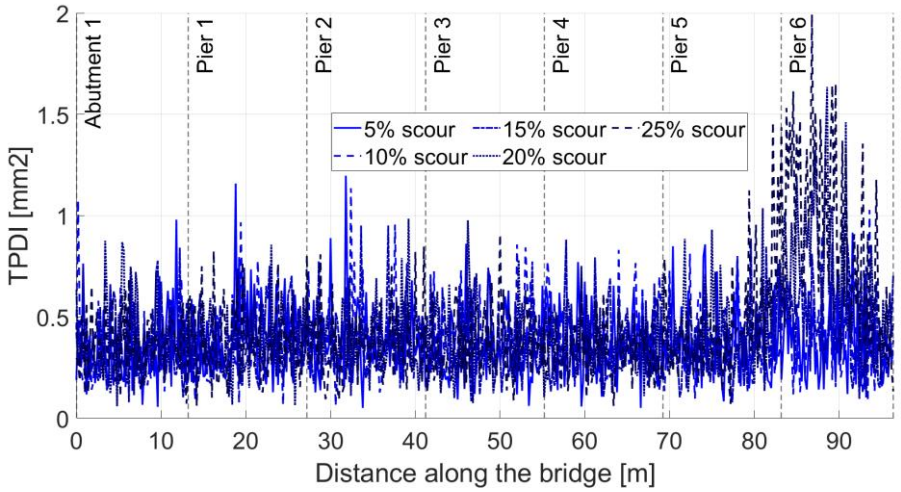


Figure 3.43. TPDI values from Track 3 rail irregularities: RILA front-mounted (reprinted from Tola et al., 2025).

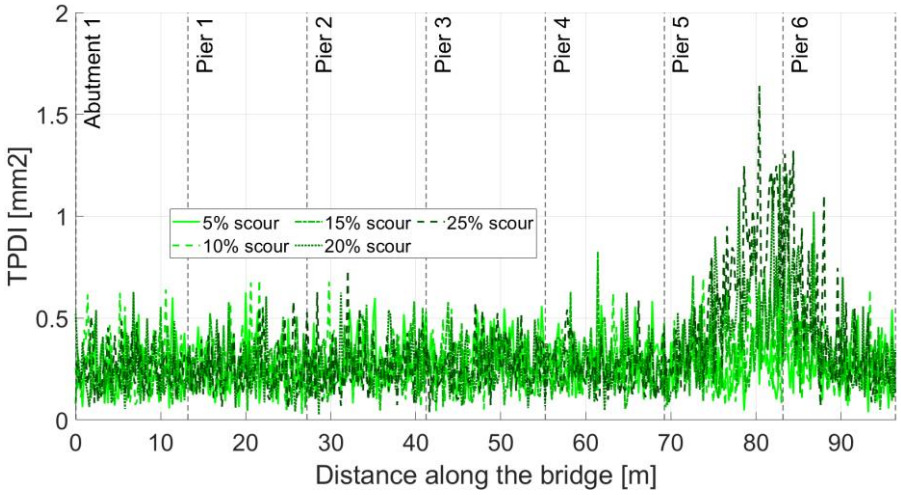


Figure 3.44. TPDI values from Track 3 rail irregularities: RILA rear-mounted (reprinted from Tola et al., 2025).

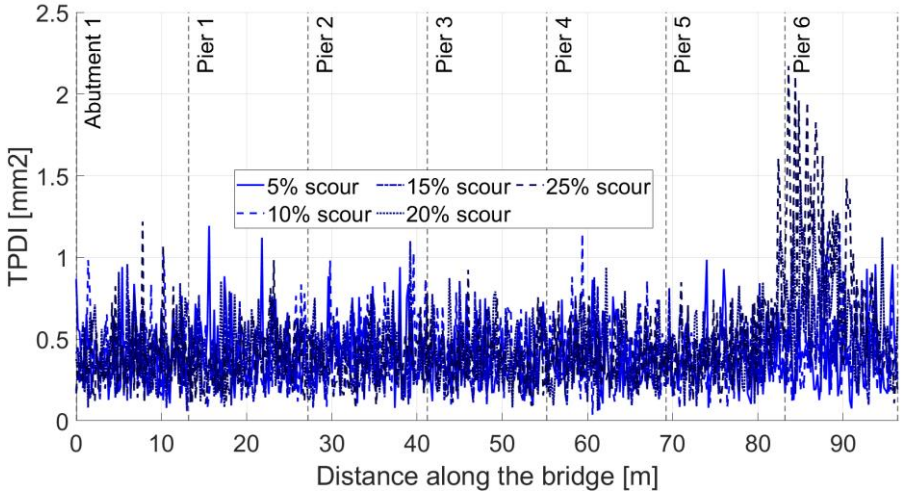


Figure 3.45. TPDI values from Track 4 rail irregularities: RILA front-mounted.

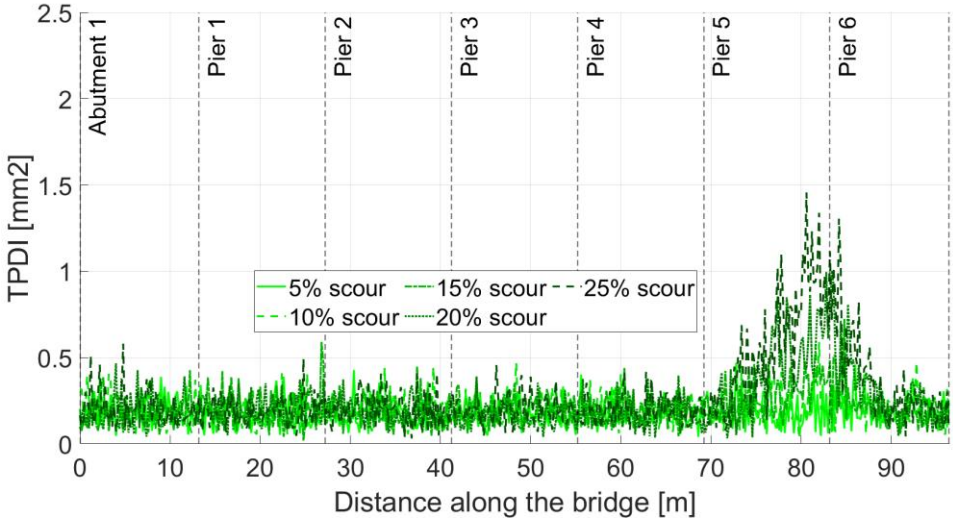


Figure 3.46. TPDI values from Track 4 rail irregularities: RILA rear-mounted.

3.4.7 Assessment of Results and Discussion

Due to practical constraints encountered during the study, measurements were only taken from the after-repair stage of the railway bridge, i.e., the healthy stage. Despite this limitation, field data has been utilized to demonstrate repeatability, indicate resolution and accuracy, and emphasize the significance of the proposed approach. To address the absence of data from the scoured stages, synthetic track profiles have been generated. In the healthy state, the CE algorithm minimises the differences between measurements and the theoretical track profile, which incorporates static bridge simulations and rail irregularities. In the scoured state, measurements are yielded by superimposing rail irregularities obtained from field measurements and synthetically generated bridge displacements.

A CE optimisation that minimises the sum of squared differences between measured and theoretical track profile is used to determine the best-fitting pier stiffness. The static bridge displacements of the theoretical track profile are calculated with a model that uses the unit load theorem to establish MR-ILs for a 6-axle train crossing the bridge, mimicking RILA longitudinal level measurements. When simulating bridge displacements, scour is represented by a reduction in vertical pier stiffness. The disparity in optimised pier stiffness between the damaged and scour-free scenarios serves as an indicator of scour. A sensitivity analysis comprising one hundred runs, demonstrates that as the vertical support stiffness decreases to simulate a more significant percentage of scour, the differences between healthy and scoured stage track profiles become more pronounced. For noise levels of 1% or higher, the sensitivity analysis indicates that the TPDI values of multiple unscoured piers tend to approach the TPDI value of the scoured pier. Consequently, the methodology encounters difficulties in accurately detecting the scoured pier.

3.5 Conclusion

The proposed methodology of this research validates the concept of drive-by scour detection (Fitzgerald et al., 2019; Prendergast et al., 2016) by conducting a case study consisting of a combination of synthetically generated and field data for track profiles. The implementation of this method will facilitate the development of enhanced maintenance strategies and enable the implementation of timely, targeted interventions. These interventions will contribute to the development of safer and more resilient railway infrastructure.

It is important to note that the proposed methodology focuses on the structural consequences of scour, as observed through displacement measurements, rather than on the complex hydrodynamic processes causing it. It is acknowledged that temperature fluctuations can exert an influence on the flexural rigidity of concrete bridges. However, the impact of temperature on foundations is relatively small. Consequently, the impact of temperature is not a significant concern for the proposed approach.

The application of the proposed framework is contingent upon the utilization of a numerical model that calculates the static displacements of a theoretical bridge track profile. These displacements are computed with a model that does not consider dynamic effects. A more robust numerical model would better represent the true behaviour of the structure. The present study focuses solely on determining the 'optimised' pier stiffnesses rather than calculating the true values, thereby reducing dependency on the model being a perfect representation of reality. The findings of the case study, in conjunction with the sensitivity analysis, show that the utilization of a simplified numerical model was adequate for the objective.

The field data of the case study was obtained from a 6-axle vehicle data crossing a multi-span simply supported bridge. It is recommended that subsequent studies investigate and further validate the methodology with diverse field data obtained from bridges exhibiting distinct structural characteristics and other types of trains. The case study has been seen to detect scour with a noise of 0.5%. Future research should involve a more thorough examination of methodology with additional field data. Furthermore, further research is recommended for bridges susceptible to scan at multiple piers simultaneously.

CHAPTER 4: DATA GENERATION FOR ML FRAMEWORK

The scour detection methodology explained in Chapter 3 is extended to create an ML-aided methodology that can be applied to diverse bridges and span configurations, as well as to multiple concurrent scour cases. The details of the ML framework are provided in Chapters 5 and 6. As robust ML models require large volumes of data and field measurements are scarce, the TTB-2D version 4.4 (Cantero, 2022) software is used to generate synthetic RILA-measured track profiles. The details and calculation principles of the model are presented in section 3.4.1. These longitudinal-level track profiles are then used to extract scour-sensitive features, which serve as input to ML algorithms.

This chapter begins by explaining the rationale behind generating two types of models based on sensor orientation. It then moves on to cover the Vehicle–Track–Bridge (TTB-2D) interaction model, which includes vehicle properties, track parameters, bridge parameters, rail irregularities, the integration scheme and bridge displacements. Next, scenario design and calendars are presented, addressing seasonal changes and flood progressions. Finally, the process of noise injection and scour modelling for single and multiple pier scour scenarios is explained.

4.1 The Effect of Sensor Orientation on Modelling

Two sensor configurations are modelled to emulate the RILA front-mounted and rear-mounted cases. In the front-mounted configuration, the sensor unit is mounted 2 meters ahead of the first axle, while in the rear-mounted configuration, it is mounted 2 meters behind the last axle. The front-mounted sensor records the profile experienced by the wheels, including bridge displacements as the train first engages the structure, capturing immediate local responses to scour near the first-affected supports. The rear-mounted sensor, in contrast, reflects the accumulated global dynamics after successive bogies have passed, often yielding richer signal content but also integrating longer-wavelength effects. This distinction is analogous to the differences reported between front and rear-mounted axle measurements in rolling stiffness vehicles by Berggren (2009), where front-mounted measurements emphasize direct deflections under load, while rear-mounted ones reveal delayed or propagated responses that are more sensitive to support irregularities. Modelling both sensor orientations separately, therefore, allows the proposed

framework to capture complementary scour effects and assess classifier robustness for realistic sensor placements.

4.2 Train–Track–Bridge Interaction Model Properties

The track profiles induced by vehicles crossing the railway bridges are generated using TTB-2D version 4.4 (Cantero, 2022), which incorporates both rail irregularities and bridge displacements (Tola et al., 2025). The properties of the simulation models are provided in this section.

4.2.1 Rail Irregularities

This section explains the generation of the rail irregularity component of track profiles (Eq. 3.15). Figure 4.1 shows rail irregularities of WCM15 and WCM15A bridges, respectively. Figure 4.2 illustrates the rail irregularities of the River Teign and Colonnade bridges. Vertical rail irregularities are treated as a random spatial signal, $\xi(x)$, whose statistical properties do not change with position along the track. The spatial grid is sampled with an interval Δx. The sample points are $x_1 : \Delta x : x_{end}$ with $n_x = length(x)$. For spectral synthesis, a discrete Fourier representation of length N_s is adopted:

$$N_s = 2^{\lceil log_2(n_x) \rceil} \tag{4.1}$$

enabling an efficient inverse transformation. The irregularity wavelength λ_r is restricted to a band [λ_{min}, λ_{max}]. Angular spatial frequency Ω is defined by Eq. (4.2):

$$\Omega = \frac{2\pi}{\lambda_r},\ \Omega_{min} = \frac{2\pi}{\lambda_{max}},\ \Omega_{max} = \frac{2\pi}{\lambda_{min}} \tag{4.2}$$

The one-sided Class-6 Power Spectral Density (PSD) (White et al., 1978; Federal Railroad Administration, 1983) is as seen in Eq. (4.3):

$$S(\Omega) = \frac{A_v\, \Omega_c^2}{(\Omega^2+\Omega_r^2)(\Omega^2+\Omega_c^2)} \tag{4.3}$$

where $A_v = 1.5 \cdot 10\text{-}6$, $\Omega_r = 2.06 \cdot 10^{-2}$ rad/m, $\Omega_c = 0.825$ rad/m. PSD is evaluated on the discrete angular spatial frequency Ω_k implied by Δx and N_s. A two-sided spectrum $S_2(\Omega_k)$ is formulated as in Eq. (4.4):

$$S_2(\Omega_k) = \frac{1}{2}\, S(\Omega_k) \tag{4.4}$$

with the mean, namely, zero-frequency term and the highest resolvable frequency term are set to zero to avoid drift and edge distortions. Deterministic magnitudes are assigned as seen in Eq. (4.5):

$$A_k = \sqrt{S_2(\Omega_k)\, \frac{N_s}{\Delta x}} \tag{4.5}$$

Randomness is introduced through independent phases ϕ_k (Eq. (4.6)):

$$\phi_k \sim \mathcal{U}(0,2\pi)\left(k = 1, \ldots, \frac{N_s}{2} - 1\right),\ \phi_0 = \phi_{\frac{N_s}{2}} = 0 \tag{4.6}$$

where ϕ_0 and $\phi_{(Ns/2)}$ are phases at the zero-frequency term and the Nyquist-limit term. The positive-frequency spectrum is assembled as in Eq. (4.7):

$$\hat{\xi}(\Omega_k) \ = \ A_k\, e^{\,\mathrm{i}\phi_k} \tag{4.7}$$

Conjugate symmetry is imposed for negative frequencies, and an Inverse Discrete Fourier Transform (IDFT) yields the vertical rail irregularities as in Eq. (4.8):

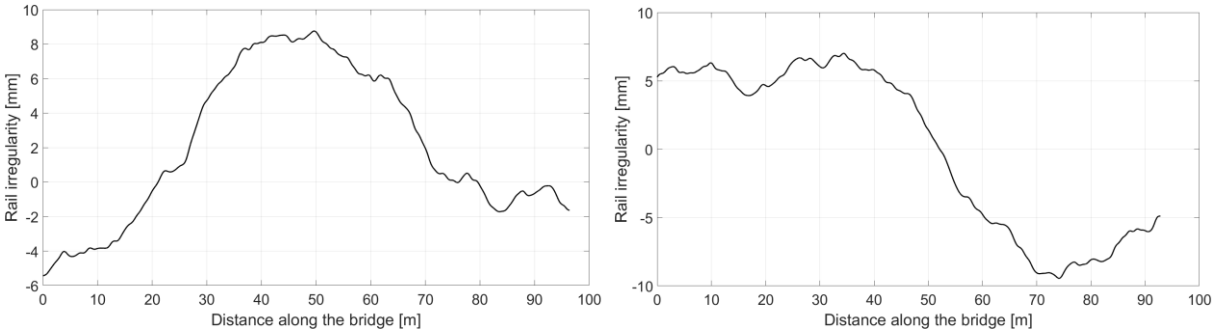
$$\xi(x_j) = IDFT\{\xi(\Omega_k)\} \tag{4.8}$$

for j=1,2,3, ..., n_x. The result $\xi(x)$ is a zero-mean, stationary vertical profile in meters whose second-order statistics match the FRA Class-6 spectrum within [λ_{min}, λ_{max}] (White et al., 1978; Federal Railroad Administration, 1983).

The range of vehicle speeds at the synthetic vehicle crossings in this study (10 m/s to 44.7 m/s) is determined from the field measurements acquired via RILA and provided by Fugro, as well as from other measurements reported by Wang et al. (2021). Using a Monte Carlo simulation, train body weights and travel speeds for each crossing are randomly generated, with train body weights constrained to ±10% of the technical specifications of the three train types. Classes 4 and 5 with higher-roughness vertical irregularities are not used, as at the upper end of the simulated speed range, permanent wheel-rail contact is lost intermittently.

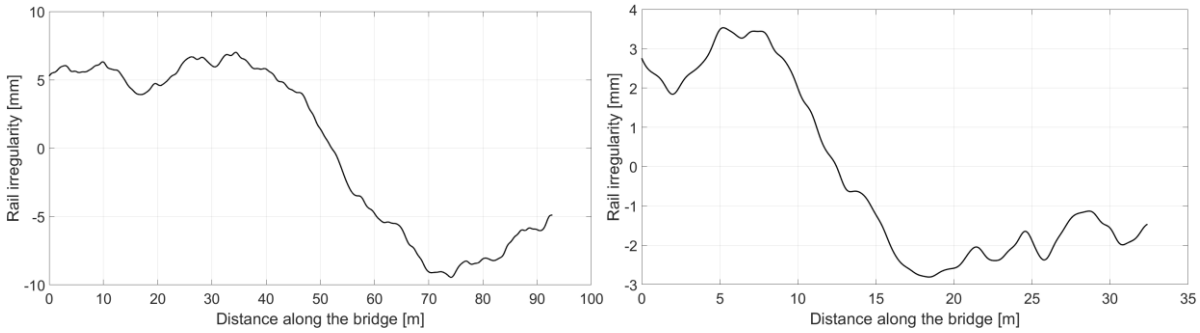
4.2.2 Vehicle and Track Properties

It is assumed that the vehicles' mechanical properties remain constant, as the measurements are believed to have been obtained from the identical vehicles. Three vehicle types are included in the simulations: the Class 66/0 Co-Co Diesel locomotive; the Eurostar trainset, consisting of a leading locomotive, seven intermediate coaches (EC2–EC8), and a trailing locomotive; and the AVE S103 ICE3 trainset, configured with four coaches (EC1, two EC2367 units, and one EC45). The vehicle properties are provided in Table 4.1. The simulations have been balanced 1:1:1 by vehicle class, assigning one-third of the crossings to AVE S103 ICE3, one-third to Eurostar, and one-third to Class 66 Co-Co Diesel. The track properties used for ML data generation match those adopted in the MR-IL track model in Section 3.1.3. The parameter values follow Table 3.2, except for the sleeper spacing. In the TTB-2D simulations, the sleeper spacing was set to 0.40 m, a multiple of the finite-element length, ensuring that sleepers coincide with mesh nodes.



(a) (b)

Figure 4.1. Rail irregularities of (a) WCM15 bridge; (b) WCM15A bridge.



(a) (b)

Figure 4.2. Rail irregularities of (a) River Teign bridge; (b) Colonnade bridge.

4.2.3 Properties of Bridge Models

Over two years, it is assumed that one RILA-implemented vehicle crosses each of the four bridges per day, resulting in 2,924 recorded passages. The dataset covers two concrete bridges in Northern England (WCM15 and WCM15A, located in Carlisle) and two steel bridges in Southern England (the River Teign near Newton Abbot and the Colonnade in Dawlish). It encompasses structural variability, including different superstructure types, a range of span counts and lengths, variations in stiffness and damping, and differing support conditions, thereby providing a heterogeneous basis for training the ML model. The structural properties of the bridge models are summarised in Table 4.2. Theoretical spring stiffness values of the piers are calculated using Eq. (3.4), as detailed in section 3.1.2.1. The properties of the railway track are likewise adopted from this study.

Table 4.1. Vehicle Properties of Simulations.

Parameter	Class 66/0 Co-Co Diesel	Eurostar (Locomotive+EC2–EC8+Locomotive)	AVE S103 ICE3 (E_C1+2×E_C2367+E_C45)
Train body mass (kg)	$72{\cdot}10^3$ – $88{\cdot}10^3$	Locomotive: $51.5{\cdot}10^3$ Coaches: 22,525	Coaches 1,2,3,4: $47.8{\cdot}10^3$ Coach 5: $41.2{\cdot}10^3$
Main body mass moment of inertia (kN·m²)	21,574	Locomotive: $1.05{\cdot}10^6$ Coaches: $810{\cdot}10^3$	Coach 1,2,3: $1.957{\cdot}10^6$ Coach 4: $1.687{\cdot}10^6$
Main body length (m)	13.34	Locomotive: 14.0. Coaches: 18.7	17.375
Additional distance (m), front and back	4.00	Locomotive: 6.28	Coach 1: [4.76, 3.70] Other coaches: [3.70, 3.70}
Number of bogies	2	2	2
Each bogie mass (kN)	8,004	Locomotive: $2.2{\cdot}10^3$ Coaches: $2.9{\cdot}10^3$	3,500
Bogie mass moment of inertia (kN·m²)	$19.9{\cdot}10^3$	Locomotive: 1900 Coaches: 2508	Coaches 1,2,3: 1,715 Coach 4: 1,372
Bogie length (m)	4.21	3.0	2.5
Number of axles	6	4	4
Each wheel mass (kg)	2498	Locomotive: 1700 Coaches: 1900	1800
Primary suspension stiffness (kN/m)	350	Locomotive: 2600 Coaches: 2000	2400
Primary suspension viscous damping (kN·s/m)	20	Locomotive: 12 Coaches: 12	20
Secondary suspension stiffness (kN/m)	1000	Locomotive: 3260 Coaches: 580	700
Secondary suspension viscous damping (kN·s/m)	150	90	40

Table 4.2. Model properties of the bridges.

Bridge Name	No of spans x span lengths (m)	A (m²)	EI (kN·m²)	Mass per unit length (kg / m)	Damping Ratio (%)	Healthy pier spring stiffness (kN/m)
WCM15	13.2, 5×14.0, 13.2	8.00	$30.9 \cdot 10^6$	22,826	1.40	467,966
WCM15A	10.8, 7×14.0, 10.8	5.43	$15.7 \cdot 10^6$	13,565	1.47	233,025
River Teign	4×23.2	2.63	$30.2 \cdot 10^6$	20,622	0.50	412,777
Colonnade	3×10.83	0.13	$3.65 \cdot 10^6$	997	1.65	215,375

Note: E = Young's modulus, A = cross-sectional area, and I = second moment of inertia.

4.2.4 Noise Injection

Gaussian noise with zero mean and variance is injected into each simulated crossing, with the standard deviation defined as in Equation 4.9 and the noisy signal expressed as in Equation 4.10:

$$\sigma = 0.005 \times RMS(\delta) \tag{4.9}$$

$$\delta_{noisy} = \delta + \eta, \quad \eta \sim N(0, \sigma^2) \tag{4.10}$$

where δ denotes the simulated track profile, η is the noise, and *RMS(δ)* is its root mean square value. Therefore, moderate noise proportional to the signal magnitude is obtained, consistent with the filtered nature of RILA field data, as seen in Figure 3.23 to Figure 3.25, and to preserve scour-sensitive signal content. Figure 4.3 shows 30 representative, synthetically generated RILA front-mounted track profiles of the Colonnade Bridge, included to illustrate the raw inputs from which the ML features are extracted. The darker-shaded curves show the track profiles of the bridge with approximately 20% scour. The RILA rear-mounted case yields results similar to the RILA front-mounted case.

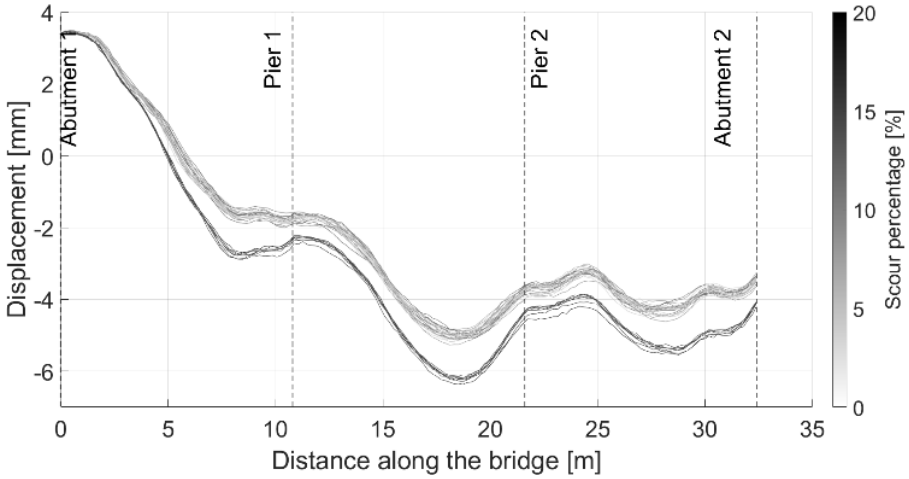


Figure 4.3. Track profiles of Colonnade bridge: RILA front-mounted.

4.3 Scour Scenario Design and Simulation Calendars

Two region-specific scour calendars covering the years 2015 and 2016 have been developed: one corresponding to the southern England (Figure 4.4) and the other to the northern England Figure 4.5), aligned with the respective bridge locations. This chronological ordering is necessary for replicating seasonal variability, storm-induced spikes, and natural recovery phases in the dataset. The years 2015–2016 are selected to construct the scour calendars, as this period captured an unusually wet winter with multiple major storm events (e.g., Storms Desmond, Eva, Frank, Imogen, and Katie) that triggered significant flooding across the UK (Centre for Ecology & Hydrology, 2016). This makes it a representative and

challenging window for simulating realistic scour progression. Although the ML models implemented do not directly learn temporal dependencies in the manner of time-series predictors, the sliding-window aggregation process requires that training crossings be arranged by their chronological order. Through this approach, the scour calendars furnish a robust temporal context for feature windowing, thereby preserving the fidelity of realistic scour progression throughout the dataset.

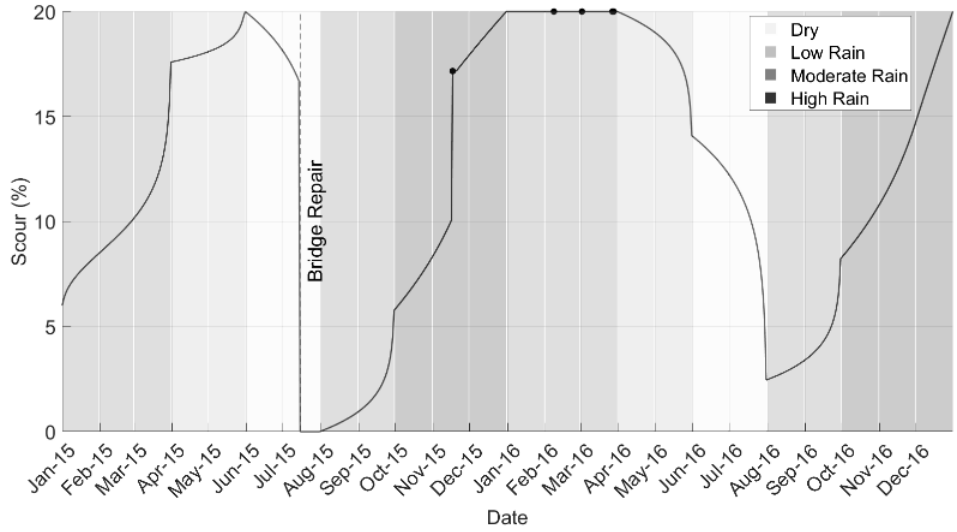


Figure 4.4. Simulated scour calendar 2015-2016, south England.

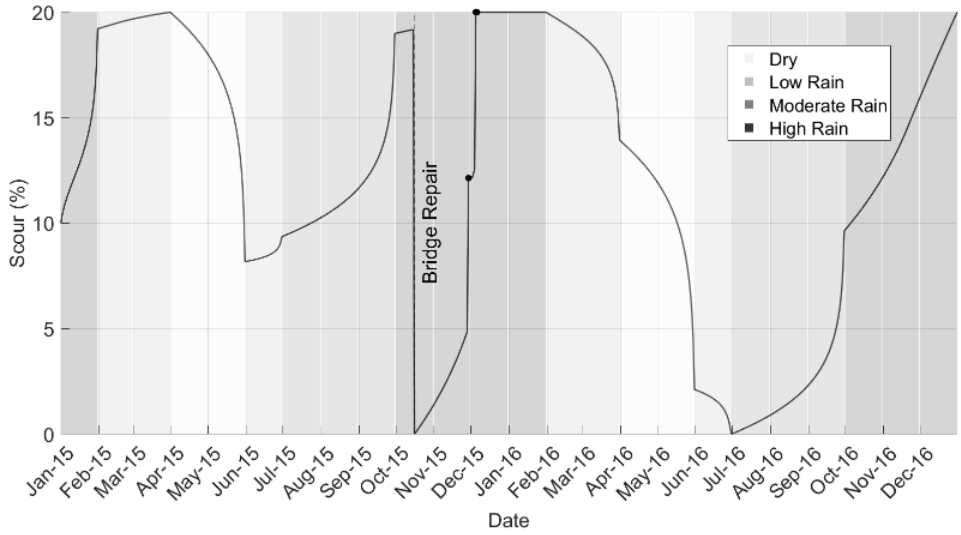


Figure 4.5. Simulated scour calendar 2015-2016, north England.

The long-term rainfall climatology of the bridge catchment zones is assessed using 30-year precipitation records (1991–2020) from the nearest UK Met Office stations, namely Teignmouth for the South bridges and Carlisle surroundings for the North bridges (Met Office, 2024). Figure 4.6 and Figure 4.7 illustrate the annual rainfall average values between 1991 and 2020 for Teignmouth and Carlisle stations, respectively. Based on this data, scour percentages that reflect monthly rainfall categories are defined as:

- High rainfall months (Oct–Dec): 10% base scour
- Moderate rainfall months (Jan–Mar, Aug–Sep in North; January–March and August–Sep in South): 6% base scour
- Low rainfall months (April–July): 3% base scour
- Dry months (region-specific, e.g., June–July in South): 3% with additional scour regression representing sediment backfilling and partial recovery of the scour hole.

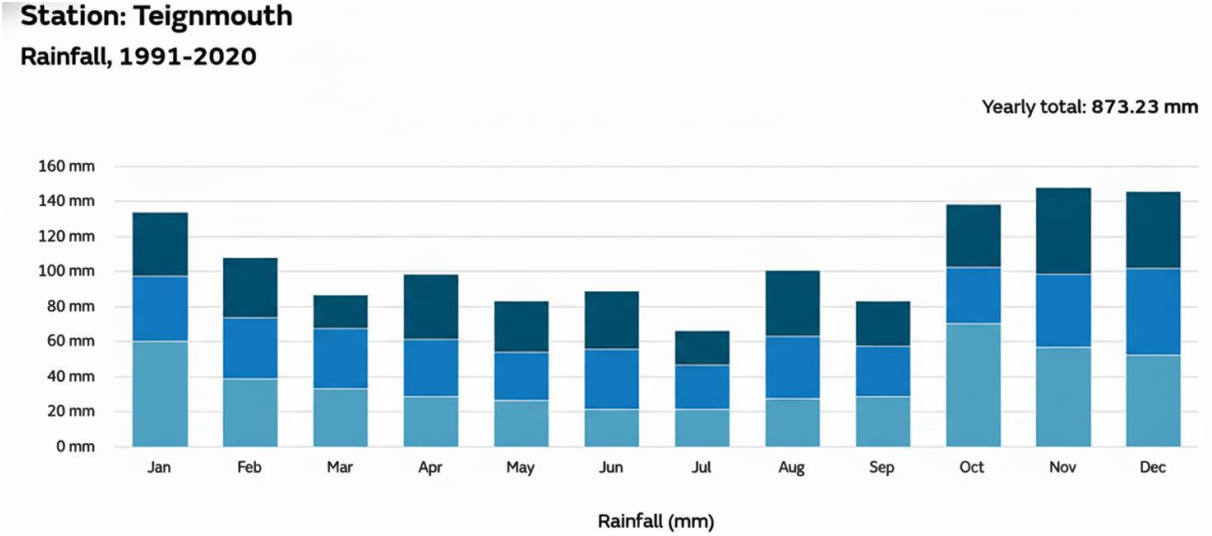


Figure 4.6. Teignmouth, UK: annual rainfall averages between 1991 and 2020 (Met Office, 2024).

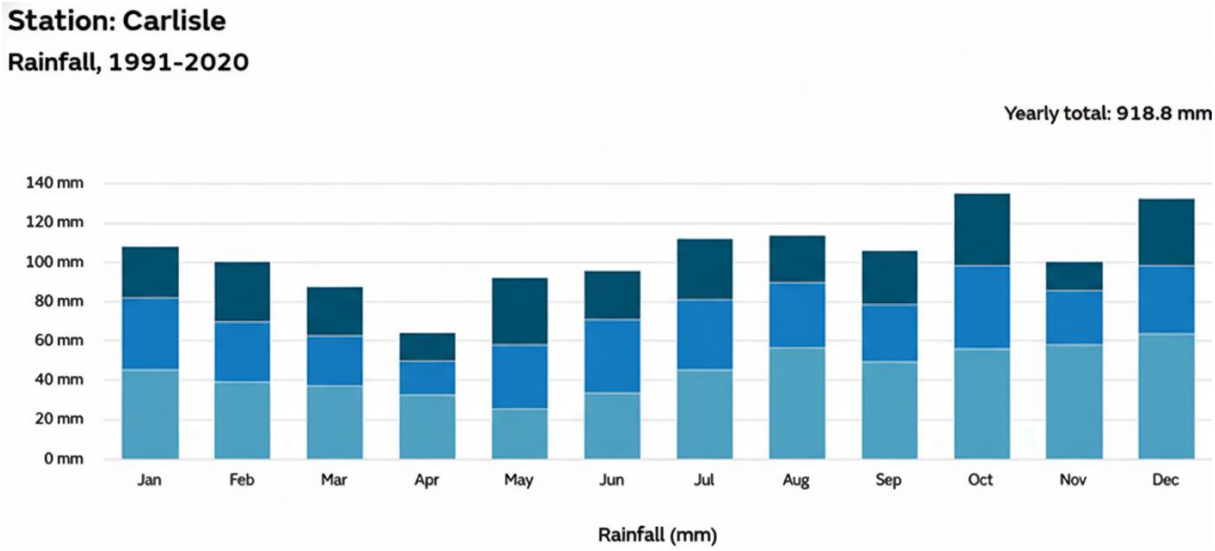


Figure 4.7. Carlisle, UK: annual rainfall averages between 1991 and 2020 (Met Office, 2024).

In Figure 4.4 and Figure 4.5, the rainfall intensity is shown with background shading: the lightest shade represents dry months, while progressively darker shades indicate low, moderate, and high rainfall months. These categories serve as nominal scour levels, from which deviations due to floods and partial

infill (represented as a reduction in the modelled scour percentage) are superimposed. Complete infills were observed in some field measurements (Mueller and Wagner, 2005). However, the soil might not be consolidated. Hence, a calendar value of 0% scour due to infills in dryer seasons is interpreted as a practical proxy for the calendar, rather than a claim of full geotechnical restoration or full stiffness recovery. With this representative infills, the "healthy" stage, namely up to 5% of the scoured stages, is covered in the calendar. Even so, most days remain classified as damaged, consistent with the high precipitation exposure and associated scour risk in the bridge zones.

Historical storms and flood records are integrated as discrete scour peaks. Floods are modelled by assigning a maximum scour percentage of 20% to specific storms that affected the study regions. For the North bridges, this included Storm Clodagh on 29 November 2015 and Storm Desmond on 5–6 December 2015. For the South bridges, the considered events are Storm Barney on 17 November 2015, Storm Imogen on 8 February 2016, Storm Jake on 2 March 2016, and Storm Katie on 27–28 March 2016. The flood dates are marked with filled dots in Figure 4.4 and Figure 4.5. These storms are selected based on their intensity and regional impact, and their inclusion ensures that the synthetic scour calendars capture the flood-induced extremes observed during the winters of 2015–2016. Flood dynamics are simulated as follows:

- ±5-day ramping: scour increases linearly up to the flood day and decreases afterwards to mimic scour-hole development and partial infill, respectively.
- Randomised spikes: a stochastic increment is applied on the flood day, dependent on the rainfall regime. During flood days, the scour percentage is increased by a randomly chosen increment, i.e., spike. The spike is sampled from a uniform distribution, as no evidence suggests that flood-induced scour increments follow a normal distribution. Therefore, the uniform case provides a simple assumption. The bounds of the spike range depend on rainfall intensity: 6–10% in high-rainfall months, 4.5–7.5% in moderate-rainfall months, and 3–5% in low- or dry-rainfall months. These limits are empirically defined to reflect observed flood severity while ensuring scour percentages never exceed 20%.
- Between floods, scour evolves according to seasonal transitions that are modelled with a logarithmic curve between the start and end of each season. Scour percentage at a time t is expressed as in Eq. (4.11):

$$y(t) = a_s log(t) + b_s \tag{4.11}$$

with coefficients derived as in Eq. (4.12):

$$a_s = log(t_2) - log(t)y_2 - y_1;\ b_s = y_1 - a_s \cdot \log(t_1) \quad (4.12)$$

where (t_1, y_1) represents the scour percentage at the start of the season, while the second point (t_2, y_2) represents the target scour percentage at the end of the season. The adoption of a logarithmic trend to represent scour progression is supported by established formulations in the literature. Ettema et al. (2011) note that, among the many predictive equations available, the logarithmic approaches of Oliveto and Hager (2005) and Kothyari et al. (2007) apply to piers with different geometries. These approaches predict an unbounded logarithmic increase in local scour depth over time, which aligns with the physical expectation that scour develops rapidly during the early stages. It progresses more slowly as an equilibrium state is approached. When the start and end values are too close for a reliable logarithmic fit, a small daily linear change (i.e., delta) is applied to approximate scour progression. The delta depends on rainfall category: in high-rainfall months, scour increased by about 0.06% per day; in moderate months, by 0.02% per day; and in low-rainfall months, by 0.01% per day. If a low-rainfall month followed a flood, scour decreased at approximately 0.02% per day, whereas in dry months, sediment infilling dominated, with a greater reduction of 0.04% per day. These reductions represent changes in the modelled scour level. They do not imply that any infilled material is consolidated or that the foundation stiffness fully returns to its pre-scour condition. With these representative decreases due to dryer seasons, less than 20% scoured stages are aimed to be incorporated in the calendar.

To replicate the temporary hydraulic equilibrium that often follows floods, a three-day pause counter is introduced, during which scour progression stops. This represents the short-term stabilisation of the scour hole, while the channel is reshaped and sediments temporarily settle. Once the counter expires, scour resumes its evolution, either following a logarithmic progression or linear delta-based changes, as explained above. The duration of the pause counter is empirically selected to be long enough to capture the observed stagnation after floods, yet short enough to avoid suppressing natural recovery trends.

Bridge scour repairs are also incorporated into the calendars, with specific reset dates assigned to each region. Scour is reset to 0% for the North bridges in October 2015 and for the South bridges in July 2015. After these dates, scour progression restarts from the seasonal baseline to ensure the influence of structural interventions is reflected. To account for realistic operational conditions, it is assumed that on public holidays, no trains cross the bridges, so the measured scour percentage remains constant.

Scour is modelled as a proportional reduction in the vertical stiffness of the pier springs, set by the scour calendar (Figure 4.4 and Figure 4.5). 0–20% scour corresponds to a 0–20% decrease in stiffness at the affected supports. This mapping is used as a simplified effective-stiffness assumption within the

simulation framework, rather than a claim of exact field stiffness recovery behaviour. The details of this modelling approach are the same as in section 3.1.2.1. In each simulation, up to three supports are modelled as scoured: the first is randomly selected. Any additional scoured supports are adjacent to this initial location. This approach is adopted because scour develops progressively around neighbouring piers rather than at distant supports.

4.4 Final Remarks

This chapter presents the principles for simulated RILA-obtained track profiles within the ML framework. Using the TTB-2D v4.4 Vehicle–Track–Bridge interaction model, synthetic longitudinal level profiles have been created for four structurally distinct bridges, under varying vehicle types, speeds, and track conditions. Rail irregularities have been synthesised from an FRA Class-6 spectrum; noise has been injected into the track profiles; and scour has been modelled as proportional reductions in pier stiffness, affecting up to three adjacent supports per crossing.

The chapter explains the use of two sensor orientations, front and rear-mounted, to emulate realistic RILA implementations and to capture complementary local and global scour responses. A chronologically ordered dataset is required for batch-based decision-making via the sliding windows method, as detailed in Chapter 5. Therefore, region-specific scour calendars have been developed for bridges in North and South England, incorporating seasonal rainfall patterns, flood-induced peaks, repair interventions and recovery periods. These calendars provide an approximation of reasonable scour evolution in the absence of field measurements under known scour states. In calendar construction, reductions in scour between flood events represent partial infill trends. These reductions do not claim that the infilled soil is consolidated or that stiffness necessarily returns to an undamaged state. Instead, with these decreases, days with less severe stages are included in the scour calendar. Most of the days in the calendars consist of damaged stages, namely 5–20% scour, to reflect the high scour risk in areas where bridges are located. Simulations have been generated using this temporal order of scour, from which scour-sensitive features are extracted. The following chapters build directly on this foundation. Chapter 5 introduces feature engineering, class balancing and ML modelling procedures for scour detection and localisation, while Chapter 6 presents the results of the ML models.

CHAPTER 5: MACHINE-LEARNING FRAMEWORK

This chapter outlines the ML framework for scour detection and localisation. It uses two complementary strategies: bridge-level detection to assess scour presence, and support-level localisation to identify scoured piers. Tasks, data splits, and batch aggregation with sliding windows are defined. Feature engineering, feature selection, and data balancing are described. Classifiers, scaling rules, and hyperparameter tuning are specified. Readouts and evaluation procedures are outlined for both strategies. Codes will be provided in the annexes.

5.1 Novelty and Methodology

The growing demand for monitoring transportation infrastructure underscores the need for network-scale approaches. Building on this foundation, the present study extends the scour detection framework provided in Chapter 3 into an ML-aided methodology capable of generalising across diverse bridges, span configurations, and multiple concurrent scour cases and of localising damage to specific supports. This extension includes (i) extracting scour-sensitive features from the longitudinal track profiles generated in Chapter 4; (ii) benchmarking multiple ML classifiers trained on synthetic RILA-measured track profiles, (iii) investigating robustness across operational conditions and sensor orientations, and (iv) implementing a support-level localisation pipeline that maps predictions to individual supports. The novel contributions are:

Integration of ML with displacement data. Unlike most vibration-based scour detection studies that rely on accelerations, this research exploits displacement profiles obtained from far-from-axle track profile measurements, an under-explored but practical data source. These displacement features are further combined with ML algorithms to enhance detection capability and generalizability.

- *Feature engineering and selection.* Scour-sensitive features are derived from Discrete Wavelet Transformation (DWT) coefficients and Markov parameters obtained from Autoregressive model with eXogenous inputs (ARX) model impulse responses. To optimise input dimensionality, two complementary feature selection strategies are implemented: a relative-importance ranking method and a genetic algorithm (NSGA-II) wrapper approach, with comparative evaluation to identify the optimal feature sets. The input features are derived from the entire track profiles for

ML models that detect scour. ML models for scour localisation are trained and tested using features derived from scour-affected bridge support zones.

- *Multi-model ML benchmarking.* Three classifiers - ANN, RF, and SVMs - are systematically trained and compared, with RFs consistently emerging as the most accurate and robust for bridge-level scour detection.
- *Sensor orientation.* A comparative analysis of RILA front- and rear-mounted configurations reveals differences in their sensitivities. Leading sensors capture more localised entry responses, as reflected by higher class-separation effect sizes (Cohen's d) at early Markov orders. In contrast, trailing sensors rely on distributed multi-order interactions that reflect global bridge dynamics.
- *Batch-based decision-making.* A sliding-window aggregation strategy is proposed to stabilise predictions across consecutive training passages, reducing sensitivity to noise and outlier crossings and supporting deployment as a practical early-warning tool. Aggregation is applied at the bridge-level and per support to obtain stable localisation readouts.
- *Support-level localisation.* A pier-based model is developed in parallel to detection. The input features are computed per support, classifiers are trained using support labels, and outputs provide pier-wise probabilities and classifications. Evaluation is performed with the same confusion-derived metrics at the support-level.

The methodology of the ML framework is illustrated in Figure 5.1. A proof-of-concept version was presented in Correia et al. (2025), where the approach was briefly mentioned without details or results. This chapter provides the full ML methodology by adding RF and SVM models, Markov-parameter features, advanced feature selection, and a comprehensive evaluation across multiple train–bridge–scour configurations. These synthetically generated RILA-measured track profiles, explained in Chapter 4, serve as raw data for feature extraction. Two complementary feature sets are extracted: (i) DWT-based descriptors that capture scour-sensitive components; and (ii) ARX-based features expressed as Markov parameters from the impulse-response sequence.

Feature derivation differs by task. For bridge-level detection, features are computed from the entire track profile along the bridge at each crossing, yielding a single bridge-level decision (healthy vs. scoured). For support-level localisation, features are computed within support-centred spatial segments covering each support's scour-affected zone and linked to pier-level labels. Accordingly, the ARX to Markov pipeline fits a single ARX model per crossing for detection, but per support-centred segment for localisation, yielding local Markov parameters. Likewise, DWT descriptors are summarised globally for detection, whereas

for localisation, they are extracted and aggregated per support segment. Feature selection uses an embedded relative-importance ranking and a genetic-algorithm wrapper (NSGA-II). The selected features are used to train and evaluate three MATLAB models: ANN, RF, and SVM. Hyperparameters are optimised via grid search for ANN and random search for RF and SVM to ensure balanced accuracy and generalisation. The performance of RILA's front-mounted and rear-mounted configurations is examined for both tasks. Localisation models are trained and evaluated per support, producing a spatial readout along the bridge. Batch-based aggregation is also applied per support to stabilise localisation across consecutive crossings.

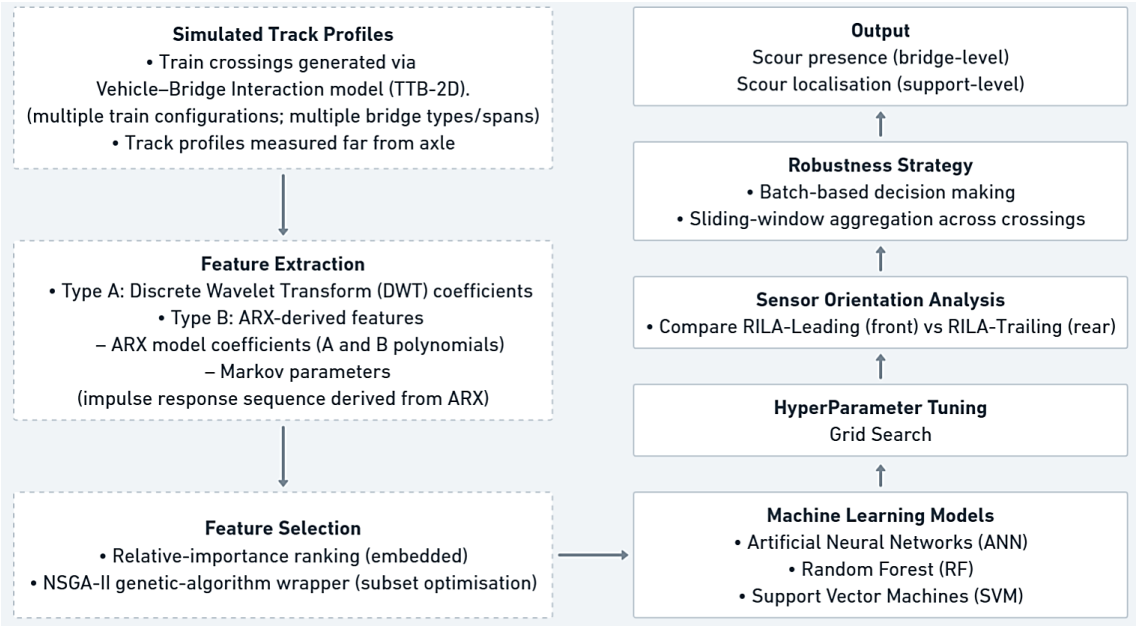


Figure 5.1. Flowchart of the methodology for scour detection, extended from Correia et al. (2025).

5.2 Task Definitions and ML Implementations

5.2.1 Detecting Presence of Scour

This section provides a concise overview of the fundamental principles underlying the binary classification algorithms used to detect scour. Details of preprocessing, decision rules, hyperparameters, and tuning within the nested CV framework are provided in Sections 5.3–5.6.

5.2.1.1 Classification labels, cross-validation protocol and data partitioning

Simulations with pier scour exceeding 5% are labelled as scoured. In contrast, those below this threshold are labelled as healthy, consistent with previous studies by Fernandes et al. (2024) and Gasparotti & Lopez (2024), which set a minimum scour level of 5%. CV is one of the most widely used resampling strategies for model evaluation and hyperparameter tuning (Berrar, 2018). In *k*-fold CV, the dataset is

partitioned into k disjoint folds of approximately equal size. The model is trained on k−1 folds and evaluated on the remaining fold. This procedure is repeated until each fold has served as a validation set once. The average of these validation results provides an estimate of the generalisation error. Compared with the single hold-out approach, k-fold CV uses the available data more efficiently and reduces the variance of performance estimates. Repeatedly testing the model on unseen partitions also mitigates the risk of overfitting, as the estimated performance reflects how well the model generalises to independent data. A nested CV extends this principle by embedding one level of CV inside another. The outer folds are used to assess generalisation performance, while the inner folds are reserved for hyperparameter tuning. This design prevents information leakage: the data used to choose hyperparameters does not overlap with the data used to evaluate the model. As a result, nested CV provides an unbiased estimate of prediction error and avoids the optimistic bias that arises if the same folds are used for both tuning and evaluation.

In this thesis, a five-fold outer CV is used to split the data in each round into an outer-training partition (four folds) and an outer-test partition (one-fold). Stratification is applied so the proportions of scoured and healthy samples are preserved in both partitions. All steps that require fitted statistics are learned on the outer-training data only and then applied to the outer-test data to avoid leakage. In practice, this affects ANN and SVM, which use z-scoring: the mean and standard deviation are computed on the outer training set and used to standardise the corresponding outer test set. RF uses the raw inputs and is not normalised. Within each outer-training partition, a three-fold inner CV is used only for hyperparameter tuning. Candidate hyperparameter settings are trained on the inner training splits and scored on the inner validation splits; scores are averaged across the three inner folds, and the best setting is selected. The model is then refit on the complete outer-training data with the selected hyperparameters and evaluated once on the test partition of the current outer fold. No information from any outer-test partition is used for model selection or preprocessing. Predictions and metrics are aggregated across the five outer folds to provide the final performance estimates. The final metrics across outer folds are computed as follows:

i. *Set-up*. For each outer fold r = 1, 2, 3..., 5, the model is fit on the corresponding training partition, and class labels are predicted on J_r, *where* J_r is the set of test observation indices for fold r. For each $i \in J_r$, the true label is $c_i \in \{0,1\}$ with 1=scoured, and 0=healthy, and the predicted label is $c_i \in \{0,1\}$, where c_i and $\hat{c}_i$ are the observed and predicted class labels for observation i.

ii. *Pooling* as in Eq. (5.1):

$$J = J_1 \cup J_2 \cup \cdots \cup J_5, N \tag{5.1}$$

Each outer-test instance appears once, as the test folds are disjoint. *N* is the total number of pooled test observations.

iii. *Confusion counts on the pooled set*. The final metrics are computed from the confusion-matrix counts *TP, FP, TN*, and *FN*, as defined in Eq. (5.2) to Eq. (5.6):

$$TP = \sum_{i \in I} [\, c_i = 1 \wedge \hat{c}_i = 1 \,] \tag{5.2}$$

$$FP = \sum_{i \in I} [\, c_i = 0 \wedge \hat{c}_i = 1 \,] \tag{5.3}$$

$$TN = \sum_{i \in I} [\, c_i = 0 \wedge \hat{c}_i = 0 \,] \tag{5.4}$$

$$FN = \sum_{i \in I} [\, c_i = 1 \wedge \hat{c}_i = 0 \,] \tag{5.5}$$

$$N = TP + FP + TN + FN \tag{5.6}$$

where *TP* is the number of scoured observations correctly classified as scoured, *FP* is the number of healthy observations misclassified as scoured, *TN* is the number of healthy observations correctly classified as healthy, and *FN* is the number of scoured observations misclassified as healthy.

iv. Final pooled metrics. Metrics used are as in Eq. (5.7) and Eq. (5.8):

$$Accuracy = \frac{TP+TN}{N}, \quad Precision = \frac{TP}{TP+FP} \tag{5.7}$$

$$Recall = \frac{TP}{TP+FN}, \qquad F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{5.8}$$

where *accuracy* measures overall correct classification; precision refers to the scoured class, namely the proportion of predicted-scoured that are truly scoured; recall refers to the scoured class, i.e., the proportion of truly scoured that are correctly detected, and *F1* is the harmonic mean of precision and recall.

v. *Training accuracy*. If J_f^{tr} denotes the outer-training indices in fold *f* and $\hat{y}^{f,tr}$, the corresponding training predictions, a pooled set of all outer-training observations across folds is as in Eq. (5.9):

$$J^{tr} = \bigcup_{f=1}^{5} J_r^{tr}, N_{tr} = | J^{tr} | \tag{5.9}$$

The fraction of correct predictions on that pooled training set is formulated as in Eq. (5.10):

$$Accuracy_{train} = \frac{1}{N_{tr}} \sum_{i \in I^{tr}} (c_i = \hat{c}_i^{tr}) \tag{5.10}$$

where N_{tr} is the size of J^{tr}.

Reproducibility. Randomness is used to assign observations to stratified *k*-fold partitions and to shuffle data within training partitions before inner CV splits. A fixed random seed is set before partitioning so that fold assignment and shuffle order are repeatable.

5.2.1.2 Input features and batch-based scour detection

Input features for ML algorithms that detect scour are computed from the full-bridge displacement profile for each train crossing. Two feature sets are used: (i) Type-A: comprises DWT-based descriptors that summarise signal content across wavelet levels. Type-B comprises Markov parameters obtained from ARX models fitted to the entire profile over the bridge length. These are the impulse-response coefficients of the identified transfer function, truncated to a fixed horizon so the vector length, i.e., the number of Type-B input features, is constant across bridges. The two sets of input features are combined into a single, fixed-dimensional feature vector for the bridge-level classifier. Full derivation details are provided in Section 5.3.

To mitigate the influence of noisy individual crossings and to strengthen robustness, a sliding-window strategy is applied to the feature space. Four batch sizes are defined: 7, 15, 31, and 61 crossings, corresponding approximately to 1 week, 2 weeks, 1 month, and 2 months of daily passages, respectively. Each window advances by one crossing, ensuring overlapping batches and a continuous sequence of training samples. For each window of length w_s, the feature construction follows two complementary steps. First, order-independent descriptors are derived by calculating the mean, standard deviation, minimum, and maximum of every feature across the w crossings. Second, order-preserving information is retained by concatenating the individual feature vectors of all m crossings in temporal sequence. The final batch representation, therefore, combines aggregated statistics with short-horizon temporal dynamics, resulting in a feature vector of dimension $(w_s+4)\cdot p$, where p denotes the number of input features. The condition of the final crossing (scoured or healthy) within the window defines the classification output of each batch. The batch scour percentages associated with the final crossing are also retained to enable subsequent analyses of scour severity. Computational details are provided in Section 5.3.

This strategy reflects a near-real-time monitoring setting, where the preceding crossings provide stabilising context, and the most recent crossing defines the decision. Window generation is first performed separately for each bridge to preserve the chronological order and avoid cross-bridge data leakage. The resulting windowed datasets comprise the feature matrices, class labels, and scour percentage vectors for each bridge. These per-bridge datasets are then concatenated to form a combined pool, which is used to train and evaluate the classifiers under both RILA front and rear-mounted configurations.

5.2.1.3 Hyperparameter tuning and evaluation metrics

ANN models are tuned by grid search, whereas RFs and SVM models are tuned by random search. A two-stage hyperparameter tuning strategy is used for ML scour detection models. First, a full search at

a 7-day batch explores wide hyperparameter ranges to identify well-performing regions. Second, transfer-tuning is applied in batches of sizes 15, 31, and 61 crossings: the search spaces are narrowed and centred around the settings found at batch size 7, reducing computation while preserving comparability across batch sizes. All tuning is performed within the inner folds of the nested CV to avoid leakage. Further computational details are provided in Section 5.6.

Performance is reported using Accuracy, Precision, Recall, and *F1* with scoured as the positive class. The formulae are given in Eqs. (5.7) and (5.8) are applied to the pooled outer-test predictions, i.e., counts *TP, FP, TN*, and *FN* are computed on the pooled set. Results are presented for each batch size (7, 15, 31, 61) using these metrics, along with batch-wise curves comparing performance across batch sizes. ROC and PR curves are not reported because evaluation is performed at a fixed operating point using hard class labels (scoured/healthy). The pipeline, by design, does not retain per-observation decision scores or probabilities. Hard-label evaluation is chosen for simplicity, comparability, and reproducibility across models and batch sizes.

5.2.1.4 ML algorithms and feature scaling

Three different ML classifiers: ANNs, RFs, and SVMs are trained and tested for scour detection. Before training, all input features are normalised to ensure comparability across dimensions. For the ANN and SVM classifiers, z-score normalisation is applied: each feature is centred by subtracting its mean and scaled by its standard deviation. This step is essential because i) ANN training is gradient-based: features on larger numerical scales would otherwise dominate the gradient updates, leading to inefficient learning and biased weight adjustments; ii) SVM with Radial Basis Function (RBF) kernel is distance-based, and unscaled features would distort the kernel space by allowing large-scale variables to overshadow smaller ones. Scaling ensures that all features contribute proportionally to the formation of the margin. For the RF classifier, implemented through MATLAB's *TreeBagger* (Matlab, 2025b), explicit feature scaling is not required. Decision trees use threshold-based splits that depend only on the ordering of feature values, making them inherently insensitive to feature magnitudes. All mathematical details for the classifiers—model definitions and decision rules; training objectives with regularisation and tuned hyperparameters; model-specific preprocessing; thresholding conventions; and tuning/validation within the nested CV framework—are provided in Sections 5.5 and 5.6.

5.2.2 Support-level Localisation of Scour

This section summarises the principles of the binary classifiers used for localising support-level scour. At the support-level, it is binary per support (scoured/healthy) with the 5% threshold. Computational

details and implementation—preprocessing, decision rules, hyperparameters, and tuning under nested CV—are provided in Sections 5.3–5.6.

5.2.2.1 Classification labels and data partition

The 5% threshold defined for scour detection in Section 5.2.1.1 is applied unchanged to the localisation algorithms. The sole difference is the unit of classification: the scoured/healthy label is assigned per support, rather than per simulation at the bridge-level. For data partition, the same nested external CV protocol and class-stratified folds described in Section 5.2.1.2 for scour detection are used for scour localisation, with the unit now a support-level batch. Folds are created on the pooled per-support batches; all preprocessing is fitted on the outer-training data and applied to the corresponding test data, and hyperparameters are tuned within the inner folds.

5.2.2.2 Feature construction, batch-based detection, localisation readout

Local deviations from the standard track profile are observed in the presence of scour, as illustrated in Figure 3.39 to Figure 3.42. The spatial interval of the observed deviated curves has been extended slightly, and the scour-affected zone of each support is considered to be $L/4$ portions of the span(s) that are supported by that pier/abutment. Consequently, the input features for the ML models that locate scour-affected supports are derived from the scour-affected zone of each support. Similar to scour detection models, two feature sets are derived for scour localisation models. Type-A comprises DWT-based descriptors computed on the scour-affected zone of each support, summarising the signal across wavelet levels. Type-B comprises Markov parameters derived from ARX models fitted to the same zone, with the impulse response truncated to a fixed horizon to give a constant length. The Type-A and Type-B vectors are then combined to form the per-support feature vector. For the support input feature, the derivation details are provided in Section 5.2.

A similar batch-based damage decision approach to scour detection, as provided in Section 5.2.1.2, is applied to the ML models that locate scoured supports. The sliding-window scheme is applied per support rather than per passage. For each bridge with n_{sup} supports and N_{sim} simulations, windows of length m are advanced by one simulation, yielding $F=N_{sim}-w+1$ windows per support and thus $F \cdot n_{sup}$ samples per bridge. Features use the same Type-A/Type-B feature sets but are computed for the target support only, i.e., from its scour-affected zone. As in detection, each batch vector combines order-independent statistics (mean, standard deviation, minimum, maximum) with the order-preserving combined per-crossing feature vectors, giving a dimension $(w+4) \cdot p$, where p is the per-crossing feature count. The batch label (scoured or healthy), i.e., the output is support-level, taken from the final crossing in the window for that support;

the corresponding support-level scour percentage and support identifier are stored for cluster-based severity evaluation (e.g., 5–10%, 10–15%, 15–20% scour), traceability, and the localisation readout. Windowed datasets are first built per bridge and then combined across bridges to train and evaluate the localisation models under both RILA front- and rear-mounted configurations.

5.2.2.3 Hyperparameter tuning and localisation readout

In the context of optimising the hyperparameters of ML algorithms employed for locating scoured support, random search is utilised for all classifiers (ANNs, RFs, and SVM-RBFs), as grid search is computationally exhaustive used to locate scoured support, random search is used for all classifiers (ANNs, RFs, and SVM-RBFs), as grid search is computationally exhaustive at the per-support-level. Initially, a comprehensive random search is conducted at a 7-batch size to identify the hyperparameter ranges that demonstrate optimal validation performance. These ranges are then used for transfer-tuning at batches 15, 31 and 61, with narrowed search spaces centred on the batch size-7 settings. All tuning is carefully executed within the nested CV, a precautionary measure to prevent leakage. Conversely, the use of the outer folds is reserved exclusively for evaluation. Further computational details are provided in Section 5.6.

After completing the external CV with fixed hyperparameters, batches that are both truly scoured and predicted as scoured (true positives) in the held-out folds are isolated. The associated support identifiers are then aligned with a standardised list of supports for each bridge (abutments and numbered piers). For each support, the number of true-positive batches is tallied and expressed as a fraction of all true-positive batches. The resulting per-support summary (support name and fraction) provides an interpretable readout of which supports are most consistently implicated by correctly detected scoured batches. Where relevant, summaries are reported separately by batch size and by sensor orientation.

5.2.2.4 ML models, feature scaling, evaluation metrics

As in Section 5.2.1.4, which summarises scour detection algorithms, the same three classifiers—ANN, RF, and SVM-RBF—are used on the support-level batch features. Unlike scour detection, the individual dataset comprises one row for each support in each simulation, yielding, per bridge, $N_{sim} \cdot n_{sup}$ observation rows. ANN and SVM use per-fold z-scoring, fit only on the outer-training partition, and are then applied unchanged to the inner-validation and outer-test data to avoid leakage; RF is trained on raw inputs. Decision rules mirror detection (hard labels at each model's default operating point). Mathematical details and tuning procedures are provided in Sections 5.5–5.6.

As in the scour detection ML models, performance is evaluated using Accuracy, Precision, Recall, and *F1* score, considering the scoured class as positive. Equations (5.7)–(5.8) are applied to pooled outer-test predictions, where *TP, FP, TN*, and *FN* are calculated over the combined dataset. Results for batch sizes 7, 15, 31, and 61 include batch-wise curves for cross-batch comparison. Evaluation uses hard class labels at a fixed operating point for its simplicity, consistency, and ease of reproduction across models and batch sizes.

5.3 Feature Engineering

This section defines the feature engineering pipeline for scour detection and localisation. It first introduces two input feature sets. Type A consists of DWT descriptors of the track profile. Type B is formed from fixed-length Markov parameters derived from ARX models, fitted either to full-bridge profiles for detection models or to the scour-affected zone around each support for scour locating models. A fallback strategy is described for cases where ARX fitting fails. The section then presents two feature selection strategies: relative-importance–based selection with grid search, and multi-objective selection using NSGA-II. Their outcomes are compared, and the adopted feature set is identified together with the normalisation rules applied to the selected parameters.

5.3.1 Deriving Input Features

Operational variables such as vehicle speed, train weight, and structural properties are not used as inputs because such metadata are often unavailable or unreliable in realistic field deployments. Instead, scour-sensitive features are derived from the simulated RILA track profiles. Two complementary feature sets are used:

- Type A: wavelet-based features selected from a discrete wavelet transform (DWT) of each track profile.
- Type B features are Markov parameters obtained from ARX models.

As explained in Section 5.2, both Type A and Type B features are derived from the entire track profile for scour detection-ML models, and from the scour-affected zones of each support for scour-locating ML models. The scour-affected zone of each support is considered to be $L/4$ portions of the span(s) that are supported by that pier/abutment. Therefore, scour detection-ML models have a single observation row for each simulation, whereas the scour-localisation models use $n_{supp}{}^{*}N_{sim}$ observation rows per simulation

5.3.1.1 Application of DWT to track profiles

The DWT is local, unlike the global FFT, enabling the detection of singularities, trends, and local variability (Cerna, 2019), which are useful for scour detection. It provides a compact, multi-resolution representation via an analysis filter bank, yielding approximation (low-frequency) and detail (high-frequency) coefficient sets (Cerna, 2019; Walnut, 2002). These sets are spatially local, thereby well-suited to capturing scour-induced shape changes. The track profile is a discrete sequence as expressed in Eq. (5.11):

$$\delta = \{\delta(x_n)\}_{n_s=0}^{N_s-1} \tag{5.11}$$

where N_s is the number of samples in δ, n is the discrete spatial sample index, and x_n is the spatial location of sample n_s on a uniform grid, $x_n = x_0 + n_s \cdot \Delta x$. The first step of the DWT analysis is to apply a low-pass filter, namely scaling filter $h[m]$, and a high-pass filter, i.e., wavelet filter $g[m]$, to the half-point symmetric extension of the track profile at the boundaries, i.e., δ^{sym}. These filters produce, respectively, a low-frequency approximation (coefficient a_{jk}) and a high-frequency detail (coefficient d_{jk}) representation that are local in space, as seen in Equations (5.12) and (5.13).

$$a_{1k} = \Sigma_{m=0}^{L_h-1} h[m] \cdot \delta^{sym}(2k-m) \tag{5.12}$$

$$d_{1k} = \Sigma_{m=0}^{L_h-1} g[m] \cdot \delta^{sym}(2k-m) \tag{5.13}$$

where k is the wavelet coefficient index at a given level j, L_h is the filter length, $L_h=2p_v$, p_v is the number of vanishing moments, k is the coefficient index: $k = 0, 1, \ldots, \lceil N/2^j \rceil - 1$, and δ^{sym} as seen in Eq. (5.14):

$$\delta^{sym}(x_n) = \{\delta(x_{-n-1}), (n<0);\ \delta(x_n),\ (0 \le n \le N-1);\ \delta(x_{2N-n-1}),\ (n \ge N)\} \tag{5.14}$$

Recursively applying the first step to the approximation yields Eq. (5.15) to (5.17); for j:

$$a_0(n) = \delta(n); \tag{5.15}$$

$$a_{jk} = \Sigma_{m=0}^{L_h-1} h[m] \cdot a_{j-1}^{sym}(2k-m) \tag{5.16}$$

$$d_{jk} = \Sigma_{m=0}^{L_h-1} g[m] \cdot a_{j-1}^{sym}(2k-m) \tag{5.17}$$

where $j = 1, 2, \ldots, J_D$, and J_D is the decomposition level. Reconstruction from level j to $j-1$ (upsample-by-2 and filter) is given by Eq. (5.18):

$$a_{j-1}(n) = \Sigma_k \underline{h}[n-2k] \cdot a_{jk} + \Sigma_k \dot{g}[n-2k] \cdot d_{jk} \tag{5.18}$$

where $\dot{h}, \dot{g}$ are the orthogonal cases: $\dot{h} = h, \dot{g} = g$. The discrete analysis/synthesis filters *h[m]* and *g[m]* are generated by the continuous two-scale relations for the scaling function ϕ and wavelet equation ψ, as seen in Eq. (5.19) and (5.20), respectively:

$$\phi(x) = \sum_{m=0}^{M-1} h[m]\sqrt{2}\phi(2x - m) \tag{5.19}$$

$$\psi(x) = \sum_{m=0}^{M-1} g[m]\sqrt{2}\phi(2x - m) \tag{5.20}$$

Wavelets with higher vanishing moments, such as Daubechies-8 (db4) or db8, yield smoother approximation coefficients that more effectively capture the signal structure compared to those with fewer vanishing moments. For healthy signals, db4 offers similarity to the original, whereas for signals affected by scour, db8 captures nonlinear and curved features more effectively. Increasing the wavelet order beyond db8 does not result in appreciable differences, indicating that an optimal balance has been reached. The 4th-level wavelet coefficients demonstrate sufficient sensitivity to scour-related changes. Therefore, each track profile is decomposed using an orthogonal Daubechies-8 wavelet to a decomposition level of J_D= 4.

5.3.1.2 DWTs for scour detection ML models

To observe scour-related changes in the wavelet coefficients, DWT coefficients of two track profiles for each bridge are presented. The coefficients of the RILA front-mounted track profiles are provided in Figure 5.2 to Figure 5.4 for the Colonnade, Figure 5.5 to Figure 5.7 for the River Teign, Figure 5.8 to Figure 5.10 for the WCM15A and Figure 5.11 to Figure 5.13 for the WCM15 bridges. The approximation coefficients (A_4) and level 4 detail coefficients, i.e., D_4, of one healthy, i.e., scour percentage is less than 5%, and one about 20% scoured-stage track profiles of the Colonnade bridge are seen in Figure 5.2. The healthy track profile coefficients are plotted in a solid curve, while the coefficients of the track profile with 20% scoured (pier 2) are shown with the dashed curve. In Figure 5.2, the scoured case shows a higher proportion of variance captured in A4, indicating that the profile is more dominated by low-frequency content, consistent with reduced global stiffness. Levels 3, 2, and 1 detail coefficients are provided in Figure 5.3 (a), Figure 5.3 (b), and 5.4. In the DWT, each coefficient contributes to the signal energy through its squared value. $D_j(i)$ denotes the i^{th} detail coefficient at level j; the contribution of that coefficient to the energy is $D_j(i)^2$. For a given detail level j, wavelet energy is the sum of these contributions over all n_z coefficients at that level, as in Eq. (5.21) for approximation, and in Eq. (5.22) for detail coefficients:

$$E_{A_j} = \sum_{i=1}^{n_z} A_j(i)^2 \tag{5.21}$$

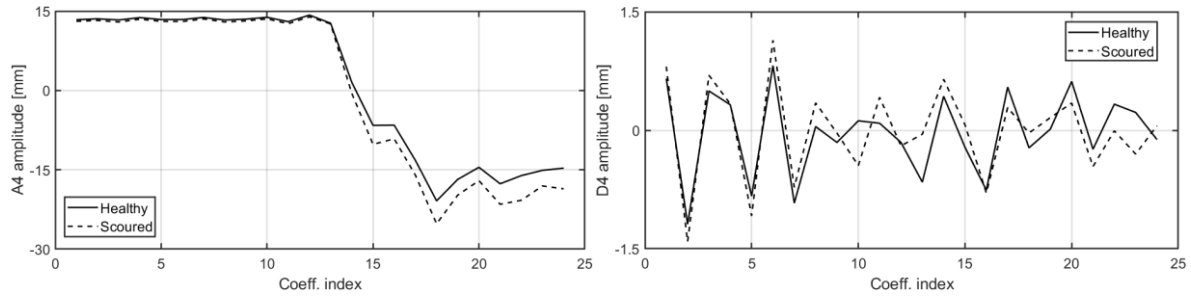
$$E_{D_j} = \sum_{i=1}^{n_z} D_j(i)^2 \tag{5.22}$$

where n_z = *length* (D_j) for detail, and n_z = *length* (A_j) for approximation coefficients. The mean absolute difference (MAD) for the approximation at level J_D is as in Eq. (5.23) for approximation and Eq. (5.24) for detail coefficients:

$$MAD_{A_{J_D}} = \left(\frac{1}{n_z}\right) \sum_{i=1}^{n_z} \left| A_{J_D}^{sco(i)} - A_{J_D}^{healthy(i)} \right| \tag{5.23}$$

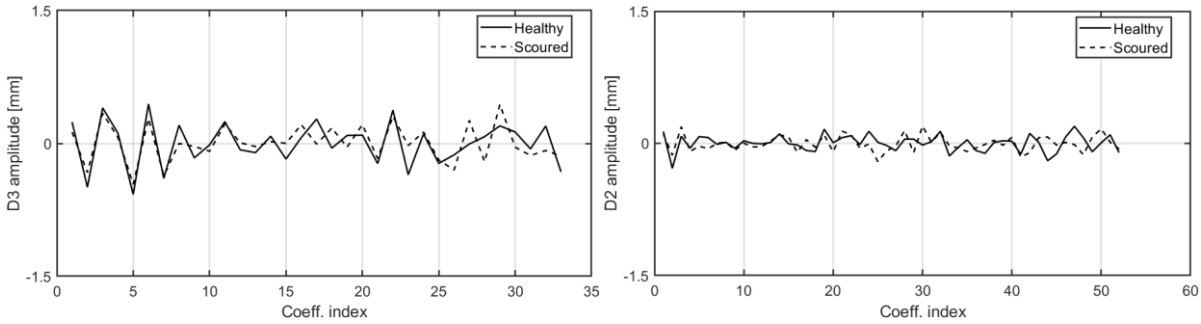
$$MAD_{D_j} = \left(\frac{1}{n_z}\right) \sum_{i=1}^{n_z} \left| D_j^{sco(i)} - D_j^{healthy(i)} \right| \tag{5.24}$$

where A_{JD}^{sco} and D_j^{sco} denote scoured, and $A_{JD}^{healthy}$ and $D_j^{healthy}$ *denote* healthy stage approximation and detail coefficients, respectively. Table 5.1 and Table 5.2 show scour percentages, mean absolute differences (MADs) and energy ratios of each detail and approximation coefficient for each bridge. The most evident scour-related differences appeared at level 4 (D_4), where the damaged case shows higher energy (ratio ≈1.25) and a larger MAD value (≈0.25 mm) compared with the healthy profile. Marked differences are also visible at level 2 (D_2), with an energy ratio of about 1.1 and a MAD value of about 0.08 mm. For the River Teign bridge, the most substantial scour-related differences are found at level 3 (D_3), where the scoured profile has approximately 1.3 times the energy of the healthy one, and a MAD value of about 0.15 mm. The following changes are most evident at level 1 (D_1), with an energy ratio of about 1.15 and a similar MAD of roughly 0.15 mm. For the WCM15A bridge, level 4 (D_4) shows the most substantial scour effect, where the damaged case has an energy ratio of about 1.7 and a MAD of about 0.19 mm. The second most pronounced difference arises at level 2 (D_2), where the energy ratio is approximately 1.25, and the MAD is about 0.12 mm. For WCM15, the most apparent scour-related differences occur at level 4 (D_4), where the scoured profile has higher energy (an energy ratio of about 1.3) and a MAD value of around 0.14 mm relative to the healthy case. A minor change is observed at level 1 (D_1), where the amplitudes show an offset with a MAD value of about 0.09 mm.



(a) (b)

Figure 5.2. DWT of front-mounted track profiles for the Colonnade bridge under healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4).



(a) (b)

Figure 5.3. DWT of front-mounted track profiles for the Colonnade bridge under healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2).

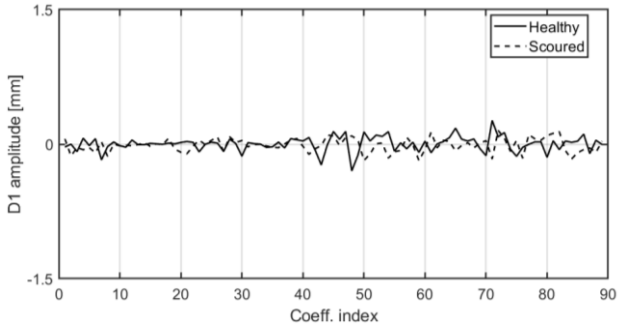
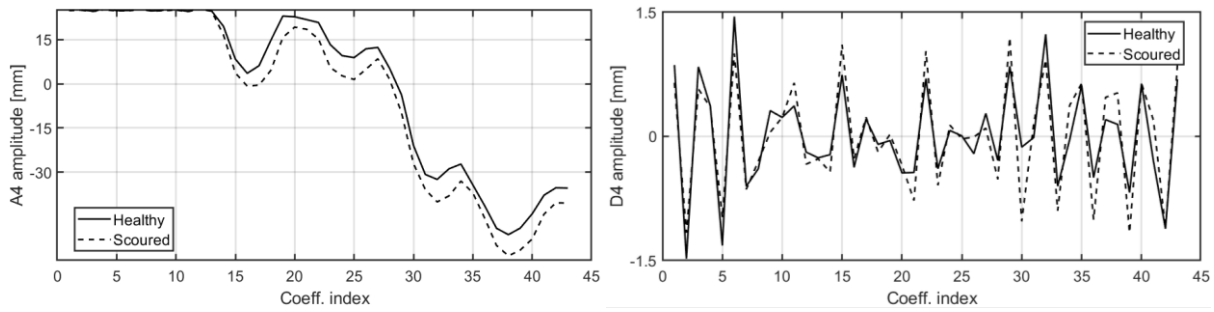


Figure 5.4. DWT of front-mounted track profiles for the Colonnade bridge under healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1).

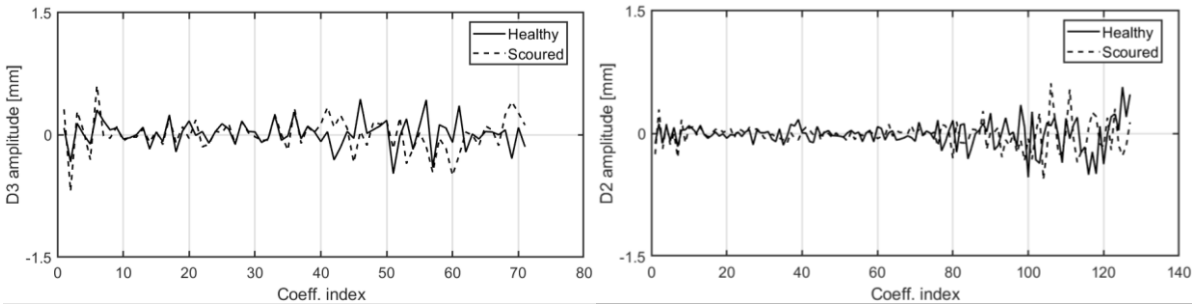
Table 5.1. Global DWT: Energy ratios and MAD values for A_4 and D_4 – RILA front-mounted case.

Bridge	**Healthy case scour percentage**	**Scoured case scour percentage**	**A4 energy ratio**	**A4 MAD (mm)**	**D4 energy ratio**	**D4 MAD (mm)**
WCM15	0.085	19.94	0.911	1.55	1.285	0.137
WCM15A	0.002	19.85	1.108	2.89	1.659	0.194
River Teign	0.010	19.95	1.167	4.21	1.143	0.227
Colonnade	0.023	19.91	1.223	1.69	1.193	0.232



(a) (b)

Figure 5.5. DWT of front-mounted track profiles for the River Teign bridge under healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4).



(a) (b)

Figure 5.6. DWT of front-mounted track profiles for the River Teign bridge under healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2).

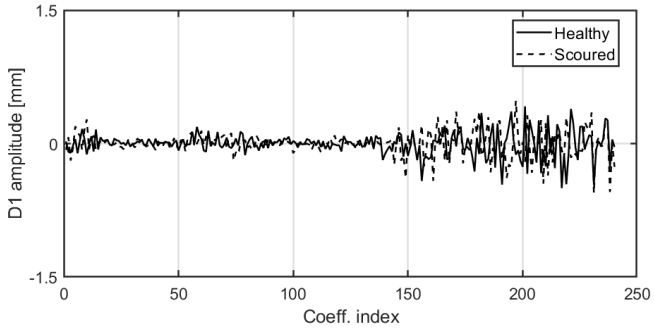
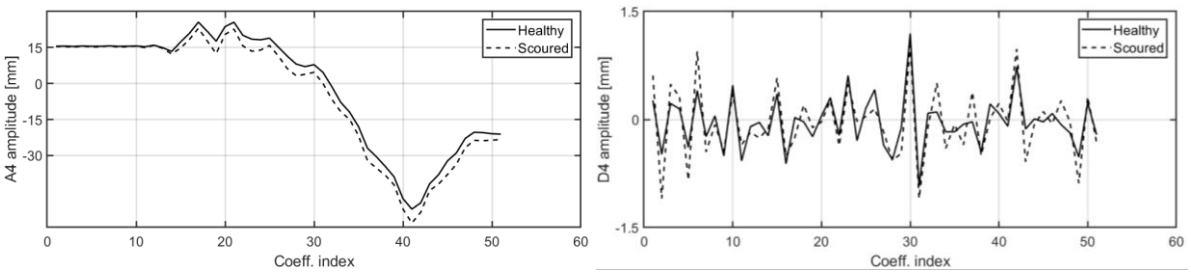


Figure 5.7. DWT of front-mounted track profiles for the River Teign bridge under healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1).

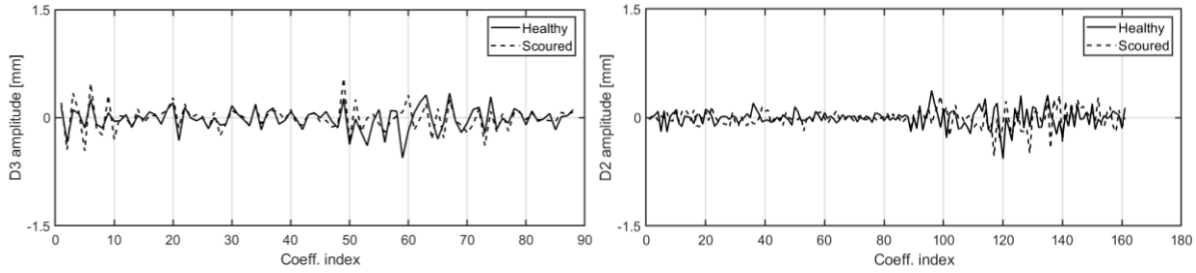
Table 5.2. Global DWT: Energy ratios and MAD values for D_3, D_2 and D_1 – RILA front-mounted case.

Bridge	D3 energy ratio	D3 MAD (mm)	D2 energy ratio	D2 MAD (mm)	D1 energy ratio	D1 MAD (mm)
WCM15	0.611	0.0944	0.867	0.082	0.867	0.0861
WCM15A	1.070	0.116	1.250	0.121	0.606	0.136
River Teign	1.335	0.145	0.917	0.156	1.146	0.147
Colonnade	0.670	0.128	1.084	0.0808	0.893	0.0849



(a) (b)

Figure 5.8. DWT of front-mounted track profiles for the WCM15A bridge under healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4).



(a) (b)

Figure 5.9. DWT of front-mounted track profiles for the WCM15A bridge under healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2).

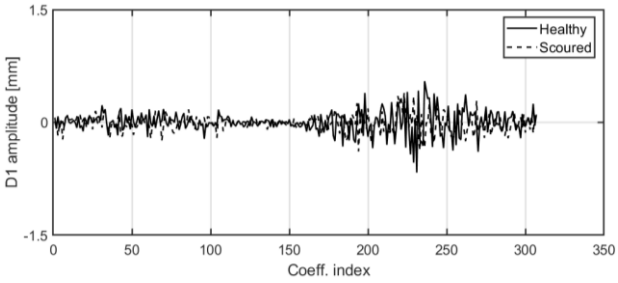
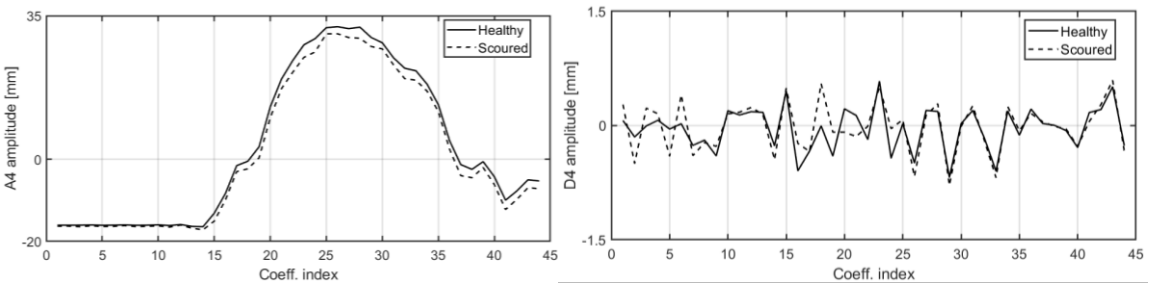
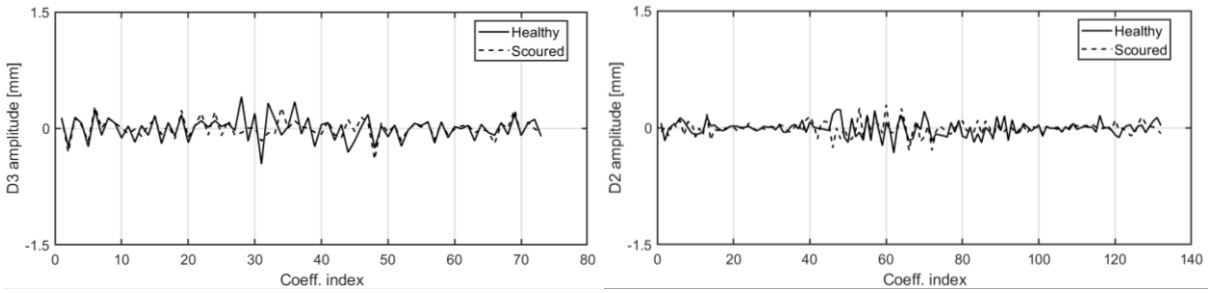


Figure 5.10. DWT of front-mounted track profiles for the WCM15A bridge under healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1).



(a) (b)

Figure 5.11. DWT of front-mounted track profiles for the WCM15 bridge under healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4).



(a) (b)

Figure 5.12. DWT of front-mounted track profiles for the WCM15 bridge under healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2).

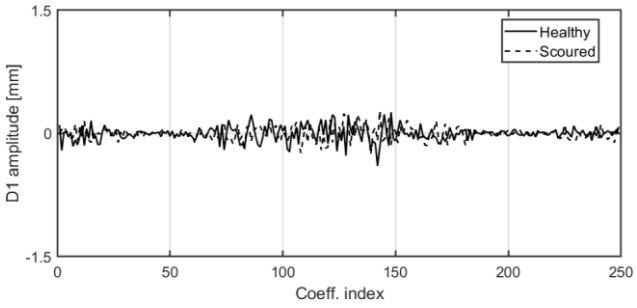
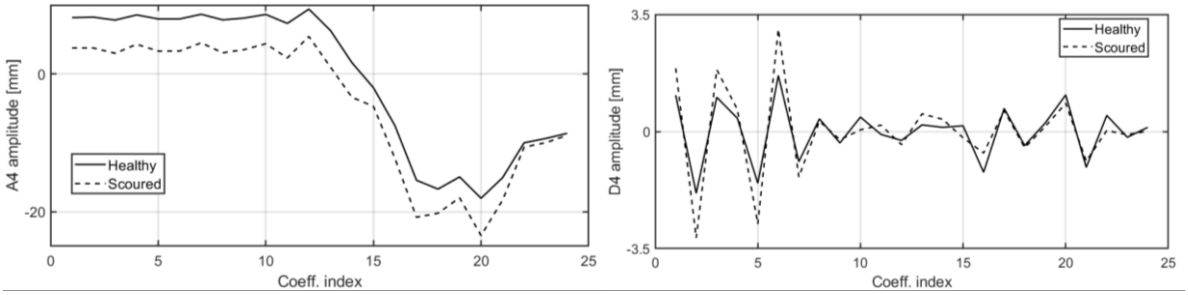


Figure 5.13. DWT of front-mounted track profiles for the WCM15 bridge under healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1).

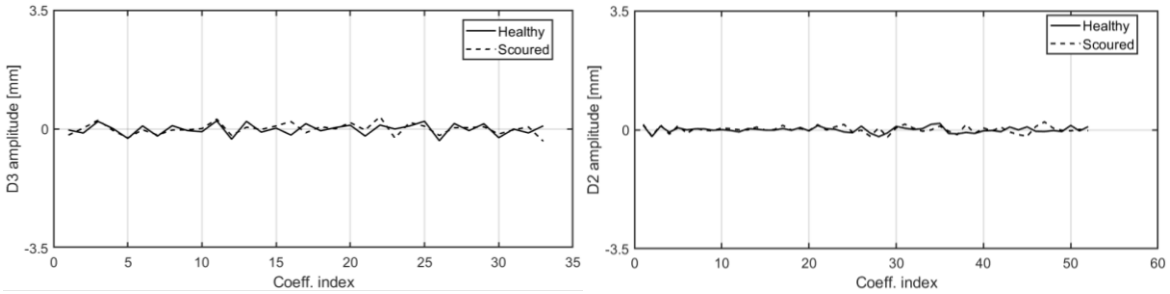
A similar analysis is conducted for rear-mounted track profiles. The coefficients of the RILA rear-mounted track profiles are provided in Figure 5.14Figure 5.16 for the Colonnade, Figure 5.17 to Figure 5.19 for the River Teign, Figure 5.20 to Figure 5.22 for the WCM15A and Figure 5.23 to Figure 5.25 for the WCM15 bridges. Table 5.3 and Table 5.4 show scour percentages, MAD values and energy ratios of each detail and approximation coefficient for each bridge. For the RILA rear-mounted case, the Colonnade bridge shows the most evident scour-related differences at level 4 (D_4), where the damaged case has more than twice the energy of the healthy one (energy ratio ≈ 2.30) and a MAD value of approximately 0.42 mm. Additional changes are visible at level 2 (D_2), with an energy ratio of approximately 1.49 and a MAD value of about 0.08 mm. For the River Teign bridge, the dominant effects also appear in D_4, with the scoured profile exhibiting approximately twice the energy of the healthy one (energy ratio ≈ 2.03) and a MAD value of about 0.40 mm, followed by level 3 (D_3), where the energy ratio is approximately 1.39. The MAD value is about 0.16 mm. For the WCM15A bridge, level 4 (D_4) again shows the most substantial scour effect, with an energy ratio of approximately 1.28 and a MAD value of about 0.18 mm, while level 2 (D_2) shows secondary differences (energy ratio ≈ 1.14, MAD value ≈ 0.12 mm). For WCM15, the most apparent scour-related differences occur at level 4 (D_4), where the scoured profile has an energy ratio of

approximately 1.31 and a MAD value of about 0.15 mm relative to the healthy case, with more minor but still visible changes at level 3 (D_3), where the energy ratio is approximately 1.17. The MAD value is about 0.09 mm.



(a) (b)

Figure 5.14. DWT of rear-mounted track profiles for the Colonnade bridge under healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4).



(a) (b)

Figure 5.15. DWT of rear-mounted track profiles for the Colonnade bridge under healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2).

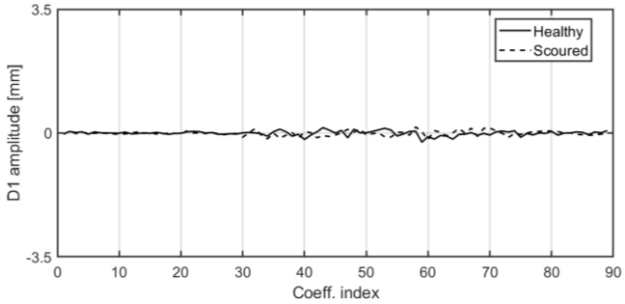
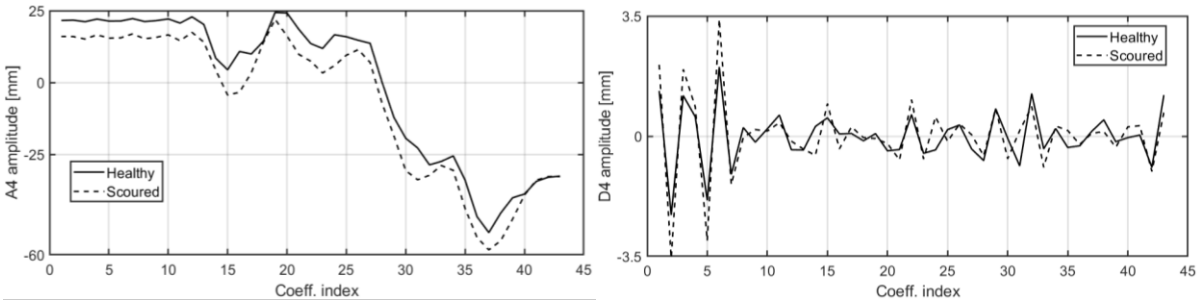
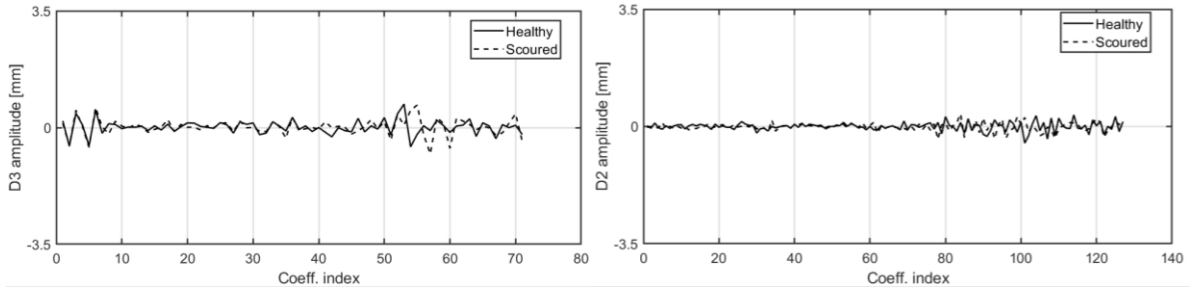


Figure 5.16. DWT of rear-mounted track profiles for the Colonnade bridge under healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1).



(a) (b)

Figure 5.17. DWT of rear-mounted track profiles for the River Teign bridge under healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4).



(a) (b)

Figure 5.18. DWT of rear-mounted track profiles for the River Teign bridge under healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2).

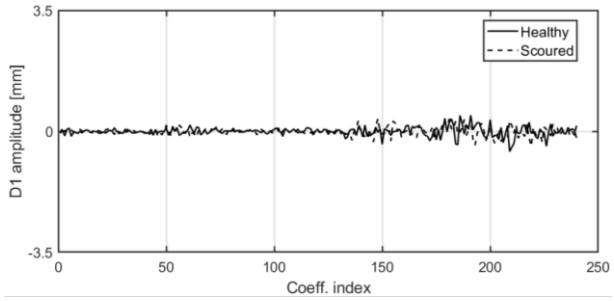
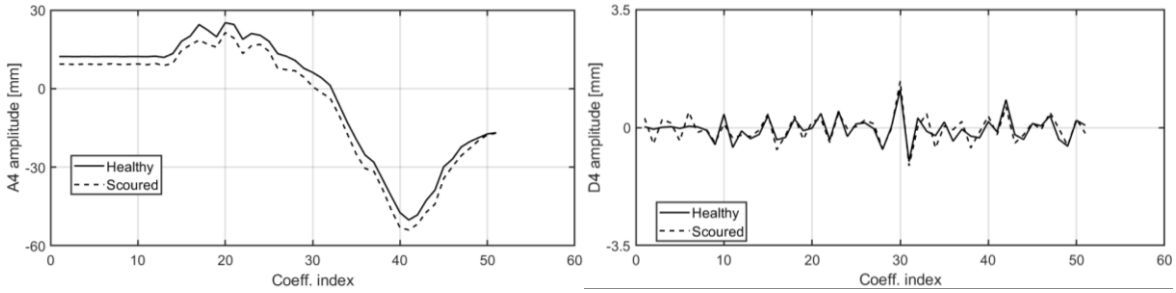


Figure 5.19. DWT of rear-mounted track profiles for the River Teign bridge under healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1).

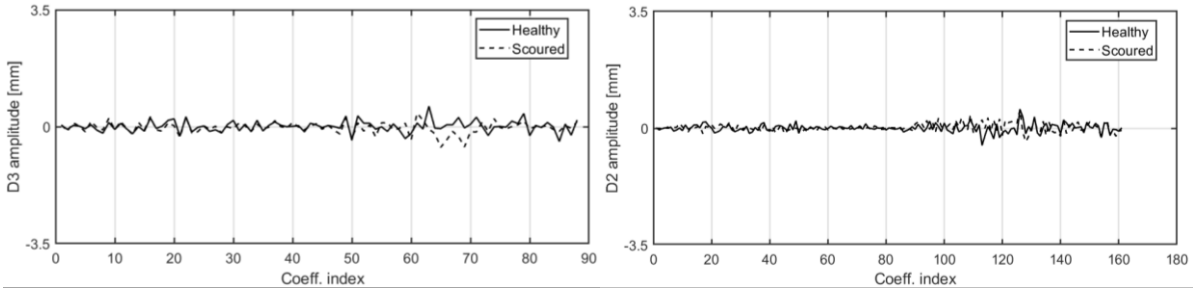
Table 5.3. Global DWT: Energy ratios and MAD values for A_4 and D_4 – RILA rear-mounted case.

Bridge	**Healthy case scour percentage**	**Scoured case percentage**	**A4 energy ratio**	**A4 MAD (mm)**	**D4 energy ratio**	**D4 MAD (mm)**
WCM15	0.031	19.983	0.968	1.93	1.309	0.151
WCM15A	0.046	19.944	1.056	3.80	1.281	0.180
River Teign	0.018	19.891	1.043	5.98	2.027	0.400
Colonnade	0.056	19.993	1.092	3.91	2.304	0.419



(a) (b)

Figure 5.20. DWT of rear-mounted track profiles for the WCM15A bridge under healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4).



(a) (b)

Figure 5.21. DWT of rear-mounted track profiles for the WCM15A bridge under healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2).

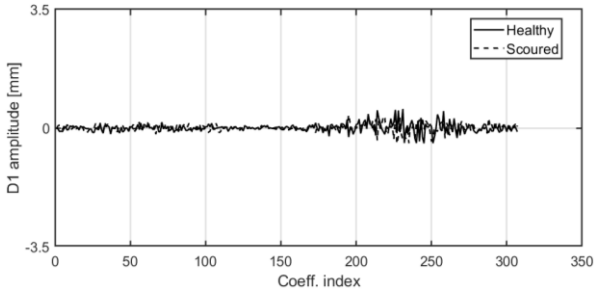
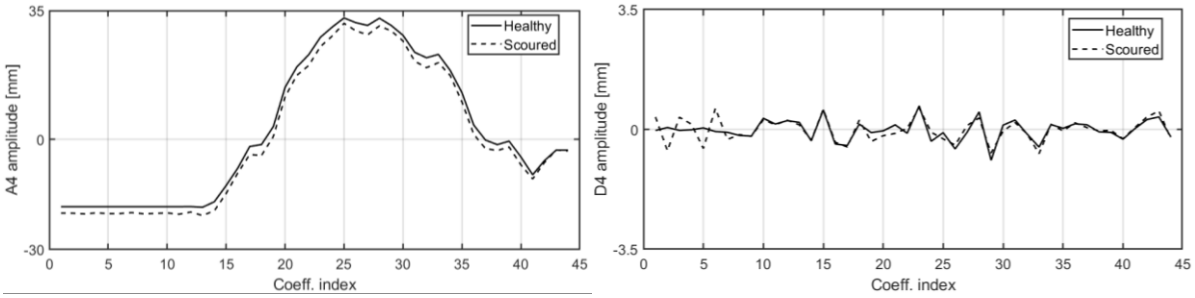


Figure 5.22. DWT of rear-mounted track profiles for the WCM15A bridge under healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1).

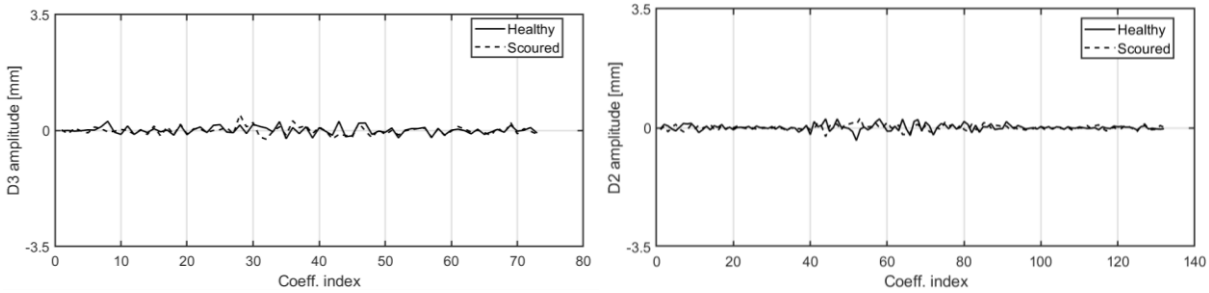
Table 5.4. Global DWT: Energy ratios and MAD values for D_3, D_2 and D_1 – RILA rear-mounted case.

Bridge	**D3 energy ratio**	**D3 MAD (mm)**	**D2 energy ratio**	**D2 MAD (mm)**	**D1 energy ratio**	**D1 MAD (mm)**
WCM15	1.166	0.0877	0.671	0.0872	0.945	0.0837
WCM15A	1.112	0.136	1.141	0.120	0.847	0.119
River Teign	1.393	0.155	0.947	0.117	0.897	0.120
Colonnade	1.016	0.131	1.488	0.0813	1.048	0.0756



(a) (b)

Figure 5.23. DWT of rear-mounted track profiles for the WCM15 bridge under healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4).



(a) (b)

Figure 5.24. DWT of rear-mounted track profiles for the WCM15 bridge under healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2).

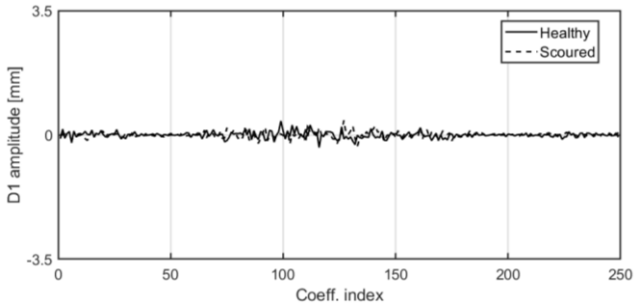


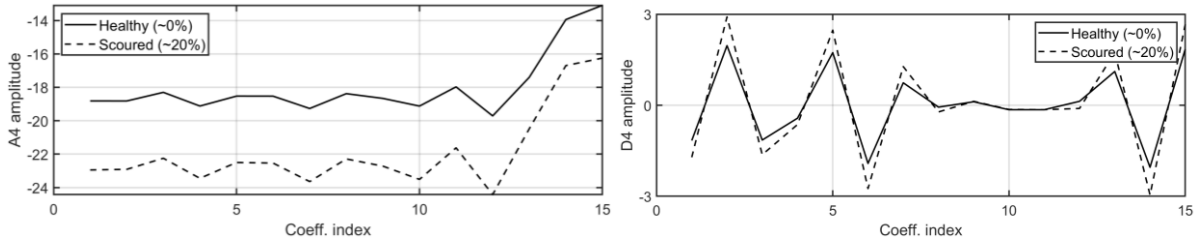
Figure 5.25. DWT of rear-mounted track profiles for the WCM15 bridge under healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1).

5.3.1.3 DWTs for scour locating ML models

In scour-locating ML models, for each train crossing, DWT is partially applied to the scoured-affected zones of each support, instead of the entire track profile. As explained earlier, scour-affected zone is defined here as the L/4 portions of the adjacent span(s) supported by that pier or abutment. Partial DWT coefficients of two supports are plotted to observe the scour effect: one support with near-zero scour, i.e., healthy, and another one approximately 20% scoured. Solid curves belong to healthy, and dashed curves belong to the scoured supports' coefficients. For the RILA front-mounted case, the coefficients of local

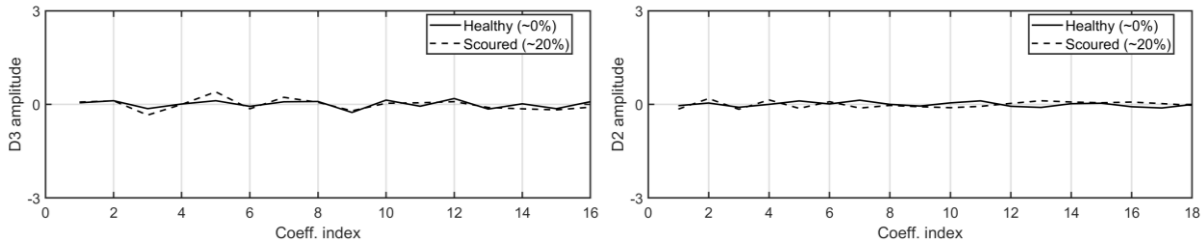
DWT are illustrated from Figure 5.26 to Figure 5.28 for the Colonnade, Figure 5.29 to Figure 5.31 for the River Teign, Figure 5.32 to Figure 5.34 for the WCM15A and Figure 5.35 to Figure 5.37 for the WCM15 bridges. Table 5.5 and Table 5.6 show scour percentages, scoured supports, MAD values and energy ratios of each detail and approximation coefficient for each bridge.

For the RILA front-mounted case, at level 4 (D_4), the scoured profile of the Colonnade bridge has more than twice the energy of the healthy one (energy ratio approximately 2.17) and a MAD value of about 0.49 mm. Marked changes also appear at levels 3, 2, and 1, with energy ratios of approximately 2.50, 1.55, and 1.91, and MAD values between about 0.09 mm and 0.11 mm. For the River Teign bridge, the most potent local effects are seen at level 3 (D_3), where the energy ratio is approximately 2.31. The MAD value is about 0.35 mm, followed by level 4 (D_4), where the scoured case has almost twice the energy (an energy ratio of approximately 1.96) and a MAD value of about 0.40 mm. For WCM15A, scour-related differences are more moderate and shift towards the approximation and lower detail levels. The A_4 coefficients show an energy ratio of approximately 1.31 with a MAD value of about 4.74 mm, while level 2 (D_2) exhibits a secondary effect (energy ratio approximately 1.24, MAD value around 0.17 mm), and D_4 displays only a mild reduction in energy. For WCM15, the main scour signature is concentrated at level 4 (D_4), where the scoured profile has an energy ratio of approximately 1.76 and a MAD value of about 0.24 mm. A_4 also shows a moderate increase (energy ratio approximately 1.45, MAD about 1.76 mm), whereas levels D_3, D_2 and D_1 display minor changes.



(a) (b)

Figure 5.26. DWT of scour affected zones of RILA front-mounted track profiles, Colonnade bridge, healthy and ≈20% scoured conditions: (a) A_4; (b) D_4.



(a) (b)

Figure 5.27. DWT of scour affected zones of RILA front-mounted track profiles, Colonnade bridge, healthy and ≈20% scoured conditions: (a) D_3; (b) D_2.

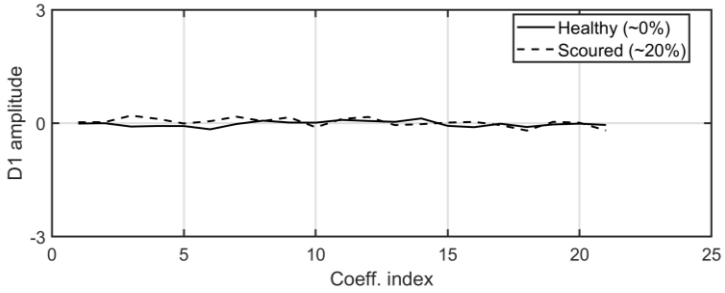
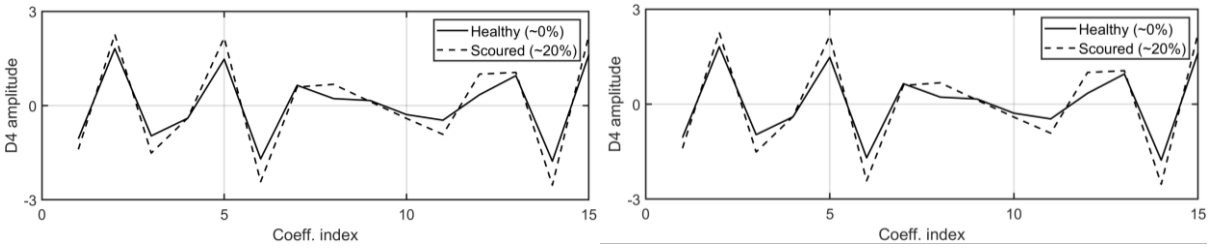
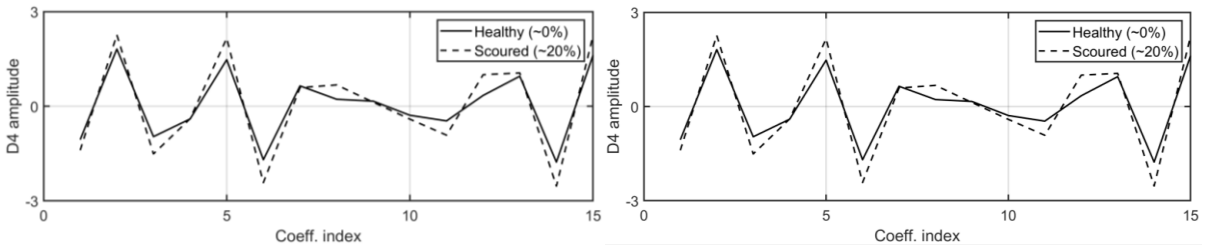


Figure 5.28. DWT of scour affected zones of RILA front-mounted track profiles, Colonnade bridge, healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1).



(a) (b)

Figure 5.29. DWT of scour affected zones of RILA front-mounted track profiles, River Teign bridge, healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4).



(a) (b)

Figure 5.30. DWT of scour affected zones of RILA front-mounted track profiles, River Teign bridge, healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2).

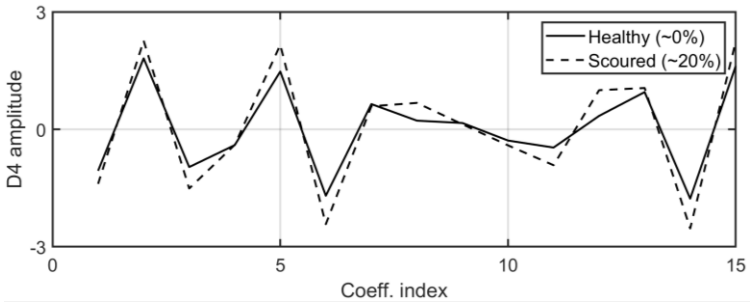
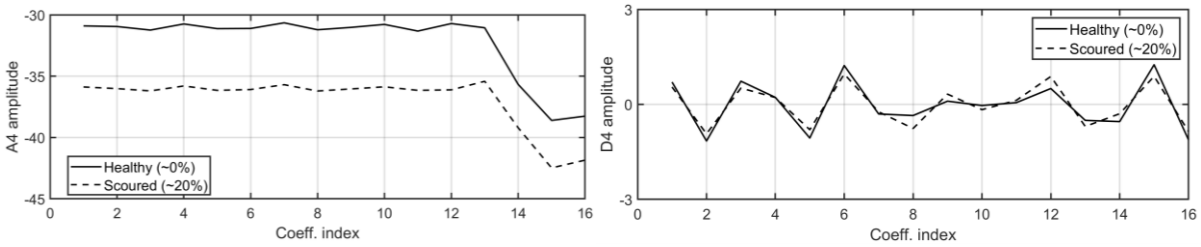
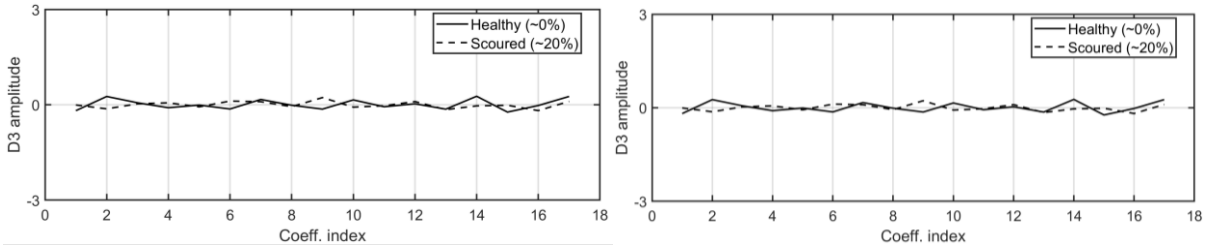


Figure 5.31. DWT of scour affected zones of RILA front-mounted track profiles, River Teign bridge, healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1).



(a) (b)

Figure 5.32. DWT of scour affected zones of RILA front-mounted track profiles, WCM15A bridge, healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4).



(a) (b)

Figure 5.33. DWT of scour affected zones of RILA front-mounted track profiles, WCM15A bridge, healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2).

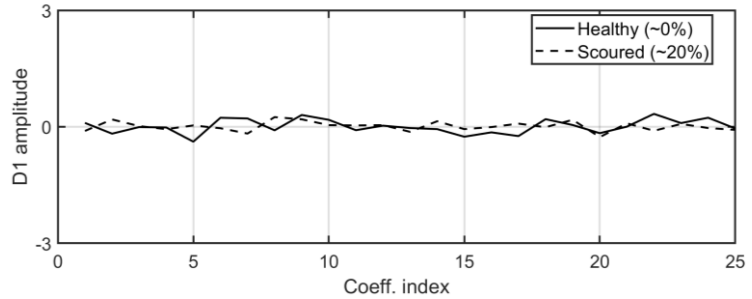
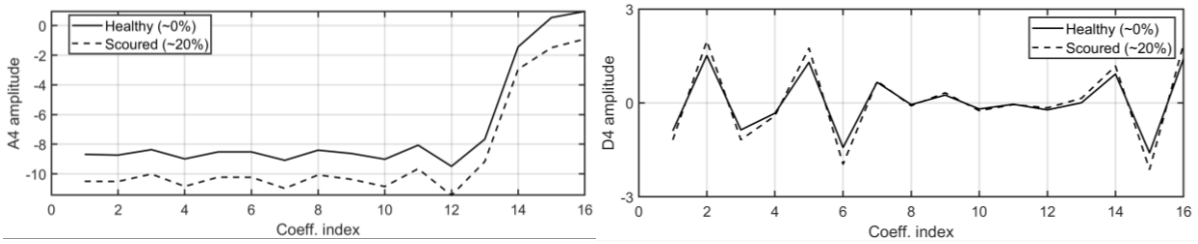


Figure 5.34. DWT of scour affected zones of RILA front-mounted track profiles, WCM15A bridge, healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1).

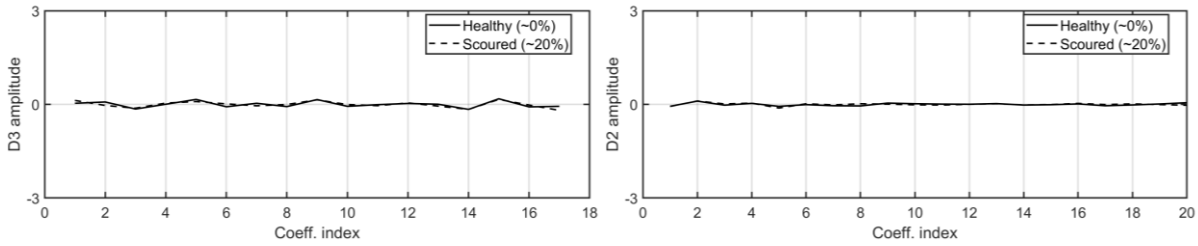
Table 5.5. Local DWT: Energy ratios and MAD values for A_4 and D_4 – RILA front-mounted case.

Bridge	Scoured support	Scour percentage	A4 energy ratio	A4 MAD (mm)	D4 energy ratio	D4 MAD (mm)
WCM15	2nd support	19.944	1.453	1.76	1.755	0.239
WCM15A	7th support	19.856	1.312	4.74	0.752	0.218
River Teign	5th support	19.955	1.401	7.76	1.959	0.404
Colonnade	3rd support	19.911	1.486	3.92	2.174	0.485



(a) (b)

Figure 5.35. DWT of scour affected zones of RILA front-mounted track profiles, WCM15 bridge, healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4).



(a) (b)

Figure 5.36. DWT of scour affected zones of RILA front-mounted track profiles, River Teign bridge, healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2).

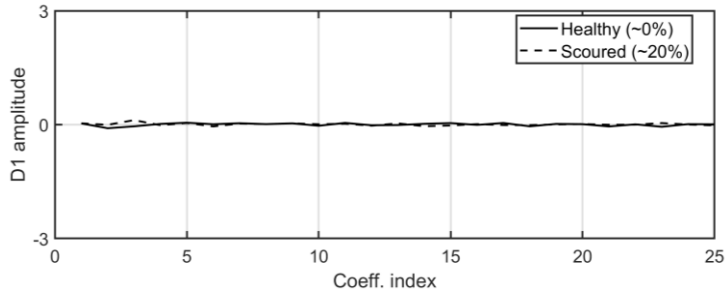


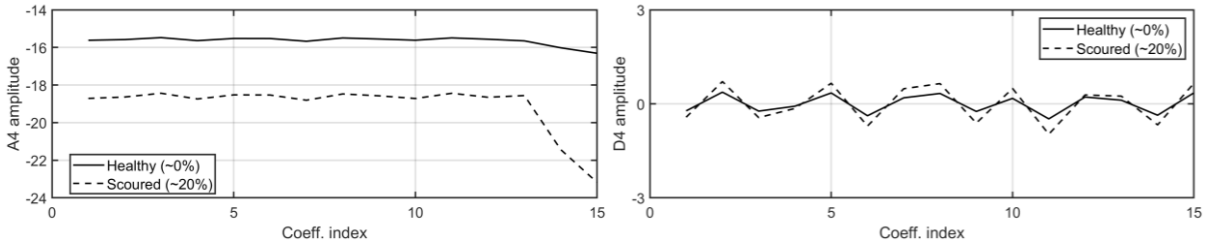
Figure 5.37. DWT of scour affected zones of RILA front-mounted track profiles, WCM15 bridge, healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1).

DWT coefficients of the partial track profiles for the RILA front-mounted case are provided in Figure 5.38 to Figure 5.16 for the Colonnade, Figure 5.41 to Figure 5.43 for the River Teign, Figure 5.44 to

Figure 5.46 for the WCM15A and Figure 5.47 to Figure 5.49 for the WCM15 bridges. Table 5.7 and Table 5.8 show scour percentages, MAD values and energy ratios of each detail and approximation coefficient for each bridge. At level 4 (D_4) of the River Teign bridge, the scoured profile has an energy ratio of approximately 14 and a MAD of approximately 0.93 mm. In contrast, the approximation A_4 changes only slightly, with an energy ratio of about 1.02 and a MAD of roughly 0.50 mm. For the Colonnade bridge, scour effects are also pronounced in D_4, where the energy ratio is about 3.90, and the MAD value is approximately 0.27 mm. These differences are reinforced at the approximation level, with A_4 showing an energy ratio of about 1.50 and a MAD of about 3.45 mm. For WCM15A, the dominant differences shift towards level 3 (D_3): the energy ratio is approximately 1.88, and the MAD is near 0.26 mm, while A4 shows a moderate increase (energy ratio about 1.27, MAD roughly 3.85 mm), and the remaining detail levels present more minor or mixed changes. For WCM15, scour-related differences are more distributed across scales. The approximation A_4 shows a moderate increase in energy (about 1.46, MAD around 1.75 mm), and the mid-level details D_2 and D_3 also rise (energy ratios approximately 1.51 and 1.41 with small MAD values), whereas D_1 exhibits a reduction in energy relative to the healthy case.

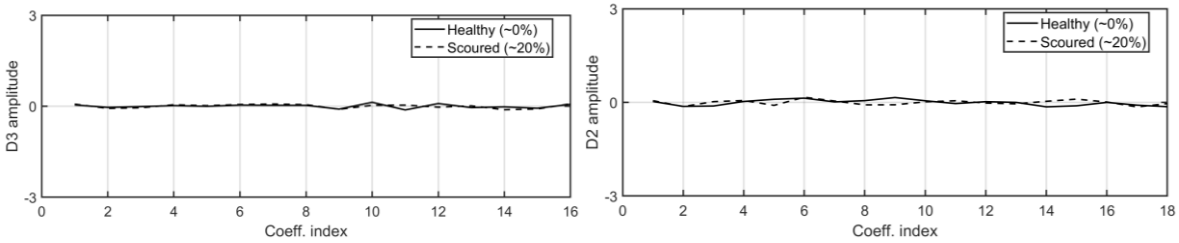
Table 5.6. Local DWT: Energy ratios and MAD values for D_3, D_2 and D_1 – RILA front-mounted case.

Bridge	D3 energy ratio	D3 MAD (mm)	D2 energy ratio	D2 MAD (mm)	D1 energy ratio	D1 MAD (mm)
WCM15	0.984	0.0563	1.070	0.0271	0.867	0.0371
WCM15A	0.451	0.160	1.242	0.167	0.449	0.187
River Teign	2.307	0.352	0.492	0.211	0.803	0.236
Colonnade	2.504	0.0938	1.549	0.109	1.905	0.0962



(a) (b)

Figure 5.38. DWT of scour affected zones of RILA rear-mounted track profiles, Colonnade bridge, healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4).



(a) (b)

Figure 5.39. DWT of scour affected zones of RILA rear-mounted track profiles, Colonnade bridge, healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2).

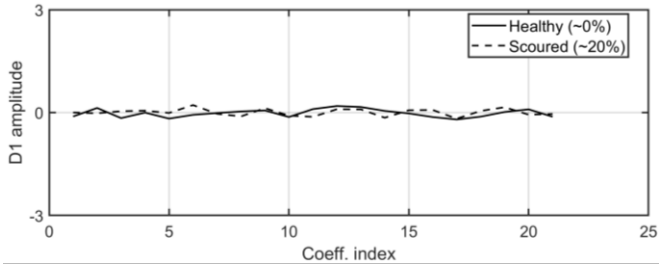
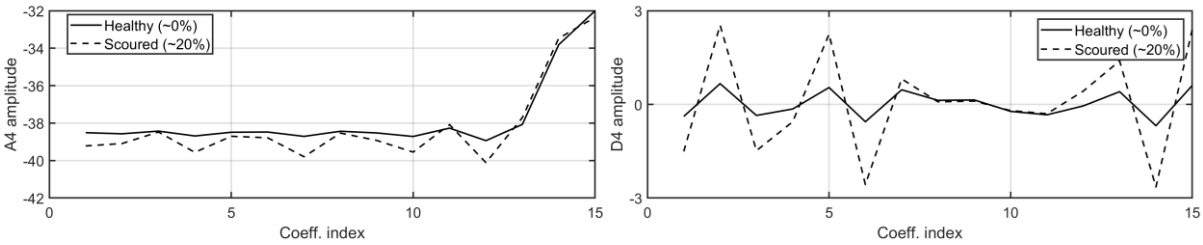
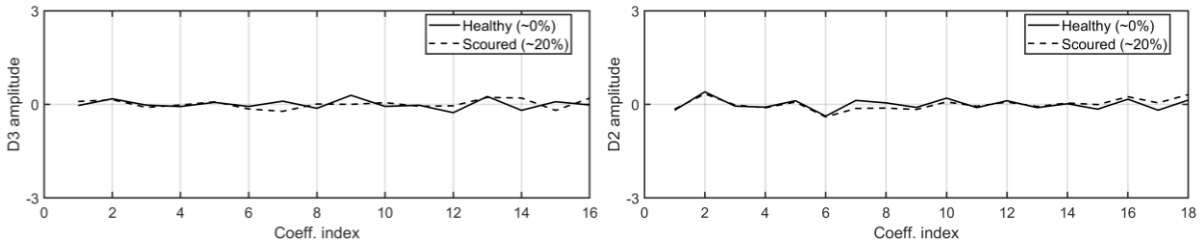


Figure 5.40. DWT of scour affected zones of RILA front-mounted track profiles, Colonnade bridge, healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1).



(a) (b)

Figure 5.41. DWT of scour affected zones of RILA rear-mounted track profiles, River Teign bridge, healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4).



(a) (b)

Figure 5.42. DWT of scour affected zones of RILA rear-mounted track profiles, River Teign bridge, healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2).

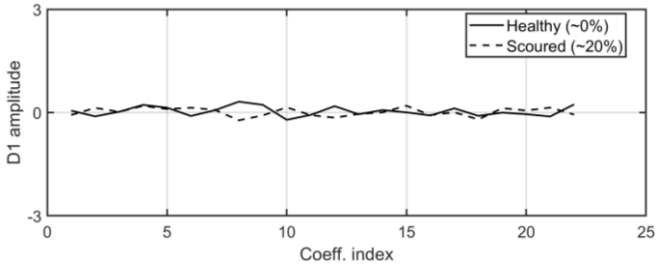
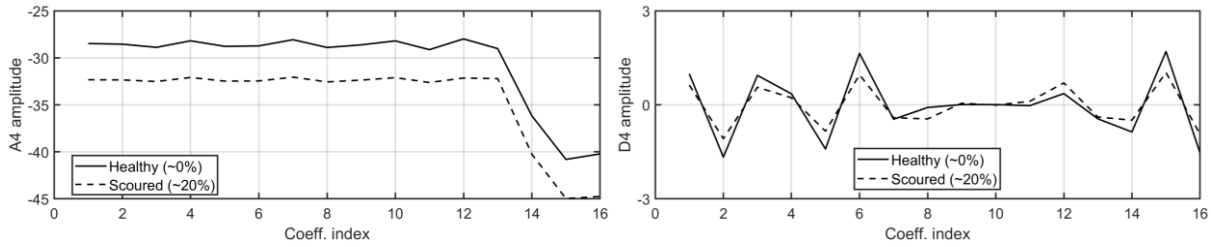
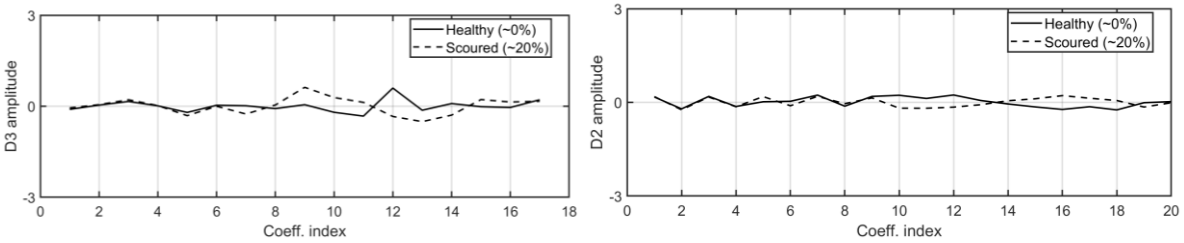


Figure 5.43. DWT of scour affected zones of RILA front-mounted track profiles, River Teign bridge, healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1).



(a) (b)

Figure 5.44. DWT of scour affected zones of RILA rear-mounted track profiles, WCM15A bridge, healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4).



(a) (b)

Figure 5.45. DWT of scour affected zones of RILA rear-mounted track profiles, WCM15A bridge, healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2).

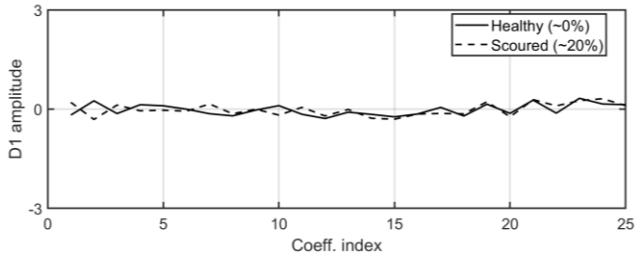
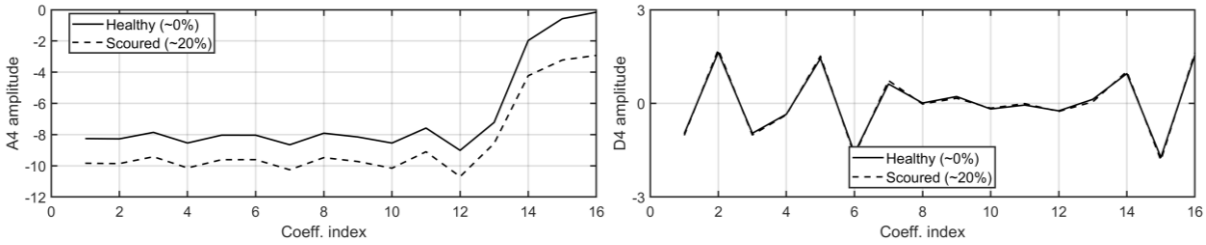
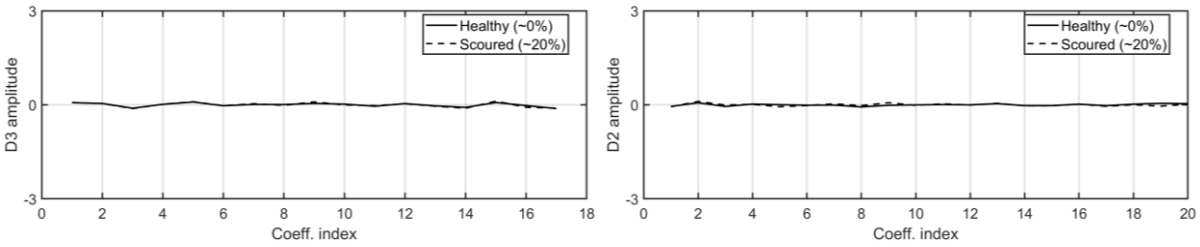


Figure 5.46. DWT of scour affected zones of RILA front-mounted track profiles, WCM15A bridge, healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1).



(a) (b)

Figure 5.47. DWT of scour affected zones of RILA rear-mounted track profiles, WCM15 bridge, healthy and ≈20% scoured conditions: (a) approximation coefficients (A_4); (b) detail coefficients at level 4 (D_4).



(a) (b)

Figure 5.48. DWT of scour affected zones of RILA rear-mounted track profiles, WCM15 bridge, healthy and ≈20% scoured conditions: (a) detail coefficients at level 3 (D_3); (b) detail coefficients at level 2 (D_2).

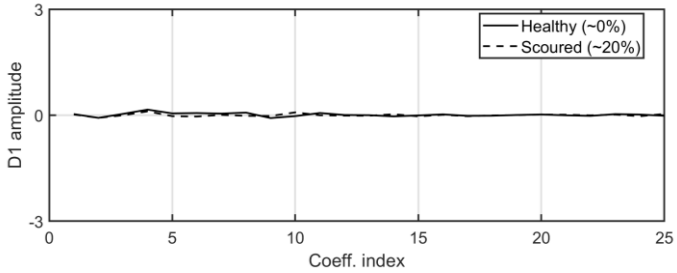


Figure 5.49. DWT of scour affected zones of RILA front-mounted track profiles, WCM15 bridge, healthy and ≈20% scoured conditions: detail coefficient at level 1 (D_1).

The coefficients with smaller apparent scour effects are not discarded. The comparisons of the energy and MAD values are based on a single healthy–scoured pair per bridge and per support. Coefficients that appear weak in this two-run view may still carry consistent, smaller-scale changes across the full range of trains, speeds and scour scenarios. If such coefficients are extracted a priori, any information contained in their higher-order statistics that are formulated in Section 5.3.1.4 and any interaction with more strongly affected levels could be irreversibly removed. By retaining all wavelet levels in the feature set, a complete local DWT representation is preserved. The decision on which coefficients are informative is deferred to the data-driven stage rather than imposed manually based on a limited energy/MAD inspection.

Table 5.7. Local DWT: Energy ratios and MAD values for A4 and D4 – RILA rear-mounted case.

Bridge	Scoured support	Scour percentage	Scoured case per-	A4 energy ratio	A4 MAD (mm)	D4 energy ratio	D4 MAD (mm)
WCM15	Pier 5	0.031	19.983	1.456	1.75	1.115	0.0614
WCM15A	Pier 6	0.046	19.944	1.265	3.85	0.423	0.335
River Teign	Abutment 1	0.018	19.891	1.021	0.5	13.976	0.931
Colonnade	Abutment 2	0.056	19.993	1.495	3.45	3.904	0.27

Table 5.8. Local DWT: Energy ratios and MAD values for D_3, D_2 and D_1 – RILA rear-mounted case.

Bridge	D3 energy ratio	D3 MAD (mm)	D2 energy ratio	D2 MAD (mm)	D1 energy ratio	D1 MAD (mm)
WCM15	1.414	0.0188	1.507	0.0273	0.532	0.0374
WCM15A	1.883	0.255	0.879	0.167	1.178	0.149
River Teign	0.864	0.152	0.973	0.0902	0.756	0.16
Colonnade	1.040	0.0511	0.770	0.0888	0.775	0.128

5.3.1.4 Type A (DWT-based) features

Each crossing is decomposed with db8 to the maximum level $J_D = 4$, yielding the level-4 approximation vector $A_4=\{a_4[k]\}$ and the detail vectors $D_j=\{d_j[k]\}$ for $j=1,\ldots,4$. Then, for all coefficient vectors $z \in \{A_4, D_1, D_2, D_3, D_4\}$ with length n_z, a total of 12 vectors are derived. These are the skewness and kurtosis of A_4 and of each D_j (five skewness + five kurtosis), the total wavelet-domain energy, and the minimum of A_4.

- Skewness of approximation and detail coefficients $A_4, D_1, \ldots, D_4$, in Eq. (5.25):

$$Skw(z) = \frac{1}{n_z}\sum_{i=1}^{n_z}\left(\frac{z_i-\mu_z}{\sigma_z}\right)^3 \tag{5.25}$$

- Kurtosis of approximation and detail coefficients $A_4, D_1, \ldots, D_4$, in Eq. (5.26):

$$Krt(z) = \frac{1}{n_z}\sum_{i=1}^{n_z}\left(\frac{z_i-\mu_z}{\sigma_z}\right)^4 \tag{5.26}$$

- Total wavelet-domain energy, by Parseval's theorem (Murguía & Rosu, 2011) in Eq. (5.27):

$$E_{tot} = \sum_{i=1}^{n_z} A_4(i)^2 + \sum_{j=1}^{J_D} \sum_{i=1}^{n_z} \left(D_j{}^2(i)\right) \tag{5.27}$$

- Minimum value of the approximation coefficient, in Eq. (5.28):

$$L_{min}^{A} = \min_{i} A_4(i) \tag{5.28}$$

where z denotes any of the DWT coefficient vectors in the set $\{A_4, D_1, D_2, D_3, D_4\}$, μ_z is the sample mean, and σ_z is the standard deviation. This set targets the two scour effects expected in the spatial track profile: (i) a span-scale downward shift and long-wavelength baseline distortion, and (ii) localised irregularities near the affected pier. Skewness captures asymmetry in the coefficient distributions, whereas kurtosis responds to heavy tails produced by sharp, local changes in slope and curvature. The total wavelet energy aggregates amplitude across scales and is sensitive to the overall strength of the deformation pattern. The minimum of A_4 isolates the most significant downward dip of the long-wavelength envelope, which deepens with scour. Levels 1 to 4 jointly cover the pier-vicinity scales (D_1 to D_3) and the span-scale trend (A_4), providing complementary sensitivity with low redundancy.

The contribution of Type A features is examined using an RI analysis following the method of Cortez & Embrechts (2013). Their framework evaluates the dispersion of sensitivity responses y_a to each input feature x_a using several measures, such as the gradient, variance, and median. The variance S_v, formulated in Eq. (5.29), is selected because it quantifies distribution in a statistically interpretable way and is less influenced by outliers than the range.

$$S_v = \frac{\sum_{j=1}^{L_e} \left(y_a^j - mean(y_a)\right)^2}{L_e - 1} \tag{5.29}$$

The RI of each parameter, r_a is then defined as seen in Eq. (5.30):

$$r_a = \frac{M_a}{\sum_{i=1}^{Mn} M_i} \tag{5.30}$$

where $j = 1,2, \ldots L_e$, L_e is the number of sensitivity evaluations carried out for each input feature, chosen L=100, i=1,2,..., M_n, M_n denotes the total number of parameters, and $M_a = S_v$ is the sensitivity measure chosen for feature a. Conducting this analysis has revealed that 1-2 Type A features carried overwhelmingly high importance in both the RILA front-mounted and rear-mounted models. The remaining parameters had negligible contributions. Such an ML model risks fitting incidental correlations specific to this dataset rather than genuine scour-sensitive behaviour, thus reducing its generalisation capability. To

overcome this limitation, an additional set of features—Type B—is introduced based on system-identification models of the track profile dynamics.

5.3.1.5 Type B (ARX-based) features

Scour detection ML models

The observed track profile along the bridge, denoted in Eq. (5.31):

$$\delta(x) = \xi(x) + z(x) \tag{5.31}$$

is composed of the rail irregularity segment $\xi(x)$ and the bridge displacement response $z(x)$. A single-input single-output ARX model is employed, with the rail irregularity serving as the external input excitation and the track profile as the output. In the scour-detection ML models, one ARX model is fitted per simulation. In ARX modelling, the input must be exogenous, i.e., independent of the system response and disturbances. Here, the rail irregularity $\xi(x)$ is exogenous to scour-related changes; namely, it remains unchanged when bridge stiffness changes. Therefore, $\xi(x)$ is used as the input and the measured track profile $\delta(x)$ as the output, which embeds the bridge displacement response $z(x)$. This choice reduces estimation bias and ensures the identified Markov orders reflect bridge dynamics rather than input–output leakage. With a spatial sampling step $T_s = 0.2^m$, the ARX model is formulated in Eq. (5.32), with Eqs. (5.33) and (5.34):

$$A(q^{-1})\delta(n) = B(q^{-1})q^{-nk}\xi(n) + e(n) \tag{5.32}$$

$$A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} + \cdots + a_n q^{-n_a} \tag{5.33}$$

$$B(q^{-1}) = b_1 + b_2 q^{-1} + \cdots + b_{n_b} q^{-(n_b-1)} \tag{5.34}$$

q^{-1} is the backward shift operator: $q^{-1}\delta(n)=\delta(n-1)$. $A(q^{-1})$ is the autoregressive polynomial that encodes the dependence of the output $\delta(n)$ on its own past values. $B(q^{-1})$ in Equation 24 is the input polynomial, representing the contribution of previous input samples, n_a and n_b are the autoregressive and input orders, n_k=1 is the fixed input delay, i.e., one spatial step, and $e(n)$ denotes the residual. ARX model orders are selected per simulation by sweeping $n_a = n_b \in [1,60]$ and choosing the minimiser of the Akaike Information Criterion (AIC). The AIC is used for model selection because it enables comparing competing models by estimating the relative information loss. Lower AIC values indicate models that achieve a better trade-off between goodness of fit and complexity. (Akaike, 1974).

When the ARX model in Eq. (5.32) cannot be reliably fitted for a given track profile, the corresponding simulation is not discarded. Instead, its ARX coefficients are reconstructed from simulations where identification succeeded. Failures typically occur when the regressor matrix built from past output and input

samples becomes nearly rank-deficient or numerically unstable. For example, when the track profile $\delta(n)$ has minimal variance or when the effective excitation from $\xi(n)$ is weak. In this case, the least-squares identification becomes numerically unstable because the solution is obtained from the normal equations, i.e., $\Phi^T\Phi\theta=\Phi^T y$, where Φ is the regressor matrix and θ collects the ARX coefficients $\{a, b\}$. The term "normal" refers to the orthogonality condition at the least-squares optimum: the residual $y-\Phi\theta$ is normal (perpendicular) to the column space of Φ, i.e., $\Phi^T(y-\Phi\theta)=0$. When Φ is ill-conditioned, the normal matrix $\Phi^T\Phi$ becomes ill-conditioned as well, so minor numerical errors can lead to significant changes in the estimated coefficients and no acceptable $\{a, b\}$ set is obtained. This situation was most pronounced in some River Teign rear-mounted simulations, where several simulations exhibited weak or degenerate excitation. To recover ARX coefficients for these failed cases, a small descriptor vector in Eq. (5.35) is defined:

$$d_j = [\sigma_\delta(j),\, r_\delta(j),\, \rho_{\delta\xi}(j)]^{\mathsf{T}} \tag{5.35}$$

is first computed for each simulation j, where $\sigma_\delta(j)$ is the standard deviation of the track profile $\delta(n)$, $r_\delta(j)$ is its root-mean-square value, and $\rho_{\delta\xi}(j)$ is the correlation coefficient between track profile $\delta(n)$ and the rail irregularity $\xi(n)$. These descriptors are standardised using the mean μ_d and standard deviation σ_d over the simulations with successful ARX fits, as in Eq. (5.36):

$$Z_j = (d_j - \mu_{\boldsymbol{d}})./\sigma_d \tag{5.36}$$

and Euclidean distances $\|Z_j-Z_k\|_2$ are used to identify a small neighbourhood N_j of K nearest neighbours (KNN) among the successful simulations. For each neighbour $k\in N_j$, the cropped ARX coefficient vectors $a_k\in R^{K_A}$ and $b_k\in R^{K_B}$ with orders truncated to common lengths K_A and K_B are available. This truncation ensures a consistent feature dimension across simulations. It retains only the lower-order coefficients—namely the first K_A autoregressive terms $a_1,\ldots, a_{KA}$ and the first K_B input terms $b_1,\ldots,b^K_B$, which are reliably identified for all successful ARX fits. The imputed coefficients for the failed simulation j are then obtained as a convex combination of the local KNN mean and the global median (med) over all successful simulations S, as in Eq. (5.37) and (5.38):

$$\tilde{a}_j = \alpha\, \frac{1}{|\mathcal{N}_j|} \sum\nolimits_{k\in\mathcal{N}_j} a_k \;+\; (1-\alpha)\, \mathrm{med}_{k\in S}(a_k) \tag{5.37}$$

$$\tilde{b}_j = \alpha\, \frac{1}{|\mathcal{N}_j|} \sum\nolimits_{k\in\mathcal{N}_j} b_k \;+\; (1-\alpha)\, \mathrm{med}_{k\in S}(b_k) \tag{5.38}$$

where $0 < \alpha < 1$ is a weight that controls the contribution of local and global information in the imputation scheme, $\tilde{a}_j$ denotes the imputed autoregressive coefficient vector for simulation j, $\tilde{b}_j$ denotes the imputed input coefficient vector for simulation j. $\tilde{a}_j$ and $\tilde{b}_j$ are replacement coefficients. The components of $\tilde{a}_j$

and $\tilde{b}_j$ are then clipped within robust bounds defined by the global medians (med) and median absolute deviations (MDAD), as in Eq. (5.39):

$$\mathrm{med}_S(a_k) \pm 6\,\mathrm{MDAD}_S(a_k), \mathrm{med}_S(b_k) \pm 6\,\mathrm{MDAD}_S(b_k) \tag{5.39}$$

Here, the factor 6 is used as a robust outlier–clipping constant. It defines a wide envelope that is approximately equivalent to ±4 standard deviations for approximately Gaussian coefficients, i.e., ±6 median absolute deviation, so that only extreme imputed values are truncated while the natural variability of the ARX coefficients is preserved. After the clipping step, a minimal perturbation term is added so that imputed coefficient vectors are not identical across simulations. This prevents numerical degeneracy without changing the coefficients. For the k^{th} *a*–coefficient of simulation *j*, a perturbation $\varepsilon_{j,k}^{(a)}$ is defined as in Eq. (5.40):

$$\varepsilon_{j,k}^{(a)} = \eta_s\,\mathrm{MDAD}_S(a_k)\sin\left(2\pi\,\varphi\,(j+\kappa_a k)\right) \tag{5.40}$$

and the stored coefficient becomes as in Eq. (5.41):

$$\hat{a}_{j,k} = \tilde{a}_{j,k} + \varepsilon_{j,k}^{(a)} \tag{5.41}$$

Similarly, for the k^{th} *b*–coefficient, the perturbation term and the b-coefficient become as in Eq. (5.42) and (5.43), respectively:

$$\varepsilon_{j,k}^{(b)} = \eta_s\,\mathrm{MDAD}_S(b_k)\cos\left(2\pi\,\varphi\,(j+\kappa_b k)\right) \tag{5.42}$$

$$\hat{b}_{j,k} = \tilde{b}_{j,k} + \varepsilon_{j,k}^{(b)} \tag{5.43}$$

where $\eta_s = 5\times10^{-4}$ is a small, dimensionless scaling factor; $\varphi = (\sqrt{5}-1)/2$ is the conjugate of the golden ratio, used to generate quasi-distinct phases across simulations and coefficient indices. κ_a and κ_a are fixed integer offsets that ensure different coefficients receive different phases. The imputed ARX polynomials for simulation *j* are as in Eq. (5.44):

$$A_j(q^{-1}) = 1 + \tilde{a}_{j,1}q^{-1} + \cdots + \tilde{a}_{j,K_A}q^{-K_A}, B_j(q^{-1}) = \tilde{b}_{j,1} + \cdots + \tilde{b}_{j,K_B}q^{-(K_B-1)} \tag{5.44}$$

This procedure ensures that every simulation, including those for which direct ARX estimation fails, notably several River Teign runs, is associated with a consistent ARX representation of fixed length.

Each simulation of four bridges yielded an ARX model with potentially different polynomial orders and, consequently, a different number of coefficients *(a,)* and *b(,)*. This variability complicates their direct use as ML input features, as the feature dimension varies across simulations and bridges. To obtain consistent-length features across simulations and bridges, the ARX models have been re-expressed in terms

of their Markov parameters, i.e., the impulse response coefficients (Phan & Longman, 1996) of the transfer function as expressed in Eq. (5.45):

$$G(q^{-1}) = \frac{B(q^{-1})}{A(q^{-1})} q^{-nk} \tag{5.45}$$

The Markov parameters are defined recursively as in Eq. (5.46):

$$g(k) = \tilde{b}(k) - \sum_{i=1}^{\min(n_a, k-1)} a_i \cdot g(k-i)\,,\ k = 1,2,\dots H \tag{5.46}$$

where k is the impulse index, the delay-aligned input coefficient $\tilde{b}(k)$ is defined as $\tilde{b}(k) = b_{k-nk+1}$ if $1 \le k-n_k+1 \le n_b$, and zero otherwise. In the context of this study, $g(k)$ quantifies how much the track profile moves $k-1$ samples after a unit impulse in the rail irregularity. For the A fixed horizon of $H = 40$ samples (approximately 8 m at $\Delta x = 0.2$ m) is adopted for all simulations. The Markov horizon is chosen based on an energy-based diagnostic of the impulse responses, performed across all simulations. For scour detection ML models, for each simulation j, the Markov vector $g_j=[g_j(1), \dots, g_j(H)]$ is first computed. The cumulative impulse-response energy up to lag h was defined as in Eq. (5.47):

$$E_{\text{cum},j}(h) = \sum_{k=1}^{h} g_j(k)^2\,; \qquad h = 1, \dots, H \tag{5.47}$$

with total impulse-response energy of simulation j, i.e., $E_{tot_imp,j} = E_{cum,j}(H)$. The fraction of energy captured by horizon h was then given by Eq. (5.48):

$$f_j(h) = \frac{E_{\text{cum},j}(h)}{E_{tot_imp,j}} \tag{5.48}$$

For each simulation, the smallest horizon H_j satisfying $f_j(H_j) \ge 0.95$ is identified; if the 95% threshold was not reached before H, the whole horizon was assigned. A robust global summary was then obtained as $H^* = \text{median}_j(H_j)$. The chosen fixed horizon, $H = 40$ samples (8 m at $\Delta x = 0.2$ m), exceed this median in all cases, as reported in the meta sheets of the ARX–Markov export files. The resulting Markov-based feature vector, computed per bridge crossing in the scour-detection models, is as in Eq. (5.49):

$$g = [g(1), g(2), \dots, g(H)] \tag{5.49}$$

with $H = 40$, for the detection models. In both cases, this provides a uniform and stable description of the bridge–track dynamics. Unlike ARX coefficients, which vary in number and scale with model order, Markov parameters provide a fixed-length, order-invariant description of the system dynamics. MATLAB Code for fitting ARX models and deriving Markov features for scour-detection ML algorithms is provided in Annex D.

Scour Locating ML models

For the scour-localisation models, each support-centred window is modelled as a discrete local linear convolution from the rail irregularity to the track profile, and the Markov parameters are taken as the first H samples of the corresponding spatial impulse response. For a given support window, the rail irregularity and track profile segments are first pre-processed by subtracting their means and applying a Hann taper, yielding the windowed rail irregularity ξw(n) and the windowed track profile $\delta_w(n)$. The local convolution model can then be written as in Eq. (5.50):

$$\delta_w(n) \;=\; \sum_{k=1}^{H} g(k)\, \xi_w(n-k) + e_w(n) \tag{5.50}$$

in matrix form in Eq. (5.51):

$$\delta_w \;=\; U_w\, g + e_w \tag{5.51}$$

where U_w is a convolution matrix built from u_w, $g=[g(1),\ldots,g(H)]^T$ is the (local) impulse-response vector, and e_w is a residual term. A Hann window is applied so that the support-centred signals decay smoothly to zero at the window edges. Therefore, edge artefacts are reduced, and the numerical stability of the local impulse-response estimation is improved.

Markov parameters are computed over short, support-centred windows. The shortest internal support window is as in Eq. (5.52):

$$N_{w,min} \;=\; \min_{s \in \text{internal supports}} N_w(s) \tag{5.52}$$

where $N_w(s)$ denotes the number of spatial samples in the window around support s. The reason for finding the shortest internal support window is because:

- Each support has a different window length $N_w(s)$, since span lengths are different and abutment windows are asymmetric.
- The Markov horizon H must be contained within every support window; otherwise, for some supports, an impulse response would be estimated that is longer than the available data in that window.

An adaptive Markov horizon was defined per bridge as in Eq. (5.53):

$$H_{adapt} = \min\,(40,\, \max\,(5,\, N_{w,min} - 2)) \tag{5.53}$$

In this way, H_{adapt} automatically adjusts to the available window length per bridge and remains fully contained within the shortest internal window. At the same time, the "−2" term provides a small margin to limit edge effects near the abutments. For WCM15 and WCM15A, this rule yielded horizons with $H_{adapt} \approx$

30–33 samples. Inspection of the resulting support-level Markov matrices across all bridges showed that, at such horizons, several supports contained trailing zero columns or noise-dominated high-order coefficients, and that the shorter bridge Colonnade displayed informative, non-zero entries only at early orders. To ensure consistency across bridges, reduce sparsity from padded or unstable tails, and retain only those parts of the impulse response that are well populated for every support, a conservative fixed horizon of $H = 13$ is therefore adopted for the localisation models.

At the spatial step $\Delta x = 0.2$m, this corresponds to an impulse-response reach of approximately 2.6m, which captures the short-to-moderate-range dynamics around each support, where scour-induced stiffness changes are most pronounced. Therefore, in scour-locating ML models, a Type B feature vector for each support window provides a uniform feature length across bridges and sensor positions while limiting the influence of poorly estimated high-order coefficients and reducing the need for imputation. For the localisation models, the scour-affected zone of each support is taken as a relatively wide support-centred window, spanning approximately one quarter of each adjacent span on either side of the pier or abutment. This window was selected based on preliminary simulations to ensure that the main scour-induced deformation patterns were fully captured. Subsequently, the Markov-parameter analysis shows that, with a fixed horizon of $H = 13$, i.e., approximately 2.6m, more than 95% of the local impulse-response energy is typically contained within a much shorter distance. This indicates that the adopted L/4 support window is conservative: it more than covers the dynamically active, scour-sensitive region while still allowing the Markov features to focus on the early, sufficiently excited part of the response.

Within each support window, a three-stage procedure is used to estimate the local impulse response vector g. First, an ARX model is fitted. If this ARX-based estimate fails the quality checks, a regularised nonparametric impulse response is used as the first fallback. If that also fails, a ridge-regularised FIR estimate is applied as a final fallback. All three identifications rely on the exact condition. Within each support window, the tapered rail irregularity $\xi_w(n)$ must exhibit sufficient variability, and this variability must be clearly reflected in the tapered track profile $\delta_w(n)$. If this is not satisfied, the convolution matrix U_w becomes nearly rank-deficient, and the identification problem is ill-conditioned. Therefore, the estimated impulse responses tend to be very small in magnitude or almost identical across different windows.

i. ARX Model

For each support, a support-centred spatial window is extracted from the rail-irregularity profile $\xi(x)$ and the track profile $\delta(x)$, covering the scour-affected zone around that support. Within this window, the discrete sequences are first demeaned and multiplied by a Hann window that decays smoothly to zero at

the edges. The resulting tapered signals are denoted $\xi_w(n)$ and $\delta_w(n)$, and the corresponding spatial segment is referred to as the tapered window *w*. A single-input single-output ARX model with one-sample input delay is fitted to the tapered window is expressed with Eq. (5.54):

$$A(q^{-1})\delta_{w(n)} = B(q^{-1})q^{-1}\xi_{w(n)} + e_{w(n)} \tag{5.54}$$

The polynomials are in Eq. (5.55) and (5.56):

$$A(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a} \tag{5.55}$$

$$B(q^{-1}) = b_1 + b_2 q^{-1} + \ldots + b_{n_b} q^{-(n_b - 1)} \tag{5.56}$$

with $n_a = n_b$ selected by sweeping $n_a = n_b \in [1, n_{max}]$ and minimising the AIC, as in the scour detection ML models. For the selected order, the transfer function is as in Eq. (5.57):

$$G(q^{-1}) = \left[\frac{B(q^{-1})}{A(q^{-1})}\right] q^{-1} \tag{5.57}$$

For the selected order, the zero-state impulse response $\tilde{g}_{ARX}$ is defined as the output sequence $\delta_w(n)$ of the system described by $G(q^{-1})$ when all initial conditions are set to zero, and the input is a unit impulse in the rail irregularity. In discrete form, the zero-state response satisfies Eq. (5.58) with zero initial state.

$$\delta_w(n) = \sum_{k=1}^{\infty} \tilde{g}_{ARX}(k)\, \xi_w(n-k) \tag{5.58}$$

For the impulse input, Eq. (5.59) is set, which represents an idealised unit "bump" in the rail irregularity at one location.

$$\xi_w(0) = 1,\ \xi_w(n) = 0 \text{ for } n \geq 1 \tag{5.59}$$

The corresponding zero-state output is as in Eq. (5.60):

$$\delta_w(n) = \tilde{g}_{ARX}(n) \tag{5.60}$$

The impulse-response vector $[\tilde{g}_{ARX}(1), \ldots, \tilde{g}_{ARX}(H)]$ is then taken and truncated to the target horizon *H*. All entries of the candidate vector must be finite, and the peak magnitude must satisfy the condition in Eq. (5.61):

$$\max_k |\tilde{g}_{ARX}(k)| \leq g_{peak}, max \tag{5.61}$$

The corresponding energy is as in Eq. (5.62):

$$E_{ARX} = \sum_{k=1}^{H} \tilde{g}_{ARX}(k)^2 \tag{5.62}$$

where *k* indexes the samples of the impulse response, i.e., Markov orders, from $k = 1$ to $k = H$, and *H* is the adopted Markov horizon. The energy must lie below an energy threshold, i.e., E_{cap}, as expressed in Eq.

(5.63), which is derived from the median and median absolute deviation over windows where identification has already succeeded.

$$E_{cap} = med_{j(E_{ARX},j)} + 6\,MDAD_{j(E_{ARX},j)} \tag{5.63}$$

If all entries of $\tilde{g}_{ARX}(k)$ are finite and the constraints in Eq. (5.58) and Eq. (5.59) are satisfied with $E_{ARX} \leq E_{cap}$. The resulting vector is then accepted as g.

ii. Regularised nonparametric impulse response

If the ARX model cannot be fit, a regularised nonparametric impulse response is obtained by solving a penalised least-squares problem. The objective is to find an impulse-response vector g that both fits the windowed track profile δ_w and remains smooth and decaying along the lag axis. Lag axis is the index k of the impulse response, i.e., spatial lag. The optimisation problem is as in Eq. (5.64):

$$\min_{g} J(g) = \| \delta_w - U_w g \|_2^2 + \lambda\, g^T K_{TC} g \tag{5.64}$$

where δ_w is the tapered track-profile vector in the current window, $\underline{U_w}$ is the corresponding convolution matrix built from the tapered rail irregularity, $\lambda > 0$ is a regularisation weight, and K_{TC} is the tuned-correlated kernel matrix. The term $\|\delta_w - U_w g\|_2^2$ measures the misfit between the observed windowed profile and the profile reconstructed from the candidate impulse response. The second term $g^T K_{TC} g$ is a quadratic penalty that encodes prior expectations about the shape of g. In a tuned-correlated tuned-correlated kernel, the entries of K_{TC} decrease as the distance between lags increases. This construction couples neighbouring coefficients $g(k)$ and $g(k+1)$ and gradually shrinks later lags towards zero. As a result, the estimated impulse response is smooth across k and decays with increasing lag. The parameter λ controls the trade-off between data fit and regularisation and is chosen automatically from the data.

The solution of the optimisation problem gives a regularised impulse-response estimate, denoted by $\tilde{g}_{IR}$. The first H samples $[g_{IR}(1), \ldots, g_{IR}(H)]$ are then retained and subjected to the same checks as for the ARX-based impulse response. All entries must be finite, the peak magnitude must satisfy the bound in Eq. (5.61), and the energy in Eq. (5.62) must remain below the threshold defined in Eq. (5.63). If these conditions are satisfied, the resulting vector is accepted as the local impulse response g or that window.

iii. Ridge-regularised finite impulse response (FIR)

If neither an ARX model can be fit and regularised, nor a nonparametric impulse response can be rejected, a ridge-regularised finite impulse response (FIR) is computed directly from the convolution

model. Let $U_w \in R(N_w \cdot H)$ denote the truncated convolution matrix built from $\xi_w(n)$ in the current window w. A ridge-FIR estimate is as in Eq. (5.65):

$$\tilde{g}_{FIR} = (U_w^T U_w + \lambda_{FIR} I)^{-1} U_w^T \delta_w \tag{5.65}$$

where $\lambda_{FIR}$is a ridge regularisation parameter scaled from trace $U_w{}^T U_w$ with a small floor to avoid numerical collapse. The same checks on finiteness, peak amplitude and energy provided in methods *i* and *ii* are applied. Before taking the first H samples as g, the energy is limited to prevent ill-conditioned windows from producing impulse responses with unrealistically large total energy. If, for a given window, the standard deviation of the tapered input $\xi_w(n)$ falls below a small threshold, or all three routes fail the guards, the corresponding row is marked as "failed" and handled by imputation rather than by forcing an unreliable impulse response.

Among the four bridges, the River Teign has the longest spans. For a fixed support-centred window and a short Markov horizon (H =13), corresponding to approximately 2.6 m, only a small portion of a long, globally deforming span is used for local identification. After subtracting the mean and applying the Hann taper, the effective central part of the window often contains a track profile $\delta_w(n)$ that varies smoothly, dominated by long-wavelength sag, and a rail irregularity $\xi_w(n)$ with relatively low variance. In such cases, both $\xi_w(n)$ and $\delta_w(n)$ become almost flat or slowly varying within the tapered window, and the corresponding convolution matrix $U_w$is close to low rank. As a result, all three identification methods tend to return minimal impulse responses or very similar vectors $g(k)$ across different windows.

This effect is most pronounced for heavily scoured River Teign simulations with the rear-mounted sensor. Scour does not change $\xi(x)$, but it reduces the local pier stiffness and increases the quasi-static, long-wavelength component of the bridge response. By the time the rear sensor passes over a heavily scoured support on a long span, the local displacement field along the short window (~2.6 m) is smooth and slowly varying, especially after tapering at the edges. The tapered rail irregularity then has low effective variance, the tapered track profile changes little over the Markov horizon, and the correlation $\rho_{\delta_w \xi_w}$ can be weak or unstable. To avoid discarding these windows while preventing repeated rows, a row-wise imputation scheme is applied directly at the Markov feature vector level. A row is flagged as "failed" if all entries are either non-finite or numerically negligible. Where g_r denotes the impulse-response vector for row r, corresponding to a given simulation and support. For all rows (successful and failed), a short descriptor vector is computed from the corresponding window as in Eq. (5.66):

$$d_r = \left[\sigma_{\delta w(r)}, r_{\delta w(r)}, \rho_{\delta w \xi w(r)} \right]^T \tag{5.66}$$

where $\sigma_{\delta_{w(r)}}$ is the standard deviation of the tapered track profile, $r_{\delta_{w(r)}}$ is its root-mean-square value, and $\rho_{\delta_w \xi_{w(r)}}$ is the correlation coefficient between the tapered output and input in that window *w*. Only the rows with successful impulse-response estimates, the descriptors are standardised as in Eq. (5.67):

$$Z_r = \frac{(d_r - \mu_d)}{\sigma_d} \tag{5.67}$$

where μ_d and σ_d are the mean and standard deviation over successful rows. Standardisation ensures that all three components of d_r contribute comparably when Euclidean distances are later used to select nearest neighbours for imputation. For each failed row r, Euclidean distances in Eq. (5.68) are computed to all successful rows ℓ.

$$D_{r\ell} = \left|\left|Z_r - Z_\ell\right|\right|_2 \tag{5.68}$$

The set of *K* nearest neighbours, N_r, is then identified, and the neighbour mean impulse response in Eq. (5.69) is blended with the global median in Eq. (5.70).

$$\bar{g}_{N_r} = \left(\frac{1}{|N_r|}\right) \sum_{\ell \in N_r} g_\ell \tag{5.69}$$

$$g_{med} = med_{(\ell \in S)(g_\ell)} \tag{5.70}$$

where *S* denotes all successful rows. The imputed vector for row *r* is in Eq. (5.71):

$$\tilde{g}_r = \alpha\, \bar{g}(N_r) + (1 - \alpha) g_{med};\ 0 < \alpha < 1 \tag{5.71}$$

To maintain robustness, each coordinate of $\tilde{g}_r$ is clipped within a wide median-absolute-deviation envelope of $g_{low}(k)$ and $g_{high}(k)$ in Eq. (5.72) and (5.73), respectively. The result after applying this envelope at coordinate *k* is in Eq. (5.74):

$$g_{low(k)} = g_{med(k)} - 6\, MDAD_{S(k)} \tag{5.72}$$

$$g_{high(k)} = g_{med(k)} + 6\, MDAD_{S(k)} \tag{5.73}$$

$$g_{est(k)} = \min\left\{\max\left[\bar{g}_{r(k)}, g_{low(k)}\right], g_{high(k)}\right\} \tag{5.74}$$

Here, $MDAD_s(k)$ is the median absolute deviation of the k^{th} component over successful rows. ±6 MDAD interval is approximately equivalent to ±4 standard deviations for near-Gaussian coefficients and only trims extreme outliers. Finally, a minimal perturbation in Eq. (5.75) is added so that imputed rows are not identical:

$$g_r^*(k) = g_{est}(k) + \eta_s MDA\, D_S(k) \cdot sin[2\pi\, \varphi\, (r + \kappa\, k)] \tag{5.75}$$

Here η_s is a small dimensionless factor of 5×10^{-4}, $\varphi = (\sqrt{5}-1)/2$ is the conjugate of the golden ratio, and κ is a fixed integer offset. This ripple is of the order of 0.05% of $MDAD_s(k)$, so it does not change the magnitude of the features, but it breaks long sequences of identical rows. The resulting $g^*_r(k)$ replaces g_r for all failed rows. This procedure was essential for the River Teign rear-mounted case, where many windows had weak excitation and short effective length. It removes long plateaus of repeated impulse-response vectors while keeping all windows in the dataset, so that the pier-based scour-localisation ML models can be trained on a complete, fixed-length Markov feature representation across bridges and sensor positions. MATLAB Code for estimating local impulse responses and deriving Markov features for scour-locating ML models is provided in Annex D.

Table 5.9 shows the distribution of impulse-response identification method usage across all bridges and sensor positions. All accepted impulse responses come from either ARX or ridge-FIR. ARX fits provide 16.6% of rows local impulse responses, while the ridge-regularised FIR route supplies the majority, with 78.6%. KNN-based imputation is only required in 4.8% of all windows. It is entirely concentrated in the River Teign rear-mounted case, where just one window is identified by ARX, 47.6% are obtained via FIR, and 52.4% are imputed. For WCM15 and WCM15A, about 15%-22% of windows are successfully identified by ARX, with the remainder handled stably by FIR with no imputation. The Colonnade bridge shows the highest ARX share, with just over 32% of windows fitted by ARX for both leading and trailing sensors and the rest covered by FIR. In contrast, for River Teign with the leading sensor, almost all windows are handled by FIR.

Table 5.9. Impulse-response identification outcomes for scour-localisation ML models.

Sensor position	Bridge	Total rows	Number of ARX models	Number of FIR models	Inputted KNN
front	WCM15	5792	1211	4581	0
rear	WCM15	5856	1259	4597	0
rear	WCM15A	7310	1086	6224	0
front	WCM15A	7260	1061	6199	0
front	River Teign	3670	1	3669	0
rear	River Teign	3655	1	1739	1915
rear	Colonnade	2976	977	1999	0
front	Colonnade	2976	959	2017	0
All	All	39495	6555	31025	1915

5.3.2 Feature Selection Methods

In the Markov parameter formulation of the scour detection models, the first coefficient, k=0, is always zero, indicating that the system has no instantaneous response to an impulse, so the corresponding

feature slot is structurally empty. As a result, although 40 Markov parameters are generated, only 39 carried usable information, yielding a consistent total of 51 features (12 Type A + 39 Type B) for all models. For the scour-locating ML models, 13 Markov parameters are generated, yielding a total of 25 input features (12 Type A + 13 Type B).

Feature selection is conducted exclusively for the ANN stage, and the same selected subset is subsequently fixed across all classifiers (ANN, RF, SVM). This ensured comparability between algorithms and reproducibility. The classification task is defined based on scour severity. Simulations with pier scour exceeding 5% are labelled as scoured, while those below this threshold are labelled as healthy, consistent with previous studies that set a minimum scour threshold of 5% (Fernandes et al., 2024; Gasparotti & Lopez, 2024). This threshold marks the onset of safety-critical stiffness loss, and it establishes the scoured class as the safety-critical condition. Performance metrics are interpreted in a class-specific manner: *F1* score for 5–20% scour measures how reliably the classifier detects scoured cases, which is critical from a safety perspective. *F1* score for 0-5% scour reflects how accurately it recognises healthy cases, thereby reducing false alarms. Two complementary strategies are applied:

5.3.2.1 Relative-importance–based selection with grid search

In this first strategy, feature relevance is estimated through perturbation-based sensitivity analysis. The variance of model outputs with respect to random perturbations of each feature is used to compute RI scores (Eq. 20). Features are then ranked in descending order of importance. Using this ranking, subsets of increasing size are formed from the top 1 feature up to the complete set of 51. Then, each subset is evaluated through external 5-fold CV of the ANN classifier.

In scour detection models, for each candidate feature subset, ANN hyperparameters are tuned by a nested grid search. The classifier is a feedforward neural network implemented with MATLAB's *fitcnet* function (Matlab, 2025a), with either one or two hidden layers. Hidden layer sizes are drawn from {5, 10, 20, 30, 40, 50, 60, 70, 80}, with a hyperbolic tangent activation function in all cases. *L2* regularisation strengths are explored over $\lambda \in \{10^{-6}, 10^{-5}, 10^{-4}\}$. Scalar hyperparameter that multiplies the sum of squared network weights in the loss function. In a feedforward ANN, each neuron first computes a combination of its inputs (weighted sum plus bias) and then passes this value through an activation function. These functions are fixed non-linear functions applied elementwise at each neuron (Dubey et al., 2022). Through introducing non-linearity, the network can transform non-linearly separable inputs into more linearly separable internal features, while keeping training stable. Good activation functions add useful curvature to the optimisation landscape, do not increase computational cost, preserve gradient flow during

backpropagation, and avoid distorting the data distribution too severely (Dubey et al., 2022). The hyperbolic tangent (tanh) activation is formulated as: in Eq. (5.76):

$$tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})} \tag{5.76}$$

The tanh function maps any real input x to the range [-1, 1]. It is smooth, S-shaped, and zero-centred: negative inputs give negative outputs, and positive inputs give positive outputs. For large positive or negative |x|, the outputs saturate near +1 or -1, which can slow learning because gradients in those regions become very small. *L2* regularisation (Krogh & Hertz, 1992) is introduced by adding a quadratic penalty on the weights to the original loss function. The regularised loss is as in Eq. (5.77):

$$E(\theta) \;=\; E_0(\theta) \;+\; \frac{1}{2}\lambda \sum_{i=1}^{p} w_i^2 \tag{5.77}$$

where $E(\theta)$ is the regularised loss that is aimed to be minimised by the training (optimisation) algorithm, $E_0(\theta)$ is the unregularized loss, $w = (w_1, w_2, \ldots, w_p)$ is the vector of network weights, $\lambda > 0$, and *L2* is the regularisation strength. The term added to the unregularized loss penalises large weights, helping reduce overfitting and overly complex models. When the network is trained using gradient-based optimisation, the *L2* penalty adds a term $-\lambda w_i$ to the update of each weight, thereby gradually shrinking the weights towards zero. This effect is often described as weight decay.

Within each of the five outer folds, the outer-training data are z-scored using training statistics only and passed to a three-fold internal CV, where each configuration is trained for up to 200 iterations and scored by mean classification accuracy. The best configuration in each outer fold is then retrained on the whole outer-training partition with an increased iteration limit of 1000 and used to generate predictions for the corresponding outer-test fold. A fixed random seed is used to ensure reproducibility of data partitions and training runs, and the inner grid search is parallelised across configurations to reduce computation time. This hyperparameter configuration is used only for feature selection, and final hyperparameter optimisation for batch sizes 7, 15, 31, and 61 is reported in Chapter 6.

In scour locating ANN models, for each outer fold and sensor orientation, the network uses two hidden layers of [35 35] or [50 20] with hyperbolic tangent activations, *L2* regularisation set to $\lambda = 10^{-4}$, and Glorot or He initialisation for the layer weights with either ones or zeros for the biases. Inputs are z-scored before training, so internal standardisation is enabled or disabled according to the stored configuration. Feature selection is then carried out via a grid over the number of retained features: for each candidate size n_{feat} from 1 to the complete set, the top-n_{feat} parameters ranked by RI are kept. Then a 5-fold CV is run with the fixed ANN to compute *F1* scores for 0–5% and 5–20%, Balanced Accuracy, Matthews Correlation

Coefficient (MCC), and overall accuracy. The same hierarchical decision rule as in the detection case is then applied to choose the final number of features. As in the scour detection models, these hyperparameters are used solely for feature selection. The final hyperparameter tuning for batch sizes 7, 15, 31, and 61 is reported in Chapter 6.

The selection criteria prioritised the *F1* score for 5–20%, as scour detection requires minimising false negatives (safety-driven). A hierarchical decision rule is applied:

i. *F1* score for 5–20% ≥ 0.85.

ii. If multiple subsets satisfy this condition, the most significant number of features is preferred.

iii. In case of ties, MacroF1, Balanced Accuracy, and MCC, in that order, are maximised. Macro F1 is defined in Eq. (5.78):

$$MacroF1 = 0.5 \cdot (F1_{5-20\%} + F1_{0-5\%}) \tag{5.78}$$

where $F1_{5\text{-}20\%}$ denotes F1 score for the 5–20% scour, corresponding to the scoured class; and $F1_{0\text{-}5\%}$ denotes F1 score for the 0-5% percentage, corresponding to the healthy class and formulated in Eq. (5.79) and (5.80), respectively:

$$F1_{5-20\%} = \frac{2 \cdot TP}{(2 \cdot TP + FP + FN)} \tag{5.79}$$

$$F1_{0-5\%} = \frac{2 \cdot TN}{(2 \cdot TN + FP + FN)} \tag{5.80}$$

Balanced accuracy and MCC are expressed in Eqs. (5.81) and (5.82), respectively:

$$Balanced\ Accuracy = 0.5 \cdot \left(\frac{TP}{(TP + FN)} + \frac{TN}{(TN + FP)}\right) \tag{5.81}$$

$$MCC = \frac{(TP \cdot TN - FP \cdot FN)}{\sqrt{[\,(TP + FP)(TP + FN)(TN + FP)(TN + FN)]}} \tag{5.82}$$

MATLAB code for relative-importance–based selection with grid search is provided in Annex E.

Results of scour detection ML models

Feature subsets obtained from relative-importance–based selection with grid search, together with their metrics, are reported in 80 (RILA front-mounted) and Table 5.11 (RILA rear-mounted) for scour detection ANN models. Results indicate that the optimal configuration for both RILA front-mounted and rear-mounted models is to retain all 51 features, achieving strong performance while ensuring a rich input space for alternative classifiers. RILA front-mounted case yielded 0.918 *F1* score for the 5–20% scour, 0.681 for *F1* score for the 0-5% scour, *MacroF1* = 0.8, and a balanced accuracy of 0.803. For the rear-

mounted case, a 0.904 F1 score for the 5–20% scour, a 0.64 *F1* score for the 0-5% scour, a *MacroF1* = 0.772, and a balanced accuracy of 0.781 are acquired.

Table 5.10. Feature subset search for RILA front-mounted case-scour detection models.

Number of features	F1 score for the 5–20% scour	F1 score for the 0-5 % scour	Balanced Accuracy	MCC	Accuracy	Training Accuracy	MacroF1
1	0.889	0.000	0.500	0.000	0.800	0.800	0.444
2	0.889	0.003	0.501	0.037	0.800	0.800	0.446
3	0.888	0.000	0.500	-0.009	0.799	0.800	0.444
4	0.889	0.000	0.500	0.000	0.800	0.800	0.444
5	0.888	0.000	0.499	-0.016	0.799	0.800	0.444
6	0.886	0.016	0.501	0.006	0.795	0.802	0.451
7	0.865	0.208	0.538	0.104	0.769	0.816	0.537
8	0.862	0.287	0.568	0.165	0.769	0.840	0.575
9	0.907	0.589	0.734	0.499	0.848	0.943	0.748
10	0.902	0.603	0.750	0.506	0.843	0.953	0.753
11	0.895	0.571	0.730	0.466	0.831	0.956	0.733
12	0.904	0.591	0.738	0.496	0.844	0.953	0.747
13	0.908	0.633	0.770	0.540	0.853	0.969	0.770
14	0.922	0.690	0.806	0.612	0.876	0.987	0.806
15	0.914	0.667	0.795	0.581	0.864	0.982	0.790
16	0.918	0.671	0.793	0.589	0.869	0.984	0.795
17	0.920	0.685	0.805	0.605	0.872	0.998	0.802
18	0.926	0.702	0.812	0.628	0.882	0.998	0.814
19	0.916	0.676	0.801	0.592	0.867	0.999	0.796
20	0.927	0.703	0.812	0.630	0.882	0.999	0.815
21	0.918	0.684	0.808	0.602	0.869	1.000	0.801
22	0.922	0.693	0.810	0.615	0.876	1.000	0.808
23	0.915	0.678	0.805	0.594	0.866	1.000	0.797
24	0.921	0.692	0.811	0.613	0.874	1.000	0.806
25	0.919	0.687	0.809	0.607	0.871	1.000	0.803
26	0.915	0.674	0.802	0.590	0.866	1.000	0.795
27	0.923	0.697	0.814	0.620	0.877	1.000	0.810
28	0.925	0.705	0.817	0.630	0.881	1.000	0.815
29	0.915	0.677	0.804	0.592	0.866	1.000	0.796
30	0.922	0.698	0.816	0.620	0.876	1.000	0.810
31	0.914	0.676	0.806	0.591	0.864	1.000	0.795
32	0.922	0.697	0.814	0.619	0.876	1.000	0.810
33	0.921	0.694	0.813	0.615	0.874	1.000	0.807
34	0.923	0.697	0.812	0.620	0.878	1.000	0.810
35	0.913	0.670	0.800	0.584	0.863	1.000	0.792
36	0.917	0.683	0.808	0.600	0.868	1.000	0.800

37	0.921	0.695	0.814	0.616	0.874	1.000	0.808
38	0.917	0.682	0.807	0.599	0.868	1.000	0.799
39	0.912	0.661	0.793	0.573	0.860	1.000	0.787
40	0.914	0.669	0.798	0.583	0.864	1.000	0.791
41	0.924	0.698	0.812	0.622	0.879	1.000	0.811
42	0.913	0.678	0.809	0.593	0.864	1.000	0.796
43	0.913	0.677	0.808	0.592	0.863	1.000	0.795
44	0.916	0.684	0.810	0.601	0.868	1.000	0.800
45	0.922	0.700	0.818	0.622	0.876	1.000	0.811
46	0.909	0.658	0.794	0.568	0.856	1.000	0.783
47	0.921	0.700	0.821	0.622	0.874	1.000	0.811
48	0.918	0.686	0.810	0.604	0.870	1.000	0.802
49	0.911	0.668	0.802	0.580	0.859	1.000	0.789
50	0.913	0.674	0.805	0.588	0.863	1.000	0.793
51	0.918	0.681	0.804	0.599	0.869	1.000	0.799

Table 5.11. Feature subset search for RILA rear-mounted case-scour detection models.

Number of features	**F1 score for the 5–20% scour**	**F1 score for the 0-5% scour**	**Balanced Accuracy**	**MCC**	**Accuracy**	**Training Accuracy**	**MacroF1**
1	0.889	0.003	0.501	0.020	0.800	0.800	0.446
2	0.888	0.003	0.500	0.005	0.799	0.800	0.446
3	0.883	0.076	0.511	0.058	0.792	0.809	0.479
4	0.877	0.348	0.599	0.247	0.793	0.852	0.613
5	0.887	0.543	0.713	0.431	0.819	0.935	0.715
6	0.899	0.591	0.743	0.490	0.838	0.959	0.745
7	0.907	0.604	0.745	0.512	0.850	0.958	0.755
8	0.897	0.592	0.745	0.488	0.835	0.958	0.744
9	0.899	0.607	0.757	0.507	0.840	0.988	0.753
10	0.894	0.603	0.758	0.498	0.832	0.990	0.748
11	0.913	0.658	0.788	0.571	0.861	0.996	0.785
12	0.916	0.660	0.786	0.576	0.865	0.992	0.788
13	0.917	0.672	0.795	0.589	0.868	0.997	0.795
14	0.919	0.675	0.795	0.594	0.871	0.999	0.797
15	0.920	0.680	0.801	0.600	0.871	0.999	0.800
16	0.925	0.695	0.808	0.620	0.879	1.000	0.810
17	0.921	0.688	0.806	0.609	0.874	1.000	0.804
18	0.918	0.675	0.798	0.593	0.869	0.999	0.797
19	0.920	0.674	0.793	0.595	0.872	1.000	0.797
20	0.921	0.683	0.801	0.603	0.873	0.999	0.802
21	0.912	0.654	0.786	0.565	0.859	0.999	0.783
22	0.909	0.656	0.792	0.566	0.856	1.000	0.782
23	0.907	0.648	0.786	0.556	0.853	1.000	0.778

24	0.909	0.653	0.789	0.562	0.855	1.000	0.781
25	0.911	0.656	0.789	0.568	0.859	1.000	0.784
26	0.914	0.662	0.791	0.576	0.863	1.000	0.788
27	0.909	0.648	0.784	0.558	0.856	1.000	0.779
28	0.910	0.653	0.788	0.563	0.857	0.999	0.781
29	0.912	0.652	0.784	0.564	0.859	0.999	0.782
30	0.911	0.656	0.789	0.567	0.858	0.999	0.783
31	0.913	0.663	0.792	0.576	0.862	0.999	0.788
32	0.909	0.656	0.791	0.567	0.857	0.999	0.783
33	0.906	0.636	0.775	0.542	0.851	1.000	0.771
34	0.908	0.641	0.779	0.549	0.853	1.000	0.774
35	0.913	0.658	0.788	0.571	0.861	1.000	0.786
36	0.908	0.652	0.789	0.561	0.855	0.999	0.780
37	0.900	0.628	0.775	0.530	0.843	1.000	0.764
38	0.904	0.638	0.781	0.543	0.848	1.000	0.771
39	0.908	0.640	0.776	0.548	0.854	1.000	0.774
40	0.906	0.647	0.787	0.555	0.852	1.000	0.777
41	0.914	0.659	0.787	0.573	0.863	1.000	0.786
42	0.909	0.644	0.780	0.554	0.855	0.999	0.777
43	0.913	0.663	0.793	0.576	0.862	1.000	0.788
44	0.919	0.673	0.794	0.592	0.870	1.000	0.796
45	0.900	0.622	0.769	0.523	0.842	1.000	0.761
46	0.918	0.668	0.790	0.587	0.869	1.000	0.793
47	0.911	0.646	0.779	0.556	0.857	1.000	0.778
48	0.905	0.650	0.790	0.557	0.851	1.000	0.777
49	0.912	0.668	0.800	0.582	0.861	1.000	0.790
50	0.906	0.637	0.777	0.543	0.850	0.999	0.771
51	0.905	0.640	0.782	0.546	0.849	1.000	0.772
52	0.889	0.003	0.501	0.020	0.800	0.800	0.446

Results of scour locating ML models

Scour locating ANN model feature subsets obtained from relative-importance–based selection with grid search are reported in Table 5.12 (RILA front-mounted) and Table 5.13 (RILA rear-mounted). The optimal configuration for both RILA front-mounted and rear-mounted models is to retain all 25 features, which achieves strong performance while ensuring a rich input space for alternative classifiers. RILA front-mounted case yielded 0.921 *F1* score for 5–20% of scour, 0.981 *F1* score for 0-5% of scour, *MacroF1* = 0.951, and a balanced accuracy of 0.95. For the rear-mounted case, 0.884 *F1* score for 5–20% of scour, 0.97 *F1* score for 0-5% of scour, *MacroF1* = 0.927, and a balanced accuracy of 0.923 are acquired.

Table 5.12. Feature subset search for RILA front-mounted case-scour locating models.

Number of features	F1 score for 5–20% of scour	F1 score for 0-5% of scour	Balanced Accuracy	MCC	Accuracy	Training Accuracy	MacroF1
1	0.000	0.890	0.500	0.000	0.801	0.801	0.445
2	0.016	0.890	0.503	0.051	0.802	0.803	0.453
3	0.067	0.892	0.517	0.147	0.807	0.808	0.479
4	0.123	0.895	0.532	0.208	0.812	0.813	0.509
5	0.274	0.900	0.577	0.303	0.824	0.827	0.587
6	0.342	0.904	0.602	0.353	0.832	0.842	0.623
7	0.353	0.903	0.606	0.353	0.832	0.842	0.628
8	0.546	0.917	0.696	0.498	0.859	0.868	0.731
9	0.702	0.936	0.792	0.647	0.895	0.916	0.819
10	0.741	0.943	0.819	0.690	0.907	0.935	0.842
11	0.758	0.945	0.831	0.707	0.911	0.942	0.851
12	0.860	0.967	0.904	0.827	0.946	0.974	0.913
13	0.870	0.969	0.911	0.839	0.950	0.978	0.919
14	0.863	0.967	0.904	0.831	0.947	0.983	0.915
15	0.865	0.967	0.910	0.833	0.947	0.982	0.916
16	0.881	0.971	0.917	0.853	0.954	0.981	0.926
17	0.908	0.977	0.940	0.885	0.964	0.991	0.943
18	0.906	0.977	0.934	0.883	0.963	0.991	0.942
19	0.915	0.979	0.947	0.894	0.966	0.995	0.947
20	0.908	0.978	0.939	0.886	0.964	0.996	0.943
21	0.917	0.980	0.945	0.897	0.967	0.996	0.948
22	0.923	0.981	0.949	0.904	0.970	0.997	0.952
23	0.922	0.981	0.948	0.903	0.969	0.997	0.951
24	0.914	0.979	0.945	0.893	0.966	0.997	0.946
25	0.921	0.981	0.950	0.902	0.969	0.997	0.951

Table 5.13. Feature subset search for RILA front-mounted case-scour locating models.

Number of features	F1 score for 5–20% of scour	F1 score for 0-5% of scour	Balanced Accuracy	MCC	Accuracy	Training Accuracy	MacroF1
1	0.002	0.882	0.500	0.020	0.790	0.791	0.442
2	0.092	0.884	0.522	0.144	0.795	0.799	0.488
3	0.133	0.887	0.534	0.199	0.801	0.805	0.510
4	0.147	0.888	0.537	0.208	0.802	0.805	0.517
5	0.191	0.890	0.551	0.250	0.807	0.811	0.540
6	0.253	0.894	0.571	0.303	0.815	0.819	0.574
7	0.262	0.894	0.573	0.304	0.815	0.821	0.578
8	0.287	0.896	0.582	0.329	0.819	0.824	0.592

9	0.288	0.896	0.582	0.324	0.818	0.825	0.592
10	0.291	0.895	0.583	0.319	0.817	0.825	0.593
11	0.435	0.901	0.639	0.403	0.832	0.839	0.668
12	0.644	0.923	0.752	0.584	0.873	0.888	0.783
13	0.629	0.921	0.744	0.570	0.869	0.887	0.775
14	0.717	0.933	0.803	0.657	0.892	0.923	0.825
15	0.759	0.942	0.830	0.706	0.907	0.934	0.851
16	0.788	0.947	0.853	0.737	0.915	0.938	0.867
17	0.795	0.949	0.855	0.747	0.919	0.944	0.872
18	0.788	0.948	0.852	0.739	0.916	0.942	0.868
19	0.856	0.963	0.900	0.820	0.941	0.972	0.910
20	0.859	0.964	0.902	0.823	0.942	0.976	0.911
21	0.874	0.967	0.915	0.842	0.948	0.977	0.921
22	0.865	0.965	0.907	0.830	0.944	0.974	0.915
23	0.861	0.964	0.904	0.826	0.943	0.971	0.913
24	0.878	0.968	0.920	0.845	0.949	0.983	0.923
25	0.884	0.970	0.923	0.854	0.952	0.982	0.927

5.3.2.2 Multi-objective feature selection using NSGA-II

The second strategy applies the Non-Dominated Sorting Genetic Algorithm II (NSGA-II), an evolutionary multi-objective optimiser that combines fast non-dominated sorting, elitism, and crowding-distance preservation (Deb et al., 2002) In this thesis, NSGA-II is used as a wrapper feature selection method with two objectives: maximising *F1* score for 5–20% (correct scour detection) and minimising the fraction of selected features (model compactness).

For the scour detection models, NSGA-II is applied as a wrapper around an ANN classifier with MATLAB's function *fitcnet* (Matlab, 2025a). Before applying NSGA-II, an initial ANN configuration was specified from earlier exploratory models. For each case (RILA front- and rear-mounted) and each outer fold, the network hyperparameters (layer sizes, *L2* regularisation strength, standardisation rule, weight and bias initialisation, and activation function) were fixed and used to train the classifier within the k-fold CV. These fold wise settings were then summarised into a single ANN configuration, which is passed to NSGA-II as a fixed base classifier. NSGA-II therefore searches only over the input feature subset; it does not tune ANN hyperparameters, which are refined later at the batch-based stage, as provide in Chapter 6.

The network uses two hidden layers, typically [35 35] or [50 20] neurons, with a hyperbolic tangent activation function. The *L2* regularisation strength is fixed at $\lambda = 10^{-4}$. Across folds, either Glorot or He weight initialisation is used and the biases are set to ones or zeros, depending on the optimised

configuration. These sets are standard schemes for setting the initial weights of neural networks so that training remains stable. In Glorot (Xavier) initialisation (Glorot & Bengio, 2010), the weights of a layer with n_{in} inputs and n_{out} outputs are initialised from a uniform distribution U in Eq. (5.83):

$$W_{ij} \sim \mathcal{U} - [\sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}}}, \sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}}}] \tag{5.83}$$

while the biases are set to zero. This choice gives the weights zero mean and variance $2/(n_{in}+n_{out})$, helping to keep activations and gradients on a similar scale across layers. In He et al. (2015), the weight variance of the neuron activations remains approximately constant as they propagate through a ReLU layer. Imposing this condition leads to Eq. (5.84):

$$\frac{1}{2}\, n_{in}\, \mathrm{Var}(w) = 1 \;\Rightarrow\; \mathrm{Var}(w_{in}) = \frac{2}{n_{in}} \tag{5.84}$$

where n_l is the number of input connections to a neuron in layer l_N. Each weight is initialized in Eq. (5.85):

$$w_{ij} \sim \mathcal{N}\,(0,\, \frac{2}{n_{in}}) \tag{5.85}$$

where w_{ij} is the weight connecting input j (from the previous layer) to neuron i in the current layer. For a fully connected layer, n_l is equal to the input size l, so this corresponds to a zero-mean normal distribution N with variance $2/l$, standard deviation $\sqrt{(2/l)}$.

Each candidate NSGA-II solution is encoded as a real-valued mask with one gene per feature, constrained to [0, 1]; genes greater than 0.5 are treated as "selected" and define the feature subset for that individual. Both for scour detection and location models, two objectives are minimised simultaneously:

i. 1-*F1* score for the 5–20% scour class

ii. The fraction of selected features

A stratified *k*-fold partition of the data is created, and the resulting training and test folds are reused for all individuals and generations so that differences in fitness arise only from the feature masks, not from changing splits. The genetic algorithm uses a population size of 100, a crossover fraction of 0.8, and a function tolerance of 10^{-3}. Depending on the chosen mode, between 40 and 60 generations are run, with a stall limit of 10–15 generations, and parallel evaluation is enabled to speed up fitness computation. Type A and Type B features are both eligible for selection. After optimisation, the Pareto set is post-processed: among all non-dominated solutions with *F1* score of scoured class ≥ 0.85, the subset with the largest number of features is chosen. If no solution meets this F1 threshold, the Pareto solution with the highest *F1* score of scoured class. In case of ties, the largest subset size is retained as the best NSGA-II feature set for detection.

For the scour locating models, the input space is encoded by one real-valued gene per feature in [0, 1], with a threshold of 0.5. A fixed random seed is used to initialise the random number generator. A stratified k-fold partition with 3 or 5 folds, depending on the fast or thorough mode is built once and reused in all evaluations. The ANN architecture used within NSGA-II is held constant, with two hidden layers of sizes [50 20], *L2* regularisation $\lambda = 10^{-4}$, hyperbolic tangent activations, Glorot weight initialisation and zero biases, and no internal standardisation since inputs are z-scored manually. NSGA-II is run with a population size of 100, crossover fraction 0.8, namely 80% of the new generation is made by recombining parents; 20% are non-crossover individuals. Function tolerance is 10^{-3}, and 40–60 generations depending on the selected mode, with parallel evaluation enabled and a Pareto plot used for monitoring. At the end of the run, the Pareto front is filtered using the same decision rule as for detection: the solution with the largest number of features among those with *F1* score for 5–20% of scour is selected. If the threshold is not met by any solution, the best available Pareto solution is chosen based on *F1* score for 5–20% of scour and subset size. MATLAB code for NSGA-II method is provided in Annex E.

Results of scour detection ML models

The results of the NSGA-II feature selection method are provided in Figure 5.50 and Figure 5.51, for RILA rear and front-mounted cases, respectively. The metrics of the best solutions are provided in Table 5.14. In Figure 5.50, each point on the plot is a non-dominated solution in the final population, representing a different trade-off between classification performance and feature-set size. The Pareto front is the set of non-dominated points: the "edge" of the cloud of solutions, one objective cannot be improved without making at least one other objective worse. Only four solutions remain on the Pareto front, indicating that the algorithm converged to a narrow set of efficient trade-offs. The chosen solution lies near the lower-left part of the front, achieving a 0.936 *F1* score for 5-20% of scour using nine features: *Skw(2), Skw(4), Krtz(1), Krtz(2), Krtz(3), Krtz(4), g(18),* E_{tot}*, g(25)*. In Figure 5.51, the RILA rear-mounted case shows a similar pattern. Only four distinct solutions lie on the Pareto front. The selected solution is the point that achieves a 0.935 *F1* score for 5–20% scour while using only five inputs: *Skw(4), Krtz(1),* E_{tot}*, g(28), and g(39).*

Results of scour locating ML models

The NSGA-II feature selection results for the support-level scour localisation models are shown for the RILA front- and rear-mounted cases in Figure 5.52 and Figure 5.53, respectively. The metrics of the best solutions are provided in Table 5.14. For the front-mounted configuration, the Pareto set contains twelve solutions spanning a range of feature fractions and *F1* values. The selected solution corresponds to an

F1 score of 0.915 for scoured class and uses 15 features: *Skw(A), Skw(1), Skw(2), Skw(3), Skw(4), Krtz(1), Krtz(2), Krtz(3), Krtz(4),* E_{tot}, L_{min}^{A}, *g(1), g(7), g(9)* and *g(11).* The RILA rear-mounted case shows a similar pattern, with eleven non-dominated solutions on the Pareto front. The chosen rear-mounted solution attains an *F1* score of 0.907 for 5–20% scour while using 14 inputs: *Skw(A), Skw(1), Skw(2), Skw(3), Skw(4), Krtz(1), Krtz(3),* E_{tot}, L_{min}^{A}, *g(1), g(3), g(4), g(5)* and *g(9).*

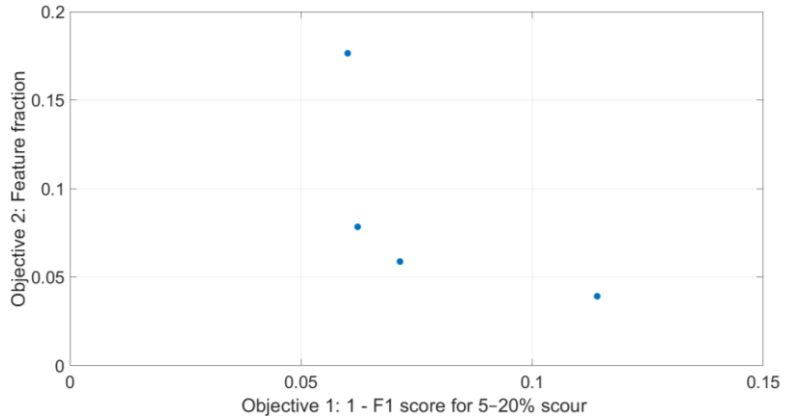


Figure 5.50. Pareto front of NSGA-II feature selection for the RILA front-mounted scour-detection model.

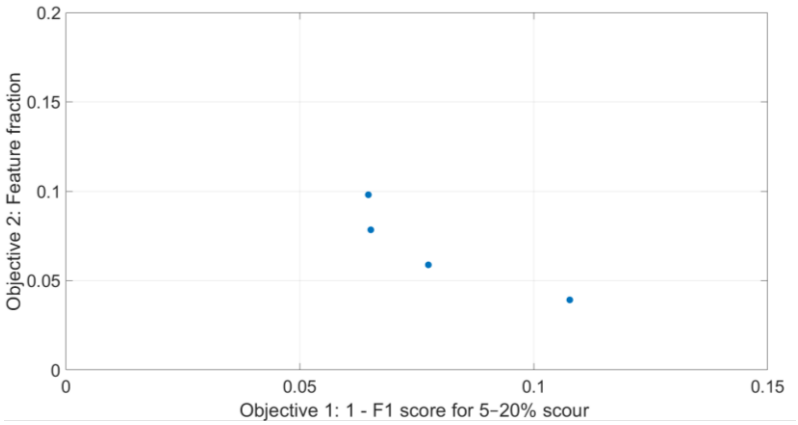


Figure 5.51. Pareto front of NSGA-II feature selection for the RILA rear-mounted scour-detection model.

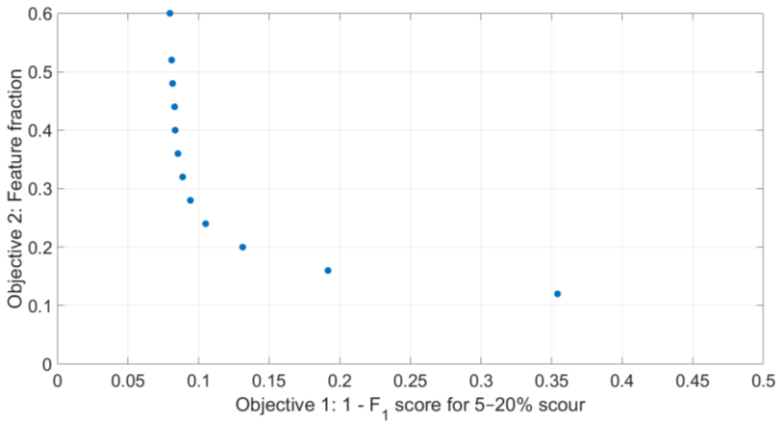


Figure 5.52. Pareto front of NSGA-II feature selection for the RILA front-mounted scour-location model.

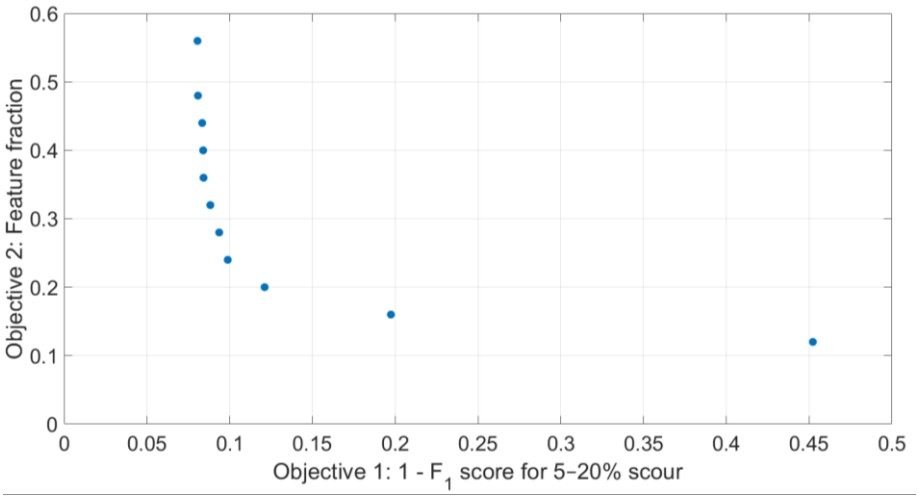


Figure 5.53. Pareto front of NSGA-II feature selection for the RILA rear-mounted scour-location model.

5.3.2.3 Assessment and decision for input features

As seen in Table 5.14, the performance of the two feature selection approaches over scour detection models is similar, but NSGA-II achieves slightly higher metrics. NSGA-II demonstrates that compact subsets can capture scour-sensitive dynamics. Table 5.15 shows that the performance of the two feature-selection approaches for scour localisation is very close to each other, with the RI-based grid-search method partly suppressing NSGA-II results. NSGA-II shows that compact subsets of 14–15 features can recover almost the same performance, confirming that scour-sensitive patterns are concentrated in a small group of DWT and ARX descriptors.

Table 5.14. Feature Selection Method Results for scour detection models.

Method	Sensor position	Selected features	MacroF1	Balanced accuracy	MCC	F1 score (5–20% scour)
RI-based & grid search	front	All	0.799	0.803	0.6	0.918
RI-based & grid search	rear	All	0.772	0.81	0.55	0.905
NSGA-II	front	Skw(2), Skw(4); Krtz(1); Krtz(2); Krtz(3); Krtz(4); g(18); E_{tot}; g(25)	0.836	0.828	0.672	0.936
NSGA-II	rear	Skw(4); Krtz(1); E_{tot}; g(28), g(39)	0.785	0.755	0.582	0.926

In both scour detection and locating ML models, for cross-model comparability and to allow RF and SVM classifiers to benefit from feature diversity, the relative-importance-based method (retaining all features) is adopted as the final feature. This ensures that differences in performance between classifiers

arise from the algorithms rather than from method-specific feature spaces. Moreover, keeping a larger set of features increases resilience when the data distribution changes between training and deployment. If certain features lose relevance under new conditions, others can still provide a predictive signal, making the overall model more robust to such shifts. This choice aligns with the learning mechanisms of the models: RF benefits from feature diversity and mild redundancy through bagging and subsampling, while an RBF-SVM uses richer representations to form discriminative margins.

Table 5.15. Feature Selection Method Results for scour localisation models.

Method	Sensor position	Selected features	MacroF1	Balanced accuracy	MCC	F1 score (5–20% scour)
RI-based & grid search	front	All	0.951	0.950	0.902	0.921
RI-based & grid search	rear	All	0.927	0.923	0.854	0.884
NSGA-II	front	Skw(A), Skw(1), Skw(2), Skw(3), Skw(4), Krtz(1), Krtz(2), Krtz(3), Krtz(4), E_{tot}, L_{min^A}, g(1), g(7), g(9), g(11)	0.947	0.947	0.894	0.915
NSGA-II	rear	Skw(A), Skw(1), Skw(2), Skw(3), Skw(4), Krtz(1), Krtz(3), E_{tot}, L_{min^A}, g(1), g(3), g(4), g(5), g(9)	0.941	0.938	0.882	0.907

5.4 Parameter Normalization

Before training the ANN and SVM classifiers, z-score normalisation is applied. Normalising the input features is essential because i) ANN training is gradient-based: features on larger numerical scales would otherwise dominate the gradient updates, leading to inefficient learning and biased weight adjustments; ii) SVM with RBF kernel is distance-based, and unscaled features would distort the kernel space by allowing large-scale variables to overshadow smaller ones. Scaling ensures that all features contribute proportionally to the formation of the margin. Each input feature is centred and scaled according to Eq. (5.86):

$$\tilde{x}_{ij} = \frac{(x_{ij} - \mu_j)}{\sigma_j} \tag{5.86}$$

where x_j is the original value of feature j, μ_j and σ_j are the sample mean and standard deviation of that feature that is computed on the training data, and $\tilde{x_j}$ is the normalised value.

For the RF classifier, implemented with *the TreeBagger* function (Matlab, 2025b), explicit feature scaling is not required. Each decision tree divides the data using threshold t rules such as "feature $x_j \leq t$ " or "$x_j > t$". These splits depend only on the ordering of the feature values, not on their absolute magnitude or units. Any monotonic rescaling of a feature, such as standardising it, preserves this ordering and therefore leaves the tree structure and the ensemble predictions unchanged.

5.5 Sliding Windows Method

This section extends the batch-based scour detection provided briefly in sections 5.2.1.2 and 5.2.2.2. In the data stream, data items arrive as an ordered sequence, and queries are evaluated continuously, rather than being evaluated once on a fixed dataset. Babcock et al. describe sliding window queries as one of the main tools for operating on such streams: instead of using the full history, the query is evaluated only on a "window" containing the most recent items, and this window slides forward as new items arrive. If a stream is written as *s(1), s(2), ... s(N);* a window of width w_s at position z contains *s(z), s(z+1), ... s(z+w_s-1). As z* increases by one, the oldest item leaves the window, and the newest item enters. The number of distinct windows is $F_w = N - w_s + 1$.

The first key property of the sliding windows is recent data, which is often more relevant for real-time decisions than older observations. Second, sliding window queries only store and process the items that fall inside the current window, rather than the entire history of the stream. In this way, sliding windows approximate a full-history query, but with a controlled and bounded memory and computational cost. When full re-computation is expensive, instead of updating the result for each individual item, the incoming stream is divided into batches. The query result is then recomputed or updated once per batch, after all items in that batch have been processed. In this thesis, each batch corresponds to one sliding window of train crossings. All crossings inside the window are used, but the decision is attached to the last crossing in that window. For scour detection and localisation, the decision, i.e., model output, is taken from the bridge condition at the last crossing in the window (scoured or healthy), while the preceding crossings are used to summarise the recent behaviour of the system and to reduce the influence of random fluctuations in any single crossing on the predicted class. At each decision time, only the most recent w_s crossings are used to construct the batch features; crossings older than this window are excluded and no longer directly affect that decision.

5.5.1 Scour Detection Models: Bridge-Level Sliding Windows

For each bridge, there are *j=1, 2, ...,* N_{sim} simulated crossings. Features *x(j)*, labels *y(j)*, and scour percentage vectors *s(j)* are first computed per crossing using the Type-A and Type-B feature sets. Sliding

windows are then applied to these per-crossing features. For a batch size w_s, the number of windows is $F_w = N_{sim}-w_s+1$. For each window index $z = 1, \ldots, F_w$, two types of input features are obtained: aggregated and individual features, and then all are combined.

i. Aggregated input features:

For each feature index $k=1, \ldots, k_p$ and window z, the statistics in Eq. (5.87) to (5.90) are computed over all crossings at that window to be used as aggregated input features in I_z:

$$\mu_k(z) = \frac{1}{w_s} \sum_{j \in I_z} x(j,k) \tag{5.87}$$

$$\sigma_k(z) = \text{std}\{x(j,k): j \in I_z\} \tag{5.88}$$

$$\min_k(z) = \min_{j \in I_z} x(j,k) \tag{5.89}$$

$$\max_k(z) = \max_{j \in I_z} x(j,k) \tag{5.90}$$

where for a window starting at crossing z, the index set of crossings inside the window is $I_z = \{z, z+1, \ldots, z+w_s-1\}$.

ii. Individual features:

The individual feature vectors of the crossings inside the window, i.e., $r(z)$ are then placed one after another in chronological order as in Eq. (5.91):

$$r(z) = [x(z)^\top, x(z+1)^\top, \ldots, x(z+w_s-1)^\top]^\top \in \mathbb{R}^{w_s p} \tag{5.91}$$

The crossings inside window z are $j = z, z+1, \ldots, z+w_s-1$; $x(z)$ is the k_p-dimensional feature vector of the crossing that is the first element in window z, as seen in Eq. (5.92):

$$x(z) = \begin{bmatrix} x(z,1) \\ x(z,2) \\ . \\ . \\ . \\ x(z,k_p) \end{bmatrix} \tag{5.92}$$

iii. Batch feature vector:

The final feature vector for window z, i.e., $X_{batch}(z)$ is obtained by combining aggregated and individual input features as in Eq. (5.93):

$$X_{\text{batch}}(z) = [\, \mu(z), \sigma(z), \min(z), \max(z), r(z) \,] \in \mathbb{R}^{(w_s+4)p} \tag{5.93}$$

iv. Batch label and scour information

The batch label (scoured or healthy) is taken from the last crossing in the window, as in Eq. (5.94):

$$Y_{batch}(z) = y(z + w_s - 1) \in \{healthy,\ scoured\} \tag{5.94}$$

where x is the input feature vector, and y is the label, i.e., output vector. The associated scour percentages for analyses of severity are taken from the same last crossing as in Eq. (5.95):

$$SCR_{batch}(z) = s(z + w_s - 1) \tag{5.95}$$

For each bridge, this process yields a windowed dataset in Eq. (5.96):

$$\{X_{batch}(z),\ Y_{batch}(z),\ SCR_{batch}(z)\}_{z=1}^{F_w} \tag{5.96}$$

Sliding windows are generated separately per bridge to keep the chronological order and to avoid information leakage between bridges. After this step, the per-bridge windowed datasets are vertically combined as seen from Eq. (5.97) to Eq. (5.99):

$$X = [X_{batch}^{(1)};\ X_{batch}^{(2)};\ \dots\ ] \tag{5.97}$$

$$Y = [Y_{batch}^{(1)};\ Y_{batch}^{(2)};\ \dots\ ] \tag{5.98}$$

$$SCR = [SCR_{batch}^{(1)};\ SCR_{batch}^{(2)};\ \dots\ ] \tag{5.99}$$

For each bridge b, with one row per window z, the windowed inputs are stored in $X_{batch}^{(b)}$, outputs in $Y_{batch}^{(b)}$ and scour percentage is $SCR_{batch}^{(b)}$. For example, the first window $z = 1$ for the first bridge uses crossings j = 1 to w_s, and takes its label and scour percentage from the last crossing in that window, that is, $Y_{batch}^{(1)} = y(w_s)$ and $SCR_{batch}^{(1)} = s(w_s)$. With 731 crossings per bridge, the number of windows is 725 for w_s = 7, 717 for w_s = 15, 701 for w_s = 31, and 671 for w_s = 61.

This combined set that consists of X, Y, and SCR is then used in the scour-detection classifiers for both RILA front-mounted and rear-mounted configurations. Each batch represents the decision "Is the bridge scoured at this last crossing?" conditioned on the previous w_s-1 crossings of the same bridge. MATLAB code details of the calculations are provided in Annex F.

5.5.2 Scour Localisation Models: Support-Level Sliding Windows

For localisation, the basic data stream is again the ordered sequence of crossings, but now each crossing *j* is resolved at the level of individual supports. For a given bridge, n_{supp} is the number of supports, and N_{sim} is the number of simulations. For each simulation $j=1,\dots,N_{sim}$ and support $p=1,\dots,n_{supp}$, $x(j,p)$ is the feature vector for support *p* at crossing *j*, $y(j,p) \in \{healthy,\ scoured\}$, and $s(j,p)$ is the scour percentage at support *p*, and $snm(p)$ is the support name, as formulated from Eq. (5.100) to Eq. (5.103):

$$x(j,p) = \begin{bmatrix} x(j,p,1) \\ x(j,p,2) \\ \vdots \\ x(j,p,n_{\text{feat}}) \end{bmatrix} \in \mathbb{R}^{n_{\text{feat}}} \tag{5.100}$$

$$y(j,p) \in \{\text{healthy, scoured}\} \tag{5.101}$$

$$s(j,p) \in \mathbb{R} \tag{5.102}$$

$$snm(p) \in S_{supp} \tag{5.103}$$

where n_{feat} is the number of input features, and S_{supp} is the set that contains all support names of the bridge. If a global row index is defined as in Eq. (5.104):

$$r_g(j,p) = (j-1)\, n_{\text{supp}} + p, j = 1, \dots, N_{\text{sim}},\ p = 1, \dots, n_{\text{supp}} \tag{5.104}$$

Then the stacked support-level data matrices are formulated as from Eq. (5.106) to (5.107):

- Feature matrix: $xC \in \mathbb{R}^{N_{\text{sim}} n_{\text{supp}} \times n_{\text{feat}}}$:

$$xC(r_g(j,p),k) = x(j,p,k), k = 1, \dots, n_{\text{feat}} \tag{5.105}$$

- Label vector: $yC \in \{\text{"healthy, scoured"}\}^{N_{\text{sim}} n_{\text{supp}}}$:

$$yC(r_g(j,p)) = y(j,p) \tag{5.106}$$

- Scour vector: $ScrC \in \mathbb{R}^{N_{\text{sim}} n_{\text{supp}}}$:

$$ScrC(r_g(j,p)) = s(j,p) \tag{5.107}$$

For a chosen window size w_s, the number of windows along the simulation axis is $F_w = N_{sim} - w_s + 1$. Windows are formed independently for each support; there are $F_w \times n_{supp}$ batches per bridge.

i. Collecting crossing indices

For a fixed window z and a fixed support p, all simulations j that belong to this window are $j = z, z+1, \dots, z+w_s-1$. For each such simulation index j, since each simulation contributes a block of n_{supp} consecutive rows. The corresponding row index in the stacked support-level data matrices xC, yC, and $ScrC$ is as in $r_g(j,p)$. For each row_{idx}, $x(j,p)$, $y(j,p)$, and $s(j,p)$ are obtained and stored in the three-dimensional window arrays for the current sample (window z, support p) as in Eq. (5.108) to Eq. (5.110):

$$window_x(b,k,s_{Idx}) = x(j,p,k);\ k = 1, \dots, n_{feat} \tag{5.108}$$

$$window_y(b,1,s_{Idx}) = y(j,p) \tag{5.109}$$

$$window_{Scr}(b, 1, s_{Idx}) = s(j, p) \tag{5.110}$$

where s_{Idx} is the index of the current sample (window z, support p), $b=1, 2, \ldots, w_s$ is the position of simulation j inside window z, therefore $j=z+b-1$.

ii. Aggregated input features:

For each feature index $k=1, \ldots, n_{feat}$, statistics are computed over the w_s values of that feature at support p across the window as: mean: $\mu_k(z,p)$, standard deviation: $\sigma_k(z,p)$, minimum: $min_k(z,p)$, maximum: $max_k(z,p)$.

iii. Individual features:

The individual feature vectors for support p across the w_s crossings are combined as in Eq. (5.111):

$$r(z,p) = [x(z,p)^\top, x(z+1,p)^\top, \ldots, x(z+w_s-1,p)^\top]^\top \tag{5.111}$$

iv. Batch feature vector:

The batch feature vector for the pair (window z, support p) is as in Eq. (5.112):

$$X_{\text{batch}}(z,p) = [\,\mu(z,p), \sigma(z,p), \min(z,p), \max(z,p), r(z,p)\,] \in \mathbb{R}^{(w_s+4)n_{feat}} \tag{5.112}$$

v. Batch label, scour value and support name:

Class the label of the batch taken from the last crossing in the window is in Eq. (5.113):

$$Y_{batch}(z,p) = y(z+w_s-1,p) \tag{5.113}$$

Scour percentage of the batch at that support, from the last crossing, is as in Eq. (5.114):

$$SCR_{batch}(z,p) = s(z+w_s-1,p) \tag{5.114}$$

Support name is as in Eq. (5.115):

$$SNM_{batch}(z,p) = snm(p) \tag{5.115}$$

For each bridge b, the localisation windowing produces a per-bridge dataset that is stored as $X_{batch}^{(b)}$, $Y_{batch}^{(b)}$, $SCR_{batch}^{(b)}$, and $SNM_{batch}^{(b)}$, with one row per sample defined by a (window z, support p) pair. For example, the first sample for support p of the first bridge is built from crossings $j = 1$ to w_s at that same support. Its input row is $X_{batch}^{(1)}(z = 1, p)$, while its label and scour percentage are taken from the last crossing in the window, that is, $Y_{batch}^{(1)}(1, p) = y(w_s, p)$ and $SCR_{batch}^{(1)}(1, p) = s(w_s, p)$. The support-name entry is $SNM_{batch}^{(1)}(1, p) = snm(p)$, which stays the same for that support across all windows.

Collecting over all windows and supports gives, for each bridge, a dataset in Eq. (5.116):

$$\{X_{batch}(z,p),\ Y_{batch}(z,p),\ SCR_{batch}(z,p),\ SNM_{batch}(z,p)\}, z = 1, \dots, F_w,\ p = 1, \dots, n_{supp} \quad (5.116)$$

As with the scour detection task, windowed datasets are created per bridge and then vertically combined across bridges to train and evaluate the scour-localisation classifiers. Semantically, each batch now answers the question: "Is support *p* scoured at the last crossing?" given the recent history of that same support across the preceding w_s-1 crossings.

5.6 Handling Class Imbalance

The scour calendars in Figure 4.4 and Figure 4.5 are naturally imbalanced, since periods of high rainfall seasons and floods dominate, and only a smaller fraction of crossings corresponds to below-5% threshold. This leads to a disproportion between the scoured and healthy classes in scour detection models, which can bias classifiers towards the majority class and reduce sensitivity to scour detection. To mitigate this, the dataset is balanced before model training. For the scour localisation models, the class distribution is defined at the support-level rather than at the crossing level. The modelling approach allows at most three scoured supports per bridge at any time step. Hence, for localisation models, the minority class, i.e., the scoured class, requires balancing.

To balance the number of samples in each class, initially, two sampling methods are implemented. Oversampling of the minority class has been performed with the Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002), while random under sampling of the majority class has also been explored for comparison (Lin et al., 2017; Shelke et al., 2017). These techniques adjust the ratio between majority and minority samples in the training sets before classifier training, and do not modify the learning algorithms themselves.

5.6.1 The Synthetic Minority Oversampling Technique (SMOTE)

SMOTE is a synthetic oversampling method that generates new minority samples by interpolating between existing ones (Chawla et al., 2002). For each minority example x_i, its *k* nearest minority neighbours are defined in feature space. Synthetic samples are then created along the line segments joining x_i to these neighbours. For a minority feature vector x_i and one of its *k* nearest neighbours x_{nn}, a synthetic point x_{new} is generated as in Eq. (5.117):

$$x_{\text{new}} = x_i + \lambda_S\,(x_{\text{nn}} - x_i) \quad (5.117)$$

where λ_s is a random number drawn uniformly between 0 and 1 (Chawla et al., 2002). This way, a random point is chosen along the line segment between x_{nn} and x_i. After oversampling, the augmented

training set is used in the usual way to train the classifier. Nearest neighbours have been obtained using the Hamming distance. The Hamming distance d_H between two feature vectors x_i *and* x_{nn} of length p_H counts the number of positions at which they differ (Bookstein et al., 2002). It can be written as in Eq. (5.118):

$$d_H(x_i, x_{nn}) = \sum_{j=1}^{p_H} I(x_{i,j} \neq x_{nn,j}) \tag{5.118}$$

where $I(\cdot)$ is an indicator that equals 1 if its argument is true and 0 otherwise. This distance treats two vectors as more similar when more of their coordinates coincide. Calculation details of the MATLAB code are provided in Annex G.

5.6.1.1 Implementation for scour detection models

For the bridge-level scour detection models, each row of the training data corresponds to a full crossing, with input features X, a binary label Y ("healthy" for below-threshold scour, "scoured" for above 5% scour), and an associated crossing-level scour percentage vector of SCR. SMOTE is applied when the majority–minority ratio differs by more than 5%, i.e., as in Eq. (5.119):

$$\max\left(N_{\text{scoured}}, N_{\text{healthy}}\right)/\min\left(N_{\text{scoured}}, N_{\text{healthy}}\right) > 1.05 \tag{5.119}$$

where $N_{scoured}$ and $N_{healthy}$ are the class counts. The minority subset X_{min} is extracted based on the smaller of the two classes, and the number of synthetic samples is set to Eq. (5.120):

$$N_{syn} = | N_{scoured} - N_{healthy} | \tag{5.120}$$

With five neighbours and Hamming distance, for each minority example, N_{syn} synthetic samples are generated. Synthetic feature vectors are generated using Eq. (5.113). All synthetic points are labelled with the chosen target class ("scoured" or "healthy") so that the final training set becomes approximately balanced. In addition to features and labels, each synthetic crossing is assigned a scour percentage drawn uniformly from a set of predefined bins.

Scour detection models have a minority class of "healthy". The target scour percentage of the synthetic cases, i.e., Scr_{new} is sampled within one of three intervals of [0,1.5], [1.5,3.5], or [3.5,5] %. The total of N_{syn} samples is split evenly across these three bins, so that each bin receives approximately $N_{syn}/3$ synthetic cases. After oversampling, the synthetic crossings are appended to the original training data as in Eq. (5.121):

$$X_{\text{balanced}} = \begin{bmatrix} X \\ X_{\text{syn}} \end{bmatrix}, Y_{\text{balanced}} = \begin{bmatrix} Y \\ Y_{\text{syn}} \end{bmatrix} \tag{5.121}$$

The balanced features and labels are then used for training the detection classifiers. Details of the MATLAB code are provided in Annex G.

5.6.1.2 Implementation for scour localisation models

For the support-level localisation models, each row of the training data corresponds to a specific support on a specific crossing. The inputs X are support-level features, the labels Y indicate whether that support is above (scoured) or below (healthy) the 5% scour threshold, and an additional vector snm stores the support names. As noted above, in the setting of localisation models, the majority class is "healthy". The same SMOTE logic with scour detection models is used but extended to carry support identifiers. The minority subset X_{min}, labels Y_{min}, scour percentages, and support names Z_{min} are extracted for the "scoured" class. The number of synthetic samples N_{syn} is taken as the difference between the majority and minority counts, and k-nearest neighbours are found within the minority subset using Hamming distance. The synthetic support is labelled as "scoured". Its scour percentage is drawn uniformly from one of the three "scoured" bins [5,10], [10,15], or [15,20] %, with the total N_{syn} again divided approximately equally across the bins. The support identifier for each synthetic sample is inherited from either the base support or its neighbour, chosen at random with equal probability. The final balanced localisation training set consists of the original support-level samples plus these synthetic "scoured" supports and is used for training the localisation classifiers.

5.6.2 Random Undersampling and Chosen Method

Under sampling handles class imbalance by reducing the number of majority-class samples. As summarised by Shelke et al. (2017), this reduction can be performed with random under sampling or using statistical or clustering information to guide which majority samples to remove. Informed under sampling methods include data-cleaning steps to refine the majority subset further.

In this thesis, for each training set, the number of majority-class observations was reduced by randomly deleting majority examples until the majority and minority counts were equal. No clustering or additional selection criteria were applied. In the earlier stages of the research, when the feature sets (Type A and Type B) had not yet been finalised, a comparison was made between SMOTE-based oversampling and random under sampling. For rear- and front-mounted sensor cases of both scour detection and localisation models, oversampling with SMOTE consistently delivered higher classification performance. In addition, random under sampling discards part of the available training data, whereas SMOTE preserves the full training set. Since larger training sets are generally expected to support more stable model training, this was a supportive reason for preferring SMOTE. Consequently, random under sampling was not

used in the final ML framework, and SMOTE-based oversampling was adopted as the main strategy for handling class imbalance in both detection and localisation tasks.

5.7 Distributions of Input Features

Histograms of the individual features are computed from the feature block corresponding to the first crossing in each sliding window. For each sliding window of length $w_s = 7$, the Type-A and Type-B feature vectors used in the histograms are taken from the first crossing in that window. In the localisation model, this is done separately for each support. Only histograms for $w_s = 7$ are presented, since larger batch sizes produce similar distributions. RILA rear-mounted case histograms are provided in Annex H.

5.7.1 Histograms for Scour Detection Models

For each feature k, the histogram is built from the set $X_{batch}(z,k)$ over all sliding windows z. The x-axis shows the $\log_{10}$-transformed batch value, i.e., $log_{10}(X_{batch}(z,k))$, and the y-axis shows how many windows fall into each histogram bin, namely the count of sliding windows with values in that range. Figure 5.54 shows log10-transformed Type-A features; Figure 5.55 and Figure 5.56 illustrate Type-B features' histograms for the RILA front-mounted case. Since the windows advance one crossing at a time along the calendar, all crossings except the last w_s-1 in each calendar appear at least once in this first position. The individual $\log_{10}$-transformed Type-A histograms show that the DWT-based skewness and kurtosis features are concentrated around 1-2 range(s), with varying degrees of skewness and spread. The total wavelet energy and the minimum approximation level display more clustered patterns, with several dominant value ranges. In contrast, many of the Type-B features exhibit more multi-modal and widespread distributions. This indicates that the Markov features are sensitive to changes in the bridge responses and capture distinct regimes across the simulated crossings.

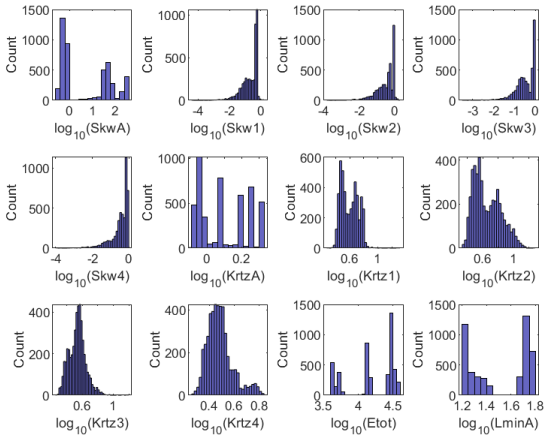


Figure 5.54. Histograms of $\log_{10}$-transformed Type-A features from the first crossing in each 7-day window of the scour detection dataset (RILA front-mounted case).

For each sliding window of length w_s = 7, Figure 5.57 andFigure 5.58 show the $\log_{10}$-transformed aggregated Type-A features, and Figure 5.59 - Figure 5.63 present the corresponding aggregated Type-B features for the RILA front-mounted case. The remaining positions within each window (second to seventh crossing) yield very similar empirical feature distributions. The mean values of $\log_{10}$-transformed skewness and kurtosis mostly lie in relatively narrow ranges, with clear differences between the approximation level and the detail levels. The mean of log-transformed total energy and the minimum of approximation coefficient features form a few distinct value bands, indicating 3-4 groups of windows with similar energy or minimum-response levels.

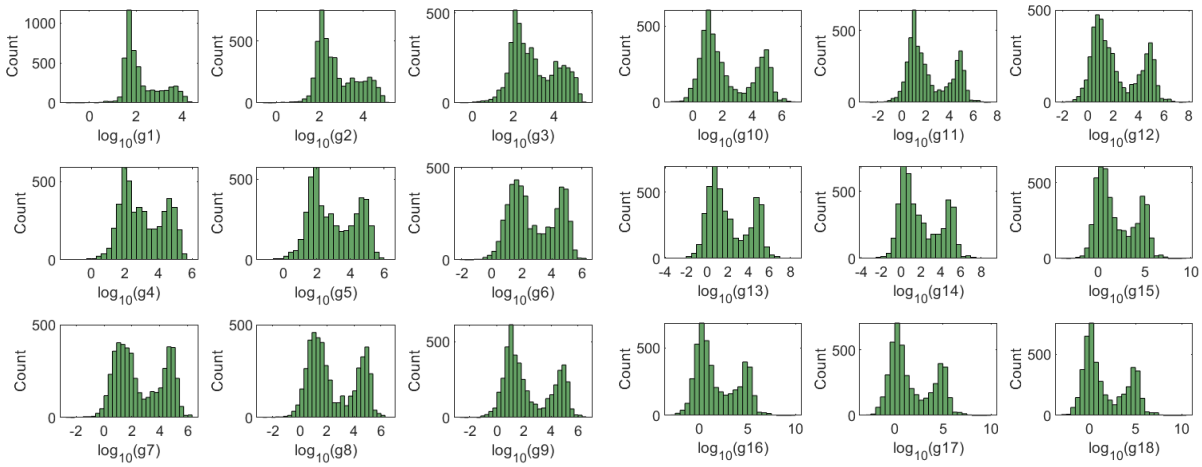


Figure 5.55. Histograms of $\log_{10}$-transformed Type-B features from the first crossing in each 7-day window of the scour detection dataset part 1 (RILA front-mounted case).

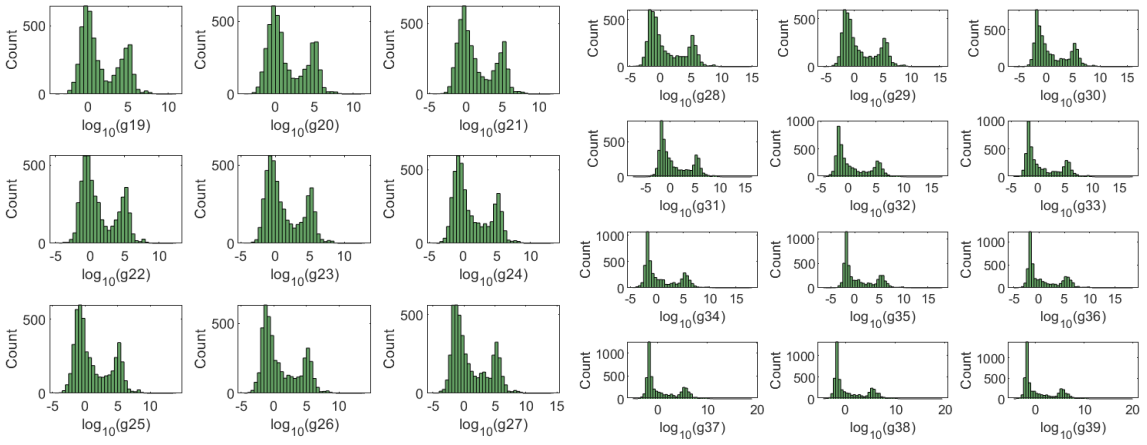


Figure 5.56. Histograms of $\log_{10}$-transformed Type-B features from the first crossing in each 7-day window of the scour detection dataset part 2 (RILA front-mounted case).

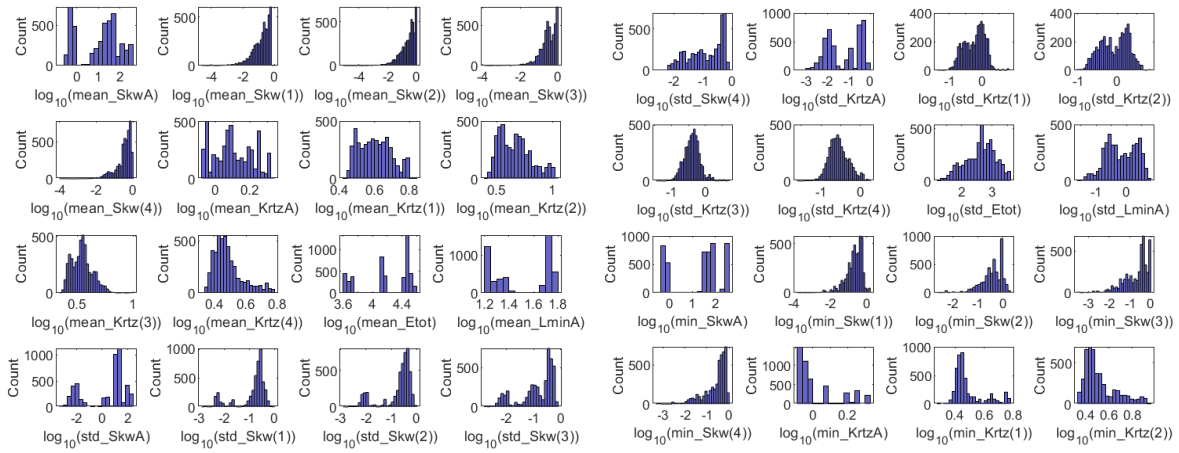


Figure 5.57. Histograms of $\log_{10}$-transformed aggregated Type-A features for batch size 7 of the scour detection dataset, part 1 (RILA front-mounted case).

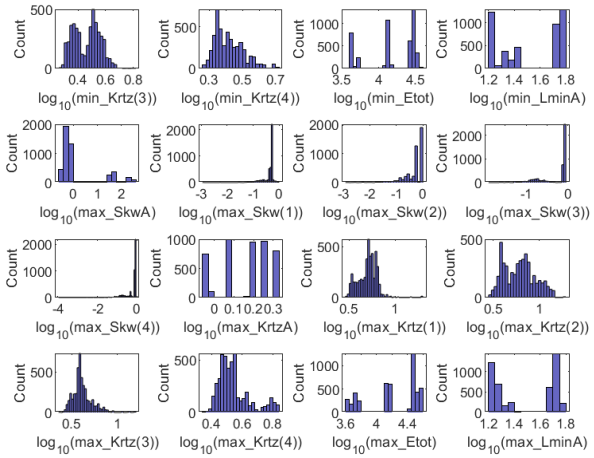


Figure 5.58. Histograms of $\log_{10}$-transformed aggregated Type-A features for batch size 7 of the scour detection dataset, part 2 (RILA front-mounted case).

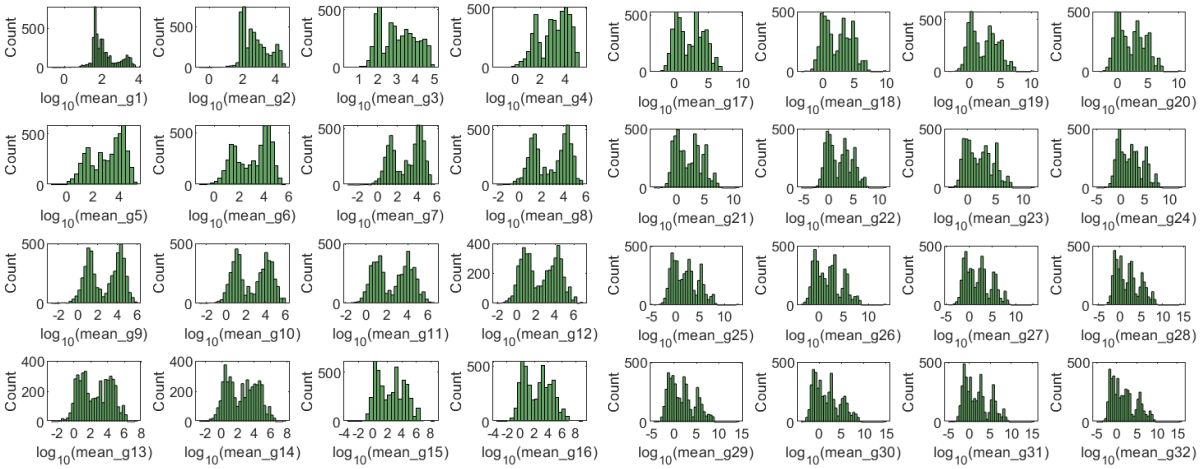


Figure 5.59. Histograms of $\log_{10}$-transformed aggregated Type-B features for batch size 7 of the scour detection dataset, part 1 (RILA front-mounted case).

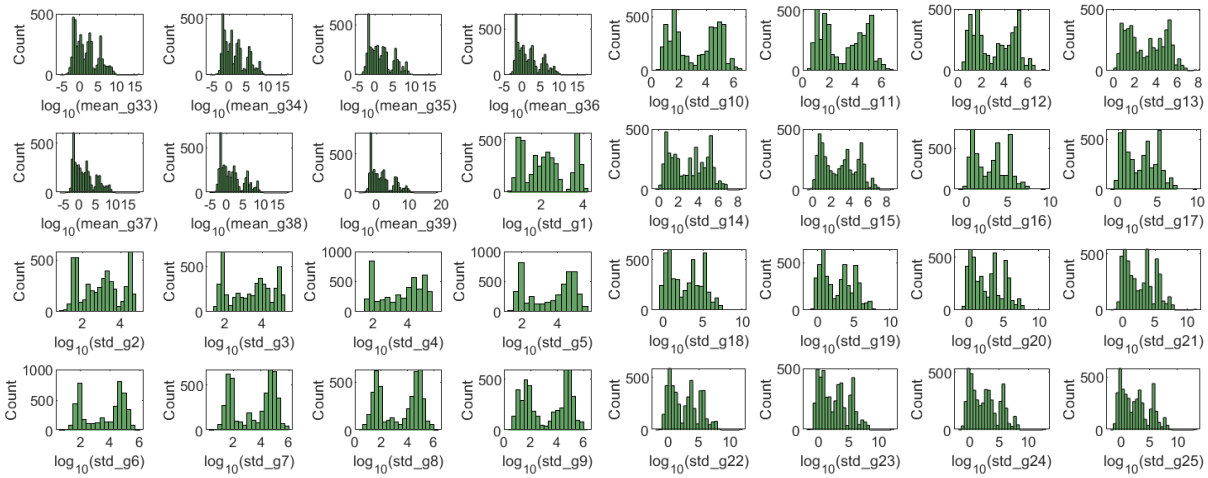


Figure 5.60. Histograms of $\log_{10}$-transformed aggregated Type-B features, batch size 7 of the scour detection dataset, part 2 (RILA front-mounted case).

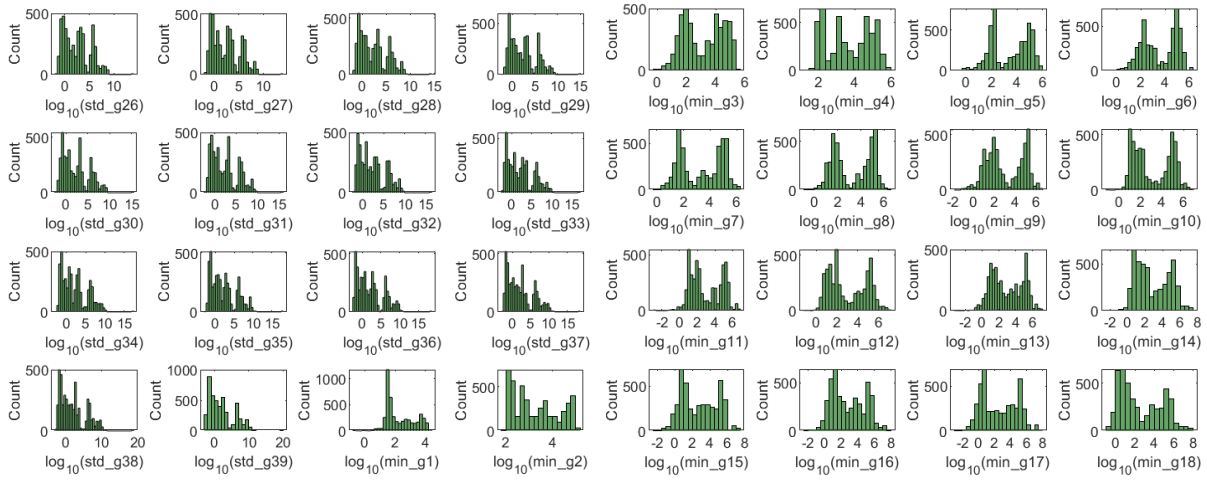


Figure 5.61. Histograms of $\log_{10}$-transformed aggregated Type-B features, batch size 7 of the scour detection dataset, part 3 (RILA front-mounted case).

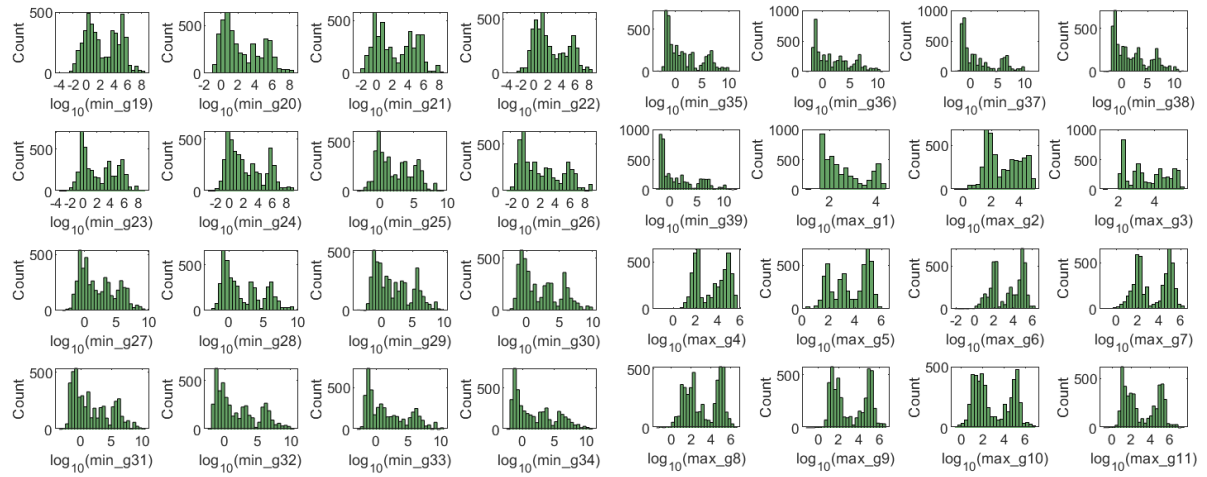


Figure 5.62. Histograms of $\log_{10}$-transformed aggregated Type-B features, batch size 7 of the scour detection dataset, part 4 (RILA front-mounted case).

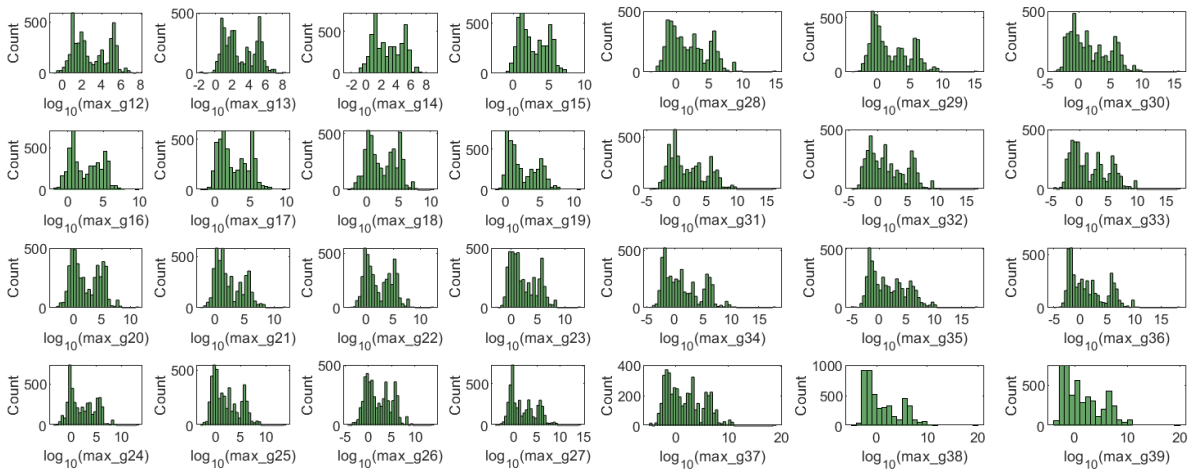


Figure 5.63. Histograms of $\log_{10}$-transformed aggregated Type-B features, batch size 7 of the scour detection dataset, part 5 (RILA front-mounted case).

5.7.1 Histograms for Scour Localisation Models

The histograms are computed from the feature block associated with the first simulation position in each sliding window of length $w_s = 7$. Since the windows advance by one crossing at a time along the calendar for each support, all crossing–support combinations except those in the last $w_s - 1$ crossings of each calendar appear at least once in this first position. For each feature index k, the histogram is built from all values of that feature across the combined batch dataset, that is, from $X_{batch}(z,p,k)$ for all windows z ranging from 1 to F_w and all supports p ranging from 1 to n_{supp}. The x-axis shows the $\log_{10}$-transformed batch value, $log_{10}(X_{batch}(z,p,k))$, and the y-axis shows how many (z,p) batches fall into each histogram bin.

Figure 5.64 shows the $\log_{10}$-transformed Type-A feature histograms, while Figure 5.65 illustrates the Type-B feature histograms for the RILA front-mounted scour localisation model. The individual Type-A descriptors model remains relatively compact after the $\log_{10}$-transform, with most samples concentrated in narrow intervals. The $\log_{10}$-transformed skewness features exhibit peaked, predominantly unimodal distributions. These clusters are close to zero on the log scale, with only a small fraction of samples in the lower tails. This indicates that, at each support, the DWT coefficients rarely exhibit extreme asymmetry across crossings and windows.

The individual Type-B descriptors, i.e., Markov parameters $g(1)$to $g(13)$, show a much more concentrated behaviour than the Type-A set. On the $\log_{10}$ scale, parameters $g(2)$ to $g(10)$ form very sharp peaks around a narrow central value, with only light side tails. This indicates that across crossing–support windows, these Markov coefficients remain within a narrow range and only occasionally deviate. $g(11)$ and $g(12)$ have broader spreads and some mild multi-peaked structure, suggesting that the dynamic response

captured at these positions varies more between healthy and scoured conditions for these parameters. The first and last Markov parameters have a significant spike at very negative $\log_{10}$ values with a small cluster at higher levels, indicating that these coefficients are close to zero in most windows and become active only in a subset of cases. Overall, the Markov features for localisation are highly structured: most positions of the Markov vector are nearly constant. At the same time, a small subset shows greater variability and may contain more discriminative information.

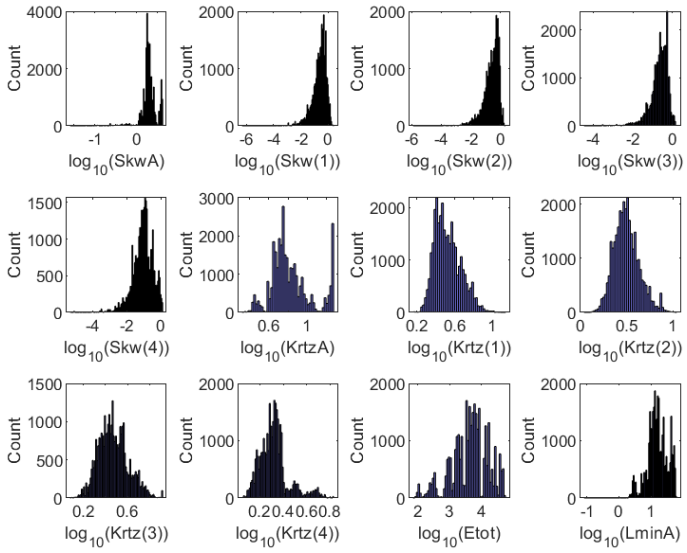


Figure 5.64. Histograms of $\log_{10}$-transformed Type-A features from the first crossing in each 7-day window of the scour localisation dataset (RILA front-mounted case).

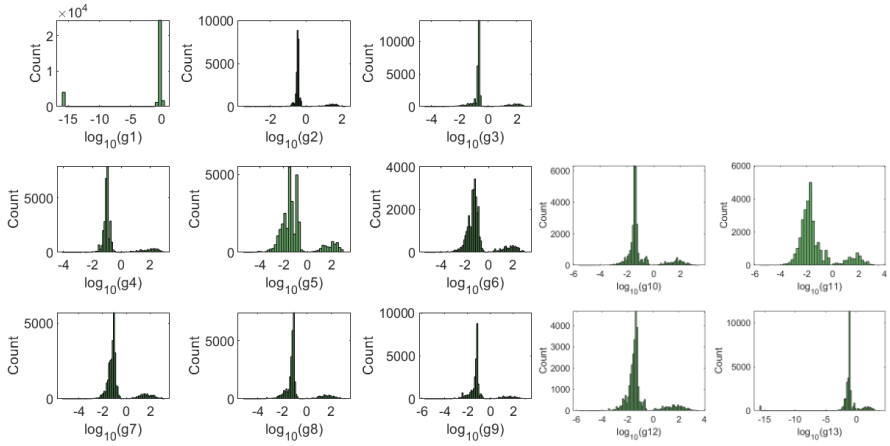


Figure 5.65. Histograms of $\log_{10}$-transformed Type-B features from the first crossing in each 7-day window of the scour localisation dataset (RILA front-mounted case).

Figure 5.66 and Figure 5.67 illustrate the $\log_{10}$-transformed aggregated Type-A features of the RILA front-mounted scour localisation model. The mean-based aggregates show narrow, mostly unimodal histograms on the $\log_{10}$ scale. This indicates a limited dispersion of the window-averaged DWT statistics

across all supports and windows. Kurtosis-related means exhibit a broader spread and mild multi-peaked shapes. The mean total energy shows wider support than the skewness- and kurtosis-based parameters, reflecting greater variation in average wavelet energy across windows. The standard-deviation aggregates are concentrated at very negative $\log_{10}$ values, with sharp spikes close to a single bin. This implies that, within a given window and support, the corresponding Type-A descriptor does not change a lot across crossings. The minimum and maximum aggregates display broader supports and occasional secondary peaks.

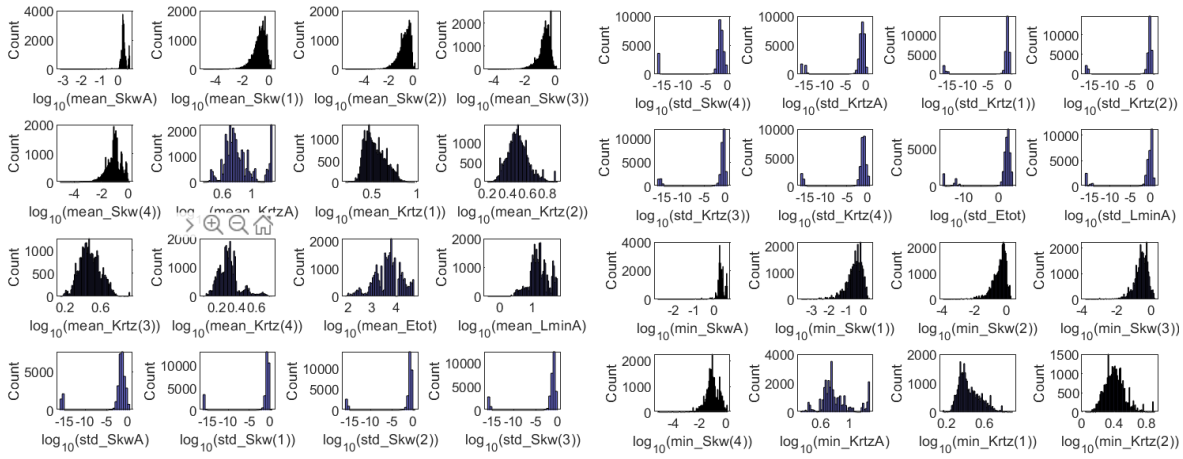


Figure 5.66. Histograms of $\log_{10}$-transformed aggregated Type-A features for batch size 7, part 1 of the scour localisation dataset (RILA front-mounted case).

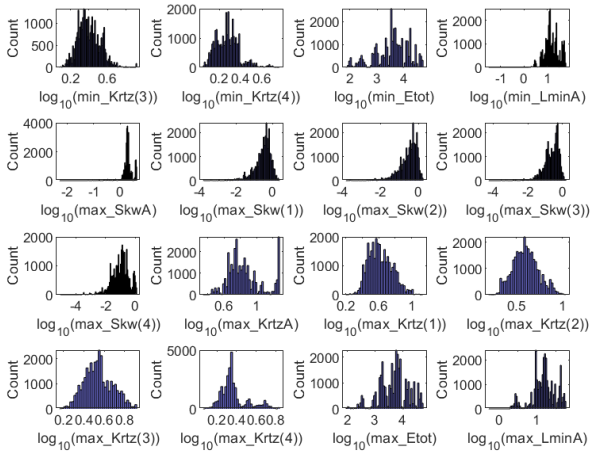


Figure 5.67. Histograms of $\log_{10}$-transformed aggregated Type-A features for batch size 7, part 2 of the scour localisation dataset (RILA front-mounted case).

Figure 5.68 to Figure 5.70 show the aggregated and log10-transformed Type-B features for the RILA front-mounted case. The mean-based aggregates of *g(2)–g(10)* form peaks with narrow tails. In contrast, *g(1)* and *g(13)* have tall spikes at very negative $\log_{10}$ levels with small secondary clusters at higher values. This shows that the first and last Markov coefficients are close to zero in most batches. The mean

histograms of *g(5), g(11)*, and *g(12)* are noticeably broader and, in places, multi-peaked, suggesting distinct regimes in which these orders exhibit different typical levels. The $\log_{10}$-transformed standard-deviation aggregates are strongly concentrated at low values for all orders, with sharp spikes and only a few secondary modes. This indicates that, within each window at a given support, the corresponding Markov coefficients exhibit minimal variation across the $w_s = 7$ crossings. The minimum and maximum aggregates of *g(2)–g(10)* remain within a relatively narrow band, while *g(1)* and *g(13)* are close to zero. The minimum and maximum histograms of *g(5), g(11)* and *g(12)* span a wider range of values.

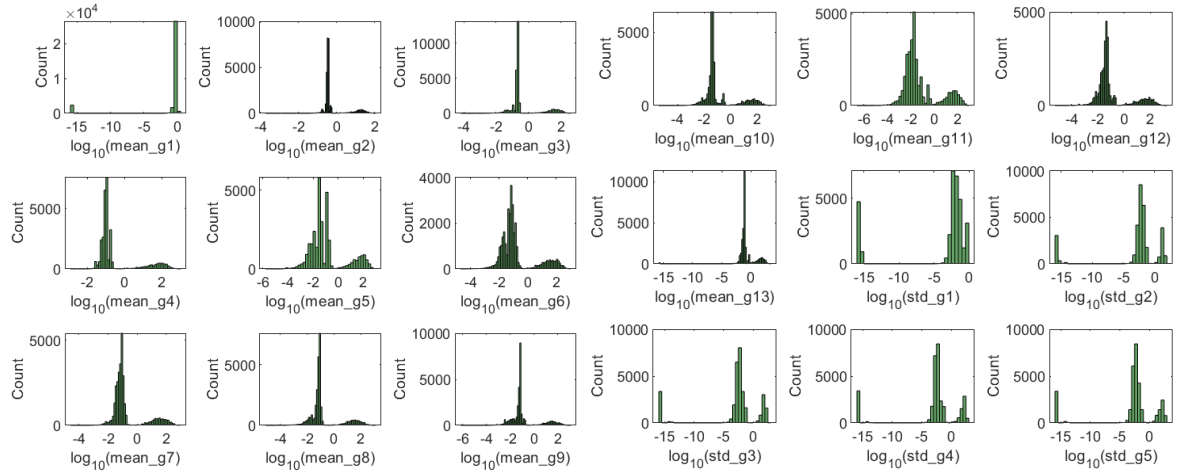


Figure 5.68. Histograms of $\log_{10}$-transformed aggregated Type-B features for batch size 7, part 1 of the scour localisation dataset (RILA front-mounted case).

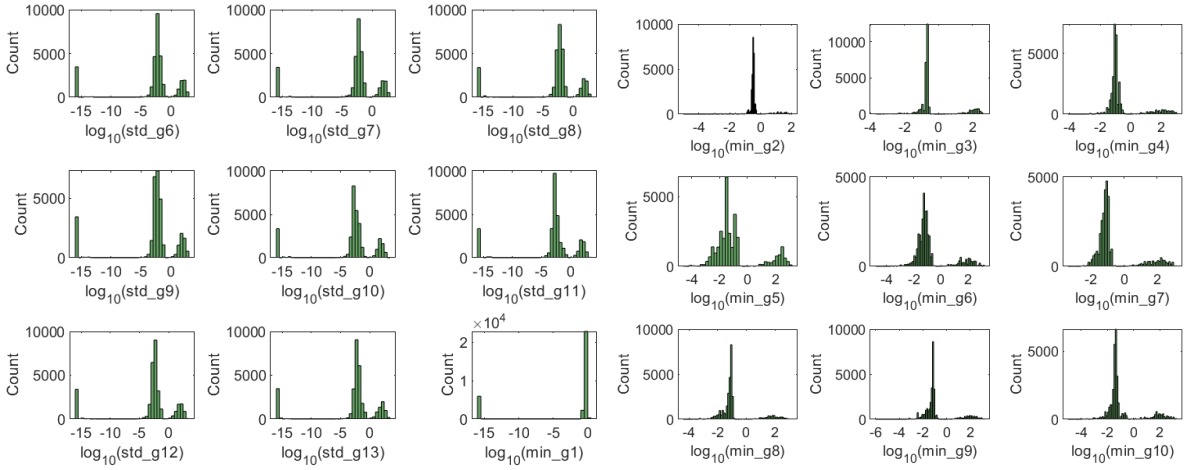


Figure 5.69. Histograms of $\log_{10}$-transformed aggregated Type-B features for batch size 7, part 2 of the scour localisation dataset (RILA front-mounted case).

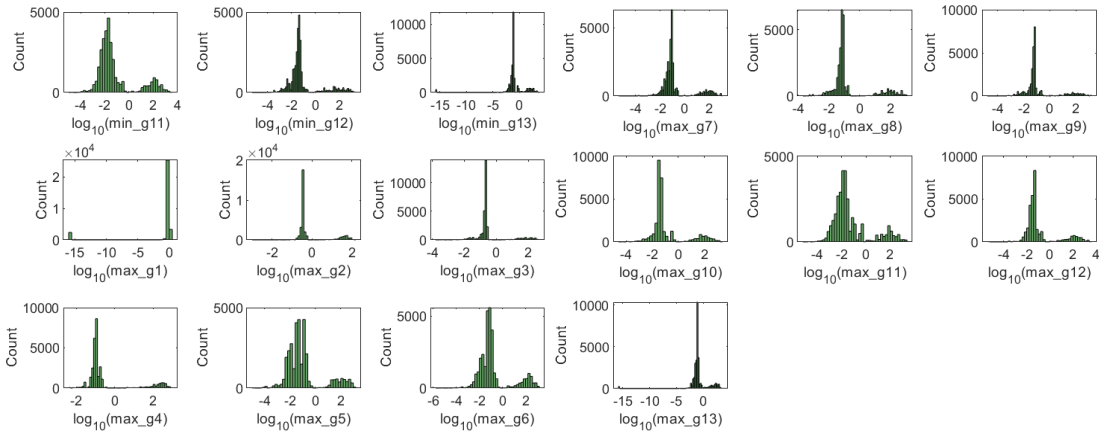


Figure 5.70. Histograms of $\log_{10}$-transformed aggregated Type-B features for batch size 7, part 3 of the scour localisation dataset (RILA front-mounted case).

5.8 Machine Learning Models and Preprocessing

This section describes the classification models and preprocessing steps used in the ML framework. It introduces the three classifiers considered: ANN, RF, and SVM with RBF kernel, outlining their decision rules and architectures. For each model, the loss function and regularisation terms are defined, along with the hyperparameters tuned. The section also explains how hard class labels are obtained and links these modelling choices to the nested CV procedure described earlier in Section 5.2.

5.8.1 Hyperparameter Tuning Principles

This subsection outlines the general principles used to train and tune all ML models in the scour detection and localisation pipelines. At batch size $w_s = 7$, a full hyperparameter search is carried out, and the best regions of this space are then used to initialise narrower "transfer" searches for the larger batches (15, 31, 61). The training and hyperparameter-tuning procedure consist of the following steps:

i. Building sliding-window features for each bridge and stacking per-bridge sliding-window features into global arrays X_{batch}, Y_{batch}, SCR_{batch} that are used for training the ML models, as explained in Section 5.5.

ii. Balancing the classes with SMOTE, as explained in section 5.6.1.

iii. Defining the nested CV scheme by creating an outer stratified 5-fold partition on Y_{batch}.

iv. Looping over outer folds:

- Splitting X_{batch}, Y_{batch}, SCR_{batch} into outer-train and outer-test sets for the current fold.

- For ANN and SVM, computing z-score statistics as explained in section 5.4 on the outer-train features and applying them to both outer-train and outer-test, keeping RF inputs in their raw form.
- Creating an inner stratified CV on the outer train set.

v. Tuning hyperparameters within inner folds:

- Fix random seeds and search budgets to ensure reproducibility across folds.
- For each hyperparameter configuration in the grid or random search, training the model on the inner-training data and evaluating it on the inner-validation data.
- Computing the mean validation accuracy across inner folds and selecting the configuration with the best mean accuracy.

vi. Selecting the best hyperparameters: For the current outer fold, choosing the hyperparameter configuration with the highest mean inner-fold validation accuracy.

5.8.2 Artificial Neural Network (ANN) Classifier

5.8.2.1 Model structure and decision rule

For both scour detection and scour localisation, the ANN is used as a binary classifier acting on a feature vector $x \in R^{nfeat}$. The classifier implements a scalar decision function f_{ANN} in Eq. (5.122):

$$f_{\text{ANN}}: \mathbb{R}^{n_{\text{feat}}} \rightarrow [0,1] \tag{5.122}$$

The estimated probability that the observation belongs to the scoured class ("scoured") is denoted as in Eq. (5.123):

$$p_{\text{scour}}(\text{x}) = f_{\text{ANN}}(\text{x}) \tag{5.123}$$

Hence, the probability of the healthy class "healthy" is denoted with $1\text{-}p_{scour}(x)$. The (binary) class label is Y, and the feature vector is X. For a given input x, the network output $p_{scour}(x)$ is an estimate of the posterior probability, i.e., the probability of the scoured class after observing the features x, denoted as in Eq. (5.124):

$$p_{\text{scour}}(\text{x}) \approx \mathbb{P}(Y = \text{“scoured”} \mid \text{X} = \text{x}) \tag{5.124}$$

The binary decision rule is obtained by thresholding this probability at a fixed level $t \in (0,1)$. The classifier prediction is expressed in Eq. (5.125):

$$\hat{y}(\mathrm{x}) = \begin{cases} \text{“Scoured”}, & \text{if } p_{\text{scour}}(\mathrm{x}) \geq t \\ \text{“Healthy”}, & \text{if } p_{\text{scour}}(\mathrm{x}) < t \end{cases} \tag{5.125}$$

In all experiments, a fixed threshold $t=0.5$ is used, so $\hat{y}(x)$ corresponds to the most probable class under the ANN model. In decision-theory terms, this threshold corresponds to the case where false positives (i.e., false scour alarms) and false negatives (i.e., missed scour cases) are assumed to have equal costs, and the overall misclassification rate is to be minimised.

5.8.2.2 Network architecture

A feedforward ANN can be viewed as a directed graph G without cycles, $G = (V_N, EN)$, whose nodes are neurons and whose directed edges carry scalar outputs (Shalev-Shwartz & Ben-David, 2014). A sketch of a feedforward ANN is illustrated in Figure 5.71. Here, V_N denotes the set of neurons, and E_N shows the set of directed edges, namely the connections between neurons. Each directed connection is an ordered pair $e=(k,j)$, with $k,j \in V_N$, so that there is an arrow from source neuron k in layer l_N-1 to target neuron j at layer l_N. Each connection $(k,j) \in E_N$ has an associated weight w_{ij}, and the input to neuron j is the weighted sum of the outputs of all its predecessor neurons i for which $(i,j) \in E_N$.

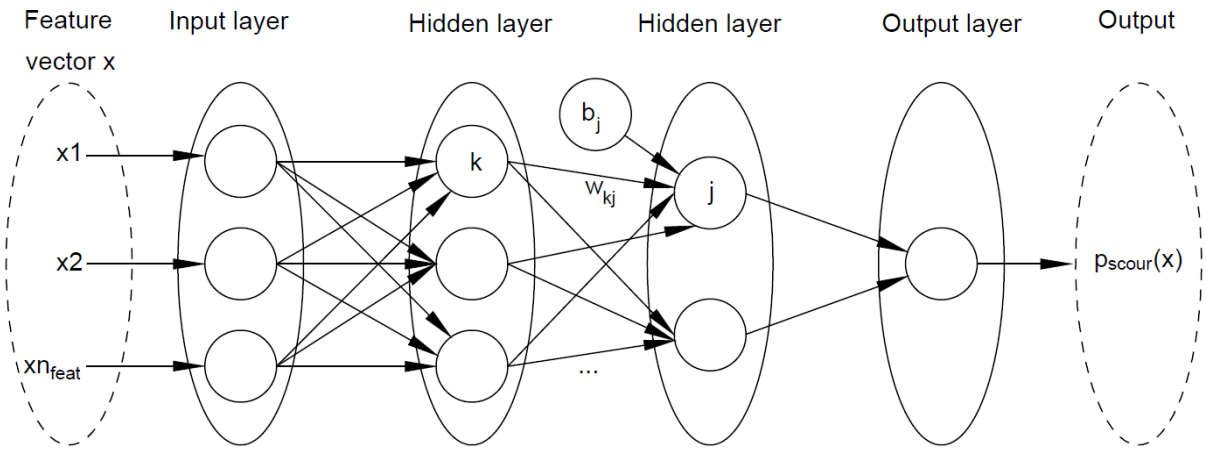


Figure 5.71. Feedforward ANN architecture.

In a layer-wise description, the neurons are organised into layers. The input layer (i.e., layer 0) holds the feature vector x. Each directed connection from neuron k in layer l_N-1 to neuron j in layer l_N has weight $w_{jk}^{(l_N)}$. The input to neuron j is the weighted sum of the outputs of all its previous neurons, as in Eq. (5.126):

$$z_j^{(l_N)} = \sum_k w_{jk}^{(l_N)} \, h_k^{(l_N-1)} + b_j^{(l_N)} \tag{5.126}$$

where $b_j^{(l_N)}$ is the bias of neuron j in layer l_N. For each subsequent layer $l_N = 1, \ldots, l_F$, each neuron first computes the pre-activation $z_j^{(l_N)}$ as above and then the post-activation as in Eq. (5.127):

$$h_j^{(l_N)} = \sigma^{(l_N)}(z_j^{(l_N)}) \tag{5.127}$$

by applying the activation function $\sigma^{(l_N)}(\cdot)$ elementwise. $h_j^{(l_N)}$ is the output of neuron j in layer l_N after the activation function. In vector form, these layer-wise computations can be written as in Eq. (5.128):

$$h^{(0)} = x,\ z^{(l_N)} = W^{(l_N)}h^{(l_N-1)} + b^{(l_N)},\ h^{(l_N)} = \sigma^{(l_N)}(z^{(l_N)}) \tag{5.128}$$

where $W^{(l_N)}$ and $b^{(l_N)}$ are the weight matrix and bias vector of layer l_N. Hence, pre-activation and post-activation are two operations that each neuron performs: a linear combination of its inputs followed by a fixed non-linear activation, respectively. Activation functions introduce nonlinearity, enabling compositions of layers to represent complex, nonlinear decision boundaries (Dubey et al., 2022). Using only the identity activation would collapse the network to an equivalent linear model, so non-linear activations are essential for the extra expressive power of deep architectures.

The ANN classifier used as the first classifier in this thesis is implemented using MATLAB's *fitcnet* function (Matlab, 2025a), which trains a fully connected, feedforward neural network for tabular classification. The architecture consists of

- an input layer of size n_{feat}
- one or two hidden layers with n_{l_N} neurons each
- a single output neuron with a sigmoid activation that produces $p_{scour}(x)$.

For a given input x, the final scalar score $s(x) = z^{(l_F)}$ for the scoured class as a linear combination of the last hidden layer l_F's outputs; this score is the pre-activation of the single output neuron. The corresponding estimated probability $p_{scour}(x)$ is then obtained by applying the logistic sigmoid (Shalev-Shwartz & Ben-David, 2014) at the output layer, as in Eq. (5.129):

$$p_{\text{scour}}(x) = \sigma_{\text{sig}}(s(x)) = \frac{1}{1+\exp(-s(x))} \tag{5.129}$$

which maps the real-valued score to (0,1) so it can be interpreted as a probability. In the binary case, the output of logistic sigmoid function represents the posterior probability of the "scoured" class. The standard element-wise non-linear candidate activation functions that hidden layers use, which are tuned in grid search, are:

- the hyperbolic tangent (tanh), whose explicit formula and properties (range $[-1,1]$, smooth S-shape, zero-centred output) are given in Eq. (5.72) and discussed in Section 5.3.2.1.
- the Rectified Linear Unit (ReLU), defined in Eq. (5.130), which is piecewise linear and leaves positive inputs unchanged while truncating negative inputs to zero:

$$ReLU(v) = \{ v \; if \; v \geq 0, \; 0 \; if \; v < 0 \} \tag{5.130}$$

where the input $v = z^{(l)}$.

5.8.2.3 Loss function, regularisation and training setup

Let $\{(x_i, y_i)\}_{i=1}^{n}$ denote the training set in outer CV fold, where $x_i \in \mathbb{R}^{n_{feat}}$ is the feature vector for sample i and $y_i \in \{0,1\}$ is the class label, with $y_i = 1$ for "scoured" and $y_i = 0$ for "healthy". In matrix form, these samples are stacked into the feature matrix $X \in \mathbb{R}^{N \times n_{feat}}$ and label vector $Y \in \{0,1\}^N$, whose i^{th} row and entry are x_i^T and y_i, respectively. The network outputs $p_i = p_{scour}(x_i)$ for each sample. The empirical loss function used for training is the standard binary cross-entropy, i.e., negative log-likelihood loss, as in Eq. (5.131) (Aggarwal, 2018):

$$E_0(\theta) = -\frac{1}{N} \sum_{i=1}^{N} [\, y_i \log \, p_i + (1 - y_i) \log \, (1 - p_i) \,] \tag{5.131}$$

where θ collects all weights and biases. When the outputs are interpreted as class probabilities, the binary cross-entropy is the negative log-likelihood of the observed class labels under the model. Minimising this loss is therefore equivalent to maximising the likelihood of the training data.

To reduce overfitting, *L2* regularisation is added to the loss, as already introduced in Eq. (5.73) in Section 5.3.2.1. The regularised objective is as in Eq. (5.77) and Eq. (5.132):

$$E(\theta) = E_0(\theta) + \frac{1}{2} \lambda \sum_{i=1}^{p_w} w_i^2 \tag{5.132}$$

where $\lambda > 0$ is the *L2* regularisation strength and $w_1, \ldots, w_p$ are the network weights. This quadratic penalty discourages substantial weights and yields the familiar "weight decay" effect in gradient-based training. During training, the gradient of $E(\theta)$ with respect to all weights and biases is computed by the backpropagation algorithm, which applies the chain rule efficiently through the sequence of linear and activation operations in each layer.

In MATLAB's *fitcnet* function (Matlab, 2025a), the minimisation of the regularised cross-entropy loss is performed by a limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) quasi-Newton algorithm. This is a full-batch, second-order method that uses approximate curvature information to accelerate convergence compared with plain gradient descent, while avoiding the explicit storage of the full Hessian. A quasi-Newton (L-BFGS) is as in Eq. (5.133):

$$\theta_{k+1} = \theta_k + \alpha_k p_k \tag{5.133}$$

where the search direction in parameter space is as in Eq. (5.134):

$$p_k = -H_k\, g_k \tag{5.134}$$

θ is the vector collecting all network parameters (weights and biases), $\ell_{reg}(\theta)$ is the regularised cross-entropy loss, $g_k = \nabla\ell_{reg}(\theta_k)$ is the full-batch gradient at iteration k, and H_k is a limited-memory BFGS approximation of the inverse Hessian $(\nabla^2\ell_{\text{reg}}(\theta_k))^{-1}$. The BFGS update for the inverse Hessian approximation is as in Eq. (5.135):

$$H_{k+1} = (I - \rho_k s_k y_k^{\top})H_k(I - \rho_k y_k s_k^{\top}) + \rho_k s_k s_k^{\top} \tag{5.135}$$

where s_k, y_k, and ρ_k are as in Eq. (5.136):

$$s_k = \theta_{k+1} - \theta_k,\ y_k = g_{k+1} - g_k,\ \rho_k = \frac{1}{y_k^{\top} s_k} \tag{5.136}$$

where s_k and y_k denote, respectively, the change in the parameter vector and in the gradient between iterations k and $k+1$, and $\rho_k=1/(y_k^T s_k)$ is a scalar factor used in the BFGS update. In L-BFGS, MATLAB does not store H_k. Instead, it keeps only the last m pairs (s_i,y_i). It implicitly applies the same BFGS update via a two-loop recursion, reducing the memory and computational cost from $O(p^2)$ to $O(pm)$, where p is the number of network parameters.

For each outer CV fold, the *fitcnet* function (Matlab, 2025a) runs the L-BFGS solver until a prescribed iteration limit is reached. Hyperparameters such as the number of hidden layers, layer widths, activation type (tanh/ReLU), and *L2* regularisation strength are selected by the nested CV/grid search procedure described in Section 5.8.1. Once the best hyperparameter combination has been chosen in the inner folds, a final model is retrained on the complete outer-training set and used to generate ANN predictions on the corresponding outer-test set, which are then used to compute global performance metrics for both scour detection and scour localisation.

5.8.2.4 Hyperparameters and tuning — scour detection models

Batch size 7 (full grid search)

For the bridge-level scour-detection model with batch size $w_s = 7$, both for the RILA front and rear-mounted cases, the ANN classifier hyperparameters were tuned via nested CV, with a grid search in the inner loop and a 5-fold partition for external performance estimation. After the sliding-window step described in Section 5.5.1, each bridge yields the batch-level dataset. $\{X_{batch}(z), Y_{batch}(z), SCR_{batch}(z)\}$ for $z = 1, \ldots, F_w$. The per-bridge datasets are then vertically combined as in Eqs. (5.96) - (5.98) to form the design matrix X and the label vector Y. In matrix form, these samples are stacked into a feature matrix $X \in \mathbb{R}^{N \times n_{feat}}$ and a label vector $Y \in \{healthy, scoured\}^{N_b}$, where N_b is the total number of batches across all bridges for either the RILA front-mounted or rear-mounted

configuration. The i^{th} row of X is the batch feature vector $X_{batch}(z_i)^T$, and the i^{th} entry of Y is the corresponding batch label $Y_{batch}(z_i)$. These X and Y are the inputs to the tuning procedure, and SCR stores the associated scour percentages $SCR_{batch}(z)$.

i. *Nested CV structure*

Hyperparameters are tuned by nested CV.

- Outer loop (external CV):

The batch-level dataset (X, Y) is split into $k_{ext} = 5$ outer folds. This yields five disjoint test sets and corresponding outer-training sets. The partitioning is stratified, so the "scoured"/"healthy" ratio is approximately preserved in each outer fold.

- Inner loop (internal CV):

For each outer training set, a second partition with $k_{int} = 3$ folds is used to define internal training/validation splits for hyperparameter tuning via grid search.

ii. *Pre-processing within each outer fold*

For each outer fold: $k_{ext} = 1, ..., 5$:

a. X_{train} and Y_{train} denote the outer-training indices, while X_{test} and Y_{test} denote the corresponding outer-test indices.

b. Each feature column of X_{train} is standardised to zero mean and unit variance:

- $\mu_{tr} \in \mathbb{R}^{n_{feat}}$ is the vector of training means.
- $\sigma_{tr} \in \mathbb{R}^{n_{feat}}$ is the vector of training standard deviations.

c. The standardised matrices $\tilde{X}_{train}$ and $\tilde{X}_{test}$ are the inputs to the neural network in that outer fold.

d. In MATLAB's *fitcnet* function (Matlab, 2025a), the 'Standardise' option is set to false, because standardisation is performed explicitly using (μ_{tr}, σ_{tr}) from the outer-training set, avoiding any information leakage.

iii. *Grid search*

Grid search is a simple, non-adaptive hyperparameter optimisation strategy (Bergstra & Bengio, 2012). For each component $k = 1, ..., K$, a finite candidate set of values $L^{(k)}$ is chosen in advance. The grid of trial configurations is then the Cartesian product as in Eq. (5.137):

$$H_s = L^1 \times L^2 \times \ldots \times L^K \tag{5.137}$$

where θ denotes the vector of K hyperparameters and H_s the whole hyperparameter space. The total number of trials S_N is the product of the number of candidate values along each hyperparameter dimension, as in Eq. (5.138):

$$S_N = \prod_{k=1}^{K} |L^k| \tag{5.138}$$

For each grid point $\theta^{(s)}$ in H_{grid}, the learning algorithm $A_{\theta^{(s)}}$ is trained, and its performance is evaluated on a validation set via a hyperparameter response function $acc\bar{v}al(\theta)$, for example, mean validation accuracy across inner folds. Here, $A_{\theta^{(s)}}$ denotes the ANN learning algorithm, i.e., *fitcnet* function (Matlab, 2025a), which, given a hyperparameter vector θ, returns a trained classifier. The grid search then selects the configuration that maximises $acc\bar{v}al$ over the grid, as expressed in Eq. (5.139):

$$\hat{\theta} = \arg \max_{\theta \in H_{grid}} ac\bar{c}val(\theta) \tag{5.139}$$

Equivalently, if S_N grid points are indexed as $\{\theta^{(1)}, \ldots, \theta^{(S_N)}\}$, grid search can be written as in Eq. (5.140):

$$\hat{\theta} = \arg \max_{s = 1,\ldots,S_N} ac\bar{c}val(\theta^{S_N}) \tag{5.140}$$

In this section, θ corresponds to the ANN hyperparameters $\theta = (n_{IN}, l_N, \sigma(\cdot), \lambda)$, so grid evaluates all combinations of the predefined values for hidden-layer width, number of hidden layers, activation function and *L2* regularisation strength, and returns the one with the highest mean inner-fold validation accuracy.

iv. <u>*Hyperparameter grid*</u>

For each outer fold, the following hyperparameter grid is explored:

- Hidden-layer width n_{IN}: Number of neurons in a single layer is selected from the set: {5, 10, 20, 30, 40, 50, 60, 70, 80}.
- Number of hidden layers l_N: Number of layers is selected from the set: {1, 2}.

  If $l_N = 1$, *the network has a single hidden layer of size* l_N; if $l_N = 2$, both hidden layers have size n_{IN}, so the layer-size vector is $[n_{IN}, n_{IN}]$.
- Activation function $\sigma(\cdot)$: Activation function is selected as the rectified linear unit (ReLU) or the hyperbolic tangent (tanh).
- *L2* regularisation strength λ: is selected from the set: $\{10^{-6}, 10^{-5}, 10^{-4}\}$.
- Training iteration limits:
  - Inner-loop models use 200 iterations to speed up grid search.

- Final models in each outer fold use 1000 iterations for more thorough training.

v. *Inner-loop grid search (per outer fold)*

Within each outer fold, the 3-fold internal CV operates on (X_{train}, Y_{train}):

a. $k_{int} = 3$ defines three internal training/validation splits.

b. For each candidate hyperparameter vector $\theta_c = (n_{lN}, l_N, \sigma(\cdot), \lambda)$ in the grid, the following steps are carried out:

- The corresponding layer-size vector is constructed: $[n_{lN}:]$ if $l_N = 1$, $[n_{lN}:, n_{lN}:]$ if $l_N = 2$.
- For each internal fold $k_{int} = 1, 2, 3$. A model is trained on the internal training subset, with the layer size, activation, and *L2* regularisation strength chosen from the sets mentioned above in the subsection hyperparameter grid. Then, predictions are obtained on the internal validation subset. The internal validation accuracy for that fold is the ratio of the number of correctly classified batches over the number of batches in the validation fold.
- The inner-loop performance of θ_c is summarised by the mean validation accuracy $\overline{acc}_{val}(\theta)$ of 3 inner folds.

c. The grid search selects the hyperparameter vector $\hat{\theta}$ that maximises $\overline{acc}_{val}(\theta)$ over all candidates in the grid.

Parallel execution is used to accelerate the inner-loop grid search.

vi. *Final training and prediction (per outer fold)*

For each outer fold:

a. The best configuration $\hat{\theta}$ is found by the inner grid search.

b. A final ANN model is trained on the full outer-training data (X_{train}, Y_{train}) using $\hat{\theta}$ and a 1000 iteration limit.

c. This final model is then used to:

- Predict labels on X_{train}, to compute training accuracy.
- Predict labels on X_{test}, providing outer-test predictions for the performance metrics of the model with $w_s = 7$.

d. The true and predicted labels for all outer folds are concatenated.

Batch sizes 15, 31 and 61 (transfer tuning)

For larger batch sizes *(w_s = 15, 31, and 61),* the same nested CV structure as in the batch-7 case is used, with k_{ext} = 5 outer folds and k_{int} = three inner folds. The main difference is that the hyperparameter search is no longer a full grid over architectures and activations. Instead, the best configuration found for w_s = 7 is treated as an anchor, and only a small neighbourhood around its regularisation strength is explored.

i. Anchor configuration from batch size 7

For each sensor configuration, the batch-size-7 tuning produces a best ANN configuration, which is denoted by Eq. (5.141):

$$\hat{\theta}^{(7)} = (\hat{n}^{l_N}, \hat{l}_N, \hat{\sigma}(\cdot), \hat{\lambda}) \tag{5.141}$$

This anchor configuration is stored and then reused as the centre of the search for all larger batch sizes.

ii. Reduced hyperparameter neighbourhood

In the transfer-tuning runs, the network architecture and activation are fixed to the anchor values in Eq. (5.133). Only the *L2* regularisation strength is re-tuned. A small grid around the anchor value $\hat{\lambda}$ is defined, including the values in Eq. (5.142):

$$\lambda \in \{0, 0.5\hat{\lambda}, \hat{\lambda}, 2\hat{\lambda}\} \tag{5.142}$$

with non-finite or negative candidates discarded. The same inner-loop procedure as in the batch-7 case is then applied: for each candidate λ in this reduced grid, models are trained on the inner-training splits and validated on the inner-validation splits. Inner accuracies are averaged across the three folds to obtain $\bar{acc}_{val}(\lambda)$. The value $\hat{\lambda}^{(w_s)}$that maximises $\bar{acc}_{val}(\lambda)$is selected for that outer fold and batch size.

iii. Final training and prediction for each batch size

For each outer fold and each batch size $w_s \in \{15,31,61\}$:

- The ANN is retrained on the complete outer-training set using the fixed architecture $(\hat{n}_{\ell N}, \hat{l}_N, \hat{\sigma}(\cdot))$ and the selected regularisation $\hat{\lambda}^{(w_s)}$;
- Predictions are obtained on both the outer-training and outer-test sets and stored in pooled collectors as in the w_s = 7 case.
- The per-fold best hyperparameters are recorded.

Hyperparameter tuning results

Table 5.16 summarises the final ANN hyperparameter choices for the scour detection models across all batch sizes. Across all batches, a single hidden layer with 60 neurons for the front-mounted and 50 neurons for the rear-mounted case is found to be optimal. Hyperbolic tangent activation yielded the best results across all models. For the RILA front-mounted model, λ increases slightly for batches 15 and 31 relative to batch 7, then decreases again for batch 61. For the RILA rear-mounted model, λ remains close to its anchor value across all batches.

Table 5.16. Hyperparameter tuning results of scour detection ANN models.

Sensor position	Shared settings	Batch-specific settings (batch size: value)
front	$\sigma(\cdot)$ = Hyperbolic tangent. l_N = 1, n_{IN} = 60	λ ={7: 1e-5, 15: 2e-5, 31: 2e-5, 61: 5e-6}
rear	$\sigma(\cdot)$ = Hyperbolic tangent, l_N = 1, n_{IN} = 50	λ ={7: 1e-6, 15: 1e-6, 31: 1.8e-6, 61: 2e-6}

5.8.2.5 Hyperparameters and tuning — scour localisation models

For the pier-based scour localisation models, the same nested CV framework as for the detection models is used: five outer folds and three inner folds, stratified by class. This procedure is applied separately to the RILA front-mounted and rear-mounted localisation models, using the support-level feature matrix and labels defined in Section 5.5.2. Compared with the detection models, the localisation dataset contains many more instances (one per support per batch) and a richer set of hyperparameters. For this reason, hyperparameters are tuned by random search rather than by a full grid search.

Batch size 7 (Random search)

Random search is a simple, non-adaptive alternative to grid search for hyperparameter optimisation. Instead of evaluating all points on a Cartesian grid, several S_N candidate hyperparameter vectors are randomly sampled from a distribution over the hyperparameter space H_s, namely independent uniform distributions for each component. Random search chooses independent and identically distributed samples $\theta^{(1)}, \ldots, \theta^{(S)} \sim p(\theta)$ on H, and then selects the configuration with the best validation performance as in Eq. (5.143):

$$\{\hat{\theta}\} = \arg \max_{\{s = 1,\ldots,S\}} \Psi\left(\theta^{(s)}\right) \tag{5.143}$$

where $\Psi(\theta)$ denotes the hyperparameter response function (e.g. mean inner-fold validation accuracy), Bergstra & Bengio (2012) show that when Ψ effectively depends on only a few hyperparameters (low "effective dimensionality"), random search is much more efficient than grid search: it spends most of its

trials exploring different values along the important dimensions, instead of exhaustively combining values along unimportant ones.

Within each outer fold, the outer-training data are used to run a random search over candidate ANN configurations. A fixed number of trials, $S_N = 50$, in the present runs. For each trial $s = 1, \ldots, S_N$:

- A hyperparameter configuration is drawn at random,
- That configuration is evaluated by inner CV on the outer-training set,
- Its performance metrics are recorded.

The inner CV uses a 3-fold CV. In each inner fold, per-feature z-scoring is computed from the inner-training split and applied to the corresponding validation split. In MATLAB, the *fitcnet* function (MATLAB, 2025a) is used with the 'Standardise' option set to false, so that no additional internal scaling is applied. As in the detection models, standardisation is always learned from the relevant training split (inner or outer) and then applied to the corresponding validation or test split, to avoid information leakage.

Hyperparameters and search space

Each random-search trial samples set θ an ANN configuration in Eq. (5.144):

$$\theta = \left(\hat{n}^{l_N}, \hat{l}_N, \hat{\sigma}(\cdot), \hat{\lambda}, w_{init}, b_{init}\right) \tag{5.144}$$

where, in addition to grid search, the weight and bias initialisation schemes Glorot/Xavier or He-type initialisers, as in Eq. (5.79) to Eq. (5.81), denoted by w_{init} and b_{init}, passed to the *fitcnet* function (Matlab, 2025a) are tuned, as well. The sampling distribution $p(\theta)$ is used: The number of hidden layers l_N is drawn from the set {1, 2, 3} with probabilities 0.25, 0.55 and 0.20, respectively. n_{lN} are drawn with replacement from the set of neuron counts {32, 48, 64, 96, 128} and then sorted in descending order, so that typical architectures range from a single medium-width layer to three wider layers. $\sigma(\cdot)$ is chosen at random between the hyperbolic tangent and the rectified linear unit (ReLU) with equal probability. λ is sampled from a log-uniform distribution between 10^{-6} and 10^{-2}, so that small and large values are explored on a multiplicative scale. The weight-initialisation scheme w_{init} is set to a He-type initialiser when ReLU is used, and to a Glorot/Xavier-type initialiser when tanh is used. The bias initialisation b_{init} is chosen to be either all zeros or all ones, with equal probability. These choices define the sampling distribution $p(\theta)$ from which the S random trials are drawn in each outer fold.

For each sampled configuration θ, the inner three-fold CV proceeds as follows:

- in each inner fold, an ANN is trained on the z-scored inner-training data using the *fitcnet* function (Matlab, 2025a) with the sampled hyperparameters and an iteration limit of about 400 iterations.
- Predictions are obtained on both the inner-training and inner-validation splits.
- The true and predicted labels from all inner folds are concatenated, and performance metrics are computed.

For both training and validation, the accuracy, balanced accuracy, *F1*-score for the scoured class, *F1*-score for the healthy class, and MCC are obtained.

Selection criterion

For each trial s, $S_{val}(s)$ and $S_{tr}(s)$ denote the validation and training metrics, respectively. The primary selection objective is the *F1*-score of the scoured class on the validation set. To decide between two or more hyperparameter settings that look almost equally successful on the *F1*-score of the scoured class, i.e., break ties between configurations with similar metrics, a lexicographic ordering is applied using:

- higher validation *F1*-score of the scoured class,
- validation macro-*F1*
- validation balanced accuracy,
- validation MCC,
- a small train–validation accuracy gap to favour less overfitted models.

Among all valid trials in the outer fold, the configuration with the best combination of these criteria is selected as the fold-specific best configuration, denoted with θ^*.

In the bridge-level scour-detection models, the hyperparameter selection rule uses only the mean inner-fold validation accuracy on the training folds. In this setting, the candidate grid is relatively small and restricted to basic architectural and regularisation choices. Using mean validation accuracy as the objective is therefore sufficient to discriminate between candidate configurations. By contrast, the pier-level localisation models use a richer random search over a higher-dimensional hyperparameter space and operate at the support-level, aiming to identify scoured supports and avoid overconfident, overfitted models. For that reason, the localisation tuning employs a lexicographic selection rule driven by *F1* for the scoured class, macro-*F1*, balanced accuracy, MCC and the train–validation gap, rather than a single accuracy-based criterion.

Final training and pooled predictions

For each outer fold:

i. The best configuration θ^* is taken from the random search for that fold.

ii. The outer-training data are z-scored using the outer-training mean and standard deviation, and the ANN is retrained on the full outer-training set with theta* and a higher iteration limit of 1200 iterations.

iii. This final model is used to predict on both the outer-training and outer-test sets.

- The pooled outer-training predictions are used to compute an overall training accuracy and to assess the generalisation gap.
- The pooled outer-test predictions across all outer folds are used to compute the final localisation metrics.

iv. Alongside the labels and predictions, the corresponding scour-percentage vectors and support names and z-scored test features are stored. These pooled quantities are later used for support-level performance analysis and for relative-importance evaluations.

Batch sizes 15, 31 and 61 (transfer tuning)

For larger batch sizes *(w_s = 15, 31, and 61)*, the same nested CV and random-search framework is retained. However, the hyperparameter search is performed in a transfer-tuning mode rather than as a full random search. The goal is to exploit the information already obtained at w_s = 7 and restrict the search for each larger batch size to a neighbourhood of the best batch-7 configurations.

For each sensor configuration, the batch-7 runs produce a set of fold-specific best configurations θ^* (one per outer fold). Before tuning a larger batch size, these stored configurations are summarised into a single "centre" configuration θ_{centre} by taking:

- the most frequent hidden-layer pattern across folds as the centre architecture,
- the median of the selected λ values as the centre regularisation strength,
- and the most frequent weight and bias initialisation schemes are the centre initialisers.

A default centre configuration with a two-layer [50, 20] architecture, $\lambda \approx 10^{-4}$, Glorot weights and zero biases is used. For each batch size $w_s \in \{15, 31, 61\}$, random search is then run with a reduced trial budget of S_N = 20 trials per outer fold, using a sampling distribution $p_{TT}(\theta_{centre})$ that perturbs the centre

configuration instead of sampling from the broader prior used at $w_s = 7$. Each transfer-tuning trial draws an ANN configuration $\theta = (n^{IN}, l_N, \sigma(\cdot), \lambda, w_{init}, b_{init})$ as follows:

- The activation function is fixed to the hyperbolic tangent, i.e. $\sigma(\cdot) = tanh$, since this was found to be best in batch 7.
- The hidden-layer sizes start from the centre layer-size vector. They are multiplied by positive random perturbation factors on a logarithmic scale, rounded to integers and clipped to a reasonable range between 8 and 128 neurons per layer. When the centre architecture has a single hidden layer, a second layer of similar size is occasionally added to explore both one- and two-layer variants.
- λ is sampled by perturbing the centre value on the $\log_{10}$ scale with a moderate random offset and then clipping to the interval $[10^{-6}, 10^{-2}]$. This concentrates the search around the regularisation strengths that worked well for $w_s = 7$ while allowing variation over approximately half an order of magnitude.
- The weight-initialisation scheme w_{init} is chosen at random as either a Glorot/Xavier-type or a He-type initialiser, and the bias initialisation b_{init} is chosen as either all zeros or all ones, each with equal probability.

For each sampled configuration θ, the inner three-fold CV, metric computation and lexicographic selection rule are as described for the batch-size-7 localisation models. Thus, for $w_s = 15$, 31, and 61, the localisation models are tuned using a transfer-tuning random search that reuses the batch-7 results as an anchor, while allowing controlled variability in architecture, regularisation, and initialisation.

Hyperparameter tuning results

Table 5.18 summarises the final ANN hyperparameter choices for the scour localisation models. Across all batch sizes, both the front-mounted and rear-mounted models converge to a similar architecture with three hidden layers of size [128, 128, 64] and hyperbolic tangent activation. For the front-mounted model, the median λ values show a modest increase for batches 15 and 31 relative to batch 7, followed by a reduction for batch 61. The rear-mounted model exhibits a slightly smoother evolution of λ across batch sizes, returning to its batch-7 value at batch 61. Weight initialisation is predominantly Glorot/Xavier for the front-mounted case, and predominantly He-type for the rear-mounted case, with biases initialised mainly to zero in both models.

Table 5.17. Hyperparameter tuning results of scour localisation ANN models.

Sensor position	Shared settings	Batch-specific settings (batch size: value)
front	σ(·)=Hyperbolic tangent, Layer Sizes ≈ [128, 128, 64]	λ (median across outer folds) = {7: 1.1e-3, 15: 2.3e-3, 31: 2.4e-3, 61: 1.4e-3}
rear	σ(·)=Hyperbolic tangent, Layer Sizes ≈ [128, 128, 64]	λ (median across outer folds) = {7: 1.1e-3, 15: 2.0e-3, 31: 1.7e-3, 61: 1.1e-3}

5.8.3 Random Forest (RF) Classifier

5.8.3.1 Random Forest formulation and generalisation properties

Random forests are ensemble classifiers that combine many decision trees grown under controlled randomness. Breiman (2001) introduced RFs as collections of tree-structured input features as in Eq. (5.145):

$$h(x, \Theta_k);\ for\ k\ =\ 1, \dots, T_R \tag{5.145}$$

where each Θ_k is a random vector that governs how the k^{th} tree is built, and all trees vote for a class label at any input x. The randomness comes from two primary sources: drawing a bootstrap sample of the training data for each tree (i.e., bagging) and, at every node, selecting a random subset of input variables from which the best split is chosen. This combination of bagging and random feature selection produces trees that are individually reasonably accurate but only weakly correlated, which is crucial for the ensemble to improve over a single tree.

In Breiman's formulation, an RF classifier consists of tree predictors $h(x, \Theta_1), \dots, h(x, \Theta^T)$. Each tree maps a feature vector x to a class label. For a given input x, each tree casts one vote for a class, and the forest prediction is the class receiving the highest number of votes. As the number of trees T_R increases, the generalisation error of the forest converges to a limit rather than diverging, so adding more trees does not lead to overfitting (Breiman, 2001). This convergence is analysed using the margin function, along with two key ensemble properties: strength and correlation. For an input–output pair (x,y), where y is the true class label, the (finite-forest) margin is defined as the difference between the proportion of trees voting for the true class and the largest proportion voting for any other class. The margin $m_T(x,y)$ can be written as in Eq. (5.146):

$$m_T(x,y) = \frac{1}{T}\ \sum_{k=1}^{T_R} I(h(x,\Theta_k) = y)\ \ \max_{\text{classes } c \neq y}\ \frac{1}{T}\ \sum_{k=1}^{T_R} I(h(x,\Theta_k) = c) \tag{5.146}$$

where $I(\cdot)$ is the indicator function, equal to 1 if its argument is true and 0 otherwise; y is the true class, and c is the generic class label. A large positive margin means that the forest strongly favours the correct

class, whereas a negative margin indicates that the majority vote is incorrect. In the limit as T becomes large, the proportions of votes converge to their population counterparts, and the margin $m(x,y)$ converges to the value in Eq. (5.147):

$$m(x,y) = P(h(X,\Theta) = y) \; - \max_{\text{classes } c \neq y} P(h(X,\Theta) = c) \tag{5.147}$$

with X and Θ denoting the random input vector and the random tree-building vector, respectively. The generalisation error of the forest PE is then the probability that this limiting margin is negative, as in Eq. (5.148):

$$PE = P_{XY}(m(X,Y) < 0) \tag{5.148}$$

The strength of the forest is defined as the expected margin, i.e., the average value of $m(X, Y)$ over the joint distribution of inputs X and labels Y. It measures, on average, how confidently the ensemble favours the correct class. In the binary "scoured/healthy" setting, the correlation term ρ can be written as in Eq. (5.149):

$$\rho = corr_{X,Y}(I(h(X,\Theta) = Y), I(h(X,\Theta') = Y)) \tag{5.149}$$

where Θ and Θ' are two tree-building random vectors with the same distribution, generated from independent random draws. Each represents the randomness used to build a different tree in the forest. This correlation term ρ quantifies how similarly two randomly chosen trees in the forest tend to vote for the correct class across the data set. Breiman (2001) derives an upper bound on the generalisation error that decreases when the trees are individually strong and only weakly correlated. Random feature selection at each node is specifically designed to reduce correlation between trees, while still allowing each tree to exploit informative variables and maintain sufficient strength. This mechanism makes random forests particularly effective in high-dimensional problems and in settings with many weak or noisy predictors.

An additional advantage of RF algorithms is that they provide built-in, out-of-bag (OOB) estimates. Because each tree is trained on a bootstrap sample, approximately one-third of the training observations are left out of that tree's bootstrap set. For a training set $\{(X_i, Y_i)\}_{i=1}^{n}$, if S_i denotes the set of tree indices for which (X_i, Y_i) is out-of-bag, and $|S_i|$ denotes the number of such trees, the OOB prediction for observation i is then the majority vote over these trees, as in Eq. (5.150):

$$\hat{Y}_i^{\text{OOB}} = \arg\max_c \; \frac{1}{|S_i|} \sum_{k \in S_i} I(h(X_i, \Theta_k) = c) \tag{5.150}$$

The OOB estimate of the generalisation error is formulated as in Eq. (5.151):

$$PE_{\mathrm{OOB}} = \frac{1}{n}\sum_{i=1}^{n_t} I(\hat{Y}_i^{\mathrm{OOB}} \neq Y_i) \tag{5.151}$$

where Y_i is the true class label of the i^{th} training observation, n_t denotes the total number of batch samples in the training set, namely the number of rows in the stacked feature matrix X and label matrix Y defined in Eqs. (5.93)–(5.94). The generalisation error provides an internal estimate of the forest error without requiring a separate hold-out data set (Breiman, 2001). The same OOB framework is used in this thesis to construct measures of variable importance by monitoring how PE_{OOB} changes when the values of a given predictor are randomly permuted in the OOB samples.

5.8.3.2 Model structure and majority-vote decision rule

In this thesis, the RF is implemented by the *TreeBagger* function (Matlab, 2025b), configured as a binary classifier for scour detection and localisation. Details are provided in sections 5.8.3.1 and 5.8.3.2. For both scour detection and scour localisation, the RF classifier acts on the sliding-window feature vectors $X_{batch}(z)$ and batch labels $Y_{batch}(z)$ defined in Section 5.7. Given a batch feature vector $x = X_{batch}(z)$, the k^{th} tree of RF produces a class prediction as in Eq. (5.152):

$$\hat{y}_k(x) = h(x, \Theta_k) \in \{\text{healthy, scoured}\} \tag{5.152}$$

The empirical fraction of trees voting for the “scoured” class can be expressed as in Eq. (5.153):

$$\hat{p}_{\text{scoured}}(x) = \frac{1}{T}\sum_{k=1}^{T_R} I(h(x, \Theta_k) = \text{scoured}) \tag{5.153}$$

This fraction is interpreted as the RF estimate of the posterior probability that the batch belongs to the scoured class. The final RF decision for the batch is obtained by a fixed 0.5 threshold on this quantity, which is equivalent to a majority vote in the binary case is in Eq. (5.154):

$$\hat{Y}_{\text{batch}}(z) = \begin{cases} \text{scoured}, & \text{if } \hat{p}_{\text{scoured}}(X_{\text{batch}}(z)) \geq 0.5 \\ \text{healthy}, & \text{otherwise} \end{cases} \tag{5.154}$$

Thus, each tree contributes one vote for "healthy" or “scoured, and the RF prediction corresponds to the class receiving at least half of the votes. The same decision rule is applied to all batch samples, both at the bridge-level (scour detection) and at the support-level (scour localisation). This model structure is realised using the *TreeBagger* function (Matlab, 2025b) in classification mode.

5.8.3.3 Tree structure and training objective

Each tree in the RF is a binary decision tree grown on a bootstrap sample of the training batches. Internal nodes correspond to split tests on single input features, and leaves correspond to terminal regions

in feature space where the tree outputs a class prediction. An illustrative structure of the RF model structure is shown in Figure 5.72.

At a given internal node, a random subset of input features of size m_{try} is first drawn from the complete set of input features. For each candidate feature x_k in this subset, the algorithm considers splits of the form as in Eq. (5.155):

$$x_k \leq \tau \tag{5.155}$$

for suitable thresholds τ, which partition the node's samples into left and right children. Among all such candidates, the chosen split is the one that maximises the reduction in impurity. The impurity at a node is measured by the Gini index. For a node ℓ containing N_ℓ training samples. The Gini impurity is as in Eq. (5.156):

$$G(\ell) = 1 - \hat{p}_{\text{healthy}}(\ell)^2 - \hat{p}_{\text{scoured}}(\ell)^2 \tag{5.156}$$

where $\hat{p}_{\text{healthy}}(\ell)$ and $\hat{p}_{\text{scoured}}(\ell)$ denote the class proportions of the healthy and scoured classes.

If a candidate split produces left and right child nodes L and R with N_L and N_R samples, respectively, the corresponding impurity after the split is the weighted sum of the child-node impurities, as in Eq. (5.157):

$$G_{\text{split}} = \frac{N_L}{N_L+N_R}\, G(L) + \frac{N_R}{N_L+N_R}\, G(R) \tag{5.157}$$

where $G(L)$ and $G(R)$ are the Gini impurities of the left and right child. The training objective at each node is to choose the feature and threshold τ that minimises G_{split}, i.e., to maximise the impurity reduction relative to the parent node. This top–down splitting continues until stopping criteria are met, such as a minimum leaf size or a limit on the number of splits, which are controlled by the *TreeBagger* (Matlab, 2025b) options.

At a leaf node ℓ, the tree stores the class counts $N_{healthy}$ and $N_{scoured}$ of the training batches that fall into that leaf. The predicted class for any new batch routed to leaf ℓ is the majority class at that leaf, and the empirical leaf proportions in Eq. (5.158) define the per-tree class probabilities used in the ensemble voting rule in Section 5.8.3.1.

$$\hat{p}_{\text{healthy}}(\ell) = \frac{N_{\text{healthy}}(\ell)}{N_{\text{healthy}}(\ell)+N_{\text{scoured}}(\ell)},\ \hat{p}_{\text{scoured}}(\ell) = \frac{N_{\text{scoured}}(\ell)}{N_{\text{healthy}}(\ell)+N_{\text{scoured}}(\ell)} \tag{5.158}$$

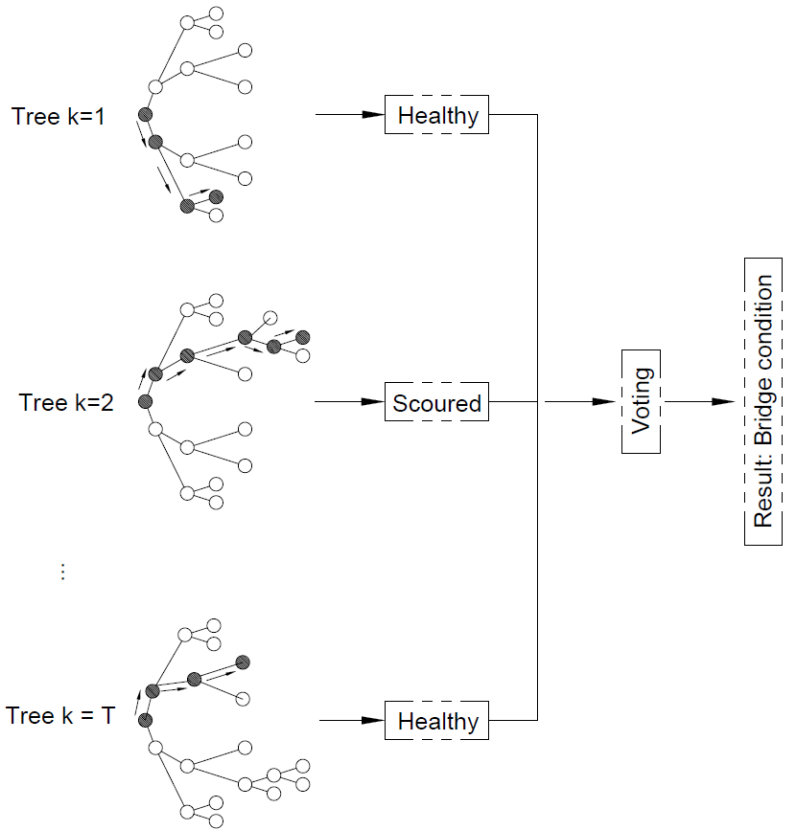


Figure 5.72. Random Forest model structure.

5.8.3.1 Hyperparameters and tuning — scour detection models

Nested CV

For all batch sizes of the scour-detection models, random forest hyperparameters are tuned by nested CV. X and Y in Eqs. (5.93) -(5.95) serve as inputs to the RF tuning procedure, and SCR stores the corresponding scour percentages. No scaling is applied to the input features, since decision trees are invariant under monotone transformations of individual features. Hyperparameters are tuned by nested CV:

- Outer loop (external CV):

The batch-level dataset (X, Y) is split into $k_{ext}=5$ outer folds using stratified partitioning, so that the "healthy"/"scoured" ratio is approximately preserved in each outer fold. For each fold, this yields an outer training set and a disjoint outer test set.

- Inner loop (internal CV):

For each outer-training set, a second partition with $k_{int}=3$ folds is used to define internal training/validation splits for hyperparameter tuning. Only the outer-training data are used in this inner loop; the outer-test fold remains untouched for external performance estimation.

Random search hyperparameter space

The following parameters are tuned in the RF model *Treebagger* (Matlab, 2025b):

- number of decision trees in the forest, i.e., T: larger T gives more stable predictions but increases computation time.
- minimum leaf size (Min_{Leaf}): minimum number of training samples allowed in a terminal node
- maximum number of splits per tree (Max_{Splits}): controls the depth and overall complexity of the tree.
- number of input features sampled at each split (m_{try}): controls the amount of randomisation and decorrelation between trees.
- in-bag fraction ($IB_{Fraction}$): fraction of training samples drawn with replacement to build each tree
- impurity measure ($s_{Criterion}$): splitting criterion used to evaluate candidate splits; guiding how nodes are partitioned.
- leaf-merging option (M_{Leaves}): controls whether neighbouring leaves are merged after the tree is grown to simplify the tree.
- predictor-selection strategy ($Pr_{Selection}$): controls how splits are chosen given the m_{try} candidate input feature.

During the inner random search, some of these are fixed, while others are tuned. The tuning uses a budget of $S = 40$ random trials per outer fold. For each trial $s=1,...,S$, a candidate hyperparameter vector $\theta^{(s)}$ is drawn by:

- sampling the minimum leaf size from the discrete set {2,4,8,16,32,64}
- sampling Max_{Splits} uniformly from an interval $[n_{feat}, M_{max}]$, M_{max} is an upper cap depending on n_{feat} and N_b
- sampling m_{try} uniformly from an interval around $\sqrt{n_{feat}}$. If m_{try} is large, close to n_{feat}, trees are strong but similar, which increases correlation and reduces the benefit of averaging. If m_{try} is small, trees become weak because splits are based on insufficient information. Choosing m_{try} in a neighbourhood of $\sqrt{n_{feat}}$ provides a compromise: each node sees enough input features to form informative splits, while random feature selection keeps trees less correlated, which is favourable for generalisation (Botta et al., 2014).

The remaining hyperparameters are kept fixed during tuning:

- The number of trees in the inner loop is set to a moderate value of 150, which is large enough to stabilise inner-fold performance estimates while keeping the computational cost of the nested CV manageable; the final models in each outer fold are later retrained with more trees.
- The in-bag fraction is fixed to $IB_{Fraction}$ = 1.0, corresponding to classical bootstrap sampling as in Breiman's (2001) formula, which is for each tree, n samples are drawn with replacement from n training observations.
- The split criterion is chosen as the Gini impurity (Eq. (5.152), a widely used and computationally cheaper index.
- Leaves are not merged, in line with *TreeBagger* function (Matlab, 2025b) recommendations, to avoid post-hoc leaf merging that would interfere with the explicit control of tree complexity via Max_{Splits} and Min_{Leaf}.
- Predictor Selection $Pr_{Selection}$ is set to all splits so that, at each node, the best split is chosen among the m_{try} candidate predictors without introducing additional randomness in the split search. "all splits" option performs a complete search over all split points for each candidate input feature.

Thus, the effective search space focuses on the depth and width of the trees via Max_{Splits} and Min_{Leaf}, and on the degree of randomisation at each split via $m_{try.}$

Inner-loop random search and selection rule

For each outer-training set and for each random trial, denoted by the candidate hyperparameter vector $\theta^{(s)}$, the inner 3-fold CV operates on (X_{train}, Y_{train}) of that outer fold:

- The inner partition defines three internal training/validation splits.
- For each inner fold, an RF model is trained on the inner-training subset using the candidate $\theta^{(s)}$ with 150 trees and the fixed options described above.
- Predictions are obtained on the corresponding inner-validation subset. For that fold, the validation accuracy and the F1-score of the scoured class are computed. In addition, the out-of-bag error of the fitted model is recorded.

For each candidate $\theta^{(s)}$, the inner-loop performance is summarised by:

- the mean validation *F1*-score of the scoured class, denoted with $\bar{F1}_{\text{val}}(\theta^{(s)})$

- the mean validation accuracy, denoted $\bar{acc}_{val}(\theta^{(s)})$
- the mean out-of-bag error, denoted $\bar{PE}_{\mathrm{OOB}}(\theta^{(s)})$

all averaged over the three inner folds. The random search then ranks the S candidates primarily by decreasing $\bar{F1}_{\mathrm{val}}$; ties are broken by higher $\bar{acc}_{val}$ and, if needed, by lower $\bar{PE}_{\mathrm{OOB}}$. The selected hyperparameter vector, denoted $\hat{\theta}$ for that outer fold is the one with the best rank among the 40 trials.

Final training and global recommended hyperparameters

For each outer fold, the best configuration $\hat{\theta}$ from the inner random search is used to train a final RF model on the complete outer-training set, with a larger number of trees, i.e., $T_R = 500$, and with out-of-bag prediction and predictor importance enabled. This final model results in:

- predictions on the outer-training set, used to compute training accuracy
- predictions on the outer-test set, used to compute the external-CV accuracy and F1-score

The true and predicted labels from all outer folds are concatenated to form pooled estimates of performance. In parallel, the best hyperparameter vectors $\hat{\theta}$ from the five outer folds are stored.

Hyperparameter tuning results

Table 5.18 summarises the final RF hyperparameter choices for the scour-detection models across all batch sizes. For both sensor positions, the minimum leaf size is $Min_{Leaf} = 2$ for all batch sizes, allowing relatively deep trees. The tuned maximum number of splits, i.e., Max_{Splits}, increases with batch size up to $w_s = 31$ for both front- and rear-mounted sensors, and then drops for $w_s = 61$, indicating that the random search favoured more complex trees for intermediate batch lengths. The tuned m_{try} values remain moderate relative to the total number of input features and are larger for intermediate batch sizes.

Table 5.18. Hyperparameter tuning results of scour detection RF models.

Sensor position	**Shared settings**	**Batch-specific settings (batch size: value)**
front	T_R = 500; $IB_{Fraction}$ = 1; $s_{Criterion}$ = Gini Diversity Index. M_{Leaves} = off; $Pr_{Selection}$ = All splits	Min_{Leaf} = 2 (all); m_{try} = {7: 25, 15: 46, 31: 74, 61: 23} Max_{Splits} = {7: 2576, 15: 5855, 31: 15146, 61: 1361};
rear	T_R = 500; $IB_{Fraction}$ = 1; $s_{Criterion}$ = Gini Diversity Index. M_{Leaves} = off; $Pr_{Selection}$ = All splits	Min_{Leaf} = 2 (all); m_{try} = {7: 33, 15: 34, 31: 57, 61: 31} Max_{Splits} = {7: 2376, 15: 4833, 31: 13101, 61: 1200};

5.8.3.2 Hyperparameters and tuning - scour localisation models

The random forest hyperparameters for the scour localisation models are tuned using the same nested CV protocol as in the scour-detection case provided in Section 5.8.3.1. The batch-level feature

matrix X and label vector Y entering Eq. (5.96) and (5.97) are now built from the pier-based pipeline, after SMOTE balancing, and are accompanied by two additional batch-level descriptors: the scour percentage vector *SCR* and the support-name vector *SNM*. These two vectors are not used as inputs to the RF classifier. However, they are carried through all outer folds so that localisation metrics and support-level summaries can be computed later.

Nested CV

The scour localisation RFs use the same 5×3 nested CV scheme as the scour-detection models. The batch-level dataset (X, Y) is partitioned into stratified outer folds and, within each outer-training set, into inner folds for hyperparameter selection. The definitions of the performance metrics, such as *F1*-score of the scoured class, accuracy and out-of-bag error, and the ranking rule for candidate hyperparameters, are the same as in the scour-detection tuning section.

Random search for batch length $w_s = 7$

For the smallest batch size (w_s = 7 crossings), a full random search is performed for the localisation models, separately for the front and rear-mounted sensors.

For each outer fold:

- The outer-training subset (X_{tr}, Y_{tr}) is repartitioned into three inner folds.
- A random search with 40 trials explores the RF hyperparameter space. In each trial, a candidate configuration is drawn, trained, and evaluated using an inner 3-fold CV.

The hyperparameter set and ranges mirror those used for scour detection:

- The tuned quantities are: minimum leaf size, i.e., Min_{Leaf}, maximum number of splits per tree, namely Max_{Splits}, and number of predictors sampled at each split, i.e. m_{try}.
- The remaining *TreeBagger* (Matlab, 2025b) function options are fixed during tuning: number of trees in the inner loop set to 150, in-bag fraction equal to 1.0 (classical bootstrap sampling), Gini impurity as splitting criterion, leaf merging disabled, predictor selection set to “all splits”.

Within each trial and each inner fold, an RF model is fitted on the inner training subset and evaluated on the corresponding inner validation subset. The inner-loop performance for a given trial is summarised by:

- the mean *F1*-score of the scoured class, averaged over the three inner folds,
- the mean validation accuracy,

- the mean out-of-bag error.

The 40 candidates are then ranked using the same rule as for the detection models: primary ranking by decreasing mean *F1*-score of the scoured class, with ties broken by higher mean accuracy and, if needed, by lower out-of-bag error. The top-ranked configuration for that outer fold is stored as the best hyperparameter vector for $w_s = 7$.

Transfer tuning for longer batch lengths (w_s = 15, 31, 61)

For the larger batch sizes (w_s = 15, 31, and 61 crossings), the localisation models do not undergo a fresh, full random search from scratch. Instead, a transfer-tuning strategy is adopted to control

computational cost and to keep the RF configuration consistent across batch lengths:

- The nested 5×3 CV structure is retained, so that external performance can still be estimated for each w_s.
- The RF settings selected at $w_s = 7$ for a given sensor position are treated as the anchor configuration and are transferred to the localisation models at w_s = 15, 31 and 61.
- For these larger batch sizes, the outer folds therefore use a fixed RF configuration (per sensor orientation) inherited from the $w_s = 7$ tuning, with 500 trees and the same fixed options (in-bag fraction, Gini criterion, "all splits" predictor selection and no leaf merging).

Hyperparameter tuning results for scour localisation models

Table 5.19 summarises the final random forest hyperparameters used by the scour localisation models for each batch length and sensor position. For both front-mounted and rear-mounted sensors, the minimum leaf size is $Min_{Leaf} = 2$ across all batch sizes, allowing relatively deep trees and leaving overall tree complexity to be controlled mainly by Max_{Splits} and m_{try}. For the front-mounted case, the best Max_{Splits} values lie in a band between 1475 and 1823 for all batch lengths w_s, and the tuned m_{try} values remain between 24 and 32. This indicates that, once Min_{Leaf} is fixed to a small value, the nested CV tends to favour a relatively stable tree depth and a moderate number of candidate predictors per split, with only mild adjustments as w_s increases.

In contrast, for the rear-mounted case, both Max_{Splits} and m_{try} increase substantially with batch length. The selected Max_{Splits} grows from 1473 at $w_s = 7$ to 8803 at $w_s = 61$, and m_{try} rises from 24 to 54 over the same range. Thus, for larger windows, the localisation models based on the rear-mounted sensor use

deeper trees and consider more predictors at each split, suggesting that the random search favours more complex RF configurations in this orientation when more crossings are aggregated in each batch.

Table 5.19. Hyperparameter tuning results of scour localisation RF models.

Sensor position	Shared settings	Batch-specific settings (batch size: value)
front	T_R = 500, $IB_{Fraction}$ = 1, $s_{Criterion}$ = Gini diversity index, M_{Leaves} = off, $Pr_{Selection}$ = All splits	Min_{Leaf} = 2 (all), m_{try} = {7: 24, 15: 31, 31: 31, 61: 32} Max_{Splits} = {7: 1475, 15: 1630, 31: 1630, 61: 1823}
rear	T_R = 500, $IB_{Fraction}$ = 1, $s_{Criterion}$ = Gini diversity index, M_{Leaves} = off, $Pr_{Selection}$ = All splits	Min_{Leaf} = 2 (all), m_{try} = {7: 24, 15: 32, 31: 40, 61: 54} Max_{Splits} = {7: 1473, 15: 2541, 31: 4673, 61: 8803}

5.8.4 Support Vector Machine classifier

5.8.4.1 Decision function and geometric margin

SVMs are the third margin-based ML classifiers used in this thesis that learn a separating hyperplane between two classes, here "damaged" and "healthy" bridge conditions. They search for a linear predictor that not only separates the training data but does so with a large margin, that is, with all points lying as far as possible from the decision boundary in feature space (Shalev-Shwartz & Ben-David, 2014). If Eq. (5.159) shows the training set:

$$\{(x_1, y_1), \ldots, (x_m, y_m)\} \tag{5.159}$$

with feature vectors $x_i \in \mathrm{R}^{n_{\text{feat}}}$ and labels $y_i \in \{-1,+1\}$, where we map the damaged class to $y_i = +1$ and the healthy class to $y_i = -1$. A linear SVM learns a weight vector $w \in R^{nfeat}$ and bias $b \in R$ that define the decision function in Eq. (5.160):

$$f(x) = \langle w, x \rangle + b \tag{5.160}$$

For a given pair (w,b), the functional margin of a sample (x_i, y_i) is as in Eq. (5,161):

$$\gamma_i = y_i f(x_i) = y_i(\langle w, x_i \rangle + b) \tag{5.161}$$

and the geometric margin of the hyperplane on the training set is the minimum Euclidean distance from any training point to the hyperplane, which can be expressed as in Eq. (5.162)

$$\gamma_{geom}(w, b) = \min_{i=1,2,\ldots m} \frac{y_i(\langle w, x_i \rangle + b)}{\|w\|} \tag{5.162}$$

A larger value of this geometric margin means that all damaged and healthy examples are not only correctly classified but also lie further away from the separating hyperplane, as sketched in Figure 5.73. This schematic two-dimensional figure is included only to illustrate the notion of a separating hyperplane and

its margins. As a result of hyperparameter tuning, the final SVM model for scour detection and localisation was an RBF-kernel SVM, which operates in a higher-dimensional feature space and induces a non-linear decision boundary there.

In the ideal Hard-SVM formulation, the data are assumed to be linearly separable, and a separating hyperplane with the largest margin is sought. The optimisation problem is as in Eq. (5.163):

$$\min_{w,b} \frac{1}{2} \| w \|^2 \text{ subject to: } y_i(\langle w, x_i \rangle + b) \geq 1 \tag{5.163}$$

for all *i=1, 2, ..., m_t*, where m_t is the number of training samples, and $\|w\|$ denotes the Euclidean norm, i.e., the length of the weight vector w as in Eq. (5.164):

$$\| w \| = w_1^2 + w_2^2 + \cdots + w_{n_{feat}}^2 \tag{5.164}$$

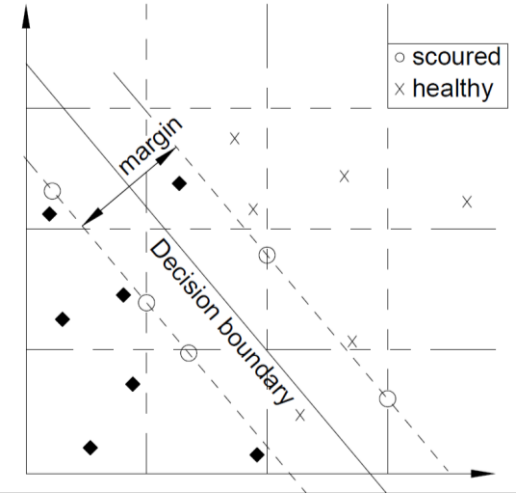


Figure 5.73. Geometric interpretation of a linear SVM classifier.

Among all hyperplanes that classify the training data correctly, Hard-SVM selects the one with minimal norm $\|w\|$, which is equivalent to maximising the margin. However, real data are rarely perfectly separable, hence the Soft-SVM formulation is implemented. This allows margin violations but penalises them via the hinge loss. In the regularised risk-minimisation form used by Shalev-Shwartz and Ben-David (2014), the soft-margin SVM solves the objective function Eq. (5.165):

$$\min_{w,b} [\frac{\lambda}{2} \| w \|^2 \quad + \quad \frac{1}{m} \sum_{i=1}^{m_t} \max \{0, 1 - y_i(\langle w, x_i \rangle + b)\}] \tag{5.165}$$

where $\lambda > 0$ controls the trade-off between margin maximisation (*small $\|w\|$*, as seen in Eq. (5.158)) and hinge-loss penalties on misclassified or low-margin points. The hinge loss term of one training point *(x_i, y_i)* is as in Eq. (5.166):

$$\ell_{hinge}(x_i, y_i; w, b) = max\{0,1 - yi(\langle w, xi \rangle + b)\} \tag{5.166}$$

is zero for points correctly classified with a margin of at least 1 and increases linearly when points are on the wrong side of the margin or misclassified.

A key property of the SVM solution is that the optimal weight vector lies in the span of the training examples as in Eq. (5.167):

$$w = \sum_{i=1}^{m_t} \alpha_i \, y_i x_i \tag{5.167}$$

for coefficients $\alpha_i \geq 0$. Only a subset of the training points has $\alpha_i > 0$; these are the support vectors, which lie on or inside the margin and fully determine the decision boundary. Substituting this representation into the decision function in Eq. (5.160) results in Eq. (5.168):

$$f(x) = \langle w, x \rangle + b = \sum_{i=1}^{m_t} \alpha_i \, y_i \langle x_i, x \rangle + b \tag{5.168}$$

where $y_i\langle x_i, x\rangle$ can be interpreted as the similarity between the new point x (e.g. a test-set sample) and the training point x_i." The SVM is therefore a margin-based classifier: the sign and magnitude of $f(x)$ encode both the predicted class and the distance to the separating hyperplane. In this thesis, a *hard* decision rule based on the sign of $f(x)$ is used so that:

- predict "damaged" if $f(x) \geq 0$
- predict "healthy" if $f(x) < 0$

Both the scour detection (bridge-based) and scour localisation (support-based) models use MATLAB's *fitcsvm* function (Matlab, 2025c) to implement soft-margin SVM classifiers. The models use hinge-loss regularisation as in Section 5.8.4.1 and allow several kernel functions (linear, Gaussian RBF, polynomial). Hyperparameters are selected by nested CV with random search. The procedure is identical for detection and localisation; the scour-detection case is described first.

5.8.4.2 Hyperparameter tuning for scour detection SVM models

For the scour-detection models, hyperparameters are selected by nested CV with a random search over candidate SVM configurations. The outer loop uses five folds. In each outer fold, the training portion is split into three inner folds. For every outer fold:

i. *Outer split.* The data are partitioned into five outer folds. A stratified sampling so that each fold preserves approximately the same proportion of 'scoured' and 'healthy' samples as the complete data set.

ii. Random search in the inner loop with:

- A budget of 60 random configuration trials is evaluated.
- For each trial, the following hyperparameters are sampled:
  - Kernel type $k \in$ {"linear", "rbf", "polynomial"}.
  - Box constraint $C > 0$ (MATLAB BoxConstraint), sampled on a log scale in the range of $[10^{-3}, 10^{3}]$. The regularisation parameter λ in Eq. (5.161) controls the trade-off between the margin term $(\lambda/2)\|w\|^2$ and the average hinge loss. A larger λ puts more weight on keeping $\|w\|$ small, which favours wider margins and tolerates more margin violations. A smaller λ puts more weight on reducing hinge loss, allowing larger $\|w\|$ and narrower margins to fit the training labels better. In the *fitcsvm* function (Matlab, 2025c) implementation, the same trade-off is controlled by the Box Constraint parameter $C > 0$ in the primal objective in Eq. (5. 169):

$$\min_{w,b,\xi} [\frac{1}{2} \| w \|^2 + C \sum_{i=1}^{m_t} \xi_i] \tag{5.169}$$

    subject *to* $y_i((w,x_i)+b) \geq 1-\xi_i$ and $\xi_i \geq 0$ measures how much the margin condition is violated:

  - If $\xi_i = 0$: the point is correctly classified and lies outside or on the margin, i.e., it satisfies the condition $y_i(\langle w,x_i\rangle+b) \geq 1$.
  - $0 < \xi_i \leq 1$: the point is correctly classified; it lies inside the margin strip.
  - $\xi_i > 1$: the point is misclassified, namely on the wrong side of the decision boundary.

    C functions similarly to λ, because it scales the penalty on margin violations. Up to a rescaling of the objective, large values of C serves similar to a small effective λ: namely a weak regularisation, narrow margin, and closer fit to the training data; whereas small values of C serve like a large effective λ, i.e., strong regularisation, wider margin, more tolerance for training errors (Shalev-Shwartz & Ben-David, 2014; MathWorks, 2025c).

  - Kernel scale for the RBF kernel is chosen as follows: The feature vectors are z-scored within each training fold, so most coordinates have variance close to 1. For RBF SVMs on such standardised data, typical kernel-scale values are around 1, with smaller values yielding very narrow kernels and larger values yielding smoother, broader kernels. Exploring the interval [0.01, 100] on a base-10 logarithmic scale therefore covers narrow to broad kernels without wasting trials on clearly unreasonable scales. With probability 0.25, Kernel Scale is set to 'auto', so MATLAB's built-in heuristic is included in the search; if this heuristic is already close

to optimal for the data, the random search can recover it quickly. In the remaining 75% of trials, kernel scale is drawn as a numeric value on a $\log_{10}$ scale between 0.01 and 100. Overall, this mixed strategy is a compromise between coverage and computational budget.

- Polynomial order of kernel *q* is sampled from the set {2,3,4}.

- For each sampled configuration, a 3-fold inner CV is run. In each inner split, the features are standardised using z-scores, using only the inner training fold. The SVM is then trained with the *fitcsvm* function (Matlab, 2025c) using:

  a) the sampled *C*,

  b) the sampled kernel scale (when applicable),

  c) the sampled polynomial order (when applicable),

  d) the chosen kernel type ('linear', 'gaussian' for RBF, or 'polynomial')

A kernel is a symmetric function *K(x,x')* that can be written as an inner product in feature space (Shwartz & Ben-David, 2014). Given an embedding ψ from the original input space *X* to a Hilbert space, the kernel is defined by Eq. (5.170):

$$K(x, x') = \langle \psi(x), \psi(x') \rangle \tag{5.170}$$

This way, the SVM learns a linear separator in the feature space, while the decision boundary can be non-linear in the original feature coordinates.

- *Linear kernel:* The simplest case is the identity embedding $\psi(x)=x$. This yields the linear kernel in Eq. (5.171):

$$K_{\text{lin}}(x, x') = \langle x, x' \rangle \tag{5.171}$$

which corresponds to the standard linear SVM decision function introduced in Eq. (5.166).

- *Polynomial kernel:* Shalev-Shwartz and Ben-David (2014) define the degree-*k* polynomial kernel as in Eq. (5.172):

$$K_{\text{poly}}(x, x') = (1 + \langle x, x' \rangle)^q \tag{5.172}$$

This kernel is equivalent to mapping each input vector $x = (x_1, \ldots, x_{nfeat})$ to a higher-dimensional feature vector whose components are all products of the form $x_1^{a1}, x_2^{a2}, \ldots, x_{nfeat}^{anfeat}$, where the exponents $a_1, \ldots, a_{nfeat}$ are non-negative integers smaller than *q*. The SVM then learns a linear separator in this enlarged polynomial feature space. The random search selects the polynomial order $q \in \{2, 3, 4\}$ as one of the hyperparameters.

- *Gaussian Radial Basis Function (RBF) kernel:* A Gaussian kernel can be obtained by mapping the input to an infinite-dimensional feature space and taking inner products there. It can be a formula in Eq. (5.173):

$$K_{\text{RBF}}(x, x') = \mathrm{e}^{\frac{\|x-x'\|^2}{2\sigma}} \tag{5.173}$$

where $\sigma > 0$ is a scale parameter controlling which distances are considered "close". When $\gamma = 1/(2\sigma)$ is implemented, the formulation becomes the form in Eq. (5.174):

$$K_{\text{RBF}}(x, x') = \mathrm{e}^{(-\gamma \|x-x'\|^2)} \tag{5.174}$$

A larger value of γ makes the kernel more local: only similar feature vectors give kernel values close to 1, and the decision boundary can adapt to small-scale variations in the damaged / healthy labels. Smaller γ values yield smoother decision boundaries that change more slowly across the feature space.

e) For each inner validation fold, predictions are obtained and the *F1*-score for the "scoured" class. The performance of a trial is the mean of the *F1*-scores for the "scoured" class across the three inner folds.

f) the Sequential Minimal Optimisation (SMO) solver:

The soft-margin SVM used in this work is trained with the SMO solver. In dual form, the kernel SVM problem solved by SMO can be written as a quadratic programme in the dual coefficients $\alpha = (\alpha_1, \ldots, \alpha_m)^T$, as in Eq. (5.175):

$$\min_{\alpha \in \mathbb{R}^m} \Phi(\alpha) = \frac{1}{2} \alpha^\top Q \alpha - \mathbf{1}^\top \alpha \tag{5.175}$$

subject to Eq. (5.176):

$$0 \le \alpha_i \le C, i = 1, \ldots, m, \sum_{i=1}^{m_t} y_i \alpha_i = 0 \tag{5.176}$$

where $\mathbf{1}$ is the unit vector, $y_i \in \{-1, +1\}$ are the class labels: +1 for damaged, -1 for "healthy", $Q \in R^{m \times m}$ is the symmetric kernel matrix with entries Q_{ij} in Eq. (5.177):

$$Q_{ij} = y_i y_j K(x_i, x_j) \tag{5.177}$$

where $K(x_i, x_j)$ is the chosen kernel function. The primal weight vector w is recovered from the dual solution in Eq. (5.178):

$$w = \sum_{i=1}^{m_t} \alpha_i y_i x_i \tag{5.178}$$

Therefore, only points with $\alpha_i > 0$, namely the support vectors, contribute to the decision function.

A direct solution of this m-dimensional optimisation problem with a quadratic objective function would require storing and manipulating the whole kernel matrix Q. Instead, SMO uses a decomposition method: at each iteration, it modifies only a small "working set" of dual variables while keeping the others fixed. In the classical SMO scheme, the working set has size 2, $B = \{i, j\}$. If α^k denotes the current dual vector at iteration k, and split the indices into the working set $B=\{i,j\}$ and its complement $N=\{1,\ldots,m\} \setminus B$. Keeping $\alpha_t=\alpha_t^k$ fixed for all $t \in N$, the dual objective restricted to α_i,α_j becomes a two-variable quadratic function in Eq. (5.179):

$$\min_{\alpha_i,\alpha_j} \ \frac{1}{2}\begin{pmatrix}\alpha_i\\ \alpha_j\end{pmatrix}^{\top}\begin{pmatrix}Q_{ii} & Q_{ij}\\ Q_{ij} & Q_{jj}\end{pmatrix}\begin{pmatrix}\alpha_i\\ \alpha_j\end{pmatrix}+\begin{pmatrix}g_i\\ g_j\end{pmatrix}^{\top}\begin{pmatrix}\alpha_i\\ \alpha_j\end{pmatrix} \tag{5.179}$$

subject to Eq. (5.180):

$$0 \le \alpha_i, \alpha_j \le C, \ y_i\alpha_i + y_j\alpha_j = -\sum_{t\in N} y_t\, \alpha_t^k \tag{5.180}$$

The linear terms g_i, g_j collect the contributions of all fixed dual variables, as in Eq. (5.181):

$$g_i = -1 + \sum_{t\in N} Q_{it}\ \alpha_t^k, g_j = -1 + \sum_{t\in N} Q_{jt}\ \alpha_t^k \tag{5.181}$$

Since only α_i and α_j are free variables, and they must satisfy a single linear equality constraint, the feasible set reduces to a line segment in the (α_i, α_j) plane. The two-variable optimisation problem with a quadratic objective function therefore has a closed-form solution, obtained by projecting the unconstrained optimum onto this line segment and clipping it to the box constraints [0, C]. This is the core idea of SMO: each iteration solves a subproblem and updates just a pair of coefficients, while progressively decreasing the global dual objective $\Phi(\alpha)$.

An important component of SMO is the working-set selection rule, i.e. how the pair (i, j) is chosen. Fan, Chen and Lin (2005) describe SMO-type methods that select (i, j) using the Karush–Kuhn–Tucker (KKT) optimality conditions for the dual problem. The gradient of the dual objective can be expressed as in Eq. (5.182):

$$\nabla\Phi(\alpha) = Q\alpha - \mathbf{1} \tag{5.182}$$

Indices of the training samples, namely the dual variables, are classified into sets $I_{up}(\alpha)$ and $I_{low}(\alpha)$ depending on whether they can move upwards or downwards without violating the box constraints. The "violating pairs" (i, j) are those for which the KKT conditions are most strongly violated. The function *fitcsvm* (Matlab, 2025c) implements an SMO-type decomposition algorithm: it chooses two dual variables at each step associated with a violating pair, updates (α_i, α_j) by solving the corresponding two-variable optimisation problem with a quadratic objective function and repeats until all KKT violations fall below a tolerance.

ii. *Selection of the best configuration.* After the 60 trials, the configuration with the highest mean inner *F1*-scores for the "scoured" class is selected as the best SVM setting for that outer fold.

iii. *Retraining and outer-test evaluation.* The outer-training data are standardised again (using all outer-training samples), and a final SVM is retrained on this scaled outer-training set with the selected hyperparameters. This model is then applied to the scaled outer-test set to obtain:

e) the outer-test accuracy,

f) the outer-test *F1* score for the scoured class,

g) and the outer-training accuracy.

Across the five outer folds, these metrics are averaged to give the mean accuracy, the mean *F1* score for the scoured class, and the mean training accuracy.

Hyperparameter tuning results for scour detection models

Table 5.20 the hyperparameter tuning results of the random search for scour detection models. Across all batch sizes and both sensor positions, the tuned models consistently select Gaussian RBF kernels, confirming that non-linear decision boundaries are beneficial for scour detection. For the front-mounted case, the box constraints remain in a moderate range of 4.8–73, whereas for the rear-mounted case, they vary more widely, from about 4.8 to 509. Short and intermediate windows sometimes favour a tighter fit to the training labels: large *C*, especially rear-mounted w_s = 7. The longest window selects smaller *C* values, corresponding to more regularised, smoother decision boundaries.

Table 5.20. Hyperparameter tuning results of scour detection SVM models.

Sensor position	Shared settings	Batch-specific settings: (batch size: value)
RILA front	Kernel = RBF	Box Constraint = {7: 30.562, 15: 72.858, 31: 7.230, 61: 4.829} Kernel Scale = {7: 20.579, 15: 30.518, 31: 36.638, 61: 58.217}
RILA rear	Kernel = RBF	Box Constraint = {7: 508.955, 15: 11.892, 31: 24.727, 61: 4.762} Kernel Scale = {7: 24.236, 15: 17.382, 31: 37.293, 61: 65.070}

5.8.4.3 Hyperparameter tuning for scour localisation SVM models

For the pier-based scour localisation models, the SVM hyperparameters are tuned using the same nested CV structure as in the detection case, but with an additional transfer-tuning step for the larger window lengths. For the shortest window length w_s=7, a full random search is run in the inner loop with a budget of 60 trials per outer fold. In each trial, a candidate configuration is drawn from a broad sampler that chooses:

- a kernel type in {linear, RBF, polynomial},
- a box constraint $C > 0$ on a logarithmic scale over a wide range,
- for RBF kernels, either a numeric kernel scale sampled on a log scale or the 'auto' heuristic,
- for polynomial kernels, an integer degree in a small range (typically between 2 and 5).

For each sampled configuration, the SVM is trained and evaluated by a 3-fold inner CV. Predictions from all inner-validation folds are pooled, and several metrics summarise performance: F1-score for the scoured class, macro-averaged *F1*, ie.., mean of *F1* scores of scoured and healthy classes, balanced accuracy and MCC, together with a simple generalisation-gap measure, namely the difference between inner-train and inner-validation accuracy. Within each outer fold, the primary selection criterion is the *F1* score for the scoured class. Ties are broken by preferring larger macro-*F1*, larger balanced accuracy, larger MCC, and a smaller absolute generalisation gap. The resulting best SVM configuration for that fold is stored.

For the larger window lengths $w_s \in \{15,31,61\}$, a smaller transfer-tuning random search is used to reduce computational cost while re-using information from the $w_s=7$ results. First, the best-per-fold configurations selected at $w_s=7$ are called, and a prior configuration is constructed by taking the most frequent kernel type and the median values of C, kernel scale and polynomial degree across the five outer folds. For each trial in the transfer-tuning runs, a configuration is then sampled as follows:

- With probability 0.70, the sampler perturbs the prior:
    - The kernel is kept equal to the prior kernel with high probability (~85%), and only occasionally switched to another choice in {linear, RBF, polynomial}.
    - The box constraint C is drawn from a log-normal distribution centred on the prior C, in $\log_{10}$ space with a standard deviation of 0.35, so that it varies only within a narrow band around the $w_s=7$ solution.
    - For RBF kernels, the kernel scale is either kept in 'auto' mode (if the prior used 'auto') or drawn from a log-normal distribution around the prior numeric scale, with a lower probability of toggling between 'auto' and numeric values.
    - For polynomial kernels, the degree q is restricted to the neighbourhood $\{max(2, q_0-1), q_0, min(6, q_0+1)\}$ of the prior degree q_0. The lower bound 2 avoids the linear case, which is already covered by the separate linear kernel, and the upper bound 6 prevents

high-degree polynomials, which tend to be numerically unstable and overly flexible for the available sample sizes.

- With probability 0.30, the sampler performs a broad random search used at $w_s=7$, so that configurations far from the prior are also tried.

Hyperparameter tuning results for scour localisation models

Table 5.18Table 5.21 shows tuned hyperparameters of scour localisation models. Across all batch sizes and both sensor positions, the SVM localisation models consistently selected the RBF kernel. This indicates that a non-linear decision boundary is needed to separate "scoured" and "healthy" supports in the feature space. The tuned C values for the front-mounted sensor lie in the range of 30–65, with kernel scales between approximately 15 and 35. The changes in C and kernel scale across batch sizes (7, 15, 31, 61) are relatively modest, suggesting that the complexity required for scour localisation does not depend strongly on the window length.

For the rear-mounted case, the selected RBF SVMs also use similar kernel-scale values, but with generally larger box constraints, especially at smaller batch sizes. This suggests slightly less regularisation and more flexible decision boundaries for the rear-mounted models, consistent with the more challenging, heterogeneous feature patterns of the rear sensor. Overall, the hyperparameters in Table 5.21 show that both sensor configurations favour RBF SVMs of comparable smoothness, with the rear-mounted models allowing slightly sharper boundaries to capture the additional variability in the rear-mounted profiles.

Table 5.21. Hyperparameter tuning results of scour localisation SVM models.

Sensor position	Shared settings	Batch-specific settings: (batch size: value)
front	Kernel = RBF	Box Constraint = {7: 40.874, 15: 34.022, 31: 63.234, 61: 57.981} Kernel Scale = {7: 25.821, 15: 14.533, 31: 23.031, 61: 33.591}
rear	Kernel = RBF	Box Constraint = {7: 175.132, 15: 66.508, 31: 62.294, 61: 82.074} Kernel Scale = {7: 10.010, 15: 13.791, 31: 15.849, 61: 26.618}

5.9 Concluding remarks

This chapter has defined the complete ML framework used for scour detection and localisation in this thesis. It has formalised two complementary tasks: bridge-level detection, which classifies each crossing as "healthy" or "scoured", and support-level localisation, which assigns the same binary label to each support using a 5% scour threshold. A nested CV protocol with outer and inner folds has been specified to separate model selection from performance evaluation, together with pooled confusion-matrix metrics as the main readouts.

On the input side, the chapter has introduced a displacement-based feature engineering strategy built entirely from simulated RILA track-geometry profiles. Type-A features are wavelet-domain descriptors extracted from DWTs of the profiles, while Type-B features are Markov parameters derived from ARX models fitted either to full-bridge profiles (detection) or to scour-affected zones around each support (localisation). Two complementary feature-selection strategies have then been developed: an embedded relative-importance ranking with grid search, and a multi-objective NSGA-II wrapper that trades off *F1* score against feature-set size. Their outputs have been compared and used to define the feature sets passed to the classifiers.

On the modelling side, three classifiers—ANN, RF, and SVM with an RBF kernel—have been formulated in detail, including their decision rules, loss functions, regularisation, and hyperparameters. The chapter has described how features are normalised for ANN and SVM, left unscaled for RF, and how class imbalance is addressed using SMOTE within the nested CV scheme. A sliding-window, batch-based strategy has then been defined for both bridge- and support-level models, using windows of 7, 15, 31 and 61 crossings to combine aggregated statistics with order-preserving individual features and to stabilise decisions over time. The whole procedure has been specified for both RILA front-mounted and rear-mounted configurations, ensuring consistent treatment of sensor orientation.

Together, these elements have provided a complete and reproducible ML methodology that links the simulated RILA data from Chapter 4 to the classification results. The next chapter reports ML-based scour detection and localisation results, presenting bridge-level and support-level performance for single-pass and batched models, along with hyperparameter tuning outcomes, sensitivity analyses, and feature-importance patterns.

CHAPTER 6: MACHINE LEARNING MODEL RESULTS

This chapter presents the results of the ML models developed for scour detection and localisation. It reports the full outcomes for both bridge-level detection and multi-pier (support-level) localisation and compares single-pass decisions against batched decisions. First, the performance of the scour detection models across algorithm types and sensor positions is presented. Then the scour localisation model results, including model performance metrics. The results of the feature-importance and Markov-order sensitivity analyses are also provided. Finally, the discussion and final remarks address batch-size effects, algorithm differences, and model interpretability.

6.1 Scour Detection Model Results

6.1.1 Model Metrics

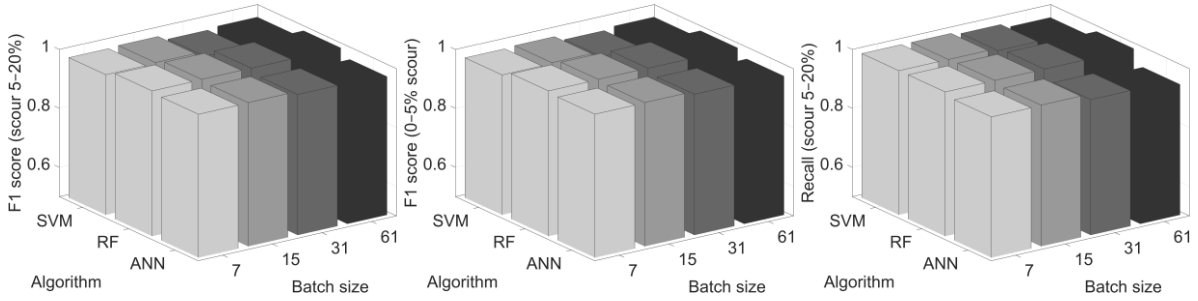
The predictive performance of the proposed framework is evaluated across temporal batch sizes of 7, 15, 31, and 61 crossings. For each configuration, three ML algorithms are assessed using standard classification metrics, including accuracy, precision, recall, and *F1* score. Figure 6.1 to Figure 6.3, and Table 6.1 show RILA front-mounted, while Figure 6.4 to Figure 6.6 and Table 6.2 illustrate rear-mounted model performances. Across both front and rear-mounted configurations, RF models achieved the highest and most stable performance, with strong accuracy, balanced *F1* scores across scour levels, and minimal generalisation gaps. ANN models also performed well, showing slightly lower recall and higher gap, yet maintaining high precision and stable metrics. SVM models exhibited greater variability and the largest generalisation gaps, indicating slightly less robustness compared to ANNs and RFs. Overall, RF offered the most consistent detection, while ANN and SVM delivered competitive but less stable results.

Overfitting has been assessed by comparing training and test performance across all models and by examining metrics across the external cross-validation folds. The generalisation gap in Figure 6.3(b) and 6.6(b) measures the difference between the test and training accuracies. For ANN and SVM, the gaps are less than 4% for the front-mounted and less than 3% for the rear-mounted configurations. RF shows the smallest gaps, less than 1%. Metrics are stable across batch sizes, indicating limited fold-to-fold fluctuation. For the best RF models, the F1 variance across folds stays below 1%, and the Out Of Bag (OOB)

error aligns with external cross-validation results. Taken together, these results indicate no overfitting and good generalisation.

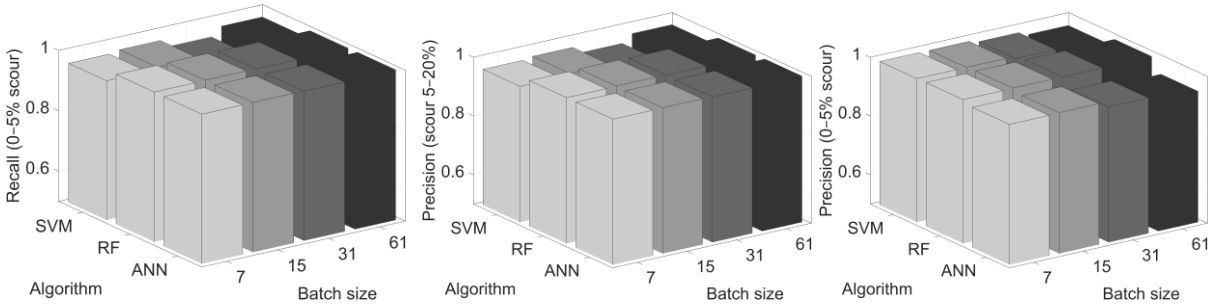
Table 6.1. Scour detection ML model metrics- RILA front-mounted.

Model	w_s	Accuracy	Scoured class (5-20 % scour)			Healthy class (0-5 % scour)			Macro-F1	Training accuracy	Generalisation gap
			Precision	Recall	F1 score	Precision	Recall	F1 score			
ANN	7	0.99	1.00	0.98	0.99	0.98	1.00	0.99	0.99	1.00	0.01
ANN	15	0.99	1.00	0.98	0.99	0.98	1.00	0.99	0.99	1.00	0.01
ANN	31	0.98	1.00	0.96	0.98	0.96	1.00	0.98	0.98	1.00	0.02
ANN	61	0.97	1.00	0.94	0.97	0.95	1.00	0.97	0.97	1.00	0.03
RF	7	0.99	1.00	0.99	0.99	0.99	1.00	0.99	0.99	1.00	0.01
RF	15	0.99	1.00	0.99	0.99	0.99	1.00	0.99	0.99	1.00	0.01
RF	31	0.99	1.00	0.99	0.99	0.99	1.00	0.99	0.99	1.00	0.01
RF	61	0.99	1.00	0.99	0.99	0.99	1.00	0.99	0.99	1.00	0.01
SVM	7	0.98	0.96	0.99	0.98	0.99	0.96	0.97	0.98	1.00	0.02
SVM	15	0.98	0.97	0.99	0.98	0.99	0.97	0.98	0.98	1.00	0.02
SVM	31	0.96	0.94	0.98	0.96	0.98	0.94	0.96	0.96	1.00	0.04
SVM	61	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	1.00	0.03



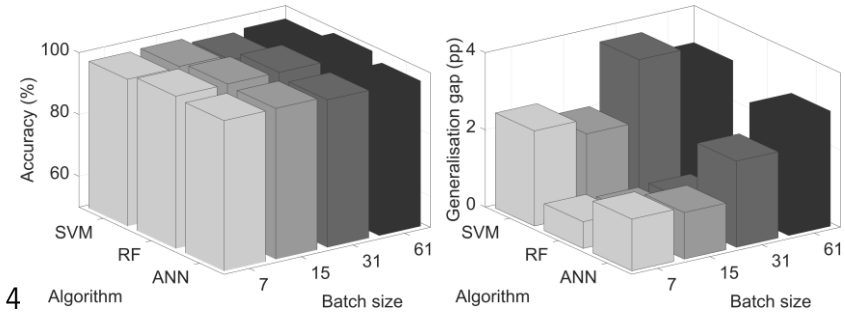
(a) (b) (c)

Figure 6.1. RILA front-mounted: (a) F1 score for 5–20%; (b) F1 score for 0-5%; (c) recall for 5–20%.



(a) (b) (c)

Figure 6.2. RILA front-mounted: (a) recall for 0-5; (b) precision for 5-20%; (c) precision for 0-5%.

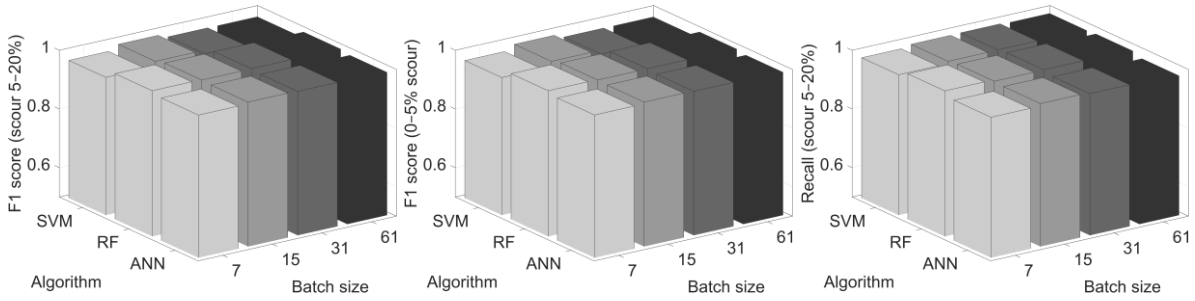


(a) (b)

Figure 6.3. RILA front-mounted: (a) accuracy; (b) generalisation gap.

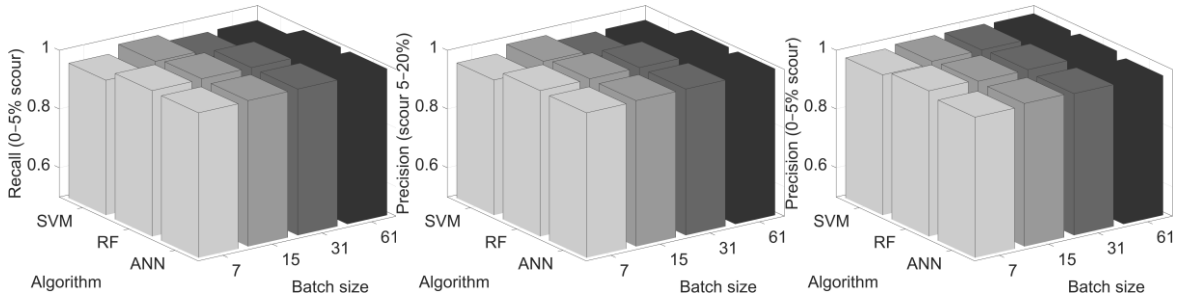
Table 6.2. Scour detection ML model metrics- RILA rear-mounted.

Model	w_s	Accuracy	Scoured class (5-20 % scour)			Healthy class (0-5 % scour)			Macro-F1	Training accuracy	Generalisation gap
			Precision	Recall	F1 score	Precision	Recall	F1 score			
ANN	7	0.99	0.99	0.98	0.99	0.98	0.99	0.99	0.99	1.00	0.01
ANN	15	0.99	1.00	0.99	0.99	0.99	1.00	0.99	0.99	1.00	0.01
ANN	31	0.99	1.00	0.98	0.99	0.98	1.00	0.99	0.99	1.00	0.01
ANN	61	0.99	1.00	0.97	0.99	0.98	1.00	0.99	0.99	1.00	0.01
RF	7	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
RF	15	0.99	1.00	0.99	0.99	0.99	1.00	0.99	0.99	1.00	0.01
RF	31	1.00	1.00	0.99	1.00	0.99	1.00	1.00	1.00	1.00	0.01
RF	61	0.99	1.00	0.99	0.99	0.99	1.00	0.99	0.99	1.00	0.01
SVM	7	0.97	0.96	0.98	0.97	0.98	0.96	0.97	0.97	1.00	0.03
SVM	15	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1.00	0.01
SVM	31	0.97	0.96	0.99	0.97	0.99	0.96	0.97	0.97	1.00	0.02
SVM	61	0.97	0.96	0.98	0.97	0.98	0.96	0.97	0.97	1.00	0.03



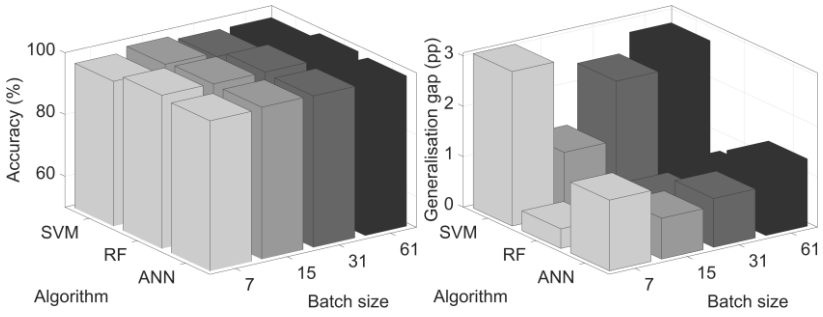
(a) (b) (c)

Figure 6.4. RILA rear-mounted: (a) F1 score for 5–20%; (b) F1 score for 0-5%; (c) recall for 5–20%.



(a) (b) (c)

Figure 6.5. RILA rear-mounted: (a) recall for 0-5%; (b) precision for 5-20%; (c) precision for 0-5%.



(a) (b)

Figure 6.6. RILA rear-mounted: (a) accuracy; (b) generalisation gap.

Across both front-mounted and rear-mounted configurations, all three algorithms achieve very high detection performance at all batch sizes. RF is the strongest and most stable detector, with accuracy and Macro-F1 remaining at approximately 0.99–1.00 and generalisation gaps typically 0-0.01. ANN also performs strongly, with high precision and slightly lower recall, and small but visible gaps around 0.01–0.03 for front-mounted, and about 0.01 for rear-mounted. SVM achieves competitive results but shows the largest variability across batch sizes and the largest gaps up to about 0.04 in the front-mounted case.

Batch aggregation does not affect detection performance because the models already perform well at $w_s = 7$. Longer windows yield only small, sometimes non-monotonic changes. For example, RF remains nearly constant across all batch sizes, while the front-mounted ANN shows a mild reduction as the batch size increases, with the rear-mounted ANN remaining essentially steady. Sensor-orientation effects are small for detection, but rear-mounted detection is slightly more uniform across batch sizes for ANN and SVM, whereas RF remains very high for both orientations.

6.1.2 Performance Based on Scour Severity

The scour severity is further examined for the best-performing algorithm, i.e., RF. Figure 6.7 and Figure 6.8 present the performance of the RILA front-mounted and rear-mounted models, respectively,

for different scour severities and batch sizes. Scour levels are classified as mild for 5-10%, moderate for 10–15%, and severe for 15–20% scour (vertical stiffness losses). The RF model maintained near-perfect classification performance across all scour severities and temporal windows. Precision and *F1* scores exceeded 0.98 for both RILA front- and rear-mounted cases, indicating that false positives are very few. Recall values remained above 0.97 for front-mounted and 0.96 for rear-mounted cases, confirming that scour severity is reliably detected.

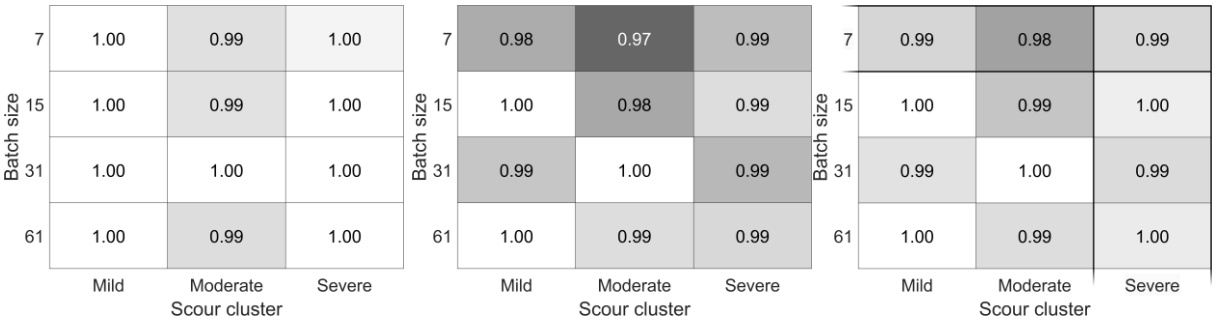


Figure 6.7. Scour severity performance of the RF model for the RILA front-mounted configuration across batch sizes: (a) precision; (b) recall; and (c) *F1* score.

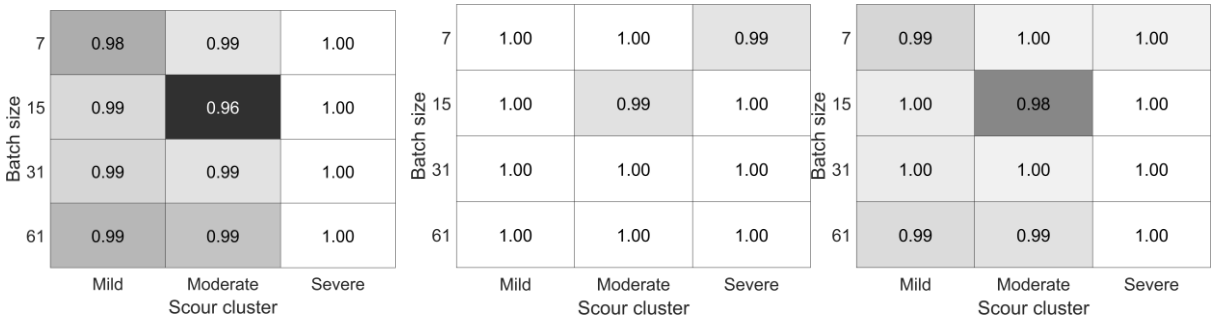


Figure 6.8. Scour severity performance of the scour detection RF model for the RILA rear-mounted configuration across batch sizes: (a) precision; (b) recall; and (c) *F1* score.

Overall, for RF models, performance remains consistently high across mild, moderate, and severe scour clusters, with precision and F1 above 0.98–0.99 and recall above 0.96–0.97, confirming reliable detection across severities and batch sizes.

6.2 Scour Localisation Model Results

6.2.1 Model Metrics

The predictive performance of the support-level scour localisation models is evaluated across temporal batch sizes of 7, 15, 31, and 61 crossings for the RILA front-mounted case, as summarised in Table 6.3, and illustrated in Figure 6.9 to Figure 6.11. The same evaluation is performed for the rear-mounted configuration, as shown in Table 6.4, Figure 6.12 to Figure 6.14. Three ML algorithms (ANN, RF, and SVM) are compared using accuracy, precision, recall, and F1 score for both classes (scoured and healthy supports), along with *Macro-F1*. Across both sensor positions, RF models achieved the highest localisation performance and the most consistent class balance. For the front-mounted case, RF accuracy ranges from 0.954 to 0.968, and *Macro-F1* follows a similar range. ANN and SVM accuracies remain around 0.908-0.927. For the rear-mounted case, RF accuracy ranges from 0.946 to 0.964, whereas ANN and SVM remain around 0.898 to 0.920.

ANN models are more "sensitive" to the class with the scoured label. In other words, these models capture a larger share of truly scoured supports (higher Recall for the scoured class), but they also trigger more false alarms by labelling some healthy supports as scoured (lower Precision for the scoured class). By contrast, in the rear-mounted configuration, SVM behaves more conservatively. When SVM predicts scour, it is more often correct (higher Precision for the scoured class), but it misses a larger share of truly scoured supports (lower Recall for the scoured class). In other words, SVM models produce fewer scour detections overall, but with more missed scoured supports. Overall, RF offers the strongest and most balanced localisation across batch sizes, while ANN and SVM remain competitive but at lower levels.

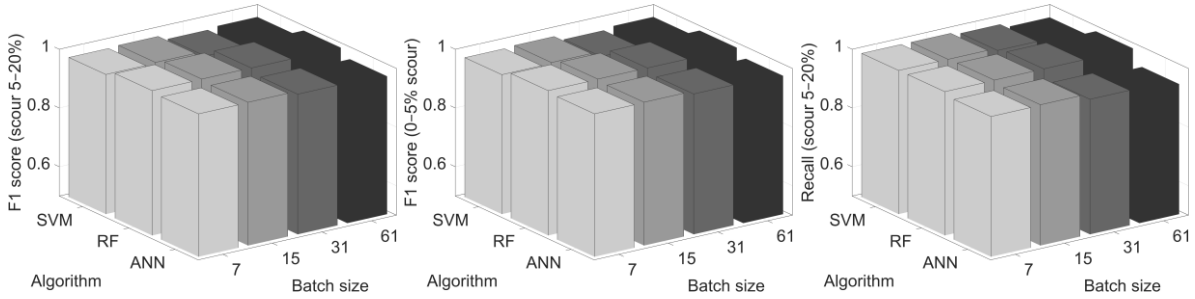
Overfitting is assessed by comparing training and test performance through the generalisation gap, defined as the difference between training and test accuracies. For localisation, the gaps are larger than those observed for bridge-level detection. RF yields the smallest gaps, approximately 3 to 5% across both sensor orientations. ANN and SVM show larger gaps, approximately 7 to 10%, indicating slightly weaker generalisation than RF for the localisation task. Despite these differences, the results remain relatively stable across batch sizes. For ANN and SVM, performance generally improves with greater batch sizes (most clearly at $w_s = 61$), whereas RF remains high across all batch sizes but shows a slight reduction at longer windows in both the front-mounted and rear-mounted configurations.

Support-level localisation is consistently more demanding than bridge-level detection. Across both sensor orientations and all batch sizes, RF achieves the highest localisation performance and the best class balance. For the front-mounted case, RF accuracy ranges from 0.954 to 0.968. For the rear-

mounted case, RF accuracy ranges from 0.946 to 0.964. ANN and SVM perform at lower levels, with accuracies typically around 0.908–0.927 for front-mounted and 0.898–0.920 for rear-mounted.

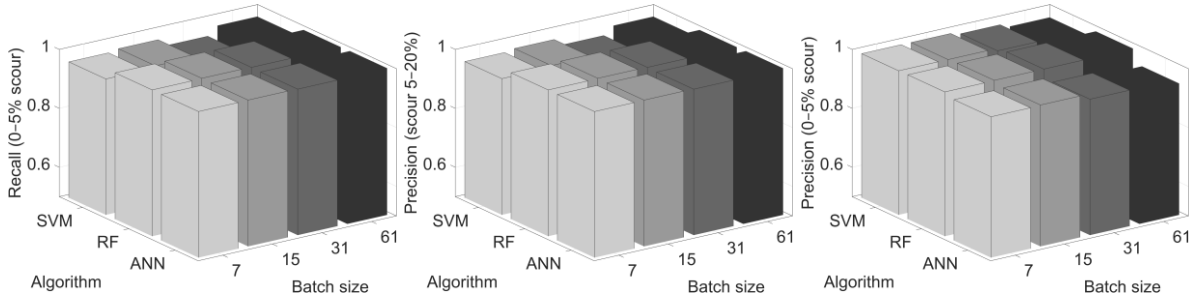
Table 6.3. Scour localisation ML model metrics – RILA front-mounted.

Model	ws	Accuracy	Scoured class (5-20 % scour)			Healthy class (0-5 % scour)			Macro-F1	Training accuracy	Gap
			Precision	Recall	F1 score	Precision	Recall	F1 score			
ANN	7	0.91	0.89	0.94	0.92	0.94	0.88	0.91	0.91	1.00	0.09
ANN	15	0.91	0.87	0.95	0.91	0.95	0.86	0.90	0.91	1.00	0.09
ANN	31	0.92	0.88	0.97	0.92	0.96	0.87	0.92	0.92	1.00	0.08
ANN	61	0.93	0.89	0.98	0.93	0.97	0.88	0.92	0.93	1.00	0.07
RF	7	0.97	0.96	0.97	0.97	0.97	0.96	0.97	0.97	1.00	0.03
RF	15	0.97	0.96	0.97	0.97	0.97	0.96	0.97	0.97	1.00	0.03
RF	31	0.96	0.95	0.97	0.96	0.97	0.95	0.96	0.96	1.00	0.04
RF	61	0.95	0.95	0.96	0.95	0.96	0.95	0.95	0.95	1.00	0.05
SVM	7	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.99	0.07
SVM	15	0.92	0.92	0.93	0.92	0.92	0.91	0.92	0.92	1.00	0.08
SVM	31	0.92	0.91	0.94	0.92	0.94	0.90	0.92	0.92	1.00	0.08
SVM	61	0.93	0.92	0.94	0.93	0.94	0.91	0.93	0.93	1.00	0.07



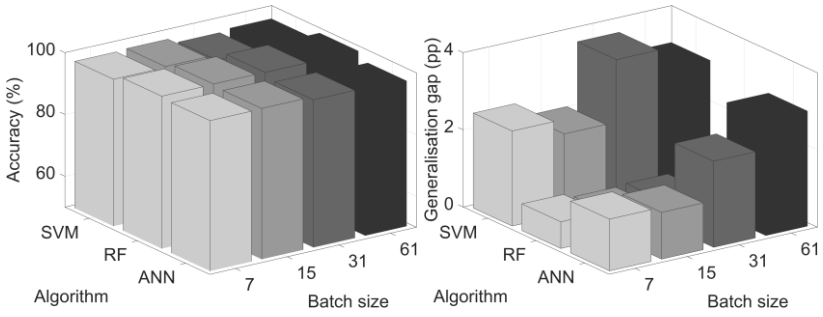
(a) (b) (c)

Figure 6.9. RILA front-mounted: (a) F1 score for 5–20%; (b) F1 score for 0-5%; (c) recall for 5–20%.



(a) (b) (c)

Figure 6.10. RILA front-mounted: (a) recall for 0-5%; (b) precision for 5-20%; (c) precision for 0-5%.

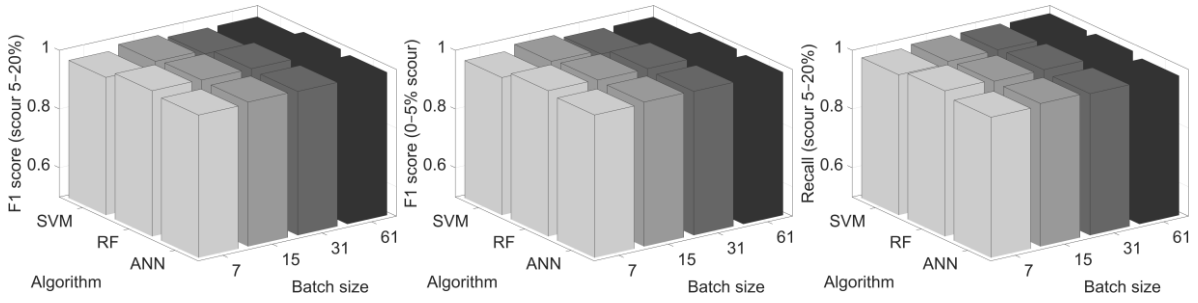


(a) (b)

Figure 6.11. RILA front-mounted: (a) accuracy; (b) generalisation gap.

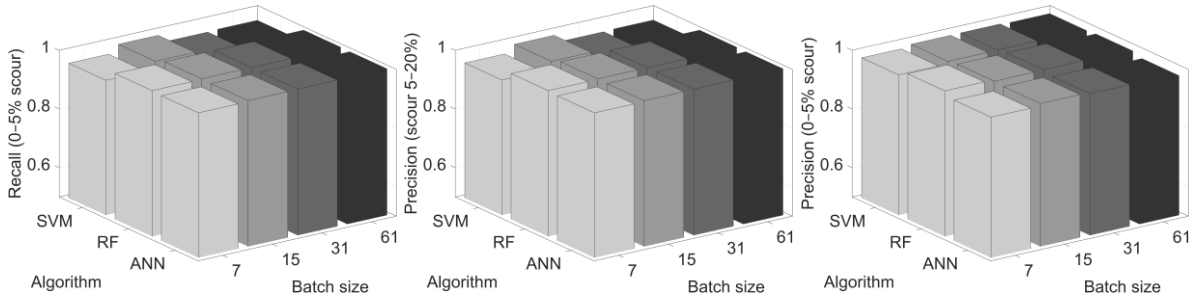
Table 6.4. Scour localisation ML model metrics – RILA rear-mounted.

Model	ws	Accuracy	Scoured class (5-20 % scour)			Healthy class (0-5 % scour)			Macro-F1	Training	Gap
			Precision	Recall	F1 score	Precision	Gap	F1 score		accuracy	
ANN	7	0.90	0.88	0.93	0.91	0.93	0.88	0.90	0.90	1.00	0.10
ANN	15	0.90	0.87	0.94	0.90	0.93	0.86	0.89	0.90	1.00	0.10
ANN	31	0.91	0.87	0.96	0.91	0.95	0.86	0.90	0.91	1.00	0.09
ANN	61	0.92	0.88	0.97	0.92	0.96	0.87	0.92	0.92	1.00	0.08
RF	7	0.96	0.96	0.97	0.96	0.97	0.96	0.96	0.96	1.00	0.04
RF	15	0.95	0.94	0.96	0.95	0.96	0.94	0.95	0.95	1.00	0.05
RF	31	0.95	0.93	0.96	0.95	0.96	0.93	0.95	0.95	1.00	0.05
RF	61	0.95	0.94	0.96	0.95	0.95	0.94	0.95	0.95	1.00	0.05
SVM	7	0.90	0.91	0.90	0.90	0.90	0.91	0.90	0.90	0.99	0.09
SVM	15	0.90	0.91	0.90	0.90	0.90	0.91	0.90	0.90	1.00	0.10
SVM	31	0.91	0.91	0.89	0.90	0.90	0.92	0.91	0.91	1.00	0.09
SVM	61	0.91	0.93	0.90	0.91	0.90	0.93	0.92	0.91	1.00	0.08



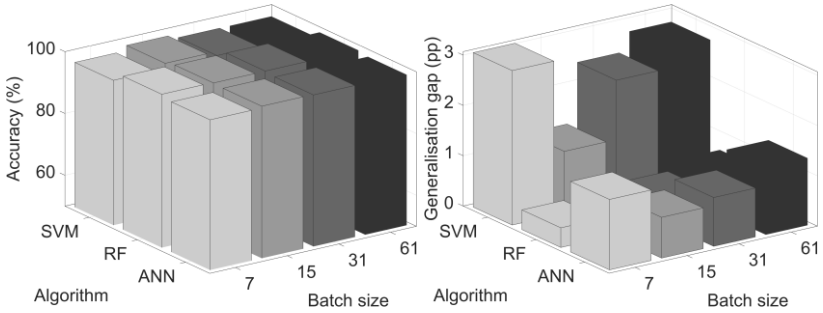
(a) (b) (c)

Figure 6.12. RILA rear-mounted: (a) F1 score for 5–20%; (b) F1 score for 0-5%; (c) recall for 5–20%.



(a) (b) (c)

Figure 6.13. RILA rear-mounted: (a) recall for 0-5%; (b) precision for 5-20%; (c) precision for 0-5%.



(a) (b)

Figure 6.14. RILA rear-mounted: (a) accuracy; (b) generalisation gap.

ANN is more 'sensitive' to scour. It achieves higher recall for the scoured class but lower precision, indicating more false alarms on healthy supports. In contrast, SVM is comparatively more conservative in the rear-mounted configuration, with higher scoured-class precision than recall, meaning fewer false alarms but more missed scoured supports. Generalisation gaps are larger for localisation than for detection. RF shows the smallest gaps around 3–5%, while ANN and SVM show larger gaps of about 7–10%, indicating weaker generalisation for localisation relative to RF.

Batch size has an algorithm-dependent effect. ANN and SVM show modest improvements with increasing batch size, most clearly at $w_s = 61$. In contrast, RF remains strong for short windows and shows a slight reduction at longer windows. For example, in the front-mounted case, accuracy is about 0.97 at $w_s = 7–15$ and about 0.95 at $w_s = 61$. Overall, localisation is slightly stronger in the front-mounted configuration than the rear-mounted configuration, particularly for RF.

6.2.2 Performance Based on Scour Severity

The scour severity is further examined for the best-performing algorithm, i.e., RF. Figure 6.15 and Figure 6.16 present the performance of the RILA front and rear-mounted models for different scour severities across different batch sizes. Scour levels are classified as mild for 5-10%, moderate for 10–15%,

and severe for 15–20% scour (vertical stiffness loss). For scour localisation, the RF models remain strong across the three scour-severity clusters (mild, moderate, and severe) for both RILA front-mounted and rear-mounted configurations. Precision stays at approximately 0.99 to 1.00 across all batch sizes and severities, indicating that when a support is localised as scoured, false positives are very few. *F1* scores are also consistently high, ranging from 0.98 to 0.99 for both sensor positions, indicating stable overall localisation quality across severities. Recall shows the expected reduction compared to bridge-level detection, since localising the correct support is harder than detecting scour presence. Yet this metric remains consistently high: approximately 0.96 to 0.98 for the front-mounted case and approximately 0.95 to 0.97 for the rear-mounted case. The lowest recalls appear in the rear-mounted configuration for the moderate and severe clusters at longer windows (around 0.95). The corresponding *F1* values remain above roughly 0.97, confirming that localisation performance is robust across scour severities and temporal batch sizes.

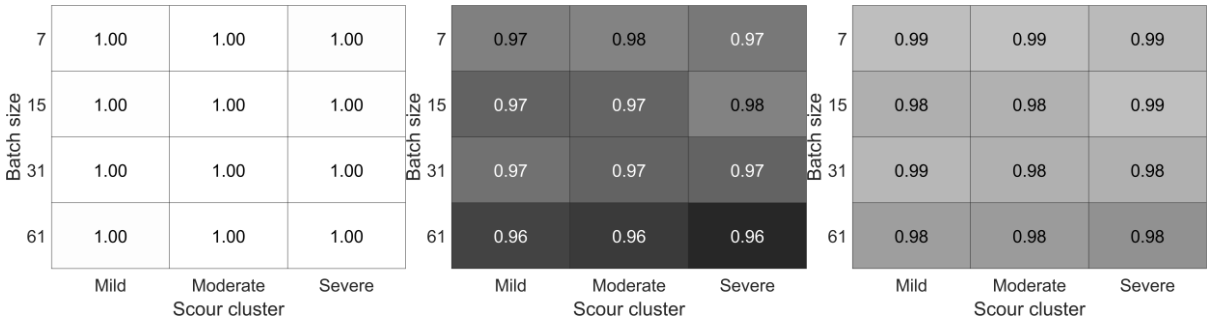


Figure 6.15. Scour severity performance of the scour localisation RF model for the RILA front-mounted configuration across batch sizes: (a) precision; (b) recall; and (c) *F1* score.

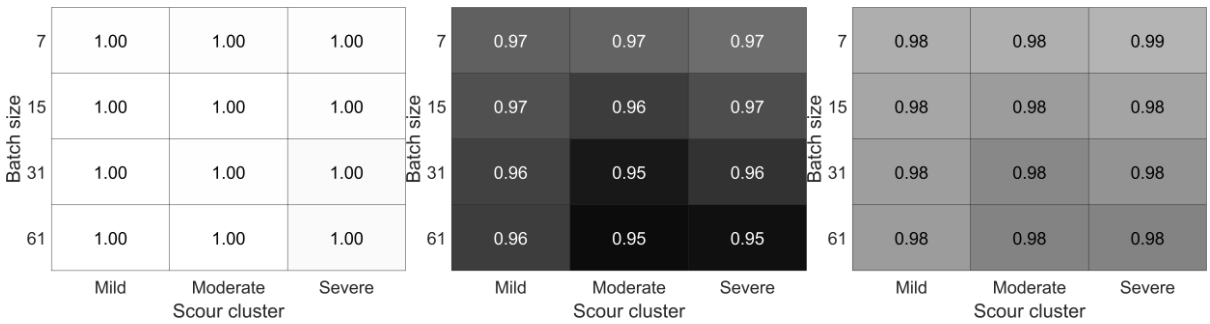
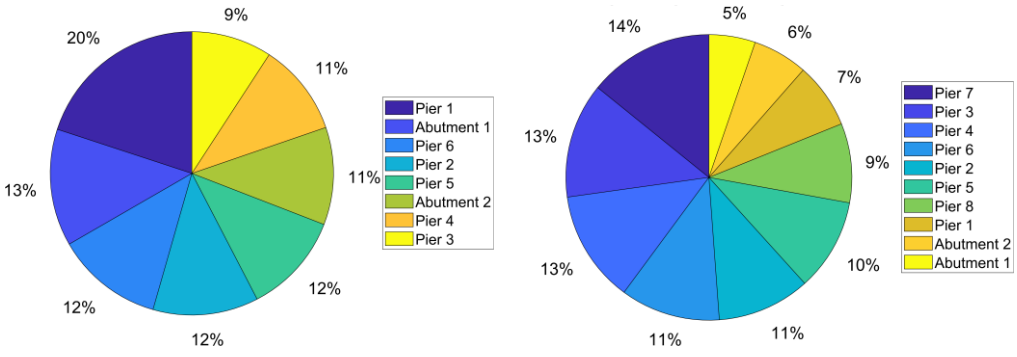


Figure 6.16. Scour severity performance of the scour localisation RF model for the RILA rear-mounted configuration across batch sizes: (a) precision; (b) recall; and (c) *F1* score.

6.2.3 True predicted scoured supports

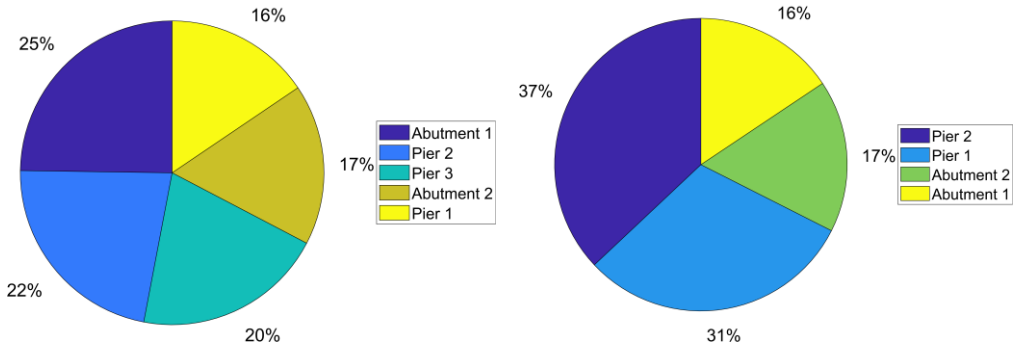
For the most successful model, RF, Figure 6.17 to Figure 6.20 illustrate how the true-positive (TP) localisation cases are distributed across bridge supports under the front-mounted and rear-mounted configurations, and across a batch size of 7 crossings. The TP values indicate how many times that support was correctly predicted as scoured. The scoured supports detected by all algorithms, sensor configurations, and batch sizes are provided in Annex I. Across all four bridges at a batch size of 7, the TP-localisation support fractions are highly consistent between ANN and RF, with differences typically within about 1 percentage point for most supports. This indicates that, when localisation is correct, ANN and RF attribute the correct cases to essentially the same support sets in very similar proportions, for both the front-mounted and rear-mounted configurations. The largest algorithmic deviations occur in SVMs and are not uniform across bridges. On the River Teign and Colonnade bridges, SVM largely preserves the same dominant-support structure as ANN/RF, with only modest reweighting among the leading supports. In WCM15 and WCM15A, the SVM assigns a smaller share of correctly localised TP cases to the abutments (especially Abutment 1) than ANN and RF do. This indicates an abutment-to-pier shift in the TP-localisation composition for SVM, while ANN and RF remain very similar.

At $w_s = 15$, ANN and RF show near-identical TP-localisation distributions across supports for both sensor orientations. SVM follows the same dominant supports, but with selective redistribution: the share of abutments is decreased, particularly for Abutment 1, as in the WCM15 front-mounted and WCM15A rear-mounted cases. The fractions of the supports of River Teign remain stable across all three algorithms, indicating minimal algorithm-driven redistribution for that bridge. At $w_s = 31$ and 61, ANN and RF remain closely aligned, while SVM again introduces the clearest shifts. The most pronounced and repeatable difference is still the down-weighting of abutments in SVM, especially for WCM15A rear-mounted configurations. For the Colonnade bridge rear-mounted configuration, SVM further reduces the Abutment 1 share within the correctly localised TP set. River Teign continues to show only minor cross-algorithm changes. The abutments consistently exhibit the lowest true-positive localisation rates. This behaviour may be attributed to their geometric position at the beginning and end of the bridge, and their structural configuration, which is typically stiffer than that of piers and therefore less sensitive to variations in stiffness.



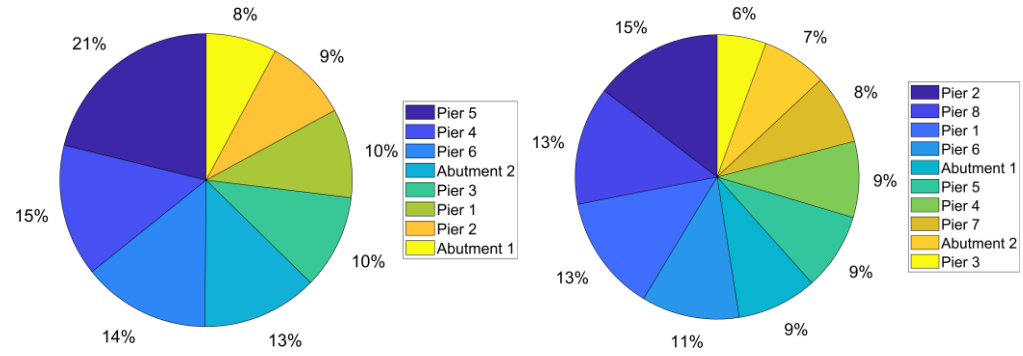
(a) (b)

Figure 6.17. True-positive localisation fractions for the RILA front-mounted configuration, w_s = 7: (a) WCM15 bridge; (b) WCM15A bridge.



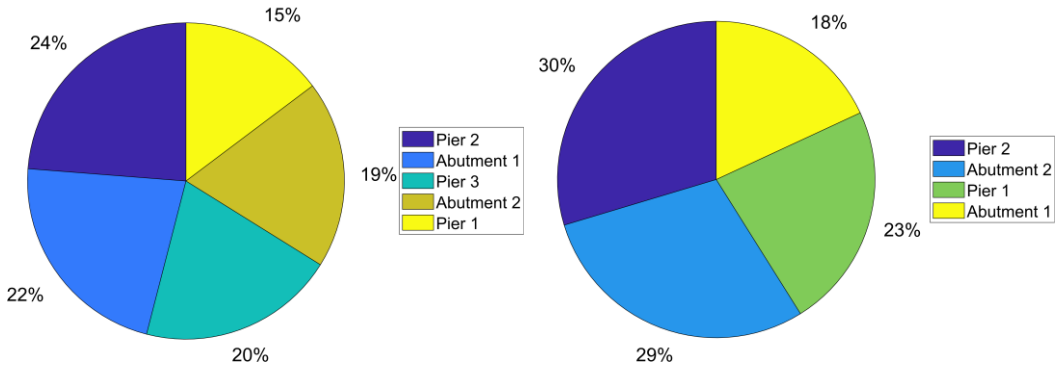
(a) (b)

Figure 6.18. True-positive localisation fractions for the RILA front-mounted configuration, w_s = 7: (a) River Teign bridge; (b) Colonnade bridge.



(a) (b)

Figure 6.19. True-positive localisation fractions for the RILA rear-mounted configuration, w_s = 7: (a) WCM15 bridge; (b) WCM15A bridge.



(a) (b)

Figure 6.20. True-positive localisation fractions for the RILA rear-mounted configuration, $w_s = 7$: (a) River Teign bridge; (b) Colonnade bridge.

ANN's TP-localisation composition is stable with batch size. For most supports, the within-bridge TP fraction changes by only about 1-2% as batch size increases. The main exceptions are:

- WCM15 front-mounted case: a small redistribution occurs with Pier 6 fraction decreasing from 12.9% at batch size 7 to 10.8% at batch size 61, while Pier 1 fraction increases 20.1% to 22.4% from batch size 7 to 61.
- WCM15A front-mounted case: the top-ranked support changes across batch size from Pier 7 at batch size 7 to Pier 4 at batch size 15, and to Pier 3 at batch size 61.
- WCM15A rear-mounted case: there is a gradual shift where Pier 2 fraction reduces from 14.9% to 13.0%, while Pier 8 fraction increases from 13.2% to 15.4% as batch size increases.

RF shows similar stability to ANN: most support fractions change slightly with batch size. The clearest batch-size trends are:

- WCM15 front-mounted case: a progressive increase in Pier 1 fraction from 20.0% to 23.3% and a reduction in the fraction of Abutment 1 from 13.4% to 11.9%, indicating a mild re-weighting.
- The WCM15A front-mounted case shows that the leading supports shift from Pier 7, with 14.2% at batch size 7, towards Pier 4, with 14.3% at batch size 61.
- WCM15A rear-mounted case: Pier 8 fraction increases at batch size 61 from 13.5% to 15.7%, leading to a top-rank change relative to smaller batch sizes.

SVM is the most batch-sensitive of the three, although many support fractions remain stable. The key differences are:

- Colonnade rear-mounted case: Abutment 1 fraction drops from 13.2% at batch size 7 to 7.4% at batch size 61, while Pier 2 fraction increases from 30.9% to 34.8%. TP-localisation distribution is more concentrated on the dominant pier at a larger batch size.
- WCM15 front-mounted case: Pier 1 fraction increases as batch size increases from 22.1% to 25.1%.
- WCM15A front-mounted case: Pier 2 fraction steadily decreases from 11.5% to 8.6%, while the fraction of Pier 4 increases to become the top contributor at batch size 14.9%.
- WCM15A rear-mounted case: Abutment 1 fraction reduces from 6.1% to 3.5%, and the fraction of Abutment 2 decreases from 4.9% to 3.5% with increasing batch size.

Across batch sizes, ANN and RF behave similarly in terms of how correctly localised TP cases are distributed across supports. For most bridge–orientation combinations, both models retain the same dominant supports as the batch size increases, and the support-wise fractions change by only a few percentage points at most. Where changes occur, they mainly appear as small reweightings within the front-mounted group. There are minor rank switches in bridges where several supports have comparable TP fractions, such as WCM15A.

In contrast, SVM shows the largest sensitivity to batch size and the clearest redistribution of TP-localisation fractions. SVM often produces noticeable shifts in the balance between supports as the batch size increases. The most pronounced example is the Colonnade rear-mounted case, where SVM progressively reduces the share of correct localisations attributed to Abutment 1 at larger batch sizes. A milder redistribution is also observed in WCM15A and WCM15, where SVM tends to shift a portion of correct localisations away from abutments and towards specific piers as batch size increases. These results indicate that batching has only a limited effect on the TP-localisation composition for ANN and RF, whereas SVM is more prone to batch-dependent redistribution. Overall, the RF models are assessed as the most suitable choice, because they achieve a stable TP-localisation distribution across supports even at smaller batch sizes. This indicates limited dependence on batching to obtain consistent localisation behaviour.

6.3 Sensitivity and Feature Importance Analysis

6.3.1 Scour Detection Models

6.3.1.1 Feature importance analysis

Feature importance has been computed in a model-appropriate manner. For ANN and SVM, a perturbation-based sensitivity analysis following Cortez & Embrechts (2013) has been applied: each feature was perturbed 100 times. The variance of the resulting sensitivities S_v is computed (Eq. 5.29), and relative importance is obtained by normalising $M_a = S_v$ via Eq. 5.30, so that importances sum to one. For RF, MATLAB's *TreeBagger* out-of-bag (OOB) permutation importance is used. For each tree t, the OOB error ε_t is measured. For each predictor x_j used to split that tree, values are randomly shuffled on its OOB samples to obtain ε_{tj} and the difference $d_{tj} = \varepsilon_{tj} - \varepsilon_t$ is recorded. Here ε_t is the baseline OOB error of tree to its own OOB samples, and ε_{tj} is the OOB error of the same tree t after only feature x_j is randomly shuffled across the OOB samples, leaving all other features unchanged. Across trees, the mean and standard deviation of d_{tj} are computed, and importance is reported as μ_j/σ_j MATLAB's *TreeBagger* function.

The results of the sensitivity analysis are reported in Table 6.5. For the ANN, relative-importance values are almost uniform across batch sizes and sensor positions. The spread narrows as the batch window increases, indicating that the ANN draws on a wide set of predictors with broadly similar weights rather than relying on a few dominant variables. For the RF, out-of-bag permutation importances are more uneven. A small subset of predictors carries most of the weight, while others have very low contribution. Slightly negative permutation importances arise from sampling variability of the OOB procedure and predictor collinearity; they should be interpreted as near-zero importance. This concentration is most evident in shorter batch windows, whereas larger windows yield a flatter importance distribution but still show a clear separation between influential and low-utility predictors. For the SVM, importances are generally small, with only occasional modest peaks. A few features are slightly more influential without becoming dominant. Overall, RF concentrates modelling capacity on a limited set of strong predictors, while ANN and SVM distribute importance more evenly. Table 6.5 supports the modelling choice about the feature selection in Section 5.3.2.3: ANN and SVM exhibit smooth, low-dispersion importance across windows and positions. Selecting the smaller subsets identified by NSGA-II would risk removing cumulatively informative predictors and could result in reduced performance, especially for ANN and SVM. RF shows a more concentrated pattern, but the ensemble down-weights low-utility variables, retaining all features, is acceptable. The computational cost of keeping all features is manageable; hyperparameter tuning via grid search dominates runtime.

Table 6.5. Feature importance analysis results for scour detection models.

Algorithm	Sensor Position	7 day-batch	15 day-batch	31 day-batch	61 day-batch
ANN	Front	Range: 0.00121–0.00232 mean 0.0017; std 0.0001	Range: 0.00084–0.00199 mean 0.0014 std 0.00014	Range: 0.00038–0.00091 mean 0.0006. std 0.00007	Range 0.00021–0.0004 mean 0.0003 std 0.00003
ANN	Rear	Range: 0.00058–0.00235 mean 0.0018; std 0.0002	Range: 0.00042–0.00168 mean 0.0013. std 0.0001	Range: 0.00032–0.00091 mean 0.0006 std 0.00007	Range 0.00014–0.00044 mean 0.0003 std 0.00003
RF	Front	Max 0.832 mean 0.215. std 0.118	Max 0.623. mean 0.147 std 0.101. negatives 6	Max 0.487. mean 0.110 std 0.079. negatives 25	Max 0.392. mean 0.081 std 0.060. negatives 130
RF	Rear	Max 0.782. mean 0.218. std 0.116	Max 0.598; mean 0.150 std 0.098; negatives 8	Max 0.463; mean 0.115 std 0.076; negatives 22	Max 0.409; mean 0.087 std 0.055; negatives 75
SVM	Front	Range: 0.00000–0.03254 mean 0.0018. std 0.0031	Range: 0.00000–0.02081 mean 0.0012. std 0.0022	Range: 0.00000–0.0154 mean 0.0008. std 0.0016	Range 0.000–0.011610 mean 0.0003. std 0.00073
SVM	Rear	Range 0.00000–0.02917 mean 0.0017. std 0.0028	Range 0.00000–0.01950 mean 0.0012. std 0.0021	Range: 0.000–0.01398 mean 0.0007. std 0.0015	Range 0.0000–0.010122 mean 0.0003. std 0.00071

Across the four batch sizes, the ten most important predictors change with batch aggregation. Within this top ten set, input features appear either as aggregated statistics across the batch or as an individual in-window value from a specific crossing position inside the batch. For the front-mounted models, the most persistent descriptors are *Krt(2)* and E_{tot}, each appearing in the top ten across three of the four batch sizes. The remaining top ten features vary with batch size and include several Type-B features that recur only intermittently. For the rear-mounted ANN models, the top ten set is less stable: no descriptor appears in the top ten across three or four batch sizes, and repeated entries typically occur only in two of them.

In the front-mounted RF models, the top ten set is more stable across batch sizes: *Krt(1)*, *Skw(1)*, and *Skw(2)* remain within the top ten at every batch size, with *Krt(2)* and L_{min}^{A} also showing strong

persistence over three batch sizes. In the rear-mounted RF models, the stability is stronger: *Krt(1)*, *Krt(4)*, *Skw(1)*, *Skw(2)*, and *Skw(3)* are present in the top ten at all four batch sizes, and L_{min}^{A} remains in the top ten in three batch sizes. Overall, RF shows a relatively consistent ranking of key Type-A descriptors across batches. For the front-mounted SVM models, the top ten features are *Krt(A)*, *Krt(1)*, and *Krt(4)*, which appear at every batch size, and *Skw(2)*, which recurs at three batch sizes. For the rear-mounted SVM models, *Krt(1)* and *Krt(4)* remain in the top ten across all batch sizes, while L_{min}^{A}, *Skw(1)*, and *Skw(3)* are also persistent across three batch sizes. The top ten predictors remain entirely within the Type-A descriptor set across all batch sizes.

6.3.1.2 Markov-order sensitivity analysis

Further feature analysis of the best models, i.e., RFs, is performed by assessing Type B features. Type-B (Markov) features are ablated in an order-wise manner by removing early Markov orders 1–5 and late Markov orders 35–39. Here, 'ablation' refers to systematically excluding a specified subset of Markov-order features from the input set and recomputing the evaluation metrics under the same protocol, to isolate the contribution of those orders. For each ablation and both front and rear-mounted configurations, balanced accuracy is recomputed to quantify changes in classification performance. In parallel, Cohen's *d* is calculated per order and summarised within each group as the median to characterise class separation. Thus, overall performance is captured by balanced accuracy, while Cohen's *d* clarifies where the discriminative information resides early, near-support orders or later, span-wide, i.e., accumulated response. Early orders (1–5) characterise the short-duration, near-support response at the start of the crossing (localised entry dynamics). Later orders (35–39) characterise the span-wide response after several bogies have passed, reflecting accumulated global dynamics. Cohen's *d* effect size (Cohen, 1988), which measures the standardised mean difference between classes, is calculated as seen in Eq. (6.1):

$$d_k = \frac{(\mu_{scour,k} - \mu_{healthy,k})}{s_{pooled,k}}; \quad s_{pooled,k} = \frac{(n_{scour}-1)\cdot\sigma^2_{scour,k} + (n_{healthy}-1)\cdot\sigma^2_{healthy,k}}{n_{scour}+n_{healthy}-2} \tag{6.1}$$

where μ and σ^2 denote the mean and variance of feature *k,* respectively*, n* is the number of samples in the class (scoured and healthy), and $s_{pooled,k}$ is pooled standard deviation of feature *k*. For the front-mounted sensor, removing early orders lowers balanced accuracy from 0.991 to 0.960; removing late orders lowers it by about 0.014. Early-order dynamics therefore carries most of the discriminative signal at entry, with later orders contributing secondarily. The median Cohen's *d* across early-order features is 0.030, versus 0.029 across late-order features, consistent with the sensor's position at the bridge entry where scour-related stiffness loss first appears. For the rear-mounted sensor, importance is more evenly

distributed across early and late Markov orders. Removing early orders lowers balanced accuracy by approximately 0.025; removing late orders lowers it by approximately 0.023. Per-order class separation is small, with Cohen's d about 0.012 for an early order and −0.030 for a late order. This indicates that the rear sensor relies on interactions accumulated across multiple orders as the train traverses the bridge.

6.3.2 Scour Localisation Models

6.3.2.1 Feature importance analysis

For the scour localisation models, feature importance is quantified using the same single-feature perturbation principle introduced in Section 6.3.1.1 (Eqs. 5.29-5.30): one predictor is perturbed while all others are held constant, and the resulting loss in predictive performance is attributed to that predictor. The implementation, however, is model-specific.

ANN model feature importances are computed by randomly shuffling one predictor at a time on the test set. First, the baseline test accuracy is computed as A_{base}. Then, for each repeat $l = 1, 2, \ldots, Le$, the values of x_j are randomly shuffled across the test samples, producing an accuracy after shuffling, i.e., $A_{j,l}$. The absolute importance of predictor j is defined as the mean accuracy drop in Eq. (6.2):

$$I_j = \frac{1}{R}\sum_{l=1}^{Le}\left(A_{\text{base}} - A_{j,l}\right) \tag{6.2}$$

where x_j denotes the predictor j, and $Le = 100$ denote the number of permutation repeats. To obtain relative importances that sum to one (consistent with Eq. (5.30)), the drops are normalised across all input features, as in Eq. (6.3):

$$RI_j = \frac{I_j}{\sum_{k=1}^{n_{feat}} I_k} \tag{6.3}$$

For the Random Forest localisation models, MATLAB *TreeBagger* function's (Matlab, 2025b) OOB permutation importance is used. Each tree t has its own OOB set, and *TreeBagger* measures a baseline mean margin on that OOB set. The baseline mean OOB margin for that tree is the average margin over its OOB observations, as in Eq. (6.4):

$$\overline{m_t} = \frac{1}{|O_t|}\sum_{i\in O_t} m_{i,t} \tag{6.4}$$

where $m_{i,t}$ is the margin computed from that tree's class scores for observation i. It is the tree's "average confidence" on the cases it did not see during training. It is formulated as in Eq. (6.5):

$$m_{i,t} = s_{i,t}(y_i) - \max s_{i,t}(c);\ for\ y_i \neq c \tag{6.5}$$

where c is the class label, y_i denotes the true class label, and $s_{i,t}(c)$ is the class score that tree t assigns to observation i for class c. In *TreeBagger,* the class score $s_{i,t}(c)$ is the class probability estimate at the leaf that observation i falls into in tree t. It is computed from the class counts of the in-bag training samples reaching that leaf, so for a binary problem, it represents the fraction of training samples in that leaf that belong to class c. Then the permuted mean margin $\bar{m}_{t,j}$ is compared against the baseline mean margin $\bar{m}_t$ computed on the same OOB samples before permutation. The reported importance in Eq. (6.6) is the average increase in classification uncertainty induced by randomly shuffling x_j, aggregated over trees:

$$\Delta_j = \frac{1}{T_R}\sum_{t=1}^{T_R}\left(\bar{m}_t - \bar{m}_{t,j}\right) \tag{6.6}$$

where $\bar{m}_t$ is the mean OOB margin for tree t, and $\bar{m}_{t,j}$ is the mean OOB margin for the same tree after randomly shuffling x_j.

Feature importance in SVM models is quantified using a one-at-a-time perturbation analysis of the trained classifier. The baseline feature vector in Eq. (6.7) is defined as the mean of the training data:

$$\bar{x} = \left(\frac{1}{Le}\right)_{\mathrm{i}} \sum_{r=1}^{Le} x_{i,j} \tag{6.7}$$

where $j = 1, 2, \ldots, n_{feat}$, $X \in \mathbb{R}^{R \times nfeat}$ is the feature matrix with R samples and n_{feat} predictors. Using the same z-score scaling adopted in training with mean vector μ and standard-deviation vector σ, the baseline input in scaled space is as in Eq. (6.8):

$$\bar{z} = (\bar{x} - \mu) / \sigma \tag{6.8}$$

The trained SVM produces a baseline hard label $\hat{\bar{y}} = \hat{y}(\bar{z})$, and a baseline decision scoured class $s_{scoured}$ as $\bar{s}_{scoured} = s_{scoured}(\bar{z})$. For each predictor $j = 1, \ldots, n_{feat}$, generate M perturbed samples by replacing only the j^{th} component of $\bar{x}$ with a value drawn from the empirical distribution of predictor j in the dataset. The n^{th} perturbed vector for predictor j *is denoted as* $x_k^{(n,j)}$, and formulated as in Eq. (6.9):

$$x_k^{(n,j)} = \{\bar{x}_k, if\ k \neq j;\ v^{(n,j)}, if\ k = j\} \tag{6.9}$$

where $v^{(n,j)}$ is sampled from $\{x_{1j}, x_{2j}, \ldots, x^{N_j}\}$. Each perturbed vector is then z-scored as in Eq. (6.10):

$$z^{(n,j)} = (x^{(n,j)} - \mu) / \sigma \tag{6.10}$$

The SVM is evaluated on each perturbed sample to obtain the predicted label $\hat{y}^{(n,j)} = \hat{y}(z^{(n,j)})$, and the scoured class score: $s_{scoured}{}^{(n,j)} = s_{scoured}(z^{(n,j)})$. Two importance indicators in Eq. (6.11), (i) flip rate and (ii) mean absolute score change in Eq. (6.12) are computed for each predictor j:

$$FP_j = (1/M) \cdot \sum_{\{n=1\}}^{M} \mathbb{1}(\hat{y}^{(n,j)} \neq \hat{\bar{y}}) \tag{6.11}$$

$$MC_j = (1/M) \sum_{n=1}^{M} |s_{scoured}^{(n,j)} - \bar{s}_{scoured}| \quad (6.12)$$

A higher FP_j indicates that perturbing predictor j alone more frequently changes the predicted class, i.e., it more often crosses the decision boundary. A higher MC_j indicates a larger change in the SVM decision score toward or away from the scoured class. In MATLAB, the class score $s_{scoured}(\cdot)$ corresponds to the decision score for the scoured class. Decision function values above zero are classified as scoured and values below zero as healthy.

For the scour localisation task, feature-importance estimation follows the same core idea as in the detection task. However, the implementation differs from the scour detection models for two practical reasons. First, the performance measure used to score "importance" and the availability of model-native importance measures. Performance is strongly affected by the per-support variability of scour localisation models. Second, computational burden prevents application of the same importance workflow used for the detection models across all localisation algorithms. In the detection setting, feature importance can be computed using a unified perturbation/sensitivity workflow because the evaluation is comparatively lightweight. In localisation, repeating full disruption experiments over many predictors becomes substantially more expensive. For this reason, localisation uses model-native or computationally efficient importance measures wherever possible, while still adhering to the single-feature disruption principle.

The results of the feature-importance analysis for the scour localisation models are reported in

Table 6.6. For the ANN, relative-importance values decrease with batch window length as expected, since the number of localisation predictors increases with larger batches. Across windows, the ANN exhibits increasing dispersion in the front-mounted case, with the most important features becoming more dominant at longer windows: the maximum relative importance increases to 0.239 at 61-day batches. This indicates that a small subset of predictors becomes increasingly influential as batch size increases. In contrast, the rear-mounted ANN shows a more stable, less uneven distribution at long windows, with a maximum of 0.064 at 61-day batches. This indicates a more uniform distribution of reliance across predictors for that sensor position.

For the RF, OOB permutation importances are all positive, and both the mean and the spread reduce as the batch window increases for both sensor positions. The maximum importance also drops from about 1.6 for 7-day batches to below 1.0 for 61-day batches. This progressive flattening indicates that, as the batch window grows, the forest distributes its decision-making across a broader set of predictors rather than concentrating strongly on a small subset. For the SVM, permutation-based importances are sparse, with many predictors exhibiting zero importance and only a few producing modest peaks. Similar

to the ANN, the front-mounted case shows occasional higher peaks than the rear-mounted case (for example, maxima around 0.10–0.12 at shorter and intermediate windows). The rear-mounted case remains more bounded: maximum 0.03875 at 7-day batches and 0.02366 at 61-day batches.

Overall,

Table 6.6 indicates that localisation importance patterns are more position-dependent than in scour detection models: front-mounted models, particularly the ANN, tend to concentrate in a small set of influential predictors at long windows, whereas rear-mounted models remain more evenly distributed. These observations support the retention of the full localisation feature set in the main analyses: for ANN and SVM, many predictors contribute weak but complementary information, and aggressive feature pruning could remove cumulatively informative cues. For RF models, the ensemble can down-weight low-utility features, because each tree only uses a subset of predictors at each split, and splits are chosen only when they improve purity. As the window length increases, feature importances become less spread out, because longer windows smooth out noise and make several predictors capture similar, overlapping information. The RF can accommodate this redundancy by selecting among these similar predictors across trees, so explicit feature elimination is not required.

Across the four batch sizes, some features are common in the top ten most important feature set across all batch sizes. These features appear either as aggregated statistics across the batch (min/mean/max/std) or as the value from a specific position within the batch, namely an individual feature of a crossing at that position. For the front-mounted ANN, the same descriptor families repeatedly appear among the ten most important predictors at every batch size, namely E_{tot}, *Krt(A)*, and *Krt(3)*. For the rear-mounted case, the only descriptors that remain in the top ten set across all batch sizes are *Skw(1*) and *Skw(2)*.

The RF shows a stable set of recurring top predictors across batch sizes, but with slightly different common features between sensor positions. For the front-mounted case, E_{tot}, *Krt(A)*, *Krt(4)*, *Skw(4)*, and the Markov-type terms *g(2)* and *g(13)* are present in the top ten importance set at all batch sizes. For the rear-mounted case, E_{tot} remains common across all batch sizes, together with *Krt(1)*, *Krt(2)*, *Krt(3)*, and *Skw(1)*. For the RBF SVM models, the most consistently recurring top predictors are Type-A features, but the common feature set across all top 10 features is narrower. For the front-mounted case, *Krt(A)*, *Krt(1)*, and *Krt(3)* appear in the top ten importance set across all batch sizes. For the rear-mounted case, the only descriptor that remains in the top ten set at all batch sizes is *Skw(A)*. This consistency is observed even though the selected predictor is not always the same: depending on the batch window, *Skw(A)*

appears either as an aggregated statistic across the window (for example, the minimum/mean/maximum of *Skw(A)*) or as a single window-indexed predictor corresponding to one specific position within the batch.

Table 6.6. Feature importance analysis results for scour localisation models.

Algorithm	Sensor Position	7 day-batch	15 day-batch	31 day-batch	61 day-batch
ANN	Front	Range: 0.00013–0.02836 mean 0.00364. std 0.00442	Range: 0.00001–0.04919 mean 0.00211. std 0.00423	Range: 0.00000–0.09229 mean 0.00114. std 0.00443	Range: 0–0.23882 mean 0.00062. std 0.00703
ANN	Rear	Range: 0.00012–0.04059 mean 0.00364. std 0.00482	Range: 0.00001–0.04896 mean 0.00211. std 0.00444	Range: 0.00000–0.06986 mean 0.00114. std 0.00439	Range: 0–0.06384 mean 0.00062. std 0.00312
RF	Front	Max 1.566 mean 0.683. std 0.226. negatives 0	Max 1.435 mean 0.557. std 0.180. negatives 0	Max 1.095 mean 0.416. std 0.130. negatives 0	Max 0.908 mean 0.329. std 0.102. negatives 0
RF	Rear	Max 1.618 mean 0.682. std 0.225. negatives 0	Max 1.284 mean 0.601. std 0.165. negatives 0	Max 1.115 mean 0.451. std 0.124. negatives 0	Max 0.752 mean 0.347. std 0.089. negatives 0
SVM	Front	Range: 0–0.10423 mean 0.00364. std 0.01052	Range: 0–0.05799 mean 0.00211. std 0.00490	Range: 0–0.11905 mean 0.00114. std 0.00493	Range: 0–0.06929 mean 0.00062. std 0.00251
SVM	Rear	Range: 0–0.03875 mean 0.00364. std 0.00662	Range: 0–0.03337 mean 0.00211. std 0.00339	Range: 0–0.04655 mean 0.00114. std 0.00335	Range: 0–0.02366 mean 0.00062. std 0.00118

6.3.2.2 Markov-order sensitivity analysis

The localisation model uses Markov parameters of order $q = 1, \ldots, H$, where $H = 13$ denotes the maximum Markov order retained per support and per window. To assess which parts of the Markov sequence carry the most discriminative information, importance was aggregated per order by summing the Random Forest OOB permutation importance over all predictors representing that order (across the mean, standard deviation, minimum, and maximum blocks, and the raw values from the first position in the window).

For the front-mounted case (w_s = 7), the aggregated RF importance shows that the most influential orders are g = [4, 5, 13, 3, 11], with a slight preference for early orders (Late/Early = 0.939). Consistently, the raw per-order class separation is higher in early orders, with median Cohen's d = 0.036 for orders 1–5 versus 0.006 for orders 9–13. Despite these differences, the ablation study yields an OOB balanced accuracy of 0.974 for the full model and removing either the early or late order sets changes balanced accuracy only to 0.975. This confirms that performance is not materially affected by excluding either order band in this configuration.

For the rear-mounted case (w_s = 7), the importance distribution is broadly spread across orders, with the most influential orders g = [9, 11, 8, 4, 13] and a near-balanced emphasis between late and early regions (Late/Early = 1.10). Median effect sizes remain small (Cohen's d = 0.020 for orders 1–5, and 0.012 for orders 9–13). The ablation study shows that the full model has OOB balanced accuracy of 0.968; removing early orders yields 0.970, while removing late orders yields 0.969. These changes are negligible and do not indicate a clear reliance on either early or late order bands.

Overall, this weak Markov-order sensitivity is expected for localisation, as the analysis windows focus on the scour-affected zone rather than the full bridge. Consequently, the Markov sequence reflects local dynamics over a short segment, and discriminative information is distributed more redundantly across orders, so removing early or late order bands has a negligible impact on balanced accuracy.

6.4 Discussions and Final Remarks

6.4.1 Effect of Batch Size on Performance

For scour detection models, the batch size determines the amount of information available to the classifiers: a 7-day batch requires only a week of crossings to produce a prediction, while a 61-day batch integrates nearly 2 months of data. As expected, larger batches yield slightly higher accuracies, since the models exploit richer input to stabilise predictions. This trend is most consistent for RFs, which maintain accuracies above 99% even at small batch sizes and improve marginally with longer windows. ANNs and SVMs also benefit from larger batches, but their gains are less consistent, reflecting higher sensitivity to data variability. The 7-batch models already achieve strong performance, showing that reliable detection is possible with only a week of data. This balance between short-horizon responsiveness and long-horizon stability highlights the framework's operational flexibility.

For the support-level scour localisation models, in general, larger batches lead to slightly higher test performance, but the gains are modest and depend on the algorithm. Across both sensor orientations, ANNs and SVMs show the clearest benefit from increasing w_s. For both rear and front-mounted cases,

ANN accuracy and *Macro-F1* increase as the batch size increases. These improvements are mainly associated with stronger identification of scoured supports for longer batches (higher Recall for the scoured class), while Precision for the scoured class remains lower (more false alarms on healthy supports) than Recall. Relative to RF and SVM, this Precision–Recall pattern indicates that the ANN is more inclined to label supports as scoured. SVM shows modest improvements with larger w_s, but with a stable level overall. For the front-mounted case, accuracy and *Macro-F1* are both around 0.92–0.93. For the rear-mounted case, accuracy remains around 0.90–0.91. Therefore, SVM Precision for the scoured class is higher than Recall, meaning that relative to ANN, SVM is more conservative: when it predicts scour, it is more often correct, but it misses a slightly larger share of truly scoured supports.

RF scour localisation models remain the strongest performers for all batch sizes, but unlike detection, they do not show a consistent increase with longer batches. For the front-mounted case, RF accuracy is already 0.97 at w_s = 7–15, then reduces slightly to 0.95 at w_s = 61. For the rear-mounted case, RF accuracy is similarly high and stable (0.96 at w_s = 7, and 0.95 for w_s = 15–61). This indicates that, for localisation, RF can achieve near-peak performance with short batches.

6.4.2 Differences in Performance Across ML Algorithms

Across both front- and rear-mounted cases, all scour detection models achieve high predictive capability, but with varying stability. Stability here reflects the model's ability to maintain consistent test performance across folds, batch sizes, and front/rear mounting—with low metric variance and a small training–testing gap. ANN models consistently yield high precision, though recall is somewhat lower, leading to moderate generalisation gaps. RF provides the most balanced and stable performance, with accuracy, *F1*, precision, and recall all remaining above 0.99, and gaps across batch sizes are negligible. SVM achieves competitive accuracies and balanced precision–recall, but its results fluctuate more and there are larger gaps (up to three percentage points). Overall, RF is the most reliable, while ANN and SVM remain effective but less stable.

For localisation models, performance is consistently lower than for bridge-level detection, reflecting the fact that these models are more demanding. The support-level decision is more demanding than bridge-level detection for two practical reasons. First, the localisation models effectively operate with a stricter tolerance: a prediction can be correct at the bridge-level while still being penalised at the support-level if scour is assigned to the wrong support or if healthy supports are incorrectly flagged. Second, particularly for the multi-pier scour cases, healthy supports adjacent to the scoured ones can show slightly altered track profile trends due to the influence of neighbouring scour, compared with healthy supports

farther away on the same bridge. This can cause the model to incorrectly label these nearby healthy supports as scoured.

RF delivers the highest and most reliable localisation performance across batch sizes and sensor orientations. For the front-mounted case, RF accuracy remains in the range 0.954–0.968, and for the rear-mounted case, it remains in the range 0.946–0.964. RF also maintains the most balanced behaviour between the scoured and healthy classes, with both Precision and Recall remaining high and relatively close to each other. In addition, RF exhibits the smallest generalisation gaps, approximately 3 to 5%, indicating stronger generalisation for the localisation task.

ANN and SVM remain competitive but at a lower overall level than RF. ANN accuracies remain around 0.908–0.927 for the front-mounted case and 0.898–0.920 for the rear-mounted case. In terms of class balance, ANN tends to prioritise catching scoured supports: Recall for the scoured class is consistently high, while Precision for the scoured class is lower, meaning that ANN identifies a larger share of truly scoured supports but triggers more false alarms by labelling some healthy supports as scoured. SVM shows the opposite tendency, most clearly for the rear-mounted configuration: Precision for the scoured class is higher than Recall, indicating a more conservative behaviour in which scour predictions are more often correct, but a larger share of truly scoured supports is missed. ANN and SVM show larger gaps, approximately 7 to 10%, indicating weaker generalisation than RF for localisation.

6.4.3 Model Interpretability

RF scour detection models offer the highest transparency, as their tree-based structure allows feature importance to be derived directly through permutation and out-of-bag measures, with results readily interpretable in terms of physical concepts such as Markov orders (system dynamics) and DWT bands (frequency content). In contrast, ANNs diffuse sensitivities across layers of hidden units, whereas SVMs define decision boundaries via support vectors, which can be particularly opaque with nonlinear kernels such as RBFs. Both models provide weaker links to individual features or structural mechanisms, making them less interpretable than RFs despite their competitive accuracy.

Amongst scour localisation models, RFs offer the highest transparency, since their tree-based structure provides model-native importance measures through OOB permutation analysis. Here, model-native means that the importance values are obtained directly from the model's own prediction mechanism (how splits and votes change when a predictor is disrupted), rather than from an external, post-processing method. This allows the contribution of individual predictors to be ranked directly, and the dominant predictors can be related to the main descriptor families used in localisation, including Type-A (DWT-

based features) and Type-B (Markov-type terms). In contrast, ANN localisation models rely on post hoc permutation importance on the test set. While this produces a usable ranking, sensitivities are still distributed across many weights and hidden units, so the link between a performance change and a specific predictor remains less direct than for RF. SVM localisation models are the least directly interpretable: with an RBF kernel, the decision boundary is defined implicitly through support vectors, and importance must be inferred using perturbation indicators (flip rate and decision-score change) rather than a model-native ranking.

CHAPTER 7: CONCLUSIONS, VALIDITY AND FUTURE WORK

Chapter 7 provides a conclusion to the thesis, outlining the primary findings and contributions, and clarifying the level of evidence supporting each conclusion. The proposed approach is compared with prior related studies, and the validation concept is discussed. A review of robustness and uncertainty is conducted with respect to noise, operational variability, and modelling choices. Finally, the chapter states the main limitations and outlines potential future work, with emphasis on targeted field campaigns and transfer across networks.

7.1 Summary of Findings and Contributions

This thesis proposes the development of a simulation-based, drive-by framework for assessing scour in railway bridges, utilising train-mounted sensing technology. The work aimed to:

i. Exploit RILA longitudinal track-profile data as the sensing input for both detection and localisation.

ii. Operate under in-service variability by using routine train passages and accounting for realistic changes in speed, axle loads, vehicle configuration, and track condition.

iii. Detect scour at bridge-level using sliding-window batching (7, 15, 31, or 61 days).

iv. Localise scour to individual supports to support targeted inspection.

v. Demonstrate external validity across bridges and time using external cross-validation and robustness analyses.

Relative to these aims, the thesis delivers an end-to-end workflow that links:

i. Developing a scour detection methodology using track profiles and application of the methodology to a case study that combines measured RILA track profiles with synthetically generated scoured profiles.

ii. Simulation-based data generation for multiple bridge types, spans, configurations, train types, travel speeds, and scour conditions

iii. Feature engineering using combined family sets (Type A and Type B) computed from the synthetic RILA-type track profiles

iv. Training and testing supervised Machine Learning (ML) algorithms to generalise the methodology to other bridges. The ML models address both scour detection and support-level localisation under different sensor orientations and batch sizes and are validated using out-of-sample splits designed to test transfer across bridges and operating conditions.

Across both front-mounted and rear-mounted configurations, the scour detection ML models achieve very high performance for all batch sizes, with Random Forest (RF) showing the strongest and most stable behaviour, Artificial Neural Network (ANN) performing strongly with slightly lower recall, and Support Vector Machine (SVM) remaining competitive but exhibiting the greatest variability across batch sizes. In parallel, the scour localisation models provide support-level predictions that enable a more granular interpretation than bridge-level detection alone, supporting the thesis goal of moving from "is scour present?" towards "where is it most likely located?" within the constraints of the available evidence.

The key scientific and practical contributions can be summarised as follows:

i. The thesis provides a structured simulation-to-ML pipeline that makes synthetic data generation explicit and reproducible for drive-by scour assessment using longitudinal track-profile inputs.

ii. It introduces and evaluates combined family sets (Type A and Type B) that capture complementary information from the synthetic RILA-type track profiles, supporting both bridge-level detection and support-level localisation.

iii. It develops and tests a sliding-window batching strategy and decision logic that stabilise inference under realistic operational variability, and it evaluates transfer across bridges, sensing configurations, and time using external cross-validation and robustness analyses.

7.2 Positioning Relative to Prior Indirect Approaches

Relative to prior indirect monitoring approaches for scour, this thesis adds a simulation-based framework that evaluates both bridge-level detection and support-level localisation using the same sensing modality and a consistent feature-engineering and learning pipeline. Rather than focusing on a single algorithm, bridge, or pass, the work compares multiple ML classifiers under matched conditions and assesses how performance changes with sensor orientation and batched decision-making.

This thesis also differs from much of the drive-by scour literature in the type and placement of the response used for learning. The proposed pipeline is built around measured and simulated displacement

responses derived from RILA longitudinal track-geometry profiles, rather than relying on axle-box or bogie-located accelerations. The sensing is not located in the wheel–rail contact region; instead, the response is acquired through a train-mounted unit positioned away from the axle, which is a more convenient sensing configuration for deployment. Many indirect approaches demonstrate feasibility only in limited scenarios, with a single scoured pier, or under a narrow set of operating conditions. Explicit investigation of multiple scoured supports remains comparatively limited in the drive-by scour detection literature. This thesis adopts a systematic, comparative evaluation across multiple bridges and controlled scour scenarios generated within the same simulation framework, including multi-support scour cases, enabling direct like-for-like comparisons between detection and localisation tasks, sensor configurations, and batch sizes.

7.2.1 Scour Detection: Recent Scour Detection Studies and this Thesis

As stated in Section 2.2.1.2, Fernandes et al. (2024), Gasparotti et al. (2024), and Minski and Lopez (2023) develop acceleration-based, drive-by ML pipelines for scour detection and/or severity classification. Fernandes et al. report performance primarily through Receiver Operating Characteristic (ROC) curves and show that discrimination improves with the number of passages (illustrated for 10, 20, and 50 passages), approaching near-perfect separation in the 50-passage case for 5 % and 10 % support stiffness loss scenarios. Gasparotti et al. report confusion matrices and mean accuracy for (healthy, 5 %, and 10 scoured cases%), with perfect classification for the healthy and 10 % classes and residual confusion concentrated at 5 %. The reported mean accuracies are approximately 93 % (DOF-6), 90 % (DOF-1), and 82 % (DOF-5), with 95 % confidence intervals of about 88.9–97.1 %, 87.9–92.6 %, and 77.8–86.5 %, respectively. Minski and Lopez report a five-class confusion matrix using 100 test passages per class (baseline, 5 %, 10 %, 20 %, 50 %). As expected, misclassifications occur mainly among the lowest-damage classes, while the 10 %, 20 %, and 50 % classes are classified cleanly (100 % in the reported matrix). They also report two binary versions of the model: one that distinguishes baseline from 5 per cent damage (average precision of about 90 per cent) and another that distinguishes baseline from 10 per cent damage (average precision of about 95 per cent).

The scour detection models presented in this thesis are evaluated using scalar classification metrics (accuracy, precision, recall, and F1 score) under external cross-validation and across batch sizes of 7–61 crossings. For both RILA front-mounted and rear-mounted configurations, performance remains consistently high, with reported accuracy in the range 0.96–1.00 and Macro-F1 in the range 0.96–0.99, together with small generalisation gaps (lowest for the RF models). These results are not intended as a direct benchmark against the acceleration-based studies above, because the sensing modality, response

type, and evaluation protocols differ. Rather, they demonstrate that high scour-detection performance can be achieved using RILA track-geometry profiles, as evidenced by consistent scalar metrics across multiple batch sizes and sensor orientations.

7.2.2 Scour Localisation: Recent Work and This Thesis

Locating scour remains less commonly reported than simple detection in recent drive-by scour studies. A clear example that includes an explicit localisation step is Zhang et al. (2022), who propose a pier index to locate the scoured pier using component wavelet energy, and validate the approach using numerical simulation and laboratory testing.

The localisation setting in the research of Zhang et al. (2022) differs from that adopted in this thesis in three main respects. First, the measured quantity differs: Zhang et al. use acceleration-based responses with sensors close to the vehicle's running gear, whereas this thesis uses displacement-based RILA track-geometry profiles in both the measured and simulated datasets. Second, the localisation target differs: this thesis frames localisation at the support-level (piers and abutments), whereas Zhang et al. report localisation through an energy-based pier index built from wavelet components. Third, the decision and learning setup differs: Zhang et al. combine statistical and wavelet-based indicators with an index-based localisation step, while this thesis evaluates supervised ML classifiers for both detection and localisation under controlled variations in sensor orientation and batch size.

Within the railway drive-by scour literature, earlier wavelet work by Fitzgerald et al. (2019b) models the scour of a multi-span bridge and reports localisation-oriented outputs, including predicted scour locations that are close to the affected piers for most tested scenarios. Post-2020 examples that explicitly report a scour location outcome using vehicle-mounted sensing remain comparatively limited, which supports the localisation emphasis of this thesis.

7.3 Data Source and External Validity

Despite the limited size of the field dataset, the 16 measured crossings were used in this thesis to develop the initial scour detection methodology. However, these measurements correspond only to the healthy stage of the case-study bridge. The scoured-stage track profile measurements were not available from field monitoring and had to be generated synthetically within the simulation framework. Nevertheless, the available field runs provided a practical reference for understanding the characteristics of in-service RILA profiles. Including crossing-to-crossing variability in travel speed and the unknown train weight, these measurements also helped identify that the measured track profiles should be grouped into four clusters, reflecting differences in the track line and sensor orientation. Since these are acquired

relatively close together, the rail irregularities are assumed to remain unchanged over the measurement period. Importantly, it has been found that these field track profiles contain both rail irregularities and bridge response. These contributions must be separated to reliably infer scour. This insight motivated the subsequent ARX-based modelling of the track profile, in which the measured response is represented as a sum of rail-irregularity and bridge-displacement components. Markov parameters derived from the fitted ARX models are then extracted and used as Type-B features in the downstream ML models.

Due to the scarcity of field measurements, ML models are trained and tested here using input features derived from simulated track profiles. It is important to discuss external validity, which refers to whether the conclusions drawn from trained ML models remain reliable outside the specific conditions used to develop them. In practice, the question is: "If the bridge, the day, or the train changes, do the same decision rules still hold?" This is important because drive-by measurements reflect not only scour-related changes, but also operational variability, measurement noise, and asset-specific characteristics.

Transfer to other bridges. Although the Type-A and Type-B feature extraction is designed to emphasise scour-sensitive changes, the measured response still depends on bridge and track characteristics. Differences in span layout, stiffness distribution, support conditions, track properties, and local irregularities can shift the distributions of features, even for the same scour state. Feature distribution change refers to shifts in the range and statistics (mean, variability, and shape) of the extracted input features when operating conditions change, which can move new data away from the patterns learned during training. As a result, performance may decrease when models are applied to bridges with geometries and boundary conditions not represented in the development set, and this transferability should be checked using cross-bridge tests.

A practical distinction between the two ML tasks is that the bridge-level scour detection framework does not require explicit bridge-geometry information during feature derivation. It relies on features extracted from the full RILA track-profile response at bridge level. In contrast, the pier-level scour localisation models require approximate span lengths and support numbering, since the input features are derived from support-centred scour-affected zones. Therefore, the localisation task has a higher requirement for bridge inventory information than the bridge-level detection task.

Transfer to other days (temporal transfer). Day-to-day operational variability can shift the input distributions for reasons unrelated to scour. This can shift the baseline feature levels so that crossings under healthy conditions produce feature values that differ from those used to train the model. Batch-based decision-making is expected to stabilise predictions by aggregating information across crossings, but

longer-term drift may still occur and should be monitored. Longer-term drift consists of gradual, persistent changes in the signal and hence in the baseline over extended periods (weeks to months), driven by operational, track condition, vehicle, or sensor changes rather than scour. Settlement due to compaction of the ballast and sub-ballast layer, and track interventions such as tamping and ballast-related works, lead to shifts in the measured rail irregularity profile over time.

Transfer to other trains (vehicle transfer). Vehicle properties influence the measured signals. In this thesis, the models are developed using responses from three distinct vehicle types, intended to improve robustness to vehicle-to-vehicle variability within the simulated operating envelope. Nevertheless, changes in axle loads, suspension and damping conditions, wheel–rail interaction and running behaviour can still shift feature values. For this reason, transfer to trainsets and operating conditions outside those represented in the development data should be verified through targeted validation.

Synthetic to field (deployment transfer). Since the available field dataset for the case-study bridge is limited to 16 crossings, it was not sufficient to train and validate the ML models directly from measurements. For this reason, the training dataset is constructed using synthetically generated runs from the simulation framework. To support generalisation within the simulated scope, the simulations cover four bridges (two steel and two concrete) with different span lengths, span numbers, and flexural rigidities, and they include vehicles with different suspension properties. While this broadens the range of conditions represented during development, simulation cannot fully reproduce all real effects.

Although the models are designed to be robust to operational variability, their performance under changes in bridge characteristics, travel speeds, and vehicle properties cannot be guaranteed based on the current dataset alone. Applying the same models to other bridges, other time periods, or other trains may introduce a distribution shift in the input features. For this reason, further validation using field measurements is required, including checks for performance drift over time and across vehicles.

7.4 Uncertainty Quantification and Robustness

Internal validity refers to whether the observed effects are due to scour rather than to non-scour changes in the measurements. In this thesis, robustness is assessed within the operating envelope considered in the simulations. In particular, the methodology is demonstrated under measurement noise levels up to 1 % and for a travel speed up to 44.7 m/s. Variations outside these conditions (e.g., higher noise levels, different speed ranges, or operational scenarios not represented in the training data) may alter the extracted Type-A and Type-B features and affect performance. Testing beyond the simulated envelope is therefore considered a limitation and left for further validation.

Robustness checks in this thesis examine whether scour-related patterns remain identifiable when common non-scour variations and realistic scenario differences are introduced in a controlled way. These checks include noise injection up to 1 %, variation in operating conditions across crossings (e.g., travel speed and train weight), comparison of different sensor orientations, and evaluation across alternative ML model choices and settings (ANN, RF, and SVM, batched decision-making). In addition, the methodology is assessed across different scour configurations (scour level and affected support(s)), which helps confirm that the learned decisions are linked to scour scenarios rather than a single fixed damage pattern.

In deployment, model "confidence" should be interpreted as the degree to which a new crossing resembles the patterns learned during development, rather than as a guarantee that scour is present. Drive-by measurements vary with travel speed, train weight, sensor orientation, and measurement noise, so the same scour state of a bridge can produce a slightly different range of Type-A and Type-B feature values. In addition, the models are trained on simulated data generated using specific choices of speed, train weight, noise level, vehicle properties, bridge configurations, and scour scenarios. Model behaviour in deployment is conditioned by these modelling assumptions and is most reliable when operating conditions remain within the simulated ranges. Consequently, confidence should be viewed as a reliability indicator: high-confidence outputs suggest strong agreement with learned scour-related patterns, whereas low-confidence outputs indicate overlap with the decision boundary, atypical inputs, or operating conditions not well represented in the development data. In practice, confidence should be considered alongside prediction stability across repeated crossings (batching) and checks that the operating conditions and feature ranges remain consistent with those represented during model development.

7.5 Limitations and Future Work

Although the results demonstrate the potential of the proposed drive-by framework for scour detection and localisation, several limitations should be explicitly acknowledged. These limitations also define the main directions for future research.

The ML framework developed in this thesis is simulation-based. Due to the limited availability of field RILA measurements, the track profiles used to train, validate, and test the ML algorithms were generated synthetically. Hence, the reported performance reflects the simulated conditions considered in the thesis. When the models are tested with field data in future work, a reduction in performance may be expected. This is because real monitoring data can include effects not fully represented in simulations, such as environmental variability, temperature, changes in track conditions, sensor and positioning uncertainties,

maintenance interventions, unknown vehicle properties, and imperfect ground truth about the actual scour state.

The operational validity of the labels should also be considered. In this thesis, “scour present” is defined by the simulated scour scenarios, namely scoured versus healthy cases, and “where is the scour?” is defined as the affected support or supports within the bridge model. These definitions enable controlled evaluation of the detection and localisation tasks. However, they remain simplified proxies for practice, where scour severity and ground truth are uncertain and are usually confirmed through inspections. Future work should therefore better align the model labels with how asset owners define actionable scour conditions.

The bridge and TTB-2D simulations used in this thesis are two-dimensional. This modelling choice is suitable for studying vertical track-profile responses and for keeping the simulation framework computationally manageable. However, it does not capture the full three-dimensional behaviour of bridges, trains, tracks, foundations, and soil. Effects related to transverse bridge response, torsion, skew, lateral track behaviour, and asymmetric foundation behaviour may influence measured responses in real structures. Future work should therefore assess whether three-dimensional VBI models modify damage-sensitive features and the resulting performance of the ML model.

Scour is represented in this thesis mainly through a reduction in vertical support stiffness. This is a reasonable and controlled simplification for modelling the structural consequences of material loss beneath the foundation, although it does not capture all possible effects of scour. In real bridges, scour may also reduce lateral resistance, alter rotational restraint, expose foundations, change embedment conditions, or cause horizontal displacement. The lateral effects of scour were not considered in the adopted scour modelling. Including these effects could change the simulated bridge response and, consequently, the extracted ML features and classification results.

Detailed soil–structure interaction was not explicitly modelled. The adopted representation uses simplified assumptions about support stiffness to describe the structural consequences of scour. However, the actual bridge response depends on the interaction among the foundation, the surrounding soil, the superstructure, the track, and the moving vehicle. More advanced soil–structure interaction models could account for nonlinear soil behaviour, degradation of lateral resistance, loss of foundation embedment, and changes in rotational or translational restraint. These effects may be important for some bridge types and foundation conditions, and should be investigated in future studies.

No hydraulic scour modelling was carried out. The thesis does not aim to simulate the hydrodynamic mechanisms that generate scour, such as flow acceleration, sediment transport, horseshoe vortices, flood duration, or riverbed evolution. Instead, it focuses on the structural consequences of scour as observed through displacement-based track-profile measurements. Therefore, the framework should not be interpreted as a hydraulic model or as a direct scour-depth prediction tool. Its purpose is to detect changes in response that may indicate scour and to support targeted inspection decisions.

The relationship between stiffness loss and established scour-severity measures should be further clarified. In this thesis, scour is introduced via stiffness reduction, but practical scour assessment often focuses on quantities such as scour depth, foundation exposure, or loss of embedment. Future research could review geotechnical evidence on how support stiffness typically changes with increasing scour depth and foundation exposure. The stiffness-loss levels used in the numerical framework could then be mapped more explicitly onto these practical measures.

The temporal resolution of the batch-based decision process remains limited by the batch sizes evaluated in this thesis. The adopted windows of 7, 15, 31, and 61 crossings provide stable decisions and reduce the effect of crossing-to-crossing variability. However, smaller windows, such as 1, 2, or 3 days, were not reported. This is relevant because scour can develop rapidly during extreme hydraulic events. The framework can be adapted to shorter decision windows by adjusting the sliding window length. Still, the resulting trade-off between faster warning and increased false-alarm sensitivity should be quantified in future studies.

This thesis reports classification performance using a fixed decision threshold of 0.5. In other words, an output score or probability of 0.5 or higher is labelled as “scour present”, while values below 0.5 are labelled as “healthy”. Using a single threshold supports consistent, like-for-like comparison across algorithms, sensor configurations, and batch sizes. However, ROC curves and threshold-sweeping analyses were not included because the primary focus was on discrete decision performance at the selected operating point. In practice, lowering the threshold would increase sensitivity and reduce missed scour cases, but it would also increase false alarms. Raising the threshold would reduce false alarms, but it could increase missed detections. Therefore, threshold selection and ROC-based operating-point optimisation should be addressed in future deployment-oriented work, in line with asset-owner risk tolerance and intervention costs.

The adopted 5 per cent scour threshold should be considered a modelling and classification assumption rather than a universal engineering threshold. In this thesis, the 5 per cent threshold is supported by

the methodology stage and the sensitivity of the proposed approach to low scour levels. However, the practical meaning of this threshold depends on the foundation type, soil conditions, embedment, bridge geometry, hydraulic exposure, and the asset owner's acceptable risk level. Future work should therefore refine this threshold using field observations, inspection records, and geotechnical evidence linking stiffness loss, foundation exposure, and scour depth.

This thesis investigates the local scour around shallow foundations of simply supported bridges. The developed drive-by scour detection method could be tested on bridges with piled foundations, where scour may modify the foundation response in ways that differ from the simplified support-stiffness representation used in the presented methodology. Scour identification through indirect monitoring could also be explored for continuous bridges and non-ballasted bridge systems, where the structural response and track–bridge interaction may differ from the cases considered in this thesis.

The influence of realistic operating conditions should be examined in more detail. This includes speed restrictions and varying train speeds near bridges, since acceleration or deceleration may alter the measured response and complicate inference. Future work could also evaluate alternative sensing options alongside RILA track profiles, including axle-box accelerometers and multiple instrumented axles. This would allow different drive-by scour detection strategies to be compared under consistent conditions.

Future research could also extend the developed scour detection approach to high-speed railways. Acquisition performance should be evaluated at higher speeds, since increased velocity can alter the frequency content and amplitude of the measured response and impose stricter requirements on sensor mounting, durability, and on-vehicle data synchronisation. This may motivate the development of dedicated sensing and mounting solutions tailored to high-speed operation.

The thesis does not investigate advanced deep learning or generative learning models. The adopted framework relies on ANN, RF, and SVM classifiers trained on explicitly engineered sets of DWT- and ARX/Markov-based features. This choice improves interpretability and keeps the modelling pipeline transparent, but it also means that the damage-sensitive representation is largely defined before model training. More advanced models, such as convolutional neural networks, recurrent or transformer-based models, deep autoencoders, variational autoencoders, adversarial autoencoders, and generative adversarial networks, could be explored in future work. Such models may learn features directly from raw or weakly processed monitoring signals, reducing the need for manually defined feature engineering and feature-selection stages. Generative models may also help augment rare damaged-state data and reduce class imbalance. Nevertheless, these methods would require larger, more representative datasets, careful

assessment of simulation-to-field transfer, and rigorous validation to ensure that the models learn scour-sensitive behaviour rather than artefacts of the simulated dataset.

The developed ML models could also be trained with external context data, including precipitation and forecast information, temperature, and independent displacement measurements from sources such as InSAR, where available. InSAR is a remote-sensing technique that tracks surface displacement over time using repeated satellite imagery. Combining these data sources with drive-by monitoring could make the models more realistic by incorporating climatic information and independent deformation indicators.

Cross-network transfer should be investigated in future work. In this setting, models trained in one operating context are adapted to another through light calibration or fine-tuning. This could include transfer across regions with contrasting hydro-climatic exposure, such as a drier corridor, a moderate-rainfall corridor, and a very wet corridor. Such transfer studies should explicitly examine how Type-A and Type-B feature baselines shift with track condition, vehicle fleet, and bridge typology, and what minimum updating is needed to maintain reliable detection and localisation.

The proposed framework should be understood as a continuous monitoring and early-warning approach. It does not aim to estimate scour depth directly. Its purpose is to detect changes in the measured track-profile response as new monitoring data are continuously incorporated into the models, and to provide bridge- and support-level warnings that help asset owners prioritise inspections. Therefore, the method is most suitable as part of a wider asset-management workflow, where model outputs are combined with engineering judgement, site knowledge, hydraulic information, and targeted field verification.

REFERENCES

Abdu, D. M., Wei, G., & Yang, W. (2023). Assessment of railway bridge pier settlement based on train acceleration response using machine learning algorithms. *Structures, 52*, 598–608. https://doi.org/10.1016/j.istruc.2023.03.167.

Aggarwal, C. C. (2018). *Neural networks and deep learning: A textbook*. Springer. https://doi.org/10.1007/978-3-319-94463-0.

Ahmad, N., Bihs, H., Kamath, A., & Arntsen, Ø. A. (2015). Three-dimensional CFD modeling of wave scour around a side-by-side and triangular arrangement of piles with REEF3D. *Procedia Engineering, 116*, 683–690. https://doi.org/10.1016/j.proeng.2015.08.355.

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control, 19*(6), 716–723. https://doi.org/10.1109/TAC.1974.1100705.

Aldwaik, M., & Adeli, H. (2014). Advances in optimization of highrise building structures. *Structural and Multidisciplinary Optimization, 50*(6), 899–919. https://doi.org/10.1007/s00158-014-1087-1.

American Society of Civil Engineers. (2000). *Prestandard and commentary for the seismic rehabilitation of buildings* (FEMA 356). Federal Emergency Management Agency.

Amini, A., Melville, B. W., & Ali, T. M. (2014). Local scour at piled bridge piers including an examination of the superposition method. *Canadian Journal of Civil Engineering, 41*(5), 461–471. https://doi.org/10.1139/cjce-2011-0389.

Amini, A., & Mohammad, T. A. (2017). Local scour prediction around piers with complex geometry. *Marine Georesources and Geotechnology, 35*, 857–864. https://doi.org/10.1080/1064119X.2017.1310242.

Anastasia, S., García-Macías, E., Ubertini, F., Gattulli, V., & Ivorra, S. (2023). Damage identification of railway bridges through temporal autoregressive modeling. *Sensors, 23*(21), 8830. https://doi.org/10.3390/s23218830.

Anghel, A., Papandreou, N., Parnell, T., De Palma, A., & Pozidis, H. (2018). Benchmarking and optimization of gradient boosting decision tree algorithms. https://arxiv.org/abs/1809.04559.

Arneson, L. A., Zevenbergen, L. W., Lagasse, P. F., & Clopper, P. E. (2012). *Evaluating scour at bridges* (Report No. FHWA-HIF-12-003, HEC-18). U.S. Department of Transportation, Federal Highway Administration. https://rosap.ntl.bts.gov/view/dot/50281.

Ataie-Ashtiani, B., Baratian-Ghorghi, Z., & Beheshti, A. A. (2010). Experimental investigation of clear-water local scour of compound piers. *Journal of Hydraulic Engineering, 136*, 343–351. https://doi.org/10.1061/(ASCE)HY.1943-7900.0000181.

Azhari, F., Tom, C., Benassini, J., Loh, K. J., & Bombardelli, F. A. (2014). Design and characterization of a piezoelectric sensor for monitoring scour hole evolution. In *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2014* (Vol. 9061, pp. 301–309). SPIE. https://doi.org/10.1117/12.2045561.

Azhari, F., & Loh, K. J. (2017). Laboratory validation of buried piezoelectric scour sensing rods. *Structural Control and Health Monitoring, 24*(7), e1969. https://doi.org/10.1002/stc.1969.

Babcock, B., Babu, S., Datar, M., Motwani, R., & Widom, J. (2002). *Models and issues in data stream systems*. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '02)* (pp. 1–16). Association for Computing Machinery.

Bao, X., Li, Y., Li, J., Shi, R., & Ding, X. (2021). Prediction of train arrival delay using hybrid ELM-PSO approach. *Journal of Advanced Transportation, 2021*, Article 7763126. https://doi.org/10.1155/2021/7763126.

Beheshti, A. A., & Ataie-Ashtiani, B. (2010). Experimental study of three-dimensional flow field around a complex bridge pier. *Journal of Engineering Mechanics, 136*(2), 143–154. https://doi.org/10.1061/(ASCE)EM.1943-7889.0000073.

Beheshti, A. A., & Ataie-Ashtiani, B. (2016). Scour hole influence on turbulent flow field around complex bridge piers. *Flow, Turbulence and Combustion, 97*(2), 451–474. https://doi.org/10.1007/s10494-016-9707-8.

Benedict, S. T., & Caldwell, A. W. (2014). *A pier-scour database: 2,427 field and laboratory measurements of pier scour* (Data Series 845). U.S. Geological Survey. https://doi.org/10.3133/ds845.

Berggren, E. (2009). *Railway track stiffness: Dynamic measurements and evaluation for efficient maintenance* [Doctoral dissertation, KTH Royal Institute of Technology]. KTH.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research, 13*, 281–305.

Bernardini, L., Carnevale, M., Somaschini, C., Matsuoka, K., & Collina, A. (2020). A numerical investigation of new algorithms for the drive-by method in railway bridge monitoring. In *Proceedings of EURODYN 2020: XI International Conference on Structural Dynamics* (pp. 1033–1043). European Association for Structural Dynamics.

Berrar, D. (2018). Cross-validation. In R. Dubitzky, M. Granzow, & D. Berrar (Eds.), *Encyclopedia of Bioinformatics and Computational Biology* (Vol. 1, pp. 542–545). Elsevier. https://doi.org/10.1016/B978-0-12-809633-8.20349-X.

Bhagat, M., & Bakariya, B. (2025). A comprehensive review of cross-validation techniques in machine learning. *International Journal on Science and Technology, 16*(1), 1–13.

Bihs, H., & Olsen, N. R. B. (2011). Numerical modeling of abutment scour with the focus on the incipient motion on sloping beds. *Journal of Hydraulic Engineering, 137*(12), 1287–1292. https://doi.org/10.1061/(ASCE)HY.1943-7900.0000401.

Bookstein, A., Kulyukin, V. A., & Raita, T. (2002). Generalized Hamming distance. *Information Retrieval, 5*, 353–375.

Botta, V., Louppe, G., Geurts, P., & Wehenkel, L. (2014). Exploiting SNP correlations within random forest for genome-wide association studies. *PLOS ONE, 9*(4), e93379. https://doi.org/10.1371/journal.pone.0093379.

Botev, Z. I., Kroese, D. P., Rubinstein, R. Y., & L'Ecuyer, P. (2013). The cross-entropy method for optimization. In C. R. Rao & V. Govindaraju (Eds.), *Handbook of statistics* (Vol. 31, pp. 35–41). Elsevier. https://doi.org/10.1016/B978-0-444-53859-8.00003-5.

Boujia, N., Schmidt, F., Chevalier, C., Siegert, D., & Pham Van Bang, D. (2021). Using rocking frequencies of bridge piers for scour monitoring. *Structural Engineering International, 31*(2), 286–294. https://doi.org/10.1080/10168664.2020.1768811.

Boulesteix, A.-L. (2006). Optimal classifier selection and negative bias in error rate estimation by cross-validation. *BMC Bioinformatics, 7,* 91. https://doi.org/10.1186/1471-2105-7-91.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5-32. https://doi.org/10.1023/A:1010933404324.

Bridge collapse shows dangers of river scour. (2023, July 11). *Rail Magazine*. https://www.railmagazine.com/news/network/2023/07/11/bridge-collapse-shows-dangers-of-river-scour.

Brincker, R., Zhang, L., & Andersen, P. (2001). Modal identification of output-only systems using frequency domain decomposition. *Smart Materials and Structures, 10*(3), 441–445. https://doi.org/10.1088/0964-1726/10/3/303.

Calderon Hurtado, A., Kaur, K., Makki Alamdari, M., Atroshchenko, E., Chang, K. C., & Kim, C.-W. (2023). Unsupervised learning-based framework for indirect structural health monitoring using adversarial autoencoder. *Journal of Sound and Vibration, 550*, 117598. https://doi.org/10.1016/j.jsv.2023.117598.

Campbell, K. E., Ruffell, A., Pringle, J., Hughes, D., Taylor, S., & Devlin, B. (2021). Bridge foundation river scour and infill characterisation using water-penetrating radar. *Remote Sensing, 13*, 2542. https://doi.org/10.3390/rs13132542.

Cantero, D. (2022). TTB-2D: Train–track–bridge interaction simulation tool for Matlab. *SoftwareX, 20*(1), 101253. https://doi.org/10.1016/j.softx.2022.101253.

CEN. (2019). *Railway applications—Track—Track geometry quality—Part 1: Characterisation of track geometry (EN 13848-1:2019).* CEN.

Center for Disaster Philanthropy. (2024). *2024 U.S. floods*. https://disasterphilanthropy.org/disasters/2024-us-floods/.

Centre for Ecology & Hydrology. (2016). *The winter floods of 2015/2016: A review*. https://www.ceh.ac.uk/sites/default/files/2015-2016%20Winter%20Floods%20report%20Low%20Res.pdf.

Černá, D. (2019). *Discrete wavelet transform, wavelets, and wavelet basis*. Retrieved August 28, 2025, from https://kma.fp.tul.cz/images/stories/files/cerna/wavelets2019/Talk1_2019.pdf

Cerna, M., & Harvey, A. F. (2000). *The fundamentals of FFT-based signal analysis and measurement* (Application Note 041). National Instruments.

Chandrasekaran, S. (2019). Structural health monitoring: An overview. In *Structural health monitoring with application to offshore structures* (pp. 1–50). World Scientific. https://doi.org/10.1142/9789811201097_0001.

Chang, W. Y. (2004). *An experimental study of local scour at bridge piers in cohesive soils* [Master's thesis, Colorado State University].

Charfi, I., Miteran, J., Dubois, J., Atri, M., & Tourki, R. (2012). Definition and performance evaluation of a robust SVM based fall detection solution. In *2012 Eighth International Conference on Signal Image Technology and Internet-Based Systems* (pp. 218–224). IEEE. https://doi.org/10.1109/SITIS.2012.37.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

Chen, Y., Tang, F., Li, Z., Chen, G., & Tang, Y. (2018). Bridge scour monitoring using smart rocks based on magnetic field interference. *Smart Materials and Structures, 27*(8), 085012. https://doi.org/10.1088/1361-665X/aac996.

Cheng, M. Y., Cao, M. T., & Wu, Y. W. (2015). Predicting equilibrium scour depth at bridge piers using evolutionary radial basis function neural network. *Journal of Computing in Civil Engineering, 29*, 04014070. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000415.

Cheng, M. Y., Liao, K. W., Chiu, Y. F., Wu, Y. W., Yeh, S. H., & Lin, T. C. (2022). Automated mobile vibration measurement and signal analysis for bridge scour prevention and warning. *Automation in Construction, 134*, 104063. https://doi.org/10.1016/j.autcon.2021.104063.

Christodoulou, S. (2005). Ant colony optimization in construction scheduling. In *Proceedings of the International Conference on Computing in Civil Engineering* (pp. 1–11). Cancun, Mexico.

Choi, J. (2014). *Qualitative analysis for dynamic behavior of railway ballasted track* [Doctoral dissertation, Technische Universität Berlin]. DepositOnce. https://doi.org/10.14279/depositonce-3921.

Chopra, A. K. (2007). *Dynamics of structures* (4th ed.). Prentice Hall International Series in Civil Engineering and Engineering Mechanics.

Chui, C. K. (1997). What are wavelets? In *Wavelets: A mathematical tool for signal analysis* (pp. 1–33). SIAM.

Claesen, M., & De Moor, B. (2015). Hyperparameter search in machine learning. *arXiv* preprint arXiv:1502.02127. https://arxiv.org/abs/1502.02127.

Cohen, J. (1988). *The t test for means*. In *Statistical power analysis for the behavioral sciences* (2nd ed., pp. 19–74). Hillsdale, NJ: Lawrence Erlbaum Associates.

Coleman, S. E. (2005). Clearwater local scour at complex piers. *Journal of Hydraulic Engineering, 131*(4), 330–334. https://doi.org/10.1061/(ASCE)0733-9429(2005)131:4(330).

Cong, S., Jia, Y., & Deng, K. (2010). Particle swarm and ant colony algorithms and their applications in Chinese traveling salesman problem. In P. Korosec (Ed.), New achievements in evolutionary computation (pp. 298–302). IntechOpen.

Corbally, R., & Malekjafarian, A. (2024, August 29–30). Machine learning applications for drive-by bridge monitoring [Paper presentation]. Civil Engineering Research in Ireland 2024 (CERI2024), Dublin, Ireland.

Corbally, R., Ghiasi, R., & Malekjafarian, A. (2025). A novel wrapper-based feature selection method for drive-by bridge anomaly detection using knowledge transfer. *Advances in Engineering Software, 210*, 104003. https://doi.org/10.1016/j.advengsoft.2025.104003.

Correia, A. G., Roshan, M. J., Moldovan, I. D., Tinoco, J., Owusu-Ansah, D., Tola, S., & Azenha, M. (2025). Advancing transportation geotechnics through digital technologies. *Soils and Rocks, 48*, e2025007525. https://doi.org/10.28927/SR.2025.007525.

Cortez, P., & Embrechts, M. J. (2013). Using sensitivity analysis and visualization techniques to open black box data mining models. *Information Sciences, 225*, 1–17. https://doi.org/10.1016/j.ins.2012.10.039

Crotti, G., & Manzoni, S. (2025). In-field assessment of bridge pier scour by means of fiber optic sensors. *Measurement*, *242*, 116030. https://doi.org/10.1016/j.measurement.2024.112345.

Dahari, M. D. (2009). *System identification theory (Chapter 3). In System identification and parametric estimation of inferential coriolis* (Doctoral dissertation, Universiti Teknologi PETRONAS). UTPedia. Retrieved August 9, 2025, from http://utpedia.utp.edu.my/id/eprint/2926/5/5-Chapter3.pdf.

Daneshvar, M. H., Gharighoran, A., Zareei, S. A., & Karamodin, A. (2021). Early damage detection under massive data via innovative hybrid methods: Application to a large-scale cable-stayed bridge. *Structure and Infrastructure Engineering, 17*(7), 902–920. https://doi.org/10.1080/15732479.2020.1777572.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197. https://doi.org/10.1109/4235.996017.

Deng, P., Yang, J. J., & Yee, T. (2024). Deep learning-based flood detection for bridge monitoring using accelerometer data. *Infrastructures, 9*(9), 140. https://doi.org/10.3390/infrastructures9090140.

de Souza, E. F., Bragança, C., Ribeiro, D., Bittencourt, T. N., & Carvalho, H. (2024). Drive-by damage detection methodology for high-speed railway bridges using sparse autoencoders. *Railway Engineering Science*. Advance online publication. https://doi.org/10.1007/s40534-024-00347-3.

Dey, S., & Barbhuiya, A. K. (2005). Time variation of scour at abutments. *Journal of Hydraulic Engineering, 131*, 11–23. https://doi.org/10.1061/(ASCE)0733-9429(2005)131:1(11).

Dhal, P., & Azad, C. (2022). A comprehensive survey on feature selection in the various fields of machine learning. *Applied Intelligence, 52*, 4543–4581. https://doi.org/10.1007/s10489-021-02628-2.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, *1*(4), 28–39. https://doi.org/10.1109/MCI.2006.329691.

Dong, H., Chen, F., Zhou, H., Guo, C., & Sun, Z. (2020). A prediction model for local scour depth around piers based on machine learning. In *IOP Conference Series: Earth and Environmental Science* (Vol. 525, No. 1, p. 012080). IOP Publishing. https://doi.org/10.1088/1755-1315/525/1/012080.

Dong, S., & Yang, J. (2022). On computing the hyperparameter of extreme learning machines: Algorithm and application to computational PDEs, and comparison with classical and high-order finite elements. *Journal of Computational Physics, 463*, 111290. https://doi.org/10.1016/j.jcp.2022.111290

Dubey, S. R., Singh, S. K., & Chaudhuri, B. B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing, 503*, 92–108.

Dzięcioł, J., & Sas, W. (2024). Estimation of the coefficient of permeability as an example of the application of the random forest algorithm in civil engineering. *Archives of Civil Engineering, 70*(2), 119–134. https://doi.org/10.24425/ace.2024.149854.

Ebtehaj, I., Sattar, A. M., Bonakdari, H., & Zaji, A. H. (2017). Prediction of scour depth around bridge piers using self-adaptive extreme learning machine. *Journal of Hydroinformatics, 19*, 207–224. https://doi.org/10.2166/hydro.2016.078.

Ebtehaj, I., Bonakdari, H., Moradi, F., Gharabaghi, B., & Khozani, Z. S. (2018). An integrated framework of extreme learning machines for predicting scour at pile groups in clearwater conditions. *Coastal Engineering, 135*, 1–15. https://doi.org/10.1016/j.coastaleng.2018.02.007.

Ebtehaj, I., Bonakdari, H., Zaji, A. H., & Sharafi, H. (2019). Sensitivity analysis of parameters affecting scour depth around bridge piers based on the non-tuned, rapid extreme learning machine method. *Neural Computing and Applications, 31*, 9145–9156. https://doi.org/10.1007/s00521-018-3894-4.

Eini, N., Janizadeh, S., Bateni, S. M., Jun, C., Heggy, E., & Kirs, M. (2024). Predicting equilibrium scour depth around non-circular bridge piers with shallow foundations using hybrid explainable machine learning methods. *Results in Engineering, 24*, 103492. https://doi.org/10.1016/j.rineng.2024.103492.

Elsaid, A., & Seracino, R. (2014). Rapid assessment of foundation scour using the dynamic features of the bridge superstructure. *Construction and Building Materials, 50*, 42–49. https://doi.org/10.1016/j.conbuildmat.2013.09.008.

Ettema, R., Melville, B. W., & Constantinescu, G. (2011). *Evaluation of bridge scour research: Pier scour processes and predictions*. Transportation Research Board.

Faber, M. H. (Ed.). (2008). *Risk assessment in engineering: Principles, system representation & risk criteria*. Joint Committee on Structural Safety (JCSS). https://www.jcss-lc.org/publications/raie/01_jcss_riskassessment.pdf.

Fan, R.-E., Chen, P.-H., & Lin, C.-J. (2005). Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research, 6*, 1889–1918. https://www.jmlr.org/papers/volume6/fan05a/fan05a.pdf.

Fang, Y., Xing, J., & Zhang, Y. (2024). Bridge damage identification with improved HHT and PSD sensitivity method based on vehicle response. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, *46*(4), 1–17. https://doi.org/10.1007/s40430-024-04783-4.

Farooq, M., Azhari, F., & Banthia, N. (2025). Photoelectric sensors for wireless monitoring of bridge scour. *Structure and Infrastructure Engineering*. https://doi.org/10.1080/15732479.2023.2261434.

Fattahi, H., & Hasanipanah, M. (2021). An integrated approach of ANFIS-grasshopper optimization algorithm to approximate flyrock distance in mine blasting. *Engineering Computations, 38*, 2619–2631. https://doi.org/10.1108/EC-03-2020-0187.

Federal Railroad Administration. (1983). *Analytical Descriptions of Track Geometry Variations* (DOT/FRA/ORD-83/03, Appendix B: Power Spectral Densities (PSD's), Vol. II: Appendices, p. B-1). U.S. Department of Transportation, Office of Research and Development.

Feldman, M. (2011). Hilbert transform in vibration analysis. *Mechanical Systems and Signal Processing, 25*(3), 735–802. https://doi.org/10.1016/j.ymssp.2010.07.018.

Feng, D., & Feng, M. Q. (2016). Output-only damage detection using vehicle-induced displacement response and mode shape curvature index. *Structural Control and Health Monitoring, 23*(8), 1088–1107. https://doi.org/10.1002/stc.1829.

Fernandes, T. M., Lopez, R. H., & Ribeiro, D. R. F. (2024). Drive-by scour damage detection in railway bridges using deep autoencoder and different sensor placement strategies. *Journal of Civil Structural Health Monitoring, 14*, 1895–1916. https://doi.org/10.1007/s13349-024-00738-y.

Fitzgerald, P. C., Malekjafarian, A., Bhowmik, B., Prendergast, L. J., Cahill, P., Kim, C. W., & O'Brien, E. J. (2019a). Scour damage detection and structural health monitoring of a laboratory-scaled bridge using a vibration energy harvesting device. *Sensors, 19*(11), 2572. https://doi.org/10.3390/s19112572.

Fitzgerald, P. C., Malekjafarian, A., Cantero, D., O'Brien, E. J., & Prendergast, L. J. (2019b). Drive-by scour monitoring of railway bridges using a wavelet-based approach. *Engineering Structures, 191*, 1–11. https://doi.org/10.1016/j.engstruct.2019.04.046.

Flayyih, S. S., Jasim, F. H., Nafe'e, O. T., & Jamel, A. A. J. (2025). An artificial intelligence-driven evaluation of scour depth around bridge piers. *Engineering, Technology & Applied Science Research, 15*(5), 26310-26316. https://doi.org/10.48084/etasr.12240.

Froehlich, D. C. (1988). Analysis of onsite measurements of scour at piers. In *Hydraulic Engineering: Proceedings of the 1988 National Conference on Hydraulic Engineering* (pp. 534–539). American Society of Civil Engineers.

Fugro. (2020). *Fugro's RILA® technology wins Network Rail Western route track survey contract*. Fugro. Retrieved April 9, 2025, from https://www.fugro.com/news/business-news/2020/fugros-rila-technology-wins-network-rail-western-route-track-survey-contract.

Funderburk, M. L., Huang, S. K., Loh, C. H., & Loh, K. J. (2019). Densely distributed and real-time scour hole monitoring using piezoelectric rod sensors. *Advances in Structural Engineering, 22*(15), 3395–3411. https://doi.org/10.1177/1369433219838394.

Gao, Y. (2005). *Structural health monitoring strategies for smart sensor networks* [Microform edition]. University of Illinois at Urbana-Champaign, Champaign, IL, United States.

Geyer, S., Papaioannou, I., & Straub, D. (2019). Cross entropy-based importance sampling using Gaussian densities revisited. *Structural Safety, 76*, 15–27. https://doi.org/10.1016/j.strusafe.2018.07.001.

Ghali, A., Neville, A. M., & Brown, T. G. (2017). *Structural analysis: A unified classical and matrix approach* (7th ed.). CRC Press.

Ghodsi, H., & Khanjani, M. J. (2020). Application of improved GMDH models to predict local scour depth at complex bridge piers. *Civil Engineering Journal, 6*, 69–84. https://doi.org/10.28991/cej-2020-03091466.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (pp. 249–256).

Gonzalez, A., Covian, E., Casero, M., & Cooper, J. (2013). Experimental testing of a cross-entropy algorithm to detect damage. *Key Engineering Materials, 569–570*, 1170–1177. https://doi.org/10.4028/www.scientific.net/KEM.569-570.1170.

Gasparotti, P., & Lopez, R. H. (2024). Convolutional neural network for highway bridge indirect structural health monitoring. In *Proceedings of the 9th International Symposium on Solid Mechanics*. Associação Brasileira de Engenharia e Ciências Mecânicas (ABCM).

Grossmann, A., Kronland-Martinet, R., & Morlet, J. (1990). Reading and understanding continuous wavelet transforms. In *Wavelets* (pp. 2–20). Springer. https://doi.org/10.1007/978-3-642-75988-8_2.

Guan, D., Chiew, Y. M., Wei, M., & Hsieh, S. C. (2019). Characterization of horseshoe vortex in a developing scour hole at a cylindrical bridge pier. *International Journal of Sediment Research, 34*, 118–124. https://doi.org/10.1016/j.ijsrc.2018.07.001.

Guan, X., Wang, Y., & He, J. A. (2017). Probabilistic damage identification method for shear structure components based on cross-entropy optimizations. *Entropy, 19*(1), 27. https://doi.org/10.3390/e19010027.

Hachem, S., Bourquin, F., & Siegert, D. (2023). Scour damage detection of bridge piers using the vibration-based method: Numerical study on a steel-concrete bridge in France. *Structural Health Monitoring 2023*. https://doi.org/10.12783/shm2023/36735.

Hajializadeh, D. (2022). Deep-learning-based drive-by damage detection system for railway bridges. *Infrastructures, 7*(6), 84. https://doi.org/10.3390/infrastructures7060084.

Han, M., & Zhao, Y. (2011). Dynamic control model of BOF steelmaking process based on ANFIS and robust relevance vector machine. *Expert Systems with Applications, 38*, 14786–14798. https://doi.org/10.1016/j.eswa.2011.05.075.

Hare, W., Nutini, J., & Tesfamariam, S. (2013). A survey of non-gradient optimization methods in structural engineering. *Advances in Engineering Software*, *59*, 19–28. https://doi.org/10.1016/j.advengsoft.2012.12.006.

Hashimoto, K., Shiotani, T., Mitsuya, H., & Chang, K. C. (2020). MEMS vibrational power generator for bridge slab and pier health monitoring. *Applied Sciences, 10*(23), 8258. https://doi.org/10.3390/app10238258.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (pp. 1026–1034).

Hoang, N. D., Liao, K. W., & Tran, X. L. (2018). Estimation of scour depth at bridges with complex pier foundations using support vector regression integrated with feature selection. *Journal of Civil Structural Health Monitoring, 8*, 431–442. https://doi.org/10.1007/s13349-018-0290-9.

Holland, D. E., Olesen, R. J., & Bevins, J. E. (2021). Multi-objective genetic algorithm optimization of a directionally sensitive radiation detection system using a surrogate transport model. *Engineering*

Applications of Artificial Intelligence, 104, Article 104357. https://doi.org/10.1016/j.engappai.2021.104357.

Holnbeck, S. R. (2011). *Investigation of pier scour in coarse-bed streams in Montana, 2001 through 2007* (Scientific Investigations Report 2011–5054). U.S. Geological Survey.

Hou, S., Jiao, D., Dong, B., Wang, H., & Wu, G. (2022). Underwater inspection of bridge substructures using sonar and deep convolutional network. *Advanced Engineering Informatics, 52*, 101545. https://doi.org/10.1016/j.aei.2022.101545.

Hou, Z., Noori, M., & St. Amand, R. (2000). Wavelet-based approach for structural damage detection. *Journal of Engineering Mechanics, 126*(7), 677–685. https://doi.org/10.1061/(ASCE)0733-9399(2000)126:7(677).

Huang, Z., Yin, X., Liu, Y., & Tang, S. (2023). Bridge scour detection method based on Siamese neural networks under bridge-vehicle-wave interaction. *Ocean Engineering*, *290*, 116327. https://doi.org/10.1016/j.oceaneng.2023.116327.

Huber, E., Anders, B., & Huggenberger, P. (2018). Quantifying scour depth in a straightened gravel-bed river with ground-penetrating radar. In *Proceedings of the 17th International Conference on Ground Penetrating Radar* (pp. 1–4). IEEE. https://doi.org/10.1109/ICGPR.2018.8441569.

Huynh, N. T., Phan, A. T. T., Trieu, T. T., Nguyen, H. H. D., & Nguyen, T. N. (2025). A CNN deep learning approach to scour depth estimation around complex bridge piers in steady flow environments. *Advances in Bridge Engineering, 6*(1), 1-29. https://doi.org/10.1186/s43251-025-00170-8.

Jafari-Asl, J., Ben Seghier, M. E. A., Ohadi, S., Dong, Y., & Plevris, V. (2021). A comparative study on the efficiency of reliability methods for the probabilistic analysis of local scour at a bridge pier in clay–sand–mixed sediments. *Modelling, 2*(1), 63–77. https://doi.org/10.3390/modelling2010004.

Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374*(2065), 20150202. https://doi.org/10.1098/rsta.2015.0202.

Johnson, P. A. (1992). Reliability-based pier scour engineering. *Journal of Hydraulic Engineering, 118*, 1344–1358. https://doi.org/10.1061/(ASCE)0733-9429(1992)118:10(1344)

Jonkman, S. N., Steenbergen, R. D. J. M., Morales-Nápoles, O., Vrouwenvelder, A. C. W. M., & Vrijling, J. K. (2015). *Probabilistic design: Risk and reliability analysis in civil engineering (Lecture notes CIE4130).* Delft University of Technology. https://repository.tudelft.nl/record/uuid:e53b8dca-a0db-4433-b9f9-e190a507f99f.

Jornal de Notícias. (2023). Tragédia da ponte de Entre-os-Rios causou 59 mortos. *Jornal de Notícias*. https://www.jn.pt/pais/artigo/tragedia-da-ponte-de-entre-os-rios-causou-59-mortos/16247814.

Kadono, T., Okazaki, S., Kabeyama, Y., & Matsui, T. (2020). Effect of angle between pier and center of river flow on local scouring around the bridge pier. *Water, 12*(11), 1–12. https://doi.org/10.3390/w12113192.

Kallias, A. N., & Imam, B. (2016). Probabilistic assessment of local scour in bridge piers under changing environmental conditions. Structure and Infrastructure Engineering, 12, 1228–1241. https://doi.org/10.1080/15732479.2015.1102295.

Kaveh, A., & Talatahari, S. (2010). An improved ant colony optimization for constrained engineering design problems. *Engineering Computations, 27*(1), 155–182. https://doi.org/10.1108/02644401011008528.

Keenahan, J., Ren, Y., & O'Brien, E. J. (2020). Determination of road profile using multiple passing vehicle measurements. *Structure and Infrastructure Engineering, 16*(9), 1262–1275. https://doi.org/10.1080/15732479.2019.1703757.

Khan, M. A., McCrum, D. P., Prendergast, L. J., O'Brien, E. J., Fitzgerald, P. C., & Kim, C. W. (2021). Laboratory investigation of a bridge scour monitoring method using decentralized modal analysis. *Structural Health Monitoring, 20*(6), 3327–3341. https://doi.org/10.1177/1475921720985122.

Khosravi, K., Sheikh Khozani, Z., & Mao, L. (2021). A comparison between advanced hybrid machine learning algorithms and empirical equations applied to abutment scour depth prediction. *Journal of Hydrology*, *596*, 126100. https://doi.org/10.1016/j.jhydrol.2021.126100.

Kim, I., Fard, M. Y., & Chattopadhyay, A. (2015). Investigation of a bridge pier scour prediction model for safe design and inspection. *Journal of Bridge Engineering, 20*, 04014088. https://doi.org/10.1061/(ASCE)BE.1943-5592.0000714.

Kirkil, G., Constantinescu, G., & Ettema, R. (2009). Detached eddy simulation investigation of turbulence at a circular pier with scour hole. *Journal of Hydraulic Engineering, 135*(10), 888–901. https://doi.org/10.1061/(ASCE)HY.1943-7900.0000101.

Klinga, J. V., & Alipour, A. (2015). Assessment of structural integrity of bridges under extreme scour conditions. *Engineering Structures, 82*, 55–71. https://doi.org/10.1016/j.engstruct.2014.07.021.

Koo, K. Y., Brownjohn, J. M. W., List, D. I., & Cole, R. (2013). Structural health monitoring of the Tamar suspension bridge. *Structural Control and Health Monitoring*, *20*(4), 609–625. https://doi.org/10.1002/stc.1481.

Kothyari, U. C. (1989). *Scour around bridge piers* (Doctoral dissertation). University of Roorkee, Roorkee, India.

Kothyari, U. C., Hager, W. H., & Oliveto, G. (2007). Generalized approach for clear-water scour at bridge foundation elements. *Journal of Hydraulic Engineering, 133*(11), 1229–1240. https://doi.org/10.1061/(ASCE)0733-9429(2007)133:11(1229).

Krogh, A., & Hertz, J. A. (1991). A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems* (Vol. 4, pp. 950–957).

Kumar, S., Oliveto, G., Deshpande, V., Agarwal, M., & Rathnayake, U. (2024). Forecasting of time-dependent scour depth based on bagging and boosting machine learning approaches. *Journal of Hydroinformatics, 26*(8), 1906-1928. https://doi.org/10.2166/hydro.2024.047.

Lan, Y., Zhang, Y., & Lin, W. (2023). Diagnosis algorithms for indirect bridge health monitoring via an optimized AdaBoost-linear SVM. *Engineering Structures, 275*, 115239. https://doi.org/10.1016/j.engstruct.2022.115239.

Landau, D. P., & Binder, K. (2014). *A guide to Monte Carlo simulations in statistical physics* (4th ed.). Cambridge University Press.

Laursen, E. M., & Toch, A. (1956). *Scour around bridge piers and abutments* (Vol. 4). Iowa Highway Research Board.

Le, T. P., & Paultre, P. (2013). Modal identification based on the time–frequency domain decomposition of unknown-input dynamic tests. *International Journal of Mechanical Sciences, 71*, 41–50. https://doi.org/10.1016/j.ijmecsci.2013.02.009.

Lee, F.-Z., Lai, J.-S., Lin, Y.-B., & Chang, K.-C. (2025). Monitoring and simulation of bridge pier scour and deposition processes in flood events. *Environmental Earth Sciences, 84*(6), 338. https://doi.org/10.1007/s12665-025-12196-2.

Ley, C., Martin, R. K., Pareek, A., Groll, A., Seil, R., & Tischer, T. (2022). Machine learning and conventional statistics: Making sense of the differences. *Knee Surgery, Sports Traumatology, Arthroscopy, 30*(1), 1–5. https://doi.org/10.1007/s00167-022-06896-6.

Li, J., Cheng, J. H., Shi, J. Y., & Huang, F. (2012). Brief introduction of back propagation (BP) neural network algorithm and its improvement. In *Advances in computer science and information engineering* (pp. 553–558). Springer. https://doi.org/10.1007/978-3-642-30223-7_86.

Li, P., Wang, Y., Hu, J., & Zhou, J. (2015). Sensors distribution optimization for impact localization using NSGA-II. *Sensor Review*, *35*(4), 409-418. https://doi.org/10.1108/SR-02-2015-0028.

Li, J., Zhu, X., Law, S.-S., & Samali, B. (2020). A two-step drive-by bridge damage detection using dual Kalman filter. *International Journal of Structural Stability and Dynamics*, 20(10), 2042006. https://doi.org/10.1142/S0219455420420067.

Liao, K. W., Muto, Y., & Lin, J. Y. (2018). Scour depth evaluation of a bridge with a complex pier foundation. *KSCE Journal of Civil Engineering, 22*, 2241–2255. https://doi.org/10.1007/s12205-017-0988-y.

Lin, T. K., & Chang, Y. S. (2017). Development of a real-time scour monitoring system for bridge safety evaluation. *Mechanical Systems and Signal Processing, 82*, 503–518. https://doi.org/10.1016/j.ymssp.2016.05.028.

Lin, J., Keogh, E., Lonardi, S., Lankford, J. P., & Nystrom, D. M. (2004, August). Visually mining and monitoring massive time series. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 460–469). ACM. https://doi.org/10.1145/1014052.1014104.

Liu, W., Zhou, W., & Li, H. (2022). Bridge scour estimation using unconstrained distributed fiber optic sensors. *Journal of Civil Structural Health Monitoring, 12*(4), 775–784. https://doi.org/10.1007/s13349-022-00553-1.

London Borough of Merton. (2024). *Bishopsford Bridge: Updates and information*. Retrieved August 3, 2025, from https://www.merton.gov.uk/streets-parking-transport/roadworks/bishopsford-bridge.

Lucero, J., & Reda Taha, M. M. (2005). A wavelet-aided fuzzy damage detection algorithm for structural health monitoring. In *Proceedings of the 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Palm Springs, CA, United States.

Lynch, J. P. (2007). An overview of wireless structural health monitoring for civil structures. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *365*(1851), 345–372. https://doi.org/10.1098/rsta.2006.1932.

Macky, G. H. (1990). Survey of roading expenditure due to scour (CR 90_09). Department of Scientific and Industrial Research, Hydrology Centre, Christchurch, New Zealand.

Madandoust, R., Bungey, J. H., & Ghavidel, R. (2012). Prediction of the concrete compressive strength by means of core testing using GMDH-type neural network and ANFIS models. *Computational Materials Science, 51*, 261–272. https://doi.org/10.1016/j.commatsci.2011.09.027.

Malekjafarian, A., & O'Brien, E. J. (2017). On the use of a passing vehicle for the estimation of bridge mode shapes. *Journal of Sound and Vibration, 397*, 77–91. https://doi.org/10.1016/j.jsv.2017.02.051.

Malekjafarian, A., Prendergast, L. J., & O'Brien, E. J. (2018, August 29–30). Detecting bridge scour using mode shapes derived from time-domain data. In *Proceedings of the Civil Engineering Research in Ireland 2018 (CERI2018) Conference*, Dublin, Ireland.

Malekjafarian, A., Kim, C.-W., O'Brien, E. J., Prendergast, L. J., Fitzgerald, P. C., & Nakajima, S. (2020a). Experimental demonstration of a mode shape-based scour-monitoring method for multispan bridges with shallow foundations. *Journal of Bridge Engineering, 27*(6), 04022032. https://doi.org/10.1061/(ASCE)BE.1943-5592.0001586.

Malekjafarian, A., Moloney, C., & Golpayegani, F. (2020b). Drive-by bridge health monitoring using multiple passes and machine learning. In *Proceedings of the European Workshop on Structural Health Monitoring* (pp. 695–703). Springer. https://doi.org/10.1007/978-3-030-64594-6_67.

Malekjafarian, A., Prendergast, L. J., & O'Brien, E. J. (2020c). Use of mode shape ratios for pier scour monitoring in two-span integral bridges under changing environmental conditions. *Canadian Journal of Civil Engineering*, *47*, 962–973. https://doi.org/10.1139/cjce-2018-0800.

Malena, M., Angelillo, M., Fortunato, A., de Felice, G., & Mascolo, I. (2021). Arch bridges subject to pier settlements: Continuous vs. piecewise rigid displacement methods. *Meccanica, 56*(4), 2487–2505. https://doi.org/10.1007/s11012-021-01397-1.

Maroni, A., Tubaldi, E., Ferguson, N., Tarantino, A., McDonald, H., & Zonta, D. (2020). Electromagnetic sensors for underwater scour monitoring. *Sensors, 20*(15), 4096. https://doi.org/10.3390/s20154096.

Marr, J. (2010). *Bridge scour monitoring technologies: Development of evaluation and selection protocols for application on river bridges in Minnesota* (No. MN/RC 2010-14). Minnesota Department of Transportation, Research Services Section.

Marsh, T., Kirby, C., Muchan, K., Barker, L., Henderson, E., & Hannaford, J. (2016). Hydrology of winter 2015/2016. In *The winter floods of 2015/2016 in the UK: A review* (pp. 15–17). Centre for Ecology and Hydrology.

Martin-Vide, J. P., Hidalgo, C., & Bateman, A. (1998). Local scour at piled bridge foundations. *Journal of Hydraulic Engineering, 124*(4), 439–444. https://doi.org/10.1061/(ASCE)0733-9429(1998)124:4(439).

Martinek, R., & Zidek, J. (2014). The real implementation of ANFIS channel equalizer on the system of software-defined radio. *IETE Journal of Research, 60*, 183–193. https://doi.org/10.1080/03772063.2014.913401.

Martinez, D., Malekjafarian, A., & O'Brien, E. (2020). Bridge flexural rigidity calculation using measured drive-by deflections. *Journal of Civil Structural Health Monitoring, 10*(5), 833–844. https://doi.org/10.1007/s13349-020-00419-y.

MathWorks. (2025a). *fitcnet: Train neural network classification model* (Statistics and Machine Learning Toolbox documentation). The MathWorks, Inc. https://www.mathworks.com/help/stats/fitcnet.html.

MathWorks. (2025b). *TreeBagger: Ensemble of bagged decision trees* (Statistics and Machine Learning Toolbox documentation). The MathWorks, Inc. https://www.mathworks.com/help/stats/treebagger.html.

MathWorks. (2025c). *fitcsvm: Train support vector machine (SVM) classifier for one-class and binary classification* (Statistics and Machine Learning Toolbox documentation). The MathWorks, Inc. https://www.mathworks.com/help/stats/fitcsvm.html.

Matsuoka, K., & Tanaka, H. (2023). Drive-by deflection estimation method for simple support bridges based on track irregularities measured on a traveling train. *Mechanical Systems and Signal Processing*, *182*, 109549. https://doi.org/10.1016/j.ymssp.2022.109549.

Maulud, D., & Abdulazeez, A. M. (2020). A review on linear regression comprehensive in machine learning. *Journal of Applied Science and Technology Trends, 1*, 140–147. https://doi.org/10.38094/jastt1137

McGeown, C., Hester, D., O'Brien, E. J., Kim, C. W., Fitzgerald, P., & Pakrashi, V. (2024). Condition monitoring of railway bridges using vehicle pitch to detect scour. *Sensors, 24*(5), 1684. https://doi.org/10.3390/s24051684.

Melville, B. W. (1997). Pier and abutment scour: Integrated approach. *Journal of Hydraulic Engineering*, *123*(2), 125–136. https://doi.org/10.1061/(ASCE)0733-9429(1997)123:2(125).

Melville, B. W., & Coleman, S. E. (2000). *Bridge scour*. Water Resources Publication.

Mendes, P. G. (n.d.). Hyper-parameter tuning using Bayesian optimization. Instituto Superior Técnico. https://web.ist.utl.pt/pedrogoncalomendes/papers/bo.pdf (Retrieved August 26, 2025).

Met Office. (2024). *Location-specific long-term averages*. https://www.metoffice.gov.uk/research/climate/maps-and-data/location-specific-long-term-averages.

Michalis, P., Tarantino, A., Tachtatzis, C., & Judd, M. D. (2015). Wireless monitoring of scour and re-deposited sediment evolution at bridge foundations based on soil electromagnetic properties. *Smart Materials and Structures, 24*(12), 125029. https://doi.org/10.1088/0964-1726/24/12/125029.

Minski, L., & Lopez, R. H. (2023). Drive-by Damage Detection In Railway Bridges Using 1d Convolutional Neural Networks. In *XLIV Ibero-Latin American Congress on Computational Methods in Engineering* (Vol. 5, No. 05).

Moscovich, A., & Rosset, S. (2022). On the cross-validation bias due to unsupervised preprocessing. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 84*(4), 1474–1497. https://doi.org/10.1093/jrsssb/qkac024.

Moss, R. J. (2020). Cross-entropy method variants for optimization. arXiv preprint arXiv:2009.09043. https://doi.org/10.48550/arXiv.2009.09043.

Mueller, D. S., & Wagner, C. R. (2005). *Field observations and evaluations of streambed scour at bridges* (No. FHWA-RD-03-052). United States, Federal Highway Administration, Office of Research, Development, and Technology.

Murguía, J. S., & Rosu, H. C. (2011). Discrete wavelet analyses for time series. In J. T. Olkkonen (Ed.), *Discrete wavelet transforms: Theory and applications* (p. 7). BoD–Books on Demand.

Murray, I. (2016). *Gaussian processes and kernels* (MLPR:w7c) [Lecture notes]. School of Informatics, University of Edinburgh. http://www.inf.ed.ac.uk/teaching/courses/mlpr/2016/ (Retrieved August 8, 2025).

Mutlu Sumer, B. (2007). Mathematical modeling of scour: A review. *Journal of Hydraulic Research, 45*, 723–735. https://doi.org/10.1080/00221686.2007.9521811.

Muzzammil, M. (2010). ANFIS approach to the scour depth prediction at a bridge abutment. *Journal of Hydroinformatics, 12*, 474–485. https://doi.org/10.2166/hydro.2010.004.

Muzzammil, M., Alam, J., Kumar, K., & Khalid, M. (2022). Reliability analysis of a complex pier against local scour. *Journal of the Institution of Engineers (India): Series A, 103*, 1237–1245. https://doi.org/10.1007/s40030-022-00699-1.

Nagayama, T., Spencer, B. F., Jr., & Rice, J. A. (2009). Autonomous decentralized structural health monitoring using smart sensors. *Structural Control and Health Monitoring, 16*(7–8), 842–859. https://doi.org/10.1002/stc.347.

Najafzadeh, M., & Azamathulla, H. M. (2015). Neuro-fuzzy GMDH to predict the scour pile groups due to waves. *Journal of Computing in Civil Engineering, 29*, 04014068. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000349.

Nassif, H., Ertekin, A. O., & Davis, J. (2002). *Evaluation of bridge scour monitoring methods*. United States Department of Transportation, Federal Highway Administration. https://cait.rutgers.edu/wp-content/uploads/2018/05/fhwa-nj-2003-009.pdf.

Network Rail. (2015a). *Existing general arrangement – West elevation (CP5 LNW Northern Structures Framework: WCM15 Eden Viaduct; 001 miles 0440 yards)* (Drawing No. 008-2005-JJA007882-UT1D, Rev. B01) [AutoCAD technical drawing]. Prepared by Hyder Consulting (UK) Ltd for Story Contracting. Unpublished internal drawing.

Network Rail. (2015b). *Infrastructure projects central: Completion report (Report No. 129038 WCM1/5 1m 0440 yds, Eden Viaduct).*

Network Rail. (2015c). *Proposed general arrangement – Plan* (CP5 LNW Northern Structures Framework: WCM15 Eden Viaduct; 001 miles 0440 yards) (Drawing No. 008-0021-UA007882-UT41D, Rev. B01) [AutoCAD technical drawing]. Prepared by Hyder Consulting (UK) Ltd for Story Contracting. Unpublished internal drawing.

Network Rail. (2015d). *Proposed general arrangement – East elevation* (CP5 LNW Northern Structures Framework: WCM15 Eden Viaduct; 001 miles 0440 yards) (Drawing No. 008-0022-UA007882-UT41D, Rev. B01) [AutoCAD technical drawing]. Prepared by Hyder Consulting (UK) Ltd for Story Contracting. Unpublished internal drawing.

Network Rail. (2020). *Eden Viaduct Main Lines Bridge detailed examination report: Underwater exam* (Exam ID 3100278). Lancs & Cumbria, UK.

Network Rail. (2022). *Carlisle Bridge: Final report* (Report No. 9221-197241-01 1.1). Utrecht, The Netherlands.

Ng, S. T., & Zhang, Y. (2008). Optimizing construction time and cost using an ant colony optimization approach. *Journal of Construction Engineering and Management, 134*(9), 721–728. https://doi.org/10.1061/(ASCE)0733-9364(2008)134:9(721)

Nordila, A., Ali, T., Faisal, A., & Badrunnisa, Y. (2014). Local scour at wide bridge piers. *International Journal of Engineering Research and Technology, 3*, 742–754.

Oben-Nyarko, K., & Ettema, R. (2011). Pier and abutment scour interaction. *Journal of Hydraulic Engineering, 137*, 1598–1605. https://doi.org/10.1061/(ASCE)HY.1943-7900.0000446.

O'Brien, E. J., Bowe, C., Quirke, P., & Cantero, D. (2017). Determination of longitudinal profile of railway track using vehicle-based inertial readings. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit, 231*(5), 518–534. https://doi.org/10.1177/09544097166649.

O'Brien, E. J., Malekjafarian, A., & Fitzgerald, P. C. (2018, August 29–30). Bridge scour detection using vehicle acceleration measurements. In *Proceedings of the Civil Engineering Research in Ireland 2018 Conference (CERI 2018)*. Dublin, Ireland.

O'Brien, E. J., McCrum, D. P., Khan, M. A., & Prendergast, L. J. (2021). Wavelet-based operating deflection shapes for locating scour-related stiffness losses in multi-span bridges. *Structure and Infrastructure Engineering, 19*(2), 238–253. https://doi.org/10.1080/15732479.2021.1937235.

Oliveto, G., & Hager, W. H. (2002). Temporal evolution of clear-water pier and abutment scour. *Journal of Hydraulic Engineering, 128*, 811–820. https://doi.org/10.1061/(ASCE)0733-9429(2002)128:9(811).

Oliveto, G., & Hager, W. H. (2005). Temporal evolution of clear-water pier and abutment scour. *Journal of Hydraulic Engineering, 131*(8), 650–661. https://doi.org/10.1061/(ASCE)0733-9429(2005)131:8(650).

Oneto, L., Bisio, F., Cambria, E., & Anguita, D. (2017). SLT-based ELM for big social data analysis. *Cognitive Computation, 9*, 259–274. https://doi.org/10.1007/s12559-017-9479-2.

Pal, M. (2019). Deep neural network-based pier scour modeling. *ISH Journal of Hydraulic Engineering, 28*, 80–85. https://doi.org/10.1080/09715010.2019.1621327.

Peng, L., Hu, J., Wang, Y., & Li, C. (2018). Sensor Distribution Optimization for Structural Impact Monitoring Based on NSGA-II and Wavelet Decomposition. *Sensors, 18*(12), 4264. https://doi.org/10.3390/s18124264.

Perry, C. A. (2000). *Significant floods in the United States during the 20th century—USGS measures a century of floods*. U.S. Department of the Interior, U.S. Geological Survey. https://pubs.usgs.gov/fs/2000/0024/report.pdf.

Perugini, E., & Tubaldi, E. (2024). Two pilot case studies for bridge-scour monitoring. *Open Research Europe*. https://doi.org/10.12688/openreseurope.15065.1.

Phan, M. Q., & Longman, R. W. (1996). Relationship between state-space and input–output models via observer Markov parameters. In *Dynamics and control of structures in space (DCSS 1996): WIT transactions on the built environment* (Vol. 19, pp. 185–200). WIT Press.

Piraei, R., Niazkar, M., Cislaghi, A., Afzali, S. H., & Mohammadi, A. (2025). Enhancing prediction of equilibrium scour depth around bridge piers using stacking machine learning models. *Earth Systems and Environment*, 1–21. https://doi.org/10.1007/s41748-025-00722-y.

Polikar, R., DePasquale, J., Mohammed, H. S., Brown, G., & Kuncheva, L. I. (2010). Learn++.MF: A random subspace approach for the missing feature problem. *Pattern Recognition, 43*(11), 3817–3832. https://doi.org/10.1016/j.patcog.2010.05.028.

Precup, D., & Coates, M. (2016, March 14). *Dimensionality reduction. PCA. Kernel PCA* [Lecture notes]. Department of Computer Science, McGill University. https://www.cs.mcgill.ca/~dprecup/courses/ML/Lectures/ml-lecture13.pdf (Retrieved August 8, 2025).

Prendergast, L. J., Gavin, K., & O'Sullivan, J. J. (2012, September 6–7). Non-intrusive bridge scour analysis technique using laboratory test apparatus. In *Proceedings of the Bridge and Concrete Research in Ireland*, Dublin, Ireland.

Prendergast, L. J., & Gavin, K. (2014). A review of bridge scour monitoring techniques. *Journal of Rock Mechanics and Geotechnical Engineering*, 6(2), 138–149. https://doi.org/10.1016/j.jrmge.2014.01.007.

Prendergast, L. J., Hester, D., & Gavin, K. (2016). Determining the presence of scour around bridge foundations using vehicle-induced vibrations. *Journal of Bridge Engineering, 21*(10), 04016065. https://doi.org/10.1061/(ASCE)BE.1943-5592.0000868.

Prendergast, L. J., Gavin, K., & Hester, D. (2017). Isolating the location of scour-induced stiffness loss in bridges using local modal behavior. *Journal of Civil Structural Health Monitoring, 7*, 483–503. https://doi.org/10.1007/s13349-017-0234-8.

Purnomo, D. A., Aspar, W. A. N., Barasa, W., Harjono, S. M., Sukamdo, P., & Fiantika, T. (2021). Initial implementation of structural health monitoring system of a railway bridge. In *IOP Conference Series: Materials Science and Engineering* (Vol. 1200, No. 1, p. 012019). IOP Publishing. https://doi.org/10.1088/1757-899X/1200/1/012019.

Quirke, P. (2017). *Drive-by detection of railway track longitudinal profile, stiffness and bridge damage* [Doctoral dissertation, University College Dublin].

Rageh, S. E., & Linzell, D. G. (2020). Steel railway bridge fatigue damage detection using numerical models and machine learning: Mitigating influence of modeling uncertainty. *International Journal of Fatigue, 134*, 105458. https://doi.org/10.1016/j.ijfatigue.2019.105458.

Raich, A. M., & Liszkai, T. R. (2012). Multiobjective optimization of sensor and excitation layouts for frequency response function-based structural damage identification. *Computer-Aided Civil and Infrastructure Engineering, 27*(2), 95–117. https://doi.org/10.1111/j.1467-8667.2011.00726.x.

Rail Accident Investigation Branch. (2014). Report 17/2010: Failure of the River Crane railway bridge near Feltham (RAIB Report No. 17/2010). https://www.gov.uk/raib-reports/failure-of-a-railway-bridge-spanning-the-river-crane-near-feltham-in-west-london.

Raju, R. D., Nagarajan, S., Arockiasamy, M., & Castillo, S. (2022). Feasibility of using green laser in monitoring local scour around bridge pier. *Geomatics, 2*, 355–369. https://doi.org/10.3390/geomatics2030020.

Ren, Y., O'Brien, E. J., Cantero, D., & Keenahan, J. (2022). Railway bridge condition monitoring using numerically calculated responses from batches of trains. *Applied Sciences, 12*(10), 4972. https://doi.org/10.3390/app12104972.

Richardson, E. V., & Davis, S. R. (2001). *Evaluating Scour at Bridges* (No. FHWA-NHI-01-001). Federal Highway Administration, Office of Bridge Technology.

Rioul, O., & Vetterli, M. (1991). Wavelets and signal processing. *IEEE Signal Processing Magazine, 8*(4), 14–38.

Rodrigues, I. D., Cunha Jr., A., & Beck, A. T. (2022). Optimization of a 2D reinforced concrete frame considering a seismic load via cross-entropy method. In *Proceedings of the XLIII Ibero-Latin American Congress on Computational Methods in Engineering* (pp. 1–9). Associação Brasileira de Engenharia e Ciências Mecânicas.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*, 533–536. https://doi.org/10.1038/323533a0.

Salcher, P., Adam, C., & Kuisle, A. (2019). A stochastic view on the effect of random rail irregularities on railway bridge vibrations. *Structure and Infrastructure Engineering, 15*(12), 1649–1664. https://doi.org/10.1080/15732479.2019.1640748.

Samsudin, R., Saad, P., & Shabri, A. (2011). River flow time series using least squares support vector machines. *Hydrology and Earth System Sciences, 15*, 1835–1852. https://doi.org/10.5194/hess-15-1835-2011.

Santos, A. D. F., Silva, M. F. M., Sales, C. S., Costa, J. C. W. A., & Figueiredo, E. (2016). Applicability of linear and nonlinear principal component analysis for damage detection. *IEEE Transactions on Instrumentation and Measurement, 65*(4), 939–946. https://doi.org/10.1109/TIM.2015.2507860.

Saputro, D. R. S., & Widyaningsih, P. (2017, August). Limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) method for the parameter estimation on geographically weighted ordinal logistic regression model (GWOLR). In *AIP Conference Proceedings* (Vol. 1868, No. 1, Article 040009). AIP Publishing. https://doi.org/10.1063/1.4995124.

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge University Press. https://doi.org/10.1017/CBO9781107298019.

Shelke, M. S., Deshmukh, P. R., & Shandilya, V. K. (2017). A review on imbalanced data handling using undersampling and oversampling technique. *International Journal of Recent Trends in Engineering & Research, 3*(4), 444–449. https://doi.org/10.23883/IJRTER.2017.3168.0UWXM.

Shen, H. W., Schneider, V. R., & Karaki, S. (1969). Local scour around bridge piers. *Journal of the Hydraulics Division, 95*, 1919–1940. https://doi.org/10.1061/JYCEAJ.0002263.

Sheppard, D. M., & Renna, R. (2005). *Bridge scour manual*. Florida Department of Transportation. https://www.fdot.gov/docs/default-source/roadway/drainage/bridgescour/FDOT-Scour-Manual.pdf.

Sim, S. H., Spencer, B. F., Jr., Zhang, M., & Xie, H. (2010). Automated decentralized modal analysis using smart sensors. *Structural Control and Health Monitoring, 17*(8), 872–894. https://doi.org/10.1002/stc.355.

Sivanandam, S. N., & Deepa, S. N. (2008). Genetic algorithm optimization problems. In An introduction to genetic algorithms (pp. 165–209). Springer.

Sreedhara, B. M., Patil, A. P., Pushparaj, J., Kuntoji, G., & Naganna, S. R. (2021). Application of gradient tree boosting regressor for the prediction of scour depth around bridge piers. *Journal of Hydroinformatics, 23*, 849–863. https://doi.org/10.2166/hydro.2021.072.

Stefenon, S. F., Ribeiro, M. H. D. M., Nied, A., Mariani, V. C., dos Santos Coelho, L., da Rocha, D. F. M., Grebogi, R. B., & de Barros Ruano, A. E. (2020). Wavelet group method of data handling for fault prediction in electrical power insulators. *International Journal of Electrical Power and Energy Systems, 123*, 106269. https://doi.org/10.1016/j.ijepes.2020.106269.

Sun, J., Wang, J., Zhu, Z., He, R., Peng, C., Zhang, C., Huang, J., Wang, Y., & Wang, X. (2022). Mechanical performance prediction for sustainable high-strength concrete using bio-inspired neural network. *Buildings, 12*, 65. https://doi.org/10.3390/buildings12010065.

Sun, Y., Zhang, J., Li, G., Wang, Y., & Jiang, C. (2019). Optimized neural network using beetle antennae search for predicting the unconfined compressive strength of jet grouting coalcretes. *International Journal for Numerical and Analytical Methods in Geomechanics, 43*, 801–813. https://doi.org/10.1002/nag.2882.

Suro, T. P., Huizinga, R. J., Fosness, R. L., & Dudunake, T. (2020). *Assessment of bridge scour countermeasures at selected bridges in the United States, 2014–18* (Scientific Investigations Report 2019-5080). U.S. Geological Survey. https://doi.org/10.3133/sir20195080.

Tan, C., Zhao, H., O'Brien, E. J., Uddin, N., Fitzgerald, P. C., McGetrick, P. J., & Kim, C. W. (2020). Extracting mode shapes from drive-by measurements to detect global and local damage in bridges. *Structure and Infrastructure Engineering, 17*(11), 1582–1596. https://doi.org/10.1080/15732479.2020.1817105.

Tang, H., Zhao, T., Yang, G., & Xue, S. (2018). Vehicle-track system identification for high-speed railway using a cross-entropy approach. In *Proceedings of the 7th Asia-Pacific Workshop on Structural Health Monitoring* (pp. 982–991). ndt.net.

Teolis, A., & Benedetto, J. J. (1998). *Computational signal processing with wavelets*. Springer.

Tien Bui, D., Shirzadi, A., Amini, A., Shahabi, H., Al-Ansari, N., Hamidi, S., Singh, S. K., Thai Pham, B., Ahmad, B. B., & Ghazvinei, P. T. (2020). A hybrid intelligence approach to enhance the prediction accuracy of local scour depth at complex bridge piers. *Sustainability, 12*, 1063. https://doi.org/10.3390/su12031063.

Tola, S., Tinoco, J., Matos, J. C., & O'Brien, E. (2023). Scour detection with monitoring methods and machine learning algorithms—A critical review. *Applied Sciences, 13*(3), 1661. https://doi.org/10.3390/app13031661.

Tola, S., O'Brien, E., Cantero, D., Tinoco, J., Matos, J. C., Bose, T., & Berkers, J. (2025). Drive-by detection of scour in a railway bridge. *Journal of Bridge Engineering, 30*, 04025043. https://doi.org/10.1061/JBENF2.BEENG-13596.

Topping, B. H. V. (1983). Shape optimization of skeletal structures: A review. *Journal of Structural Engineering*, *109*(8), 1933–1951. https://doi.org/10.1061/(ASCE)0733-9445(1998)109:8(1933).

Torrence, C., & Compo, G. P. (1998). A practical guide to wavelet analysis. *Bulletin of the American Meteorological Society, 79*(1), 61–78. https://doi.org/10.1175/1520-0477(1998)079<0061:APGTWA>2.0.CO;2.

Truong, V. H., Vu, Q. V., Thai, H. T., & Ha, M. H. (2020). A robust method for safety evaluation of steel trusses using gradient tree boosting algorithm. *Advances in Engineering Software, 147*, 102825. https://doi.org/10.1016/j.advengsoft.2020.102825.

Uddin, S., Khan, A., Hossain, M. E., & Moni, M. A. (2019). Comparing different supervised machine learning algorithms for disease prediction. *BMC Medical Informatics and Decision Making, 19*, 1–16. https://doi.org/10.1186/s12911-019-1004-8.

Ukil, A. (2007). Support vector machine. In *Intelligent systems and signal processing in power engineering* (pp. 161–226). Springer. https://doi.org/10.1007/978-3-540-37376-9_6.

Van der Auweraer, H., & Peeters, B. (2003). Sensors and systems for structural health monitoring. *Journal of Structural Control*, *10*, 117–125. https://doi.org/10.1002/stc.21.

Van Leeuwen, Z., & Lamb, R. (2014). Flood and scour-related failure incidents at railway assets between 1846 and 2013. Railway Safety and Standards Board.

Venter, G., & Sobieszczanski-Sobieski, J. (2003). Particle swarm optimization. *AIAA Journal, 41*(8), 1583–1589. https://doi.org/10.2514/2.2111.

Vieira, J., Dias, F. M., & Mota, A. (2004). Neuro-fuzzy systems: A survey. In *Proceedings of the 5th WSEAS NNA International Conference on Neural Networks and Applications* (pp. 1–6). Udine, Italy.

Walnut, D. F. (2002). The discrete wavelet transform. In *An introduction to wavelet analysis* (pp. 215–243). Birkhäuser.

Wang, H., Berkers, J., Van den Hurk, N., & Layegh, N. F. (2021). Study of loaded versus unloaded measurements in railway track inspection. *Measurement, 169*, Article 108556. https://doi.org/10.1016/j.measurement.2020.108556.

Wang, H., Liang, Q., Hancock, J. T., & Khoshgoftaar, T. M. (2024). Feature selection strategies: A comparative analysis of SHAP-value and importance-based methods. *Journal of Big Data, 11*, 44. https://doi.org/10.1186/s40537-024-00598-5.

Wang, S. C. (2003). Artificial neural network. In *Interdisciplinary computing in Java programming* (pp. 81–100). Springer. https://doi.org/10.1007/978-1-4615-0377-4_6.

Wang, R., Ming, Y., Feng, G., & Cao, X. (2025). Prediction of local scour depth in bridge piers: Physical information and machine learning based modeling. *Advances in Structural Engineering*. https://doi.org/10.1177/13694332251327844.

Wang, T., Reiffsteck, P., Chevalier, C., Zhu, Z., Chen, C.-W., & Schmidt, F. (2023). A novel extreme gradient boosting algorithm based model for predicting the scour risk around bridge piers: Application to French railway bridges. *European Journal of Environmental and Civil Engineering, 27*(3), 1104–1122. https://doi.org/10.1080/19648189.2022.2072957.

Wang, C., Zhan, J., Wang, Y., Zhang, F., & Pan, D. (2024). A drive-by methodology for rapid inspection of HSR bridge substructures using dynamic responses of passing marshaling trains. *International Journal of Structural Stability and Dynamics, 24*(06), 2450068. https://doi.org/10.1142/S0219455424500688.

Wardhana, K., & Hadipriono, F. C. (2003). Analysis of recent bridge failures in the United States. *Journal of Performance of Constructed Facilities*, *17*(3), 144–150. https://doi.org/10.1061/(ASCE)0887-3828(2003)17:3(144).

White, R. C., Limbert, D. A., Hederick, J. K., & others. (1978). *Guideway-suspension tradeoffs in rail vehicle systems* (Report ERC-R-78035). Arizona State University Engineering Research Center.

Wilson, K. V. (1995). *Scour at selected bridge sites in Mississippi* (Water-Resources Investigations Report 94–4216). U.S. Geological Survey.

Xiong, W., & Cai, C. S. (2022). Time-frequency-based bridge scour identification by trend-change detection. *Journal of Bridge Engineering, 27*(10), 04022093. https://doi.org/10.1061/(ASCE)BE.1943-5592.0001940.

Xu, L., Wang, X., Bai, L., Xiao, J., Liu, Q., Chen, E., Jiang, X., & Luo, B. (2020). Probabilistic SVM classifier ensemble selection based on GMDH-type neural network. *Pattern Recognition, 106*, 107373. https://doi.org/10.1016/j.patcog.2020.107373.

Yagci, O., Yildirim, I., Celik, M. F., Kitsikoudis, V., Duran, Z., & Kirca, V. O. (2017). Clearwater scour around a finite array of cylinders. *Applied Ocean Research, 68*, 114–129. https://doi.org/10.1016/j.apor.2017.07.006.

Yao, S., Peralta-Braz, P., Calderon Hurtado, A., Makki Alamdari, M. M., & Atroshchenko, E. (2024). Design optimisation of piezoelectric energy harvesters for simultaneous power generation and damage detection in bridge health monitoring. *Preprint*. https://doi.org/10.13140/RG.2.2.19728.93449.

Yu, T., & Zhu, H. (2020). Hyper-parameter optimization: A review of algorithms and applications. *arXiv* preprint arXiv:2003.05689. https://arxiv.org/abs/2003.05689.

Zanakis, S. H., & Evans, J. R. (1981). Heuristic "optimization": Why, when, and how to use it. Interfaces, 11(6), 84–91. https://doi.org/10.1287/inte.11.6.84.

Zhang, H., Li, Z., Reven, A., Scharfenberg, B., Chen, G., & Ou, J. (2019, August 4–7). UAV-based smart rock positioning for determination of bridge scour depth. In *Proceedings of the 9th International Conference on Structural Health Monitoring of Intelligent Infrastructure (SHMII-9)*. IWSHM/IA-FRA. St. Louis, MO, United States.

Zhang, C., Mousavi, A. A., Masri, S. F., & Gholipour, G. (2025). The state-of-the-art on time-frequency signal processing techniques for high-resolution representation of nonlinear systems in engineering. *Archives of Computational Methods in Engineering, 32*, 785–806. https://doi.org/10.1007/s11831-024-10153-z.

Zhang, Y., Xiong, Z., Liang, Z., She, J., & Ma, C. (2023). Structural damage identification system suitable for old arch bridges in rural regions: Random forest approach. *CMES—Computer Modeling in Engineering & Sciences, 136*(1), 1–??. https://doi.org/10.32604/cmes.2023.022699.

Zhang, J., & Zhao, H. (2020). A prediction model for local scour depth around piers based on CNN. In *2020 International Conference on Information Science, Parallel and Distributed Systems (ISPDS)* (pp. 318–320). IEEE. https://doi.org/10.1109/ISPDS48619.2020.00074.

Zhang, B., Zhao, H., Tan, C., O'Brien, E. J., Fitzgerald, P. C., & Kim, C.-W. (2022). Laboratory investigation on detecting bridge scour using the indirect measurement from a passing vehicle. *Remote Sensing, 14*(13), 3106. https://doi.org/10.3390/rs14133106.

Zhang, T., Zhu, J., Xiong, Z., Zheng, K., & Wu, M. (2023). A new drive-by method for bridge damage inspection based on characteristic wavelet coefficient. *Buildings*, 13(2), 397. https://doi.org/10.3390/buildings13020397.

Zhao, W., Bhushan, A., Santamaria, A. D., Simon, M. G., & Davis, C. E. (2008). Machine learning: A crucial tool for sensor design. *Algorithms, 1*(2), 130–152. https://doi.org/10.3390/a1020130.

Zheng, W., Qian, F., Shen, J., & Xiao, F. (2020). Mitigating effects of temperature variations through probabilistic-based machine learning for vibration-based bridge scour detection. *Journal of Civil Structural Health Monitoring*, *10*(5), 957-972. https://doi.org/10.1007/s13349-020-00427-y.

ANNEX A: Case Study Bridge Displacement Response

This annex provides details of the displacement component of the theoretical track profile for the case study bridge, which has seven supported spans, as seen in Figure A.1, an elevation view of the bridge before the foundations were repaired (Network Rail (2015a)). First, the details of the computational steps provided in Section 3.1.2.2 for obtaining the support-deflection component and the bending component of the displacement response for the case study bridge are presented. Then, the MATLAB code utilized for mathematical calculations is provided.

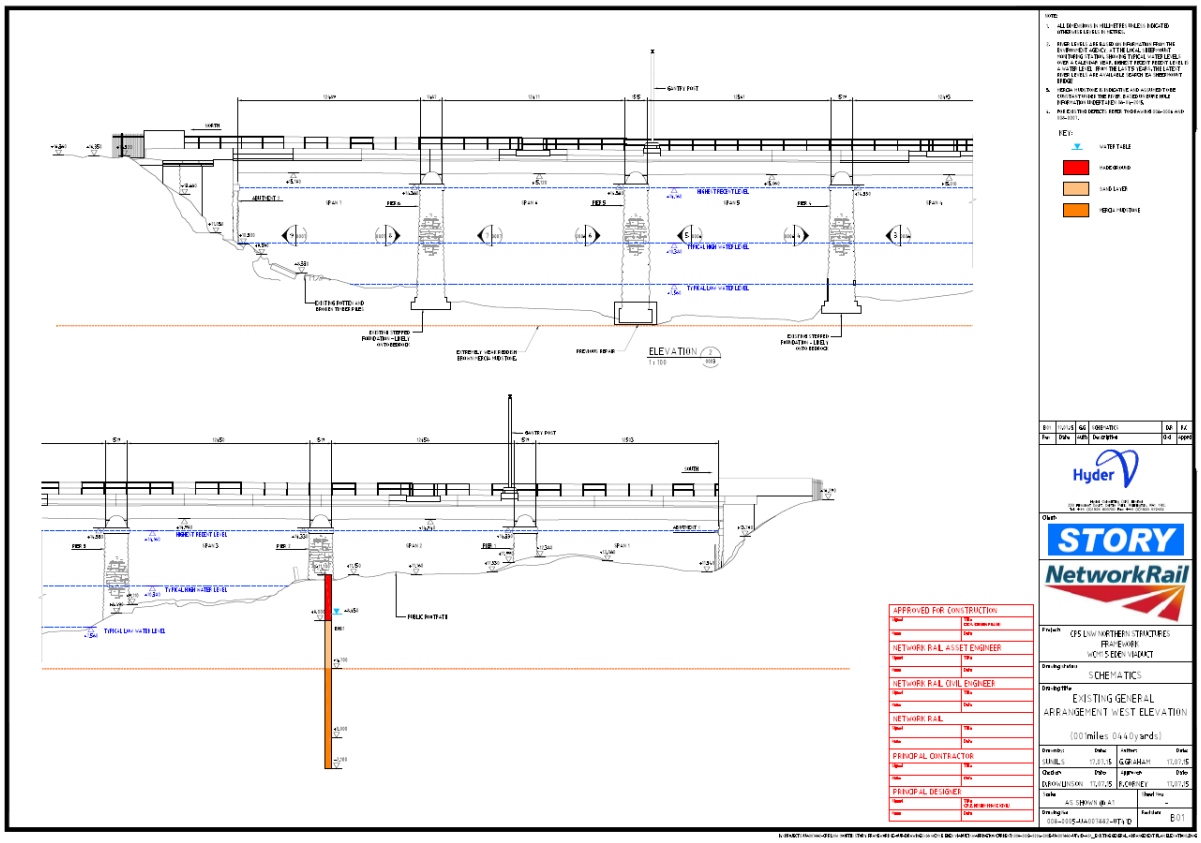


Figure A.1. Existing General Arrangement – West Elevation, Eden Viaduct Main Line railway bridge (No WCM1 5) (Network Rail (2015a)).

A.1 Deriving z_{x1} and z_{x2} For the Case Study Bridge

Computing z_{x1}. To calculate the bending component of the displacement, the MR-IL for vertical displacement is generated using the unit-load theorem. This involves applying a unit load at *x* in the virtual system (Figure A.2) while the real axle load (Figure A.3) is at *x-d*. z_{x1} is found with Eq. (A.1):

$$z_{x1} = \int_0^l \frac{M_u(x)M(x)}{EI} dx \tag{A.1}$$

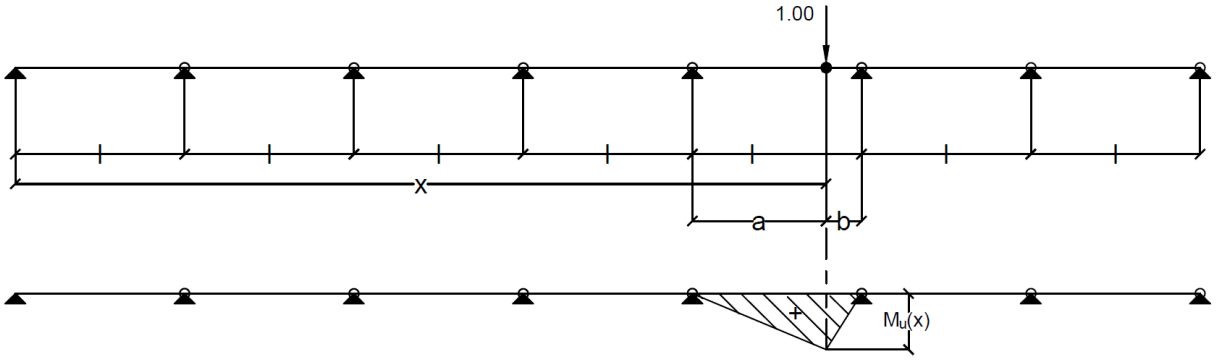


Figure A.2. Calculation of z_{x1}: Virtual system.

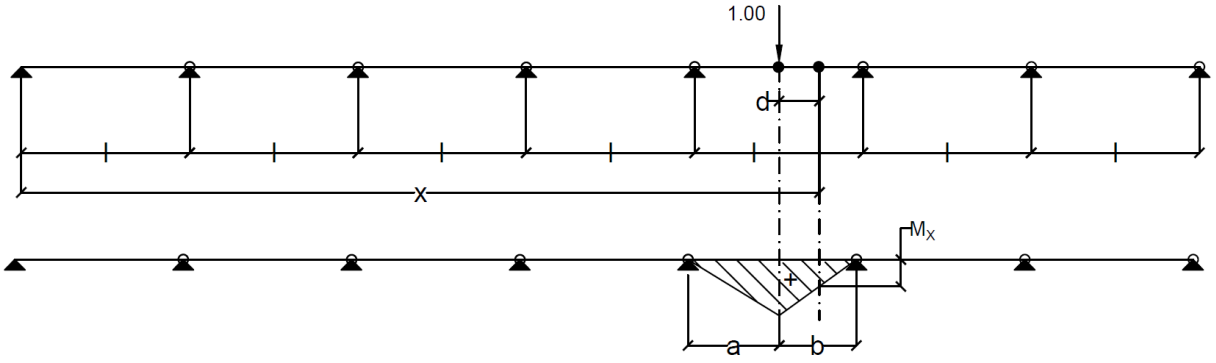


Figure A.3. Calculation of z_{x1}: Real system.

Computing z_{x2}. To calculate the displacement component due to support deflection, a 1 kN load is applied to each span, and the resulting deflections are calculated, as shown in Figure A.4 The z_{x2} component of the measurement point x is formulated as shown in Eq. (A.2):

$$z_{x2} = \frac{b}{k_L l} + \frac{(a-b)(x-(n-1)l)}{k_R l^2} \tag{A.2}$$

Where a is the distance from the left support of span *n* to the axle location *x-d*, and b is the distance from the axle location *x-d* to the right support of span *n*, k_L and k_R are the vertical stiffness of the left and right supports of the span where the load is applied, respectively.

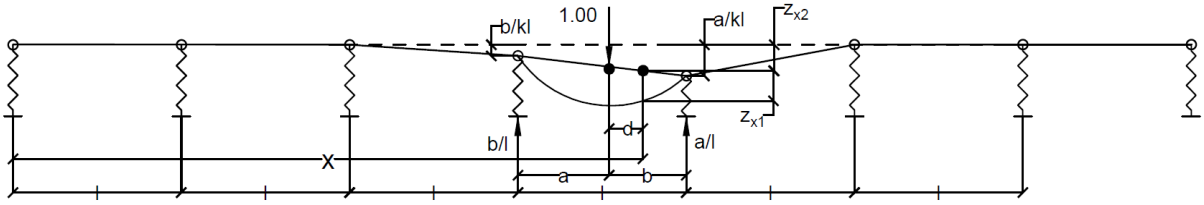
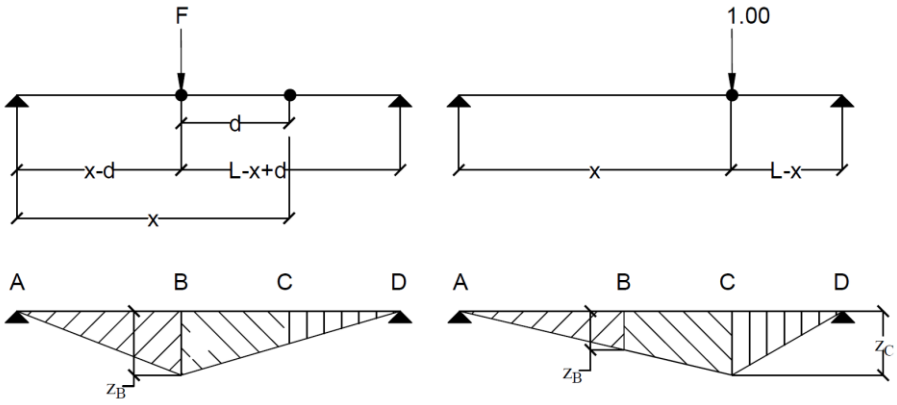


Figure A.4. Calculation of z_{x2}.

The total displacement response is thus determined by the superposition of these two components, i.e., $z_x = z_{x1} + z_{x2}$. Subsequently, the resultant displacements are to be multiplied by the axle load (it is to be noted that all axle loads are equal). Further calculations are provided in the MATLAB code presented in the following subsection.

A.2 MATLAB Code

The MATLAB model calculates the response of the Eden Viaduct Main Line railway bridge (No. WCM1 5) in the UK to the crossing of a 6-axle vehicle, namely a Class 66/0 Co-Co diesel locomotive with a RILA rear-mounted case, as provided below.



(a) (b)

Figure A.5. (a) Real system and its moment diagram M; (b) Virtual system and its moment diagram M_u.

```
% ———————— Vehicle Properties ————————————
na=6;                  % number of axles
P =181.43;             % one axle load in kN
% ———————— Beam Properties ————————————
sp=7;                  % number of spans
L=0.2;                 % length of small beam elements in meters
l=[13.2 14.0 14.0 14.0 14.0 14.0 13.2]; % Span lengths in meters
```

```
N=zeros(sp); % number of small beam elements
E = 3.4342e10;      % Young's Modulus (N/m²)
J = 0.9028;         % Second moment of area
csa = 8.954         % Cross sectional area (m²) of prestressed beam in Figure A.6.
EI = E*J;
d1=[19.4 17.4 15.2 6.2 4.0 2.0]; % distances between the unit loads (6 axles) and point x
k = 467966*ones(8,1);               % stiffness values of supports
AAB=zeros(na,ss+1);                 % Area multiplication of AB triangle in Figure A.5
ABC=zeros(na,ss+1);                 % Area multiplication of BC trapezoid in Figure A.5
ACD=zeros(na,ss+1);                 % Area multiplication of CD triangle in Figure A.5
d_xs1=zeros(na,ss+1);               % Bending component of MR-IL
d_xs=zeros(na,ss+1);                % Total displacement response
resp_x=zeros(1,ss+1);               % Bridge Displacement Response
t = [0; cumsum(l(:))];              % sum of lengths
ss=0;
for p=1:sp
   N(p)=l(p)/L; ss=ss+N(p);
end
for i = 1:ss+1
   x1(i) = fix((i-1)*L*100)/100;
   for j = 1:na  % for all "na" number of axles
      for p = 1:sp-2  % for 0 <= x <= 5L
         if round(t(p),2) <= x1(i) && x1(i) < round(t(p+1),2)
            if t(p+2) < x1(i)+d1(j) && x1(i)+d1(j) < t(p+3)
               a1(i,j) = round((d1(j)+x1(i)-t(p+2)),2);
               b1(i,j) = round((l(p+2)-a1(i,j)),2);
            elseif t(p+1) < x1(i)+d1(j) && x1(i)+d1(j) < t(p+2)
               a1(i,j) = round((d1(j)+x1(i)-t(p+1)),2);
               b1(i,j) = round((l(p+1)-a1(i,j)),2);
            elseif x1(i) == round(t(p+1)-d1(j),2)
               a1(i,j) = round(l(p),2); b1(i,j) = 0;
            elseif x1(i) == round(t(p+2)-d1(j),2)
```

```
            a1(i,j) = round(l(p+1),2); b1(i,j) = 0;
        elseif (t(p) <= x1(i)+d1(j)) && (x1(i)+d1(j) < t(p+1))
            a1(i,j) = round((d1(j)+x1(i)-t(p)),2);
            b1(i,j) = round((l(p)-a1(i,j)),2);
        end
        AAB(j,i) = ((x1(i)-t(p)).^3) .* (l(p)-(x1(i)-t(p))) .* ((l(p)-(x1(i)-t(p))-d1(j))) ./ (3*(l(p).^2));
        ABC(j,i) = (d1(j)/6) .* ( ...
            (2.*(x1(i)-t(p)).^2) .* (l(p)-(x1(i)-t(p))-d1(j)) .* (l(p)-(x1(i)-t(p))) ./ ((l(p)).^2) + ...
            (x1(i)-t(p)).^2 .* (l(p)-(x1(i)-t(p))-d1(j)).^2 / (l(p).^2) + ...
                        ((x1(i)-t(p))+d1(j)) .* (l(p)-(x1(i)-t(p))-d1(j)) .* (x1(i)-t(p)) .* (l(p)-(x1(i)-t(p))) ./
          (l(p).^2) +  2 .* ((x1(i)-t(p))+d1(j)) .* ((l(p)-(x1(i)-t(p))-d1(j)).^2) .* (x1(i)-t(p)) ./ ((l(p)).^2));
        ACD(j,i) = (x1(i)-t(p)) .* ((x1(i)-t(p))+d1(j)) .* ((l(p)-(x1(i)-t(p))-d1(j)).^3) ./ (3*(l(p).^2));
        if x1(i) > round(t(p+1)-d1(j),2) && x1(i) <= round(t(p+2)-d1(j),2)  % axles that are 1 span
in front of x
            d_xs2(j,i) = (x1(i)-t(p)) * b1(i,j) ./ (k(p+1) .* l(p) .* l(p+1));
            AAB(j,i) = 0;  ABC(j,i) = 0;  ACD(j,i) = 0;
        elseif x1(i) > round(t(p+2)-d1(j),2) && x1(i) <= round(t(p+3)-d1(j),2)  % axles that are 2
spans in front of x
            d_xs2(j,i) = 0;  AAB(j,i) = 0;  ABC(j,i) = 0;  ACD(j,i) = 0;
        elseif a1(i,j) >= b1(i,j) && fix(x1(i)*100)/100 >= t(p)-d1(j) && fix(x1(i)*100)/100 <=
round(t(p+1)-d1(j),2)  % same span with x
            d_xs2(j,i) = (b1(i,j) ./ (k(p) .* l(p))) + ((a1(i,j) ./ (k(p+1) .* l(p)) - b1(i,j) ./ (k(p) .* l(p)))
.* ((x1(i)-t(p)) ./ l(p)));
        elseif b1(i,j) > a1(i,j) && fix(x1(i)*100)/100 >= t(p)-d1(j) && fix(x1(i)*100)/100 <=
round(t(p+1)-d1(j),2)  % same span with x
            d_xs2(j,i) = (a1(i,j) ./ (k(p+1) .* l(p))) + ((b1(i,j) ./ (k(p) .* l(p)) - a1(i,j) ./ (k(p+1) .* l(p)))
.* ((t(p+1)-x1(i)) ./ l(p)));
        end
        d_xs1(j,i) = (AAB(j,i) + ABC(j,i) + ACD(j,i)) ./ (EI/1000);
        d_xs(j,i)  = -(d_xs1(j,i) + d_xs2(j,i));
    elseif round(t(sp-1),2) <= x1(i) && x1(i) < round(t(sp),2)  % 5L <= x < 6L
        if round(t(sp),2) < x1(i)+d1(j) && x1(i)+d1(j) < round(t(sp+1),2)
```

```
            a1(i,j) = round((d1(j)+x1(i)-t(sp)),2);
            b1(i,j) = round((l(sp)-a1(i,j)),2);
        elseif x1(i) == round(t(sp)-d1(j),2)
            a1(i,j) = round(l(sp-1),2); b1(i,j) = 0;
        elseif x1(i) == round(t(sp+1)-d1(j),2)
            a1(i,j) = round(l(sp),2); b1(i,j) = 0;
        elseif round(t(sp-1),2) <= x1(i)+d1(j) && x1(i)+d1(j) < round(t(sp),2)
            a1(i,j) = round(d1(j) + x1(i) - t(sp-1), 2);
            b1(i,j) = round(l(sp-1) - a1(i,j), 2);
        elseif x1(i)+d1(j) > round(t(sp+1),2)  % axles that left the bridge
            a1(i,j) = 0; b1(i,j) = 0;
        end
        AAB(j,i) = ((x1(i)-t(sp-1)).^3) .* (l(sp-1) - (x1(i) - t(sp-1))) .* ((l(sp-1) - (x1(i)-t(sp-1)) - d1(j)))
./ (3*(l(sp-1)).^2);
        ABC(j,i) = (d1(j)./6) .* (2.*(x1(i)-t(sp-1)).^2 .* (l(sp-1)-(x1(i)-t(sp-1))-d1(j)) .* (l(sp-1)-(x1(i)-
t(sp-1))) / (l(sp-1).^2) + (x1(i)-t(sp-1)).^2 .* (l(sp-1)-(x1(i)-t(sp-1))-d1(j)).^2 / (l(sp-1).^2) + ((x1(i)-t(sp-
1))+d1(j)) .* (l(sp-1)-(x1(i)-t(sp-1))-d1(j)) .* (x1(i)-t(sp-1)) .* (l(sp-1)-(x1(i)-t(sp-1))) ./ (l(sp-1).^2) + 2.*((x1(i)
- t(sp-1)) + d1(j)) .* (l(sp-1)-(x1(i)-t(sp-1))-d1(j)).^2 .* (x1(i)-t(sp-1)) ./ ((l(sp-1)).^2));
        ACD(j,i) = (x1(i)-t(sp-1)) .* ((x1(i)-t(sp-1))+d1(j)) .* ((l(sp-1)-(x1(i)-t(sp-1))-d1(j)).^3) ./
(3*(l(sp-1)).^2);
        if x1(i) > round(t(sp)-d1(j),2) && x1(i) <= round(t(sp+1)-d1(j),2)  % 1 span in front of x
            d_xs2(j,i) = (x1(i)-t(sp-1)) .* b1(i,j) ./ (k(sp) .* l(sp) .* l(sp-1));
            AAB(j,i) = 0;  ABC(j,i) = 0;  ACD(j,i) = 0;
        elseif x1(i) > round(t(sp+1)-d1(j),2)  % 2 spans in front of x (out of bridge)
            d_xs2(j,i) = 0;
            AAB(j,i) = 0;  ABC(j,i) = 0;  ACD(j,i) = 0;
        elseif a1(i,j) >= b1(i,j) && fix(x1(i)*100)/100 >= round(t(sp-1)-d1(j),2) && x1(i) <=
round(t(sp)-d1(j),2)  % same span with x
            d_xs2(j,i) = (b1(i,j) ./ (k(sp-1) .* l(sp-1))) + ((a1(i,j) ./ (k(sp) .* l(sp-1)) - b1(i,j) ./ (k(sp-
1) .* l(sp-1))) .* ((x1(i)-t(sp-1)) ./ l(sp-1)));
        elseif b1(i,j) > a1(i,j) && fix(x1(i)*100)/100 >= round(t(sp-1)-d1(j),2) && x1(i) <=
round(t(sp)-d1(j),2)  % same span with x
```

```
            d_xs2(j,i) = (a1(i,j) ./ (k(sp) .* l(sp-1))) + ((b1(i,j) ./ (k(sp-1) .* l(sp-1)) - a1(i,j) ./ (k(sp)
.* l(sp-1))) .* ((t(sp)-x1(i)) ./ l(sp-1)));
        end
        d_xs1(j,i) = (AAB(j,i) + ABC(j,i) + ACD(j,i)) ./ (EI/1000);
        d_xs(j,i)  = -(d_xs1(j,i) + d_xs2(j,i));
    elseif round(t(sp),2) <= x1(i) && x1(i) <= round(t(sp+1),2)  % 6L <= x <= 7L
        if x1(i) == round(t(sp+1)-d1(j),2)  % at the end of the bridge
            a1(i,j) = round(l(sp),2); b1(i,j) = 0;
        elseif round(t(sp),2) <= x1(i)+d1(j) && x1(i)+d1(j) <= round(t(sp+1),2)
            a1(i,j) = round(d1(j) + x1(i) - t(sp), 2);
            b1(i,j) = round(l(sp) - a1(i,j), 2);
        elseif x1(i)+d1(j) > round(t(sp+1),2)  % axles that left the bridge
            a1(i,j) = 0;  b1(i,j) = 0;
        end
        AAB(j,i) = ((x1(i)-t(sp)).^3) .* (l(sp)-(x1(i)-t(sp))) .* ((l(sp)-(x1(i)-t(sp))-d1(j))) ./ (3*(l(sp)).^2);
        ABC(j,i) = (d1(j)./6) .* (2*(x1(i)-t(sp)).^2 .* (l(sp)-(x1(i)-t(sp))-d1(j)) .* (l(sp)-(x1(i)-t(sp))) ./
(l(sp).^2) + (x1(i)-t(sp)).^2 .* ((l(sp)-(x1(i)-t(sp))-d1(j)).^2) ./ (l(sp).^2) + ((x1(i)-t(sp))+d1(j)) .* (l(sp)-(x1(i)-
t(sp))-d1(j)) .* (x1(i)-t(sp)) .* (l(sp)-(x1(i)-t(sp))) ./ (l(sp).^2) + 2.*((x1(i)-t(sp))+d1(j)) .* (l(sp)-(x1(i)-t(sp))-
d1(j)).^2 .* (x1(i)-t(sp)) ./ ((l(sp)).^2));
        ACD(j,i) = (x1(i)-t(sp)) .* ((x1(i)-t(sp))+d1(j)) .* ((l(sp)-(x1(i)-t(sp))-d1(j)).^3) ./ (3*(l(sp)).^2);
        if x1(i) > round(t(sp+1)-d1(j),2)  % 1 span in front of x (out of bridge)
            d_xs2(j,i) = 0;
            AAB(j,i) = 0;  ABC(j,i) = 0;  ACD(j,i) = 0;
        elseif a1(i,j) >= b1(i,j) && fix(x1(i)*100)/100 >= round(t(sp)-d1(j),2) && x1(i) <=
round(t(sp+1)-d1(j),2)  % same span with x
            d_xs2(j,i) = (b1(i,j) ./ (k(sp) .* l(sp))) + ((a1(i,j) ./ (k(sp+1) .* l(sp)) - b1(i,j) ./ (k(sp) .*
l(sp))) .* ((x1(i)-t(sp)) ./ l(sp)));
        elseif b1(i,j) > a1(i,j) && fix(x1(i)*100)/100 >= round(t(sp)-d1(j),2) && x1(i) <=
round(t(sp+1)-d1(j),2)  % same span with x
            d_xs2(j,i) = (a1(i,j) ./ (k(sp+1) .* l(sp))) + ((b1(i,j) ./ (k(sp) .* l(sp)) - a1(i,j) ./ (k(sp+1)
.* l(sp))) .* ((t(sp+1)-x1(i)) ./ l(sp)));
        end
```

```
            d_xs1(j,i) = (AAB(j,i) + ABC(j,i) + ACD(j,i)) ./ (EI/1000);
            d_xs(j,i)  = -(d_xs1(j,i) + d_xs2(j,i));
        end
    end
    resp_x(i) = resp_x(i) + P .* d_xs(j,i);  % response due to 6 axles
  end
end
% ———————— End of Script ————————
```

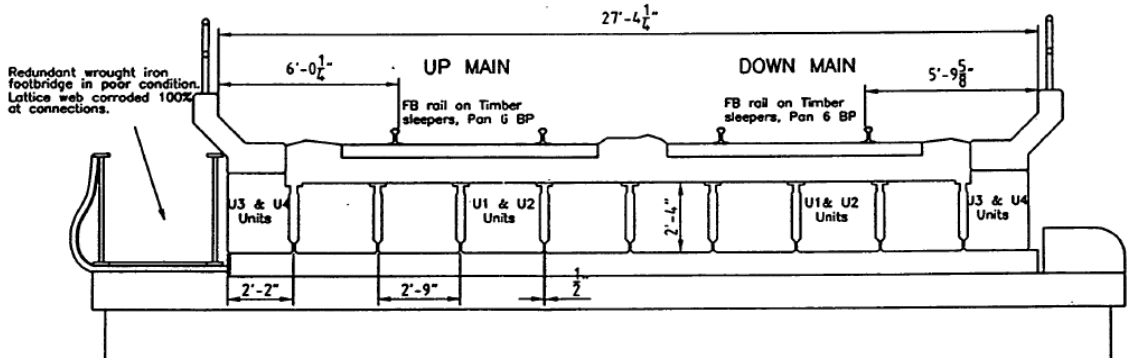


Figure A.6. Cross-section of the case study bridge.

A.3 SAP2000 Validation

To verify the MATLAB implementation of the MR-IL displacement of the bridge model, an equivalent SAP2000 model of the seven-span simply supported bridge is generated, and the displacement components are compared at representative locations under identical axle loads (6 axles, each 181.43 kN), bridge properties, and support stiffness values.

Check near a support. For x = 86.6m, calculated displacements are z_{x1} =0.819mm (Figure A.7) and z_{x2} = 0.079mm (Figure A.8) for SAP2000; a total of z_x^{SAP}= 0.898mm; while MATLAB code calculates a total of z_x^{MAT}= 0.896mm displacement (Figure A.9). *Check at midspan.* For x = 48.2m, calculated displacements are z_{x1} = 0.558mm and z_{x2} =0.729mm for SAP2000; a total of z_x^{SAP} = 1.287 mm; while the MATLAB code calculates a total of z_x^{MAT} = 1.290mm displacement. When assessing the agreement between SAP2000 and MATLAB (total displacement), the relative error is computed:

$$e_{rel} = \frac{z_x^{MAT} - z_x^{SAP}}{z_x^{SAP}} \tag{A.3}$$

The relative error is around e_{rel} = 0.228%.

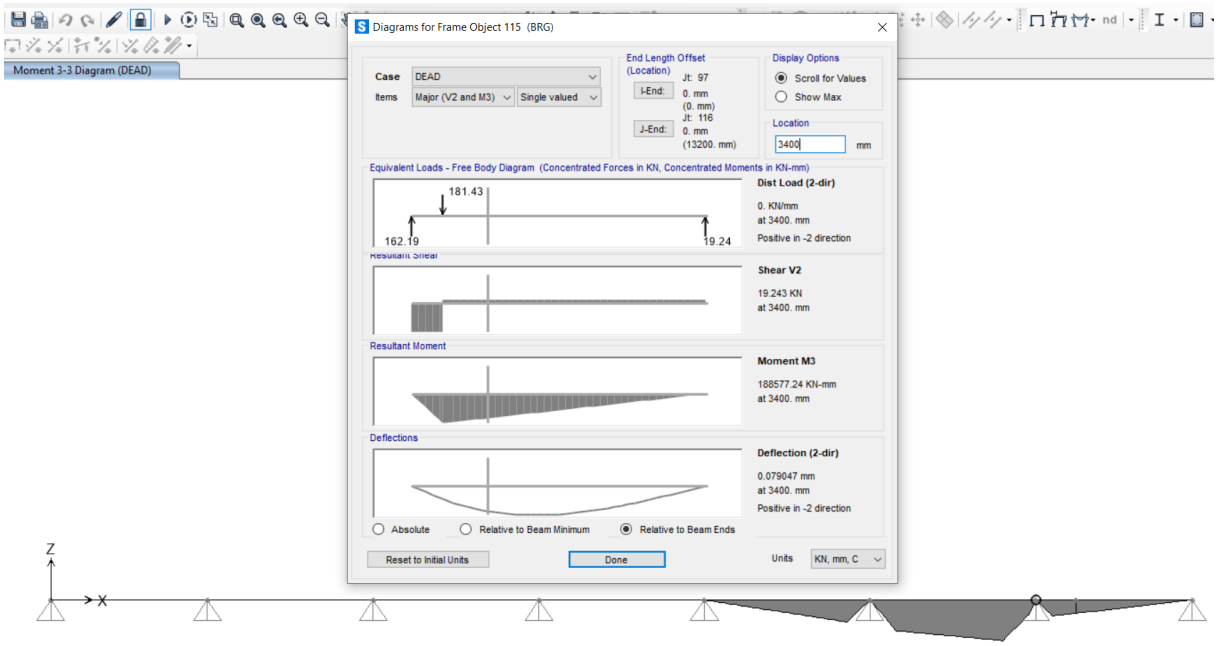


Figure A.7. SAP2000, z_{x1} component.

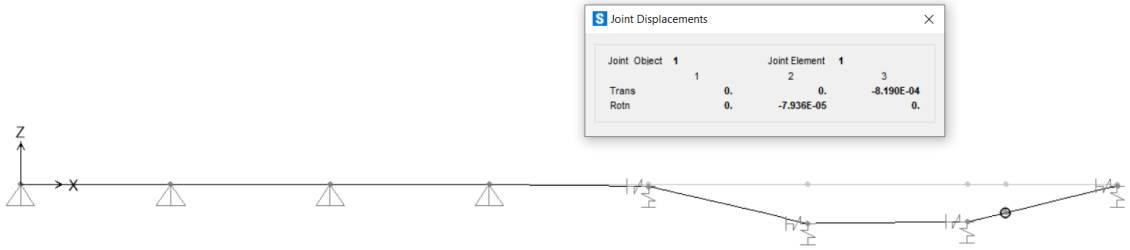


Figure A.8. SAP2000, z_{x2} component.

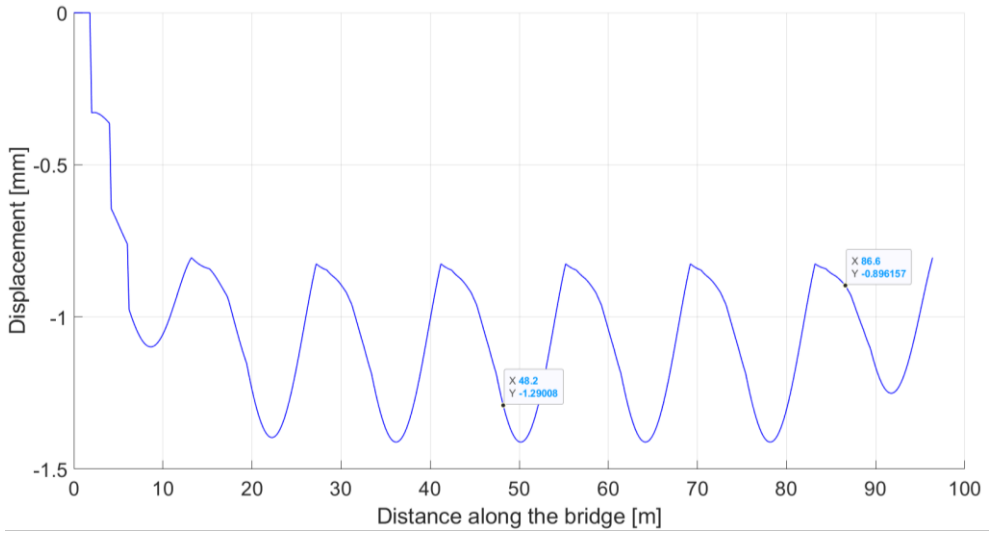


Figure A.9. MATLAB Code displacement response at x = 48.2m and x = 86.6m.

ANNEX B: Displacement Response of The Track On Soil

This annex details the calculation of the moving-reference displacement response of a 40m track on soil with 0.5m sleeper distances. It outlines the details of the computational steps presented in Section 3.1.3.2 for obtaining displacements under the moving load of a six-axle Class 66/0 Co-Co diesel vehicle. The MATLAB code used for the calculations and plots is then included. Finally, verification of the MATLAB code using SAP2000 is shown.

B.1 Obtaining Displacement Response Of The Track

Step 2: Calculating the displacement vector {D}. This step is calculated with a 1 kN load, and the results are then multiplied by the axle load, i.e., 181.43 kN. Figure B.1 Illustrates the deformed shape of the track under the load at a distance a from the left support. Displacement vector *{D}* consists of the contributions of a bending and non-bending (axial) part, denoted by $\{D_{ax}\}$ and $\{D_{ben}\}$, respectively. Only the four hinges surrounding the loaded span are non-zero.

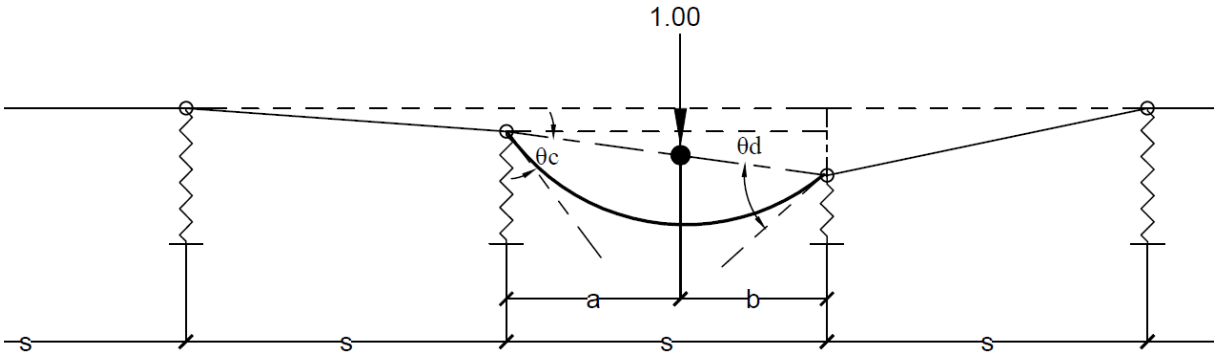


Figure B.1. Step 2.

Calculating $\{D_{ax}\}$. The non-zero terms of D_{ax} are provided in Eq. (B.1):

$$\begin{aligned} D_{\text{ax}}(j-1) &= \frac{1}{k\cdot s^2}\cdot b (\text{if } j-1 \geq 1) \\ D_{\text{ax}}(j) &= \frac{1}{k\cdot s^2}\cdot (a-2\cdot b), \\ D_{\text{ax}}(j+1) &= \frac{1}{k\cdot s^2}\cdot (b-2\cdot a), \\ D_{\text{ax}}(j+2) &= \frac{1}{k\cdot s^2}\cdot a (\text{if } j+2 \leq n), \end{aligned} \tag{B.1}$$

This matches the straight dashed and sloped element contributions in Figure B.1, where *j* is the index of the span that contains the axle, *k* is the spring stiffness, and *s* is the sleeper spacing.

Calculating $\{\Delta D_{ben}\}$. The non-zero terms are as seen in Eq. (B.2):

$$\begin{aligned} \Delta D_{\text{ben}}(j) &= \frac{a \cdot b \cdot \left(1+\frac{b}{s}\right)}{6 \cdot EI} \\ \Delta D_{\text{ben}}(j+1) &= \frac{a \cdot b \cdot \left(1+\frac{a}{s}\right)}{6 \cdot EI} \end{aligned} \tag{B.2}$$

All other components of $\{\Delta D_{ben}\}$ are zero. Θ_c and θ_d are the rotation angles in Figure B.1. When the axle is at a hinge, $\{\Delta D_{ben}\}$ =0. Resultant $\{D\}$ is as in Eq. (B.3):

$$D_{79x1} = D_{\text{ax}} + \Delta D_{\text{ben}} \tag{B.3}$$

Step 3: Calculating the flexibility matrix. In this step, unit loads are applied to the hinges, as seen in Figure B.2. Flexibility matrix $[f]_{79x79}$ is generated by combining bending and non-bending contributions of the track, denoted by $[f_{ax}]$ and $[f_{ben}]$, respectively.

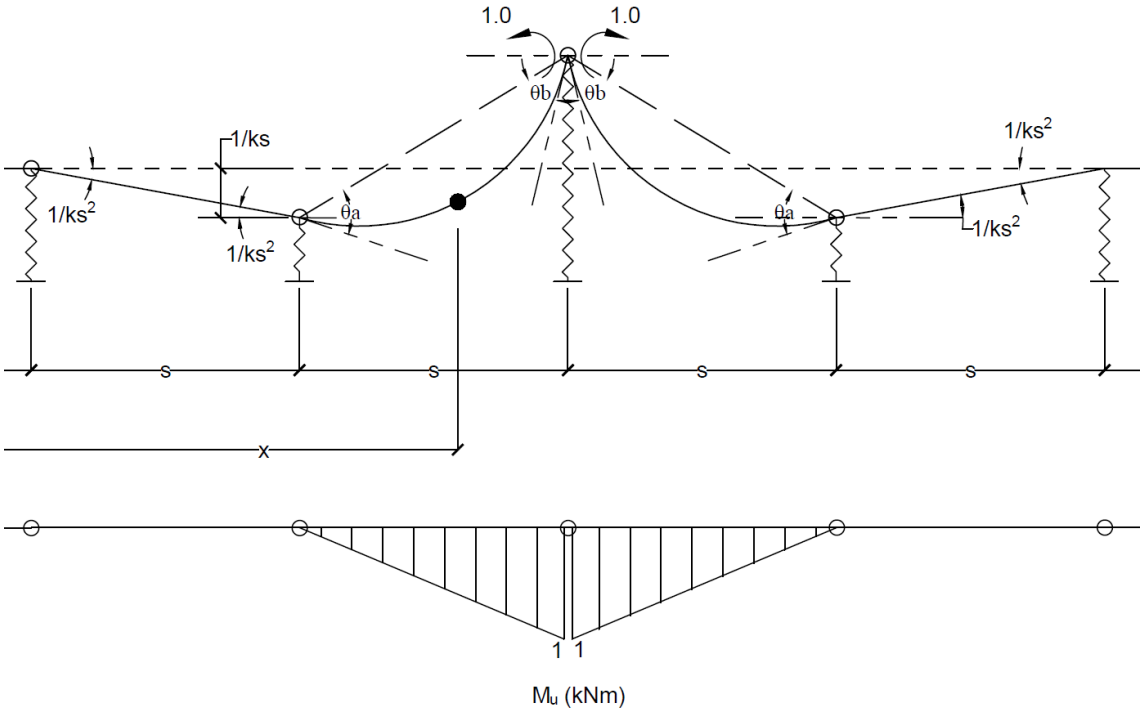


Figure B.2. Step 3.

Calculating $[f_{ax}]$. Non-zero terms of the five-diagonal Toeplitz matrix are as in Eq. (B.4):

$$\begin{aligned} f_{\text{ax}}(i,i) &= \frac{6}{k\cdot s^2}, \\ f_{\text{ax}}(i,i\pm 1) &= -\frac{4}{k\cdot s^2}, \\ f_{\text{ax}}(i,i\pm 2) &= \frac{1}{k\cdot s}, \end{aligned} \tag{B.4}$$

Calculating Δf_{ben}. For each span (i, i+1), the bending contribution is as in Eq. (B.5):

$$\Delta f_{\text{ben}}[\{i,i+1\},\{i,i+1\}] = \frac{s}{6\cdot EI}\begin{bmatrix}2 & 1\\ 1 & 2\end{bmatrix} \tag{B.5}$$

which is the standard rotational flexibility of a prismatic beam. Resultant f is as in Eq. (B.6):

$$f = f_{\text{ax}} + \Delta f_{\text{ben}} \tag{B.6}$$

For interior hinges, the combined matrix diagonals are as in Eq. (B.7):

$$\begin{aligned} f(i,i) &= \frac{6}{k\cdot s^2} + 2\cdot\frac{s}{3\cdot EI}, \\ f(i,i\pm 1) &= -\frac{4}{k\cdot s^2} + \frac{s}{6\cdot EI}, \\ f(i,i\pm 2) &= \frac{1}{k\cdot s^2}. \end{aligned} \tag{B.7}$$

At the first and last hinges, each has only one neighbouring span, so the bending add-on on the diagonal is $+\frac{s}{3\cdot EI}$ and on the first off-diagonal $+\frac{s}{6\cdot EI}$.

Step 4: Solving the hinge load vector. {F} is computed by solving Eq. (B.8):

$$\{F\} = [f]^{-1}\cdot\{-D\} \tag{B.8}$$

Step 5: Finding the total displacement of point x. Displacement response of point *x* is as in Eq. (B.9):

$$z_x = (z_x)^{step2} + \sum_{k=1}^{79}(z_x)^{step3} * F_k \tag{B.9}$$

Where *k* is the number of hinges.

B.2 MATLAB Code

The off-bridge MATLAB model calculates the structural response of a 40-meter-long track section, taking into account a 5-meter segment with poor soil conditions. This track is crossed by a 6-axle Class 66/0 Co-Co diesel vehicle equipped with a RILA front-mounted case.

```
% MR-IL & Response of track, linear reduced stiffness zone
% ———————— Vehicle Properties ————————————
na = 6;                          % Number of axles (6 axles)
Pa =181.43;                      % Each axle load in kN
% ———————— Track Properties ————————————
```

```
N = 8;                                % No. small elements (mesh size)
S =0.5;                               % Distance between sleepers
L = S/N;                              % Length of each track element
sp=80;                                % no of spans (between sleepers)
E = 2.059e11;                         % Young's Modulus for steel (N/m^2)
J = 0.00006434 ;                      % Moment of inertia (m^4) of 2 rails
EI = E*J/1000;                        % Flexural Stiffness (kNm^2)
d1=[2.0 4.0 6.2 15.2 17.4 19.4];  % 6 axles
x1=zeros(sp*N+1,1);
x1f=zeros(sp*N+1,1);                  % span scale x1
D_x=zeros(sp-1,sp*N+1);               % displacement matrix "D"
d_xs3=zeros(sp-1,sp*N+1);             % x displacements at step3
d_r=zeros(1,sp*N+1);
d_l=zeros(1,sp*N+1);
F=zeros(sp-1,sp*N+1);                 % Hinge load vector
b1=zeros(sp*N+1,na); a1=zeros(sp*N+1,na); k1=28131;% [calculated-kN/m]
resp_x=zeros(1,sp*N+1);               % MR-IL Displacement Response
sf=zeros(sp,sp*N+1);
p=zeros(sp*N+1,na);
r=zeros(sp*N+1,1);
K=k1.*ones(1,sp+1);                   % spring stiffness values along the track
z1=47;                                % start hinge of reduced stiffness zone
z3=10;                                % number of spans of the reduced stiffness zone
z2=z1+(z3/2);                         % middle hinge of reduced stiffness zone
for z=z1+1:z1+(z3/2)                  % left hand side of reduced stiffness zone
  K(z)=-0.06*k1*z+(0.06*z1+1)*k1; % value of spring stiffness
  %K(z)=k1;
end
for z=z2+1:z2+(z3/2)-1                % right-hand side of reduced stiffness zone
  K(z)=0.06*k1*z+(0.7-z2*0.06)*k1;
  %K(z)=k1;
end
```

```
fs=zeros(sp-1,sp-1); % flexibility matrix for support deflection
for m=1:sp-1
   if m==1
   fs(m,m)=(1/(S^2))*((4/K(m+1))+(1/K(m))+(1/K(m+2)));
   fs(m,m+1)=(-1/(S^2))*((2/K(m+1))+(2/K(m+2)));
   fs(m,m+2)=1/(K(m+2).*(S^2));
   elseif m==2
   fs(m,m)=(1/(S^2))*((4/K(m+1))+(1/K(m))+(1/K(m+2)));
   fs(m,m-1)=(-1/(S^2))*((2/K(m+1))+(2/K(m)));
   fs(m,m+1)=(-1/(S^2))*((2/K(m+1))+(2/K(m+2)));
   fs(m,m+2)=1/(K(m+2).*(S^2));
   elseif m==sp-1
   fs(m,m)=(1/(S^2))*((4/K(m+1))+(1/K(m))+(1/K(m+2)));
   fs(m,m-1)=(-1/(S^2))*((2/K(m+1))+(2/K(m)));
   fs(m,m-2)=1/(K(m).*(S^2));
   elseif m==sp-2
   fs(m,m+1)=(-1/(S^2))*((2/K(m+1))+(2/K(m+2)));
   fs(m,m)=(1/(S^2))*((4/K(m+1))+(1/K(m))+(1/K(m+2)));
   fs(m,m-1)=(-1/(S^2))*((2/K(m+1))+(2/K(m)));
   fs(m,m-2)=1/(K(m).*(S^2));
   elseif 3<=m && m<=sp-1
   fs(m,m)=(1/(S^2))*((4/K(m+1))+(1/K(m))+(1/K(m+2)));
   fs(m,m+1)=(-1/(S^2))*((2/K(m+1))+(2/K(m+2)));
   fs(m,m-1)=(-1/(S^2))*((2/K(m+1))+(2/K(m)));
   fs(m,m+2)=1/(K(m+2).*(S^2));
   fs(m,m-2)=1/(K(m).*(S^2));
   end
end
fb  =(diag((2*S/(3*EI))*ones(1,sp-1))+diag((S/(6*EI))*ones(1,sp-2),1)+diag((S/(6*EI))*ones(1,sp-2),-
1)); %bending component of flexibility matrix
f =fs+fb;                % total flexibility matrix
P=f^(-1);                % stiffness matrix
```

```
for i=1:sp*N+1
x1(i)=fix((i-1)*L*10000)/10000;
x1f(i)=x1(i)-(floor(x1(i)/S))*S;
d_r(1,i)=(((x1f(i).^3)/(6*S))-((x1f(i).^2)/2)+(S*x1f(i)/3))./EI;
d_l(1,i)=((-(x1f(i).^3)/(6*S))+(S*x1f(i)/6))./EI;
for j=1:na % for all six axles
  if x1(i)<d1(na)  % before the last axle  arrives to the track
   a1(i,:)=0;
   b1(i,:)=0;
   D_x(:,i)= 0;
   d_xs3(:,i)=0;
else
  if floor((round(x1(i),4)-d1(j))/S)==(round(x1(i),4)-d1(j))/S % integer, tam kat, load on the support
  a1(i,j)=S;
  b1(i,j)= round(S-a1(i,j),4);
  else
  a1(i,j)=round(x1(i)-d1(j)-(floor((x1(i)-d1(j))/S)*S),4);
  b1(i,j)= round(S-a1(i,j),4);
  end
  if (round((sp-2)*S,4)<=x1(i)) && (x1(i)<round((sp-1)*S,4)) %78S<=x<79S
   d_xs3(sp-3,i)=((sp-1)*S-x1(i))/(K(sp-1)*S^2);
   d_xs3(sp-2,i)=((1/(K(sp)*S))-(((2/K(sp-1))+(1/K(sp)))*(((sp-1)*S-x1(i))/S^2)))+d_r(i);
   d_xs3(sp-1,i)=((1/(K(sp-1)*S))-(((2/K(sp))+(1/K(sp-1)))*((x1(i)-(sp-2)*S)/S^2)))+d_l(i);
elseif (round((sp-1)*S,4)<=x1(i)) && (x1(i)<=round(sp*S,4)) %79S<=x<=80S
   d_xs3(sp-2,i)=(sp*S-x1(i))/(K(sp)*S^2);
   d_xs3(sp-1,i)=((1/(K(sp+1)*S))-(((2/K(sp))+(1/K(sp-1)))*((sp*S-x1(i))/S^2)))+d_r(i);
else
   %d_xs3(floor(x1(i)/S)-1,i)=((floor(x1(i)/S)+1)*S-x1(i))/(K((floor(x1(i)/S)+1))*S^2);
   %d_xs3(floor(x1(i)/S),i)=((1/(K(floor(x1(i)/S)+2)*S))-
(((2/K(floor(x1(i)/S)+1))+(1/K(floor(x1(i)/S)+2)))*(((floor(x1(i)/S)+1)*S-x1(i))/S^2)))+d_l(i);
   %d_xs3(floor(x1(i)/S)+1,i)=((1/(K(floor(x1(i)/S)+1)*S))-
(((2/K(floor(x1(i)/S)+2))+(1/K(floor(x1(i)/S)+1)))*((x1(i)-(floor(x1(i)/S))*S)/S^2)))+d_r(i);
```

```
        %d_xs3(floor(x1(i)/S)+2,i)=(x1(i)-(floor(x1(i)/S))*S)/(K(floor(x1(i)/S)+2)*S^2);
        r(i,1)=floor(x1(i)/S);
        d_xs3(r(i,1)-1,i)=((r(i,1)+1)*S-x1(i))/(K((r(i,1)+1))*S^2);
        d_xs3(r(i,1),i)=((1/(K(r(i,1)+2)*S))-(((2/K(r(i,1)+1))+(1/K(r(i,1)+2)))*(((r(i,1)+1)*S-
x1(i))/S^2)))+d_l(i);
        d_xs3(r(i,1)+1,i)=((1/(K(r(i,1)+1)*S))-(((2/K(r(i,1)+2))+(1/K(r(i,1)+1)))*((x1(i)-
(r(i,1))*S)/S^2)))+d_r(i);
        d_xs3(r(i,1)+2,i)=(x1(i)-(r(i,1))*S)/(K(r(i,1)+2)*S^2);
    end
      if j==na
          if round(x1(i),4)==round(d1(6),4) % En arka aksin pozisyonuna göre D matrisi
          D_x(1,i)=1/(K(1).*S);
          elseif (round(d1(na),4)<x1(i))&& (x1(i)-d1(na)<=round(S,4))  %d6<x<d6+S
              if(x1(i)-d1(na)==round(S,4)) % load on the support, no bending eff.
            D_x(1,i)=(b1(i,j)/(K(1)*S^2))-(2*a1(i,j)/(K(2)*S^2));
            D_x(2,i)=(1./(K(2).*S^2)).*a1(i,j);
              else
            D_x(1,i)=(b1(i,j)/(K(1)*S^2))-(2*a1(i,j)/(K(2)*S^2))+(a1(i,j)*b1(i,j)*(1+(a1(i,j)/S))./(6*EI);
            D_x(2,i)=(1./(K(2).*S^2)).*a1(i,j);
              end
          elseif (x1(i)-d1(na)>round(S,6)) && (x1(i)-d1(na)<round(2*S,6))  %d6+S<x<d6+2S
          D_x(1,i)=(((a1(i,j)/(K(3).*S^2))-
(2*b1(i,j)/(K(2)*S^2))))+((a1(i,j)*b1(i,j)*(1+(b1(i,j)/S))./(6*EI));
          D_x(2,i)=((1./S).*((b1(i,j)/(K(2).*S))-
(2*a1(i,j)/(K(3)*S))))+((a1(i,j)*b1(i,j)*(1+(a1(i,j)/S))./(6*EI));
          D_x(3,i)=(1./(K(3).*S^2)).*a1(i,j);
          elseif (x1(i)-d1(na)>=round(2*S,6)) && ((x1(i)<=round(sp*S,6))) % x<=80S
          p(i,j)=floor((x1(i)-d1(j))/S);
        if floor((round(x1(i),4)-d1(j))/S)==(round(x1(i),4)-d1(j))/S % when a load is on the support, no
bending eff.
          D_x(p(i,j)-1,i)=(a1(i,j)./(K(p(i,j)+1).*(S^2)));
          D_x(p(i,j),i)=-(1/(K(p(i,j)+1).*S^2)).*2*a1(i,j);
```

```
            D_x(p(i,j)+1,i)=(a1(i,j)./(K(p(i,j)+1).*(S^2)));
          else
            D_x(p(i,j)-1,i)=(b1(i,j)/(K(p(i,j)+1).*(S^2)));
            D_x(p(i,j),i)=((a1(i,j)/(K(p(i,j)+2).*(S^2)))-
(2*b1(i,j)/(K(p(i,j)+1).*(S^2))))+((a1(i,j)*b1(i,j)*(1+(b1(i,j)./S))./(6*EI));
            D_x(p(i,j)+1,i)=((b1(i,j)/(K(p(i,j)+1).*(S^2)))-
(2*a1(i,j)/(K(p(i,j)+2).*(S^2))))+((a1(i,j)*b1(i,j)*(1+(a1(i,j)./S))./(6*EI));
            D_x(p(i,j)+2,i)=(1/(K(p(i,j)+2).*(S^2))).*a1(i,j);
          end
           end
      else
         p(i,j)=floor((x1(i)-d1(j))/S);
          if floor(((fix(x1(i)*10000)/10000)-d1(j))/S)==((fix(x1(i)*10000)/10000)-d1(j))/S % when a load is
on the support, no bending eff.
            D_x(p(i,j)-1,i)=(a1(i,j)./(K(p(i,j)+1).*(S^2)));
            D_x(p(i,j),i)=-(1/(K(p(i,j)+1)*(S^2))).*2*a1(i,j);
            D_x(p(i,j)+1,i)=(a1(i,j)./(K(p(i,j)+1).*(S^2)));
          else
            D_x(p(i,j)-1,i)=(b1(i,j)/(K(p(i,j)+1).*(S^2)));
            D_x(p(i,j),i)=((a1(i,j)/(K(p(i,j)+2)*(S^2)))-
(2*b1(i,j)/(K(p(i,j)+1)*(S^2))))+((a1(i,j)*b1(i,j)*(1+(b1(i,j)./S))./(6*EI));
            D_x(p(i,j)+1,i)=((b1(i,j)/(K(p(i,j)+1)*(S^2)))-
(2*a1(i,j)/(K(p(i,j)+2)*(S^2))))+((a1(i,j)*b1(i,j)*(1+(a1(i,j)./S))./(6*EI));
            D_x(p(i,j)+2,i)=(1/(K(p(i,j)+2)*(S^2))).*a1(i,j);
          end
      end
      end
      end
      F=f^(-1)*(-D_x); % solves the eqn. f*F=-D_x
       for c=1:sp-1
          sf(c+1,i)=sf(c,i)+ d_xs3(c,i).*F(c,i); % sigma
       end
```

```
    resp_x(i)=Pa.*sf(sp,i); % displacement response due to 6 axles
end
% ———————— End of Script ————————————
```

B.3 SAP2000 Validation

To verify the MATLAB implementation of the MR-IL displacement model for the track, an equivalent SAP2000 model of a 40 m track on a Winkler (spring) foundation was developed, as illustrated in Figure B.3. Displacement components were then compared at representative locations under identical axle loads (six axles, each 181.43 kN), track properties, and support stiffness values. A summary of the MATLAB–SAP2000 displacement comparison at selected points is given in Table B.1.

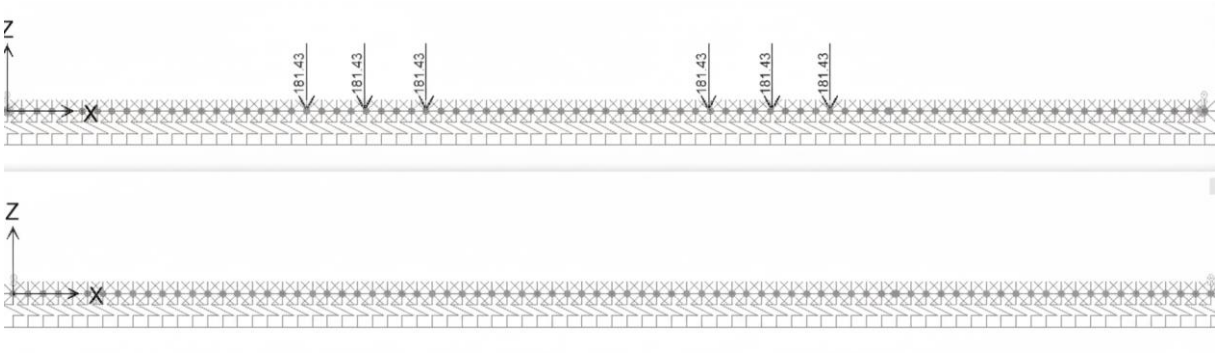


Figure B.3. SAP2000 track model.

Table B.1. Comparison of model results.

x (m)	MATLAB (mm)	SAP2000 (mm)	Relative error (%)
39.875	+0.16	+0.16	0.789
39.25	-0.06	-0.06	0.010
25.125	+0.08	+0.08	1.177

Table B.1 shows a very good agreement between the MATLAB and SAP2000 results. Relative errors (Eq. (A3)) range from 0.010% to 1.177%, with an average of approximately 0.66% across the three checkpoints. The signs and magnitudes are consistent at all locations, and the largest deviation (1.177% at x = 25.125 m) remains well within typical engineering tolerance for independent model verification. Overall, these results support the validity of the MATLAB MR-IL formulation against the SAP2000 reference.

ANNEX C: Cross-Entropy Algorithm to Optimize Pier Stiffness

The present annex details the CE algorithm employed in Section 3.4.5 to optimize pier stiffnesses. The MATLAB code utilised in this study is an adaptation of the code developed by Botev et al. (2013), which was modified to align with the case-study specific requirements of Chapter 3.

```
% — Pier 4, 20 % scoured simulations of RILA front-mounted case
% —Track profile with Track 3 rail irregularities
n=1;                                    % select dimension
Nel = 10; N = 100*n; alpha = 0.8; beta = 0.7; q = 5;
eps = 1e-8;
%mu = 0.5;                              % select initial mu
%mu=0.71;                               % 1st restart
mu=0.721;                               % 2nd restart
sigma = 5;                              % select initial sigma
mu_last = mu;
sigma_last = sigma;
X_best_overall = zeros(1,n); S_best_overall = 1E10;
%— Calling RILA front-mounted track profiles —
opts = detectImportOptions('scoured track_irr-Track3-T1.xlsx');
opts = spreadsheetImportOptions('NumVariables',51);
opts = setvartype(opts,'double');
preview('scoured track_irr-Track3-T1.xlsx',opts);
opts.Sheet = 'Sheet1';  opts.SelectedVariableNames = [1:51];
opts.DataRange = '2:484';
Rtr1=readmatrix('scoured track_irr-Track3-T1.xlsx',opts);
resp_xf=Rtr1(:,:);
resp_xo(:,:)=zeros(length(resp_xf),N);
O=zeros(N,1);
```

```
t = 0;
tic
while sigma > eps
t = t+1;
mu = alpha*mu + (1-alpha)*mu_last;
sigma= alpha*sigma + (1-alpha)*sigma_last;                % fixed smoothing
pd = makedist('Normal', 'mu', mu, 'sigma', sigma);
tr = truncate(pd, 0, 1);                                   % pier stiffness in kN/m will be optimized
rng('shuffle', 'twister');
X = random(tr, N, 1);
[numRows,numCols] = size(Rtr1);
for r=1:N
        resp_xo(:,r)=Fun111rR(X(r));
                for j=1: (numCols-1)
                        for h=1:length(resp_xf)
                        O(r,1)=O(r,1)+((resp_xf(h,j+1)./1000)-resp_xo(h,r)).^2;      % objective function
                        end
                end
end
SA = O;
[S_sort,I_sort] = sort(SA);
S_best = S_sort(1);
if (S_best < S_best_overall)
S_best_overall = S_best; X_best_overall = X(I_sort(1),:);
end
mu_last = mu;
sigma_last = sigma; Xel = X(I_sort(1:Nel),:);
mu = mean(Xel); sigma = std(Xel);
if mod(t,100)==0
fprintf('%d %5.4f',t,S_best);
fprintf(' %5.4f',mu)
fprintf('\n')
```

```
end
end
toc
fprintf('%d %9.8f\n',t,S_best_overall);
fprintf(' %9.8f',X_best_overall)
fprintf('\n')
% ———————— End of Script ———————————
function [resp_xT] = Fun111rR(y)
% MR-IL Response of the bridge, one pier scoured
Pa = 181.43;                    % 1 axle load in kN-compatible with VBI code
sp = 7;                         % number of spans
L  = 0.2;                       % beam element length (m)
Lb = [13.2 14.0 14.0 14.0 14.0 14.0 13.2]; % span lengths (m)
ss = round(sum(Lb)/L);
E  = 3.4342e10;                                 % Young's Modulus (N/m^2)
J  = 0.9028;                                    % Second moment (m^4)
EI = E*J;                                       % Flexural stiffness
na = 6;                                          % number of axles
d1 = [2.0 4.0 6.2 15.2 17.4 19.4];   % axle offsets (m), j=1..6
r  = @(yy) 467966.*yy;                          % scoured-pier stiffness (kN/m) as function of y
k  = [467966; 467966; 467966; 467966; 467966; 467966; 467966; r(y)]; % 6 piers + 2 abutments
% Span edges (supports): t(1)=0, t(2)=Lb1, ..., t(sp+1)=sum(Lb)
t = [0, cumsum(Lb)];
% Output
resp_x = zeros(1, ss+1);
% — Loop along bridge coordinate —
for i = 1:ss+1
    x_global = fix((i-1)*L*100)/100;
    [sx, x_rel, Lx] = locateSpan(x_global, t, Lb);          % response span & local x
    if sx == 0, continue; end                               % outside bridge (safety)
    % Lead-axle travel
    Xlead = (i-11)*L;   % at i=11, first axle at x=0
```

```
d_sum = 0;                                          % superposition over axles (unit load)
for j = 1:na
    % Global axle position at this step
    p = Xlead - (d1(j) - d1(1));
    % If axle not on bridge, it contributes nothing
    if p <= 0 || p >= t(end)
        continue
    end
    % Locate axle span and local coordinate
    [sa, p_rel, La] = locateSpan(p, t, Lb);
    if sa == 0, continue; end
    gap = sx - sa;
    if gap == 0
        % — (A) Same span: bending + support terms —
        % Bending (area product)
        [AAB, ABC, ACD] = triAreas(x_rel, p_rel, Lx);
        d1_bend = (AAB + ABC + ACD) / (EI/1000);
        % Support part via reactions & end settlements, then linear shape
        % Reactions for unit load at p_rel: R_L = (Lx - p_rel)/Lx, R_R = p_rel/Lx
        kL = k(sx);       % left support stiffness of span sx
        kR = k(sx+1);     % right support stiffness of span sx
        a  = p_rel;       % distance to left support
        b  = Lx - p_rel;  % distance to right support
        % Settlements at ends from unit load:
        dL =  b/(kL*Lx);  % left support settlement (unit load)
        dR =  a/(kR*Lx);  % right support settlement (unit load)
        % Deflection along span from end settlements (simply supported):
        d2_sup = dL*(1 - x_rel/Lx) + dR*(x_rel/Lx);
        d_sum = d_sum - (d1_bend + d2_sup);
    elseif gap == 1
        % — (B) One span behind/ahead: support term only (load on left, x on right) —
        % Common joint is right end of sa (== left end of sx)
```

```
            % Reaction at right of sa: R_R = a_prev/La
            a_prev   = p_rel;
            k_joint  = k(sa+1);             % common support stiffness
            delta_c  = (a_prev/La)/k_joint; % common support settlement
            d2_adj   = delta_c * (1 - x_rel/Lx); % propagate into right span
            d_sum    = d_sum - d2_adj;
        elseif gap == -1
            % — (B) One span behind/ahead: support term only (load on right, x on left) —
            % Common joint is left end of sa (== right end of sx)
            % Reaction at left of sa: R_L = b_prev/La
            b_prev   = La - p_rel;
            k_joint  = k(sa);               % common support stiffness
            delta_c  = (b_prev/La)/k_joint; % common support settlement
            d2_adj   = delta_c * (x_rel/Lx); % propagate into left span
            d_sum    = d_sum - d2_adj;
        else
            % — (C) Further away: zero —
        end
    end
    % Multiply by axle load (kN) after unit-load superposition
    resp_x(i) = resp_x(i) + Pa * d_sum;
end
resp_xb = resp_x;
% — Rail irregularities —
opts = detectImportOptions('Rail_irr-Track3-T1.xlsx');
opts = spreadsheetImportOptions('NumVariables',1);
opts = setvartype(opts,'double'); preview('Rail_irr-Track3-T1.xlsx',opts);
opts.Sheet = 'Sheet1'; opts.SelectedVariableNames = 1;
opts.DataRange = '2:484'; RM_3T1 = readmatrix('Rail_irr-Track3-T1.xlsx',opts);  % (m)
resp_xT = transpose(resp_xb) + RM_3T1./1000;
% ———— helpers ————
function [s, xr, Ls] = locateSpan(z, tvec, Lvec)
```

```
        if z < 0 || z > tvec(end)
            s = 0; xr = 0; Ls = 0; return
        end
        s = find(tvec <= z, 1, 'last');
        if s > numel(Lvec)
            s = numel(Lvec); xr = Lvec(s); Ls = Lvec(s); return
        end
        xr = z - tvec(s); Ls = Lvec(s);
        end
    function [AAB,ABC,ACD] = triAreas(x, p, Ls)
        AAB = (x - p).^3 .* (Ls - x + p) .* (Ls - x) / (3*Ls^2);
        ABC = p .* ((Ls - x).*(x - p) .* ( 2*(Ls - x + p).*(x - p) + (Ls - x + p).*x + (x - p).*(Ls - x) + 2*x.*(Ls -
x) )) / (6*Ls^2); ACD = x .* (Ls - x).^3 .* (x - p) / (3*Ls^2);
    end
    end
    % ———————— End of Script ————————————
```

ANNEX D: Deriving Type B features

This annex provides the MATLAB code details for deriving ARX-Based Type B features for scour detection and scour locating ML models that are explained in Section 5.3.1.5.

D.1 ARX–Based Type B Features for Scour Detection

This section the MATLAB code used to generate Type B features for the bridge-level scour detection models. For each train crossing, i.e., simulation, the script assumes that the track profile on the bridge, X, and the corresponding rail-irregularity segment, i.e., u, have already been assembled on a common spatial grid with step 0.2m. An optional step adds 0.5% RMS Gaussian noise to X, reproducing the noise conditions. For each simulation, a single-input single-output ARX model is then fitted with the rail irregularity as the external input and the track profile as the output. Candidate ARX orders are swept up to a maximum of 60, and the optimal model order (with common $n_a = n_b$ and delay $n_k = 1$) is selected by minimising the AIC. The resulting A and B polynomials are truncated to common lengths, re-expressed as coefficient vectors, and standardised to form ARX-coefficient features. In a second step, the corresponding impulse response is derived for each simulation and truncated to a fixed Markov horizon of 40, which was chosen based on an energy-capture analysis, namely 95% cumulative energy across simulations. These Markov parameters are then z-scored and used as the Type B features in the bridge-level scour detection models. For WCM15, WCM15A, and Colonnade the script uses this procedure directly, while for River Teign an additional KNN imputation step is applied.

```
% — Common assumptions for both blocks: —
%   X: [N x Nsims] track profiles on the bridge (mm), one column per simulation
%   u: [N x 1] rail irregularity on the bridge (mm), same spatial grid as X
%   typ: ''RILA-Leading'' or ''RILA-Trailing'' (used for naming only)
%   br_nm: bridge name string (''WCM15'',''WCM15A'',''Colonnade'',''RTeign'')
%   pres_noise: ''Added'' or ''Non'' (0.5% RMS noise on X when ''Added'')
% Outputs in both blocks:
%   X_arx: ARX coefficient features (rows = simulations)
```

```
%   X_arx_z: z-scored ARX features
%   G_ir : Markov / impulse response coefficients g(1..H_ir)
%   X_ir_z: z-scored Markov features
%   mu_arx, sg_arx: ARX feature means and std
%   mu_ir,  sg_ir: Markov feature means and std
%% BLOCK 1 – WCM15, WCM15A, Colonnade
%% ARX + Markov parameters (no KNN on Markov)
% Spatial sampling step along the bridge (m)
Ts = 0.2;
[ad, numSims] = size(X);
%% Optional 0.5% RMS noise on track profiles (as used in the ML experiments)
if pres_noise == "Added"
    perc    = 0.5/100;       % 0.5 %
    col_rms = rms(X,1);      % 1 x Nsims, RMS per simulation
    col_rms(col_rms==0) = 1; % avoid zeros
    noise   = randn(size(X)) .* col_rms .* perc;
    X       = X + noise;     % X now stores noisy track profiles
end
%% Exogenous input (rail irregularity) with one-sample delay (nk = 1)
u_raw = u(:);                % rail irregularity on bridge (mm)
u0    = [0; u_raw(1:end-1)]; % shift by one sample
u0    = u0(:);
%% ARX identification per simulation (AIC-based common order na = nb)
maxOrder = 60;
if ad <= maxOrder + 2
    error('Not enough samples along bridge (ad = %d) for maxOrder = %d.', ad, maxOrder);
end
best_na  = nan(numSims,1);
best_AIC = nan(numSims,1);
A_list   = cell(numSims,1);  % stores [1, a1, a2, ...]
B_list   = cell(numSims,1);  % stores [b1, b2, ...]
opt = arxOptions('InitialCondition','zero','Focus','prediction');
```

```
for j = 1:numSims
    yj = X(:,j);                % track profile for simulation j
    if ~any(isfinite(yj)) || var(yj,'omitnan') < 1e-12
        continue;
    end
    % Demean input and output for numerical conditioning
    yj_d = yj - mean(yj,'omitnan');
    u_d  = u0 - mean(u0,'omitnan');
    % Sweep common order na = nb, nk = 1, and pick minimum AIC
    AICs = nan(maxOrder,1);
    for na = 1:maxOrder
        nb = na;
        try
            mdl = arx(iddata(yj_d, u_d, Ts), [na nb 1], opt);
            AICs(na) = aic(mdl);
        catch
            % leave as NaN if fit fails for this order
        end
    end
    if any(isfinite(AICs))
        [amin, na_star] = min(AICs);
        best_na(j)  = na_star;
        best_AIC(j) = amin;
        mdl = arx(iddata(yj_d, u_d, Ts), [na_star na_star 1], opt);
        Ai = mdl.A(:).';                % [1, a1, a2, ...]
        if iscell(mdl.B)
            Bi = mdl.B{1}(:).';         % [b1, b2, ...]
        else
            Bi = mdl.B(:).';
        end
        A_list{j} = Ai; B_list{j} = Bi;
    end
```

```
end
%% Common-length ARX coefficient features (subset of successful sims)
A_list_all = A_list;
B_list_all = B_list;
valid0  = ~cellfun(@isempty, A_list_all) & ~cellfun(@isempty, B_list_all);
sim_idx = find(valid0);
if ~any(valid0)
    error('ARX identification produced no valid A and B polynomials.');
end
A_list_v = A_list_all(valid0); B_list_v = B_list_all(valid0);
lenA = cellfun(@numel, A_list_v);  % includes leading 1
lenB = cellfun(@numel, B_list_v);
KA = min(lenA) - 1;                % a1..a_KA
KB = min(lenB);                    % b1..b_KB
if KA <= 0 || KB <= 0
    error('Common lengths invalid (KA = %d, KB = %d).', KA, KB);
end
fprintf('Using KA = %d (a1..a_KA) and KB = %d (b1..b_KB) for all valid sims.\n', KA, KB);
numSims_feat = numel(A_list_v);
X_arx = zeros(numSims_feat, KA + KB);
for k = 1:numSims_feat
    Ai = A_list_v{k};          % [1, a1, a2, ...]
    Bi = B_list_v{k};          % [b1, b2, ...]
    X_arx(k,:) = [Ai(2:1+KA), Bi(1:KB)];
end
mu_arx = mean(X_arx, 1);
sg_arx = std (X_arx, 0, 1);
sg_arx(sg_arx < 1e-12) = 1;
X_arx_z = (X_arx - mu_arx) ./ sg_arx;
%% Markov parameters (impulse response) for all simulations
H_ir       = 40;      % fixed Markov horizon
nk_default = 1;
```

```
lambda_fir = 1e-2;    % ridge regularisation for FIR fallback
G_ir           = nan(numSims, H_ir);
succARX        = false(numSims,1);
used_impulseest= false(numSims,1);
used_fir       = false(numSims,1);
u_d = u0 - mean(u0,'omitnan');
for j = 1:numSims
   yj = X(:,j);
   if ~any(isfinite(yj)), continue; end
   if any(~isfinite(yj))
      yj = fillmissing(yj,'linear','EndValues','nearest');
   end
   yj_d = yj - mean(yj,'omitnan');
   got_g = false;
   % 1) ARX-based impulse response if model exists
   if ~isempty(A_list_all{j}) && ~isempty(B_list_all{j})
      Ai = A_list_all{j}; Bi = B_list_all{j};
        try
          mdl = idpoly(Ai, {Bi}, [], [], [], 'Ts', Ts, 'ioDelay', nk_default-1);
          g = impulse(mdl, (0:H_ir-1)*Ts);
          G_ir(j,:) = g(:).';
          succARX(j) = true;
          got_g = true;
      catch
      end
   end
   % 2) Non-parametric impulse response
   if ~got_g
      try
         d = iddata(yj_d, u_d, Ts, 'InterSample','zoh');
         sysIR = impulseest(d, H_ir, 'Regul','TC');
         g = impulse(sysIR, (0:H_ir-1)*Ts);
```

```
            G_ir(j,:) = g(:).';
            used_impulseest(j) = true;
            got_g = true;
        catch
        end
    end
    % 3) Ridge-regularised FIR fallback
    if ~got_g
        N   = length(u_d);
        U   = convmtx(u_d, H_ir);   % Signal Processing Toolbox
        U   = U(1:N, :);
        GtG = U.'*U;
        lam = lambda_fir * trace(GtG)/H_ir;
        lam = max(lam, 0);
        g   = (GtG + lam*eye(H_ir)) \ (U.'*yj_d);
        G_ir(j,:) = g(:).';
        used_fir(j) = true;
    end
end
% Standardise Markov features
mu_ir = mean(G_ir, 1, 'omitnan'); sg_ir = std (G_ir, 0, 1, 'omitnan');
sg_ir(sg_ir < 1e-12) = 1;
%% Energy-based check for the chosen Markov horizon
G_all = G_ir;
[Ns, Hmax] = size(G_all);
Ecum = cumsum(G_all.^2, 2, 'omitnan');
Etot = Ecum(:, end);
zeroEnergy = ~isfinite(Etot) | (Etot <= eps);
Etot(zeroEnergy) = NaN;
frac = Ecum ./ Etot;
target = 0.95;
H_per_sim = nan(Ns,1);
```

```
hit_target = false(Ns,1);
for i = 1:Ns
    fi = frac(i,:);
    if all(~isfinite(fi)), continue; end
    idx = find(fi >= target, 1, 'first');
    if isempty(idx)
        idx = Hmax;
    else
        hit_target(i) = true;
    end
    H_per_sim(i) = idx;
end
H_star = median(H_per_sim, 'omitnan');
if ~isfinite(H_star), H_star = H_ir; end
H_star = round(H_star);
n_zero = sum(zeroEnergy);
n_hits = sum(hit_target & ~zeroEnergy);
n_miss = sum(~hit_target & ~zeroEnergy);
fprintf('Energy-95%% median H* = %d (distance ≈ %.1f m at Ts = 0.2 m)\n', H_star, 0.2*H_star);
fprintf('  stats: hits = %d, misses = %d (forced to H = %d), zero-energy = %d out of %d sims\n', ...
        n_hits, n_miss, Hmax, n_zero, Ns);
%% BLOCK 2 – River Teign
%% ARX + KNN-imputed ARX coefficients + robust Markov parameters with KNN
% Assumptions as above:
%   X, u, typ, br_nm == "RTeign", pres_noise in {"Added","Non"}
Ts = 0.2;
[ad, numSims] = size(X);
%% Optional 0.5% RMS noise on track profiles
if pres_noise == "Added"
    perc    = 0.5/100;
    col_rms = rms(X,1);
    col_rms(col_rms==0) = 1;
```

```
    noise   = randn(size(X)) .* col_rms .* perc;
    X       = X + noise;
end
%% Exogenous input (rail irregularity) with one-sample delay (nk = 1)
u_raw = u(:);
u0    = [0; u_raw(1:end-1)];
u0    = u0(:);
%% ARX identification per simulation (same as in Block 1)
maxOrder = 60;
if ad <= maxOrder + 2
    error('Not enough samples along bridge (ad = %d) for maxOrder = %d.', ad, maxOrder);
end
best_na  = nan(numSims,1); best_AIC = nan(numSims,1);
A_list   = cell(numSims,1); B_list   = cell(numSims,1);
opt = arxOptions('InitialCondition','zero','Focus','prediction');
for j = 1:numSims
    yj = X(:,j);
    if ~any(isfinite(yj)) || var(yj,'omitnan') < 1e-12
        continue;
    end
    yj_d = yj - mean(yj,'omitnan');
    u_d  = u0 - mean(u0,'omitnan');
    AICs = nan(maxOrder,1);
    for na = 1:maxOrder
        nb = na;
        try
            mdl = arx(iddata(yj_d, u_d, Ts), [na nb 1], opt);
            AICs(na) = aic(mdl);
        catch
        end
    end
    if any(isfinite(AICs))
```

```
        [amin, na_star] = min(AICs);
        best_na(j)  = na_star; best_AIC(j) = amin;
        mdl = arx(iddata(yj_d, u_d, Ts), [na_star na_star 1], opt);
        Ai  = mdl.A(:).';
        if iscell(mdl.B)
            Bi = mdl.B{1}(:).';
        else
            Bi = mdl.B(:).';
        end
        A_list{j} = Ai; B_list{j} = Bi;
      end
end
%% Split successful / failed ARX fits and choose common ARX lengths
succ_idx = find(~cellfun(@isempty, A_list) & ~cellfun(@isempty, B_list));
fail_idx = setdiff(1:numSims, succ_idx);
if isempty(succ_idx)
    error('ARX produced no valid A/B on this bridge – nothing to learn from for imputation.');
end
lenA_s = cellfun(@numel, A_list(succ_idx));  % includes leading 1
lenB_s = cellfun(@numel, B_list(succ_idx));
KA = min(lenA_s) - 1;                        % a1..a_KA
KB = min(lenB_s);                            % b1..b_KB
if KA <= 0 || KB <= 0
    error('Common lengths invalid from successes (KA = %d, KB = %d).', KA, KB);
end
fprintf('Impute setup -> KA = %d, KB = %d from %d successful sims.\n', KA, KB, numel(succ_idx));
A_succ = zeros(numel(succ_idx), KA);
B_succ = zeros(numel(succ_idx), KB);
for ii = 1:numel(succ_idx)
    Ai = A_list{succ_idx(ii)};
    Bi = B_list{succ_idx(ii)};
    A_succ(ii,:) = Ai(2:1+KA);
```

```
    B_succ(ii,:) = Bi(1:KB);
end
%% Descriptor features for similarity (used for both ARX and Markov KNN)
% Descriptor per simulation: [std(y), rms(y), corr(y,u0)]
desc_all = nan(numSims, 3);
for j = 1:numSims
    yj = X(:,j);
    if any(~isfinite(yj))
        yj = fillmissing(yj,'linear','EndValues','nearest');
    end
    yj = detrend(yj,1);
    uj = detrend(u0,1);
    sy = std(yj);
    ry = rms(yj);
    c  = corr(yj, uj, 'rows','complete');
    if ~isfinite(c), c = 0; end
    desc_all(j,:) = [sy, ry, c];
end
% Z-score descriptors using successful sims only
mu_d = mean(desc_all(succ_idx,:), 1, 'omitnan');
sg_d = std (desc_all(succ_idx,:), 0, 1, 'omitnan');
sg_d(sg_d < 1e-12) = 1;
Z_succ = (desc_all(succ_idx,:) - mu_d) ./ sg_d;
Z_all  = (desc_all             - mu_d) ./ sg_d;
K      = min(5, size(Z_succ,1));   % number of neighbours
alpha  = 0.7;                      % weight on KNN mean (rest on global median)
% Robust centres and scales for ARX coefficients
A_med = median(A_succ, 1, 'omitnan');
B_med = median(B_succ, 1, 'omitnan');
A_mad = mad(A_succ, 1, 1);  A_mad(A_mad<1e-12) = 1e-12;
B_mad = mad(B_succ, 1, 1);  B_mad(B_mad<1e-12) = 1e-12;
%% KNN imputation for failed ARX fits (coefficients)
```

```
A_list_all      = A_list; B_list_all      = B_list;
imputed_flag_arx= false(numSims,1);
imputed_flag_arx(fail_idx) = true;
for jj = 1:numel(fail_idx)
    j = fail_idx(jj);
    z = Z_all(j,:);
    if any(~isfinite(z))
        a_est = A_med; b_est = B_med;
    else
        D = sqrt(sum((Z_succ - z).^2, 2));
        [~, ord] = sort(D, 'ascend');
        ord   = ord(1:K);
        a_knn = mean(A_succ(ord,:), 1, 'omitnan');
        b_knn = mean(B_succ(ord,:), 1, 'omitnan');
        a_est = alpha*a_knn + (1-alpha)*A_med;
        b_est = alpha*b_knn + (1-alpha)*B_med;
    end
    % Robust clipping
    lowerA = A_med - 6*A_mad; upperA = A_med + 6*A_mad;
    lowerB = B_med - 6*B_mad; upperB = B_med + 6*B_mad;
    a_est  = min(max(a_est, lowerA), upperA);
    b_est  = min(max(b_est, lowerB), upperB);
    % Tiny jitter to avoid identical imputed rows
    phi = (sqrt(5)-1)/2;
    for c = 1:KA
        a_est(c) = a_est(c) + 5e-4*A_mad(c)*sin((j+31*c)*phi*2*pi);
    end
    for c = 1:KB
        b_est(c) = b_est(c) + 5e-4*B_mad(c)*cos((j+37*c)*phi*2*pi);
    end
    A_list_all{j} = [1, a_est]; B_list_all{j} = b_est;
    end
```

```
% Safety fallback for any remaining empties / short vectors
A_mean = mean(A_succ, 1, 'omitnan');  A_mean(~isfinite(A_mean)) = 0;
B_mean = mean(B_succ, 1, 'omitnan');  B_mean(~isfinite(B_mean)) = 0;
for j = 1:numSims
    Ai = A_list_all{j}; Bi = B_list_all{j};
    if isempty(Ai), Ai = [1, A_mean]; end
    if isempty(Bi), Bi = B_mean;      end
    if numel(Ai) < 1 + KA
        Ai = [Ai, zeros(1, 1 + KA - numel(Ai))];
    end
    if numel(Bi) < KB
        Bi = [Bi, zeros(1, KB - numel(Bi))];
    end
    A_list_all{j} = Ai(1:1+KA); B_list_all{j} = Bi(1:KB);
end
%% Final ARX feature matrix (all simulations, consistent size)
X_arx = zeros(numSims, KA + KB);
for j = 1:numSims
    Ai = A_list_all{j};
    Bi = B_list_all{j};
    X_arx(j,:) = [Ai(2:1+KA), Bi(1:KB)];
end
sim_idx = (1:numSims).';
mu_arx = mean(X_arx, 1);
sg_arx = std (X_arx, 0, 1);
sg_arx(sg_arx < 1e-12) = 1;
X_arx_z = (X_arx - mu_arx) ./ sg_arx;
%% Robust Markov parameters for all simulations (no NaNs)
H_ir         = 40; nk_default   = 1;
lambda_fir   = 1e-2; lambda_floor = 1e-6; tol_u_std    = 1e-9;
G_ir           = nan(numSims, H_ir);
method_flag    = strings(numSims,1); succARX_g       = false(numSims,1);
```

```
used_impulseest = false(numSims,1); used_fir        = false(numSims,1);
imputed_flag_ir = false(numSims,1); u_d   = u0 - mean(u0,'omitnan');
u_std = std(u_d,'omitnan');
for j = 1:numSims
    yj = X(:,j);
    if ~any(isfinite(yj))
        G_ir(j,:)      = 0;
        method_flag(j) = ''sanitized'';
        continue;
    end
    if any(~isfinite(yj))
        yj = fillmissing(yj,'linear','EndValues','nearest');
    end
    yj_d = yj - mean(yj,'omitnan');
    % No-excitation guard
    if u_std < tol_u_std
        G_ir(j,:)      = 0;
        method_flag(j) = ''zero-input'';
        continue;
    end
    got_g = false;
    % 1) ARX-based impulse response
    if ~isempty(A_list_all{j}) && ~isempty(B_list_all{j})
        Ai = A_list_all{j}; Bi = B_list_all{j};
        try
            mdl = idpoly(Ai, {Bi}, [], [], [], 'Ts', Ts, 'ioDelay', nk_default-1);
            g = impulse(mdl, (0:H_ir-1)*Ts);
            if any(~isfinite(g)), error('Nonfinite g from ARX-impulse'); end
            G_ir(j,:)      = g(:).';
            succARX_g(j)   = true;
            got_g          = true;
            method_flag(j) = ''arx'';
```

```
    catch
    end
end
% 2) Non-parametric impulse estimate
if ~got_g
    try
        d = iddata(yj_d, u_d, Ts, 'InterSample','zoh');
        sysIR = impulseest(d, H_ir, 'Regul','TC');
        g = impulse(sysIR, (0:H_ir-1)*Ts);
        if any(~isfinite(g)), error('Nonfinite g from impulseest'); end
        G_ir(j,:)      = g(:).';
        used_impulseest(j) = true;
        got_g          = true;
        method_flag(j) = ''impulseest'';
    catch
    end
end
% 3) Ridge-FIR fallback
if ~got_g
    N   = length(u_d);
    U   = convmtx(u_d, H_ir);
    U   = U(1:N, :);
    GtG = U.'*U;
    lam_base = lambda_fir * max(trace(GtG)/H_ir, 0);
    lam      = max(lam_base, lambda_floor);
    g        = (GtG + lam*eye(H_ir)) \ (U.'*yj_d);
    if any(~isfinite(g))
        g = zeros(H_ir,1);
        method_flag(j) = ''sanitized'';
    else
        method_flag(j) = ''fir'';
    end
```

```
        G_ir(j,:)  = g(:).';
        used_fir(j)= true;
    end
    if any(~isfinite(G_ir(j,:)))
        G_ir(j,:)      = 0;
        method_flag(j) = ''sanitized'';
    end
end
%% KNN imputation for Markov parameters (row-wise, descriptor space)
bad_rows  = all(abs(G_ir) <= eps | ~isfinite(G_ir), 2);
good_rows = find(~bad_rows);
bad_list  = find(bad_rows);
if ~isempty(bad_list) && ~isempty(good_rows)
    G_good = G_ir(good_rows,:);
    g_med  = median(G_good, 1, 'omitnan');
    g_mad  = mad(G_good, 1, 1);
    g_mad(g_mad < 1e-12) = 1e-12;
    K_ir     = min(5, numel(good_rows));
    alpha_ir = 0.7;  % blend: KNN vs median
    Z_good = Z_all(good_rows,:);
    for ii = 1:numel(bad_list)
        j = bad_list(ii);
        z = Z_all(j,:);
        if any(~isfinite(z))
            g_est = g_med;
        else
            D = sqrt(sum((Z_good - z).^2, 2));
            [~, ord] = sort(D, 'ascend');
            ord      = ord(1:K_ir);
            g_knn    = mean(G_good(ord,:), 1, 'omitnan');
            g_est    = alpha_ir*g_knn + (1-alpha_ir)*g_med;
        end
```

```
        % Robust clipping
        g_low  = g_med - 6*g_mad;
        g_high = g_med + 6*g_mad;
        g_est  = min(max(g_est, g_low), g_high);
        % Tiny jitter to avoid identical rows
        phi = (sqrt(5)-1)/2;
        for h = 1:H_ir
            g_est(h) = g_est(h) + 5e-4 * g_mad(h) * sin((j + 29*h)*phi*2*pi);
        end
        if any(~isfinite(g_est)), g_est = g_med; end
        G_ir(j,:)        = g_est;
        imputed_flag_ir(j) = true;
        method_flag(j)   = ''imputed-knn'';
    end
end
%% Standardise Markov features and check energy horizon
mu_ir = mean(G_ir, 1, 'omitnan');
sg_ir = std (G_ir, 0, 1, 'omitnan');
sg_ir(sg_ir < 1e-12) = 1;
X_ir_z = (G_ir - mu_ir) ./ sg_ir;
G_all = G_ir;
Hmax  = size(G_all,2);
Ecum  = cumsum(G_all.^2, 2);
Etot  = Ecum(:, end);
Etot_safe = Etot;
Etot_safe(Etot_safe<=eps) = NaN;
frac  = Ecum ./ Etot_safe;
H_per_sim = nan(numSims,1);
for i = 1:numSims
    fi = frac(i,:);
    if all(~isfinite(fi))
        H_per_sim(i) = NaN;
```

```
        else
            k = find(fi >= 0.95, 1, 'first');
            if isempty(k), k = Hmax; end
            H_per_sim(i) = k;
        end
    end
    H_star = median(H_per_sim, 'omitnan');
    fprintf('Energy-95%% median H* = %d (≈ %.1f m @ Ts = 0.2 m). Fixed H_ir = %d used in features.\n',
H_star, 0.2*H_star, H_ir);
    % ———————— End of Script ————————————
```

D.2 ARX–Based Type B Features for Scour Localisation

This section presents the ARX-based scripts for the scour localisation models. The same ARX and Markov procedure with scour detection models is used, but the modelling is now carried out on the scour-affected region around each support, rather than on the full bridge length. For each train crossing and each support, a separate Single-Input Single-Output ARX model is fitted between the rail-irregularity segment acting on that zone and the localised track-profile response, with AIC again used to select the model orders and a unit delay. This gives one ARX model per simulation per support. The corresponding Markov parameter vectors are then computed and truncated to a fixed horizon H_{ir} =13 and are standardised in the same way as for the detection models, so that the feature length is consistent across simulations, supports, and bridges. These per-support Markov descriptors form the Type B feature set used by the scour localisation models.

```
    %% —— Pier-based Markov parameters (Type B features) for scour localisation ——
    % The script assumes that the following variables are already defined in the
    % workspace (by earlier geometry and simulation code):
    %   X    : [ad x numSims] matrix of bridge track profiles (mm)
    %          (columns = simulations, rows = spatial samples along bridge)
    %   u_raw: [ad x 1] rail-irregularity profile on the bridge (mm)
    %   Ts : spatial sampling step (m), e.g. Ts = 0.2 m
    %   t    : [nSup x 1] support coordinates along the bridge (m)
    %   Lb     : [nSpan x 1] span lengths (m)
    %   br_nm  : string scalar with bridge name, e.g. ''WCM15'',''WCM15A'',
```

```
%           ''Colonnade'',''RTeign''
% Output
%   Pier_Feats : table with one row per (simulation, support) and columns:
%                RunID, Support, Method, Imputed, g1...gH_ir
% Z-scoring of Markov parameters is performed later in the ML pipeline.
% Export RAW Markov parameters (impulse responses).
% 1. Markov horizon and windows
[ad, numSims] = size(X);
nSup          = numel(t);               % number of supports = spans + 1
% helper to get support-centred window in index form
getWinIdx = @(s) local_get_window_idx(s, t, Lb, Ts, ad, nSup);
% compute shortest INTERNAL window (exclude abutments) for traceability
if nSup >= 3
   Nw_int = zeros(nSup-2,1);
   for s = 2:nSup-1
      [i1,i2] = getWinIdx(s);
      Nw_int(s-1) = max(1, i2 - i1 + 1);
   end
   Nw_min_int = min(Nw_int);
else
   [i1,i2]    = getWinIdx(1);
   Nw_min_int = max(1, i2 - i1 + 1);
end
% original adaptive rule (kept for logging only)
H_rule_adaptive = min(40, max(5, Nw_min_int - 2));
% final enforced horizon for cross-bridge comparability
H_ir   = 13;                % -> g1..g13 (≈ 2.6 m at Ts = 0.2 m)
H_mode = ''fixed13'';
fprintf('Shortest internal window = %d samples; adaptive rule -> H = %d.\n', ...
      Nw_min_int, H_rule_adaptive);
fprintf('Using fixed Markov horizon H_ir = %d (mode = %s) for all bridges.\n', ...
      H_ir, H_mode);
```

```
% 2. Support naming
switch string(br_nm)
    case ''WCM15''
        supportNames = {'Abutment 1_WCM15','Pier 1_WCM15','Pier 2_WCM15', ...
                    'Pier 3_WCM15','Pier 4_WCM15','Pier 5_WCM15', ...
                    'Pier 6_WCM15','Abutment 2_WCM15'}.';
    case ''WCM15A''
        supportNames = {'Abutment 1A_WCM15A','Pier 1_WCM15A','Pier 2_WCM15A', ...
                    'Pier 3_WCM15A','Pier 4_WCM15A','Pier 5_WCM15A', ...
                    'Pier 6_WCM15A','Pier 7_WCM15A','Pier 8_WCM15A', ...
                    'Abutment 2_WCM15A'}.';
    case ''Colonnade''
        supportNames = {'Abutment 1_Colon','Pier 1_Colon', ...
                    'Pier 2_Colon','Abutment 2_Colon'}.';
    case ''RTeign''
        supportNames = {'Abutment 1_RTeign','Pier 1_RTeign','Pier 2_RTeign', ...
                    'Pier 3_RTeign','Abutment 2_RTeign'}.';
    otherwise
        supportNames = arrayfun(@(k) ''Support ''+k, (1:nSup)', 'UniformOutput', false).';
end
% ensure supportNames length matches nSup
if numel(supportNames) ~= nSup
    warning('Length(t) = %d, #supportNames = %d. Adjusting names.', nSup, numel(supportNames));
    supportNames = supportNames(:);
    if numel(supportNames) < nSup
        supportNames(end+1:nSup,1) = arrayfun(@(k) ''Support ''+k, ...
            (numel(supportNames)+1:nSup)', 'UniformOutput', false);
    end
    supportNames = string(supportNames(1:nSup));
else
    supportNames = string(supportNames(:));
end
```

```
% 3. Input signal (rail irregularity)
% u_raw is the rail irregularity on the bridge (mm)
u_raw   = u_raw(:);
u0      = [0; u_raw(1:end-1)];          % nk = 1: shift by one sample
u_d_full = u0 - mean(u0,'omitnan');     % demeaned for numerical stability
% 4. ARX + fallbacks configuration
hannWin       = @(N) (N>1) .* (0.5 - 0.5*cos(2*pi*(0:N-1)'/(N-1))) + (N==1);
tol_u_std     = 1e-9;     % "no excitation" threshold
lambda_fir    = 1e-2;     % base ridge parameter for FIR
lambda_floor  = 1e-6;     % floor so lambda never goes to zero
maxOrderCap   = 40;       % na = nb sweep cap
% impulse-response sanity and global energy cap
g_peak_max    = 5e3;      % absolute cap on |g(k)|
clip_sigma    = 12;       % energy cap = median + clip_sigma * MAD
E_med_run     = 0; E_mad_run     = 0;        E_cap_default = 1e8;
% robust imputation constants
clip_k     = 6;           % ≈ 4σ in Gaussian terms (since σ ≈ 1.4826*MAD)
jitter_eps = 5e-4;        % tiny deterministic ripple (0.05% of MAD)
jitter_h   = 11;          % small prime harmonic index
phi        = (sqrt(5)-1)/2;  % golden-ratio conjugate
% 5. Allocate result containers
nRows   = numSims * nSup; G_pier  = zeros(nRows, H_ir);
RunID   = zeros(nRows, 1); Supp    = strings(nRows, 1);
Method  = strings(nRows, 1); Imputed = false(nRows, 1);
% 6. Windowed ARX per support
row = 0;
for j = 1:numSims
    y = X(:, j);
    if any(~isfinite(y))
        y = fillmissing(y, 'linear', 'EndValues','nearest');
    end
    y = y - mean(y,'omitnan');   % demean local response
```

```
for s = 1:nSup
    row = row + 1;
    % — support-centred window and Hann taper —
    [i1,i2] = getWinIdx(s);
    if i2 < i1, i2 = i1; end
    Nw  = max(1, i2 - i1 + 1);
    yw  = y(i1:i2);
    uw  = u_d_full(i1:i2);
    w   = hannWin(Nw);
    yw  = yw .* w;
    uw  = uw .* w;
    % default: zero IR if no reliable excitation
    g_row  = zeros(1, H_ir); method = ''zero-input'';
    if std(uw,'omitnan') >= tol_u_std && any(isfinite(yw))
        got_g = false;
        % — ARX with AIC (na = nb, nk = 1) —
        nmax = min(maxOrderCap, floor(Nw/3));
        if nmax >= 1
            try
                AICs     = nan(nmax,1);
                mdl_keep = cell(nmax,1); dloc     = iddata(yw, uw, Ts);
                optA     = arxOptions('InitialCondition','zero', 'Focus','prediction');
                for na = 1:nmax
                    mdl_keep{na} = arx(dloc, [na na 1], optA);
                    AICs(na)     = aic(mdl_keep{na});
                end
                if any(isfinite(AICs))
                    [~, na_star] = min(AICs);
                    mdl  = mdl_keep{na_star}; gtmp = impulse(mdl, (0:H_ir-1)*Ts);
                    if all(isfinite(gtmp))
                        E = sum(gtmp.^2);
                        if E_med_run > 0
```

```
                E_cap = E_med_run + clip_sigma*max(E_mad_run,1e-12);
            else
                E_cap = E_cap_default;
            end
            if max(abs(gtmp)) <= g_peak_max && E <= E_cap
                g_row  = gtmp(:).';
                method = ''arx'';
                got_g  = true;
                % update running robust stats
                E_med_run = 0.9*E_med_run + 0.1*E;
                E_mad_run = 0.9*E_mad_run + 0.1*abs(E - E_med_run);
            end
        end
    end
  catch
    % fall through to fallbacks
  end
end
% — impulseest fallback (non-parametric IR) —
if ~got_g
  try
    sysIR = impulseest(iddata(yw, uw, Ts, 'InterSample','zoh'), ...
                       H_ir, 'Regul','TC');
    gtmp  = impulse(sysIR, (0:H_ir-1)*Ts);
    if all(isfinite(gtmp))
      g_row  = gtmp(:).';
      method = ''impulseest'';
      got_g  = true;
    end
  catch
    % fall through to FIR
  end
```

```
            end
            % — ridge-regularised FIR fallback —
            if ~got_g
                U   = convmtx(uw, H_ir); U = U(1:Nw, :);
                GtG = U.' * U;
                lam = max(lambda_fir * max(trace(GtG)/H_ir, 0), lambda_floor);
                gtmp = (GtG + lam*eye(H_ir)) \ (U.'*yw);
                if any(~isfinite(gtmp))
                    g_row  = zeros(1, H_ir);
                    method = ''sanitized'';
                else
                    g_row  = gtmp(:).';
                    method = ''fir'';
                end
            end
        end
        % — global energy clip —
        E_row = sum(g_row.^2);
        if E_med_run > 0
            E_cap = E_med_run + clip_sigma*max(E_mad_run,1e-12);
        else
            E_cap = E_cap_default;
        end
        if E_row > E_cap && E_row > 0
            g_row = g_row * sqrt(E_cap / E_row);
        end
        G_pier(row,:) = g_row;
        Method(row)   = method;
        RunID(row)    = j;
        Supp(row)     = supportNames(s);
    end
end
```

```
% 7. Row-wise KNN + median imputation
% "Bad" rows: all zero or non-finite impulse responses
bad_rows  = all(~isfinite(G_pier) | abs(G_pier) <= eps, 2);
good_rows = find(~bad_rows);
if ~isempty(good_rows) && any(bad_rows)
    G_good = G_pier(good_rows,:);
    g_med  = median(G_good, 1, 'omitnan');
    g_mad  = mad(G_good, 1, 1); g_mad(g_mad < 1e-12) = 1e-12;
    % descriptors per row: [std(y_w), rms(y_w), corr(y_w, u_w)]
    desc = zeros(nRows, 3);
    row  = 0;
    for j = 1:numSims
        y = X(:, j);
        if any(~isfinite(y))
            y = fillmissing(y,'linear','EndValues','nearest');
        end
        y = y - mean(y,'omitnan');
        for s = 1:nSup
            row = row + 1;
            [i1,i2] = getWinIdx(s);
            yw = y(i1:i2); uw = u_d_full(i1:i2);
            c  = corr(yw, uw, 'rows','complete');
            if ~isfinite(c), c = 0; end
            desc(row,:) = [std(yw,'omitnan'), rms(yw), c];
        end
    end
    mu_d = mean(desc(good_rows,:), 1, 'omitnan');
    sg_d = std (desc(good_rows,:), 0, 1, 'omitnan'); sg_d(sg_d<1e-12) = 1;
    Z_all  = (desc - mu_d) ./ sg_d;
    Z_good = Z_all(good_rows,:);
    K     = min(5, numel(good_rows));  % KNN size
    alpha = 0.7;
```

```
    bad_list = find(bad_rows);
    for ii = 1:numel(bad_list)
        r = bad_list(ii);
        z = Z_all(r,:);
        if any(~isfinite(z))
            g_est = g_med;
        else
            D = sqrt(sum((Z_good - z).^2, 2));
            [~, ord] = sort(D, 'ascend');
            ord    = ord(1:K);
            g_knn  = mean(G_good(ord,:), 1, 'omitnan');
            g_est  = alpha*g_knn + (1-alpha)*g_med;
        end
        % robust clipping coordinate-wise
        g_low  = g_med - clip_k * g_mad;
        g_high = g_med + clip_k * g_mad;
        g_est  = min(max(g_est, g_low), g_high);
        % tiny deterministic jitter to avoid identical long runs
        for h = 1:H_ir
            g_est(h) = g_est(h) + jitter_eps * g_mad(h) * ...
                                  sin((r + jitter_h*h) * phi * 2*pi);
        end
        if any(~isfinite(g_est))
            g_est = g_med;
        end
        G_pier(r,:) = g_est;
        Method(r)   = ''imputed-knn'';
        Imputed(r)  = true;
    end
end
% 8. Pack into a table (RAW only)
Pier_Feats = table(RunID, Supp, Method, Imputed, ...
```

```
                'VariableNames', {'RunID','Support','Method','Imputed'});
for h = 1:H_ir
    Pier_Feats.(''g''+h) = G_pier(:,h);
end
% meta-info can be printed or stored separately if desired
fprintf('Pier-based Markov features (RAW) built for %d simulations × %d supports.\n', ...
        numSims, nSup);
fprintf('Methods used: ARX = %d, impulseest = %d, FIR = %d, zero-input = %d, imputed = %d.\n', ...
    sum(Method==''arx''), sum(Method==''impulseest''), sum(Method==''fir''), ...
    sum(Method==''zero-input''), sum(Method==''imputed-knn''));
%  Helper: support-centred window function
function [i1,i2] = local_get_window_idx(s, t, Lb, Ts, ad, nSup)
%LOCAL_GET_WINDOW_IDX  Return index window around support s.
%   Window extends ±(span/4) on each side, clamped to [1, ad].
    if s > 1
        Lleft = Lb(max(1, s-1)) / 4;
    else
        Lleft = 0;
    end
    if s < nSup
        Lright = Lb(min(numel(Lb), s)) / 4;
    else
        Lright = 0;
    end
    x1 = t(s) - Lleft;
    x2 = t(s) + Lright;
    i1 = max(1, min(ad, round(x1 / Ts) + 1)); i2 = max(1, min(ad, round(x2 / Ts) + 1));
        if i2 < i1
        i2 = i1;
    end
end
```

ANNEX E: MATLAB Codes for Feature Selection Methods

This section provides the MATLAB code details of two feature selection methods: RI-based & grid search and NSGA-II explained in section 5.3.2 for both scour detection and scour locating-ML models.

E.1 RI-Based Feature Selection & Grid Search for Scour Detection ML

```
% Requires in workspace:
%   x_frs      : N x P feature matrix (all Type A + Type B parameters)
%   y_frs      : N x 1 labels (''Yes'',''No'')
%   Scr_frs    : N x 1 scour percentages (for logging only)
%   input_name : 1 x P cell array of feature names
%   relative_importance : 1 x P vector from perturbation-based RI analysis
%   netOpts    : cell array of options for fitcnet (fixed ANN architecture)
%   bestParams : struct with .Leading / .Trailing (kept for consistency)
%   typ        : ''RILA-Leading'' or ''RILA-Trailing''
% — Grid over number of retained features (1..P) —
num_original_parameters = numel(input_name);
feature_counts          = 1:num_original_parameters;
results_tbl = table('Size',[0 8], ...
   'VariableTypes', {'double','double','double','double','double','double','double','double'}, ...
   'VariableNames', {'n_feat','F1_Yes','F1_No','BalancedAcc','MCC', ...
                     'Accuracy','TrainAccuracy','Gap'});
% — Sort features by relative importance (descending) —
[sorted_imp, sorted_idx] = sort(relative_importance, 'descend');
sorted_input_names       = input_name(sorted_idx);
for n_feat = feature_counts
    % — Select top-n_feat most important features —
    selected_idx        = sorted_idx(1:n_feat);
```

```
selected_input_names = input_name(selected_idx);
X_reduced           = x_frs(:, selected_idx);
% — External 5-fold cross-validation on reduced set —
n_foldext = 5;
cvext     = cvpartition(y_frs, 'KFold', n_foldext);
all_y_testtrl   = [];
all_y_predtrl   = [];
all_x_testtrl   = [];
SCR_test_alltrl = [];
for foldext = 1:cvext.NumTestSets
    trainIdxext  = cvext.training(foldext);
    valIdxexttrl = cvext.test(foldext);
    x_traintrl = zscore(X_reduced(trainIdxext, :));
    y_traintrl = y_frs(trainIdxext);
    x_testtrl  = zscore(X_reduced(valIdxexttrl, :));
    y_testtrl  = y_frs(valIdxexttrl);
    SCR_test_alltrl = [SCR_test_alltrl; Scr_frs(valIdxexttrl,:)];
    % Fixed ANN (hyperparameters given by netOpts)
    if typ == "RILA-Leading"
        p = bestParams.Leading(foldext);
    else
        p = bestParams.Trailing(foldext);
    end
    trlMdl = fitcnet(x_traintrl, y_traintrl, netOpts{:});
    % Predictions on outer test fold
    y_pred = predict(trlMdl, x_testtrl);
    all_y_testtrl = [all_y_testtrl; y_testtrl];
    all_y_predtrl = [all_y_predtrl; string(y_pred)];
    all_x_testtrl = [all_x_testtrl; x_testtrl];
end
% — Pooled confusion matrix & metrics over all outer folds —
accuracytrl = mean(all_y_testtrl == all_y_predtrl);
```

```
    CM_alltrl   = confusionmat(all_y_testtrl, all_y_predtrl, 'Order', {'No','Yes'});
    TN = CM_alltrl(1,1); FP = CM_alltrl(1,2);
    FN = CM_alltrl(2,1); TP = CM_alltrl(2,2);
    precision_yes = TP / max(TP + FP, eps);
    recall_yes    = TP / max(TP + FN, eps);
    precision_no  = TN / max(TN + FN, eps);
    recall_no     = TN / max(TN + FP, eps);
    F1_yes = 2 * precision_yes * recall_yes / max(precision_yes + recall_yes, eps);
    F1_no  = 2 * precision_no  * recall_no  / max(precision_no  + recall_no,  eps);
    TPR         = recall_yes;
    TNR         = TN / max(TN + FP, eps);
    BalancedAcc = 0.5 * (TPR + TNR);
    MCC = (TP*TN - FP*FN) / max(sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)), eps);
    % — Train accuracy on full reduced data (diagnostic generalisation gap) —
    X_all       = zscore(X_reduced);
    trlMdl_all  = fitcnet(X_all, y_frs, netOpts{:});
    pred_train  = predict(trlMdl_all, X_all);
    trainAcc    = mean(pred_train == y_frs);
    train_val_gap = trainAcc - accuracytrl;
    fprintf('\n[n_feat=%d] Acc=%.4f | F1(Yes)=%.4f | F1(No)=%.4f | TrainAcc=%.4f | Gap=%.4f\n', ...
        n_feat, accuracytrl, F1_yes, F1_no, trainAcc, train_val_gap);
    % Append row for this n_feat
    results_tbl = [results_tbl; {n_feat, F1_yes, F1_no, BalancedAcc, ...
                                 MCC, accuracytrl, trainAcc, train_val_gap}];
end
%% === Select best number of features (safety-first rule) ===
minF1Yes = 0.85;  % floor on F1 for scoured class
minF1No  = 0.80;  % floor on F1 for healthy class
tol      = 1e-12;
% MacroF1 (mean of F1 for Yes/No)
if ~ismember('MacroF1', results_tbl.Properties.VariableNames)
    results_tbl.MacroF1 = (results_tbl.F1_Yes + results_tbl.F1_No) / 2;
```

```
    end
    % Phase A: enforce both floors
    feasible = results_tbl( ...
        (results_tbl.F1_Yes >= (minF1Yes - tol)) & ...
        (results_tbl.F1_No  >= (minF1No  - tol)), :);
    if ~isempty(feasible)
        % Among feasible sets, prefer largest n_feat, then best metrics
        maxN  = max(feasible.n_feat);
        candN = feasible(feasible.n_feat == maxN, :);
        sortVars = {'MacroF1','BalancedAcc','MCC'};
        sortDir  = {'descend','descend','descend'};
        if ismember('Gap', candN.Properties.VariableNames)
            candN.negAbsGap = -abs(candN.Gap);
            sortVars{end+1} = 'negAbsGap';
            sortDir{end+1}  = 'descend';
        end
        candN   = sortrows(candN, sortVars, sortDir);
        best_row = candN(1,:);
        rule_used = sprintf('largest n_{feat} with F1_Yes >= %.2f and F1_No >= %.2f; then maximise
MacroF1', minF1Yes, minF1No);
    else
        % Phase B: enforce F1_Yes floor only
        cand = results_tbl(results_tbl.F1_Yes >= (minF1Yes - tol), :);
        if ~isempty(cand)
            maxN  = max(cand.n_feat);
            candN = cand(cand.n_feat == maxN, :);
            sortVars = {'MacroF1','BalancedAcc','MCC'};
            sortDir  = {'descend','descend','descend'};
            if ismember('Gap', candN.Properties.VariableNames)
                candN.negAbsGap = -abs(candN.Gap);
                sortVars{end+1} = 'negAbsGap';
                sortDir{end+1}  = 'descend';
```

```
        end
        candN   = sortrows(candN, sortVars, sortDir);
        best_row = candN(1,:);
        warning('No subset met both floors (F1_Yes >= %.2f and F1_No >= %.2f). Using F1_Yes floor
only.', minF1Yes, minF1No);
        rule_used = sprintf('largest n_{feat} with F1_Yes >= %.2f; then maximise MacroF1', minF1Yes);
    else
        % Phase C: no floors met – select by metrics only
        bestF1 = max(results_tbl.F1_Yes);
        cand   = results_tbl(abs(results_tbl.F1_Yes - bestF1) <= tol, :);
        cand   = sortrows(cand, {'MacroF1','BalancedAcc','MCC','n_feat'}, ...
                          {'descend','descend','descend','descend'});
        best_row = cand(1,:);
        warning('No subset reached F1_Yes >= %.2f. Selecting by F1_Yes → MacroF1 → BalancedAcc
→ MCC.', ...
                minF1Yes);
        rule_used = 'best F1_Yes then MacroF1, BalancedAcc, MCC (no floors met)';
    end
end
best_n_feat = best_row.n_feat;
fprintf('\nOptimal number of features (rule: %s): %d\n', rule_used, best_n_feat);
disp(best_row);
if ismember('Gap', best_row.Properties.VariableNames) && abs(best_row.Gap) > 0.10
    warning('Potential overfitting: |Gap| = %.3f > 0.10.', best_row.Gap);
end
```

E.2 RI-Based Feature Selection & Grid Search for Scour Locating ML

```
% Requires in workspace:
%   x_frs, y_frs, Scr_frs, input_name, netOpts, bestParams, typ
%   all_x_test, all_y_test, finalMdl14 / finalMdl15 from localisation ANN
%   (same notation as bridge-level model, but now at pier level)
%% — Step 1: perturbation-based relative importance (RI) —
```

```
N           = 100;                  % number of perturbation samples per feature
num_features = size(x_frs, 2);
perturbedInput_all = zeros(N, num_features);
res   = zeros(N, num_features);
y_new = strings(N, num_features);
for i = 1:num_features
    for j = 1:N
        rng(''shuffle'');
        rand_idx         = randi(size(all_x_test, 1));
        perturbed_value  = all_x_test(rand_idx, i);
        perturbed_sample = mean(all_x_test, 1);
        perturbed_sample(i) = perturbed_value;
        % Use final cross-validated ANN from localisation model
        if typ == ''RILA-Leading''
            pred = predict(finalMdl14, perturbed_sample);
        else
            pred = predict(finalMdl15, perturbed_sample);
        end
        perturbedInput_all(j,i) = perturbed_value;
        y_new(j,i)              = all_y_test(rand_idx);
        res(j,i)                = strcmp(pred, y_new(j,i));
    end
end
variance_per_param  = var(perturbedInput_all);
total_variance      = sum(variance_per_param);
relative_importance = variance_per_param ./ max(total_variance, eps);
T_RI = table(input_name', relative_importance', ...
             'VariableNames', {'Parameter','RelativeImportance'});
disp('— Relative importance of input parameters (pier-level) —');
disp(T_RI);
%% — Step 2: grid search over number of features (1..P) —
num_original_parameters = numel(input_name);
```

```
feature_counts          = 1:num_original_parameters;
results_tbl = table('Size',[0 8], ...
   'VariableTypes', {'double','double','double','double','double','double','double','double'}, ...
   'VariableNames', {'n_feat','F1_Yes','F1_No','BalancedAcc','MCC', ...
                     'Accuracy','TrainAccuracy','Gap'});
[sorted_imp, sorted_idx] = sort(relative_importance, 'descend');
sorted_input_names       = input_name(sorted_idx);
for n_feat = feature_counts
    selected_idx = sorted_idx(1:n_feat);
    X_reduced    = x_frs(:, selected_idx);
    % 5-fold external CV on reduced feature set
    n_foldext = 5;
    cvext     = cvpartition(y_frs, 'KFold', n_foldext);
    all_y_testtrl   = []; all_y_predtrl   = [];
    all_x_testtrl   = []; SCR_test_alltrl = [];
    for foldext = 1:cvext.NumTestSets
        trainIdxext  = cvext.training(foldext);
        valIdxexttrl = cvext.test(foldext);
        x_traintrl = zscore(X_reduced(trainIdxext, :));
        y_traintrl = y_frs(trainIdxext);
        x_testtrl  = zscore(X_reduced(valIdxexttrl, :));
        y_testtrl  = y_frs(valIdxexttrl);
        SCR_test_alltrl = [SCR_test_alltrl; Scr_frs(valIdxexttrl,:)];
        if typ == "RILA-Leading"
            p = bestParams.Leading(foldext);
        else
            p = bestParams.Trailing(foldext);
        end
        trlMdl = fitcnet(x_traintrl, y_traintrl, netOpts{:});
        y_pred = predict(trlMdl, x_testtrl);
        all_y_testtrl = [all_y_testtrl; y_testtrl];
        all_y_predtrl = [all_y_predtrl; string(y_pred)];
```

```
        all_x_testtrl = [all_x_testtrl; x_testtrl];
    end
    % Confusion matrix and metrics
    accuracytrl = mean(all_y_testtrl == all_y_predtrl);
    CM_alltrl   = confusionmat(all_y_testtrl, all_y_predtrl, 'Order', {'No','Yes'});
    TN = CM_alltrl(1,1); FP = CM_alltrl(1,2);
    FN = CM_alltrl(2,1); TP = CM_alltrl(2,2);
    precision_yes = TP / max(TP + FP, eps);
    recall_yes    = TP / max(TP + FN, eps);
    precision_no  = TN / max(TN + FN, eps);
    recall_no     = TN / max(TN + FP, eps);
    F1_yes = 2 * precision_yes * recall_yes / max(precision_yes + recall_yes, eps);
    F1_no  = 2 * precision_no  * recall_no  / max(precision_no  + recall_no,  eps);
    TPR         = recall_yes;
    TNR         = TN / max(TN + FP, eps);
    BalancedAcc = 0.5 * (TPR + TNR);
    MCC = (TP*TN - FP*FN) / max(sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)), eps);
    % Train accuracy and generalisation gap
    X_all       = zscore(X_reduced);
    trlMdl_all  = fitcnet(X_all, y_frs, netOpts{:});
    pred_train  = predict(trlMdl_all, X_all);
    trainAcc    = mean(pred_train == y_frs);
    train_val_gap = trainAcc - accuracytrl;
    fprintf('\n[PIER  n_feat=%d]  Acc=%.4f  |  F1(Yes)=%.4f  |  F1(No)=%.4f  |  TrainAcc=%.4f  |
Gap=%.4f\n', ...
        n_feat, accuracytrl, F1_yes, F1_no, trainAcc, train_val_gap);
    results_tbl = [results_tbl; {n_feat, F1_yes, F1_no, BalancedAcc, ...
                                 MCC, accuracytrl, trainAcc, train_val_gap}];
end
%% === Select best number of features ===
minF1Yes = 0.85; minF1No  = 0.80; tol      = 1e-12;
if ~ismember('MacroF1', results_tbl.Properties.VariableNames)
```

```
        results_tbl.MacroF1 = (results_tbl.F1_Yes + results_tbl.F1_No) / 2;
    end
    feasible = results_tbl( ...
        (results_tbl.F1_Yes >= (minF1Yes - tol)) & ...
        (results_tbl.F1_No  >= (minF1No  - tol)), :);
    if ~isempty(feasible)
        maxN  = max(feasible.n_feat);
        candN = feasible(feasible.n_feat == maxN, :);
        sortVars = {'MacroF1','BalancedAcc','MCC'};
        sortDir  = {'descend','descend','descend'};
        if ismember('Gap', candN.Properties.VariableNames)
            candN.negAbsGap = -abs(candN.Gap);
            sortVars{end+1} = 'negAbsGap';
            sortDir{end+1}  = 'descend';
        end
        candN   = sortrows(candN, sortVars, sortDir);
        best_row = candN(1,:);
        rule_used = sprintf('largest n_{feat} with F1_Yes >= %.2f and F1_No >= %.2f; then maximise
MacroF1', minF1Yes, minF1No);
    else
        cand = results_tbl(results_tbl.F1_Yes >= (minF1Yes - tol), :);
        if ~isempty(cand)
            maxN  = max(cand.n_feat);
            candN = cand(cand.n_feat == maxN, :);
            sortVars = {'MacroF1','BalancedAcc','MCC'};
            sortDir  = {'descend','descend','descend'};
            if ismember('Gap', candN.Properties.VariableNames)
                candN.negAbsGap = -abs(candN.Gap);
                sortVars{end+1} = 'negAbsGap';
                sortDir{end+1}  = 'descend';
            end
            candN   = sortrows(candN, sortVars, sortDir);
```

```
        best_row = candN(1,:);
        warning('[PIER] No subset met both floors. Using F1_Yes floor only.');
        rule_used = sprintf('largest n_{feat} with F1_Yes >= %.2f; then maximise MacroF1', minF1Yes);
    else
        bestF1 = max(results_tbl.F1_Yes);
        cand   = results_tbl(abs(results_tbl.F1_Yes - bestF1) <= tol, :);
        cand = sortrows(cand, {'MacroF1','BalancedAcc','MCC','n_feat'},
{'descend','descend','descend','descend'});
        best_row = cand(1,:);
        warning('[PIER] No floor met. Selecting by F1_Yes → MacroF1 → BA → MCC.');
        rule_used = 'best F1_Yes then MacroF1, BalancedAcc, MCC (no floors met)';
    end
end
best_n_feat = best_row.n_feat;
fprintf('\n[PIER] Optimal number of features (rule: %s): %d\n', rule_used, best_n_feat);
disp(best_row);
if ismember('Gap', best_row.Properties.VariableNames) && abs(best_row.Gap) > 0.10
    warning('[PIER] Potential overfitting: |Gap| = %.3f > 0.10.', best_row.Gap);
end
```

E.3 NSGA-II Feature Selection for Scour Detection ML models

```
% Requires in workspace:
%   x_frs (N x P)      : feature matrix (Type A + Type B)
%   y_frs (N x 1)      : labels ('Yes'/'No')
%   Scr_frs            : scour percentages (used only for logging/export)
%   input_name (1 x P) : feature names
%   num_typeA          : number of Type A (DWT-based) features
%   netOpts            : cell array of options for fitcnet (fixed ANN)
%   typ                : ''RILA-Leading'' or ''RILA-Trailing''
%   bridgeNames2       : list of bridge identifiers
%   BridgeDataAllSorted.(br).x/.y/.Scr : per-bridge data (calendar-ordered)
%% — Mode toggle and stopping criteria —
```

```
fastMode = true;   % true: 3-fold, 40 generations; false: 5-fold, 60 generations
if fastMode
    n_foldext      = 3; MaxGenerations = 40; MaxStallGens   = 10;
else
    n_foldext      = 5; MaxGenerations = 60; MaxStallGens   = 15;
end
%% — Feature typing (Type A = first num_typeA, Type B = remaining) —
P         = size(x_frs, 2);
typeA_idx = 1:num_typeA;
typeB_idx = (num_typeA+1):P;
minTypeA  = 0;   % set > 0 to enforce a minimum number of Type A features
minTypeB  = 0;   % set > 0 to enforce a minimum number of Type B features
%% — GA variable setup —
nVars = P;                % one gene per feature, real in [0,1] thresholded > 0.5
lb    = zeros(1, nVars);
ub    = ones(1, nVars);
% Fix CV folds once for all evaluations
cvext = cvpartition(y_frs, 'KFold', n_foldext);
[trainMask, testMask] = local_build_folds(cvext);
%% — Objective and constraints —
% Objective: minimise [1 - F1_Yes, (#selected features / total)]
objFun = @(mask) local_nsga_obj_fun(mask, x_frs, y_frs, trainMask, testMask, netOpts, typ);
if (minTypeA > 0) || (minTypeB > 0)
    nonlcon = @(mask) local_nsga_type_constraints(mask, typeA_idx, typeB_idx, minTypeA,
minTypeB);
else
    nonlcon = [];
end
%% — NSGA-II options —
PopSize = 100;
```

```
opts = optimoptions('gamultiobj', 'PopulationSize', PopSize, 'MaxGenerations', MaxGenerations, 'MaxStallGenerations', MaxStallGens, 'FunctionTolerance', 1e-3, 'CrossoverFraction', 0.8, 'UseParallel', true, 'Display', 'iter', 'PlotFcn', @gaplotpareto);
% Output function for logging (see nsga_log_outfcn in annex)
logBase = sprintf('NSGAII_%s', typ);
opts = optimoptions(opts, ...
    'OutputFcn', @(options,state,flag) nsga_log_outfcn(options,state,flag,logBase));
%% — Run NSGA-II —
[pop_masks, pop_fvals, exitflag, output] = gamultiobj( ...
    objFun, nVars, [], [], [], [], lb, ub, nonlcon, opts);
% pop_fvals: K x 2, col1 = (1 - F1_Yes), col2 = (#features / total)
%% — Post-selection: F1_Yes >= 0.85, then prefer largest subset —
minF1         = 0.85;
F1_from_pop   = 1 - pop_fvals(:, 1);
nfeat_from_pop = sum(pop_masks > 0.5, 2);
ok = F1_from_pop >= minF1;
if any(ok)
    cand = find(ok);
    [~, mx] = max(nfeat_from_pop(ok));
    best_mask_real = pop_masks(cand(mx), :);
else
    bestF1 = max(F1_from_pop);
    tie    = abs(F1_from_pop - bestF1) <= 1e-12;
    cand   = find(tie);
    [~, mx] = max(nfeat_from_pop(tie));
    best_mask_real = pop_masks(cand(mx), :);
    warning('No solution reached F1_Yes >= %.2f. Selected best available Pareto solution.', minF1);
end
best_mask            = best_mask_real > 0.5;
important_idx        = find(best_mask);
important_input_names = input_name(important_idx);
best_n_feat          = numel(important_idx);
```

```
fprintf('\nNSGA-II selection: n_feat = %d (rule: largest n_feat with F1_Yes >= %.2f)\n', ...
        best_n_feat, minF1);
%% — Build reduced per-bridge dataset with selected features —
BridgeDataSelected = struct();
for b = 1:numel(bridgeNames2)
    br   = bridgeNames2{b};
    Xb   = BridgeDataAllSorted.(br).x;
    Yb   = BridgeDataAllSorted.(br).y;
    Scrb = BridgeDataAllSorted.(br).Scr;
    BridgeDataSelected.(br).x   = Xb(:, important_idx);
    BridgeDataSelected.(br).y   = Yb;
    BridgeDataSelected.(br).Scr = Scrb;
end
BridgeDataSelected.meta.selected_idx   = important_idx;
BridgeDataSelected.meta.selected_names = important_input_names(:);
%% — Post-hoc metrics for the selected NSGA-II mask —
% Uses the same CV folds (trainMask/testMask) and hyperparameters (netOpts)
% Guard against variables named 'max' or 'eps'
if exist('max','var'); clear max; end
if exist('eps','var'); clear eps; end
bmax = @(a,b) builtin('max', a, b);   % safe max
Xsel = x_frs(:, important_idx);
K    = numel(trainMask);
all_true = [];
all_pred = [];
for k = 1:K
    tr = trainMask{k}; te = testMask{k}; Xtr = zscore(Xsel(tr,:));
    ytr = y_frs(tr); Xte = zscore(Xsel(te,:)); yte = y_frs(te);
    mdl  = fitcnet(Xtr, ytr, netOpts{:});
    yhat = predict(mdl, Xte);
    all_true = [all_true; yte]; all_pred = [all_pred; string(yhat)];
end
```

```
order    = {'No','Yes'};
all_true = categorical(all_true, order); all_pred = categorical(all_pred, order);
CM = confusionmat(all_true, all_pred, 'Order', order);
if ~isequal(size(CM), [2 2])
   tmp = zeros(2,2);
   tmp(1:size(CM,1), 1:size(CM,2)) = CM;
   CM = tmp;
end
TN = CM(1,1); FP = CM(1,2);
FN = CM(2,1); TP = CM(2,2);
Accuracy = (TP + TN) / bmax(sum(CM(:)), 1);
PrecY  = TP / bmax(TP + FP, 1e-12);
RecY   = TP / bmax(TP + FN, 1e-12);
F1_Yes = 2 * PrecY * RecY / bmax(PrecY + RecY, 1e-12);
PrecN = TN / bmax(TN + FN, 1e-12);
RecN  = TN / bmax(TN + FP, 1e-12);
F1_No = 2 * PrecN * RecN / bmax(PrecN + RecN, 1e-12);
BalancedAcc = 0.5 * (RecY + RecN);
den = sqrt( bmax(TP+FP,1e-12) * bmax(TP+FN,1e-12) * ...
           bmax(TN+FP,1e-12) * bmax(TN+FN,1e-12) );
MCC = ((TP*TN) - (FP*FN)) / bmax(den, 1e-12);
fprintf('\n=== Metrics for selected NSGA-II mask (%d features, bridge-level) ===\n', ...
        numel(important_idx));
fprintf('Accuracy     : %.4f\n', Accuracy);
fprintf('BalancedAcc  : %.4f\n', BalancedAcc);
fprintf('MCC          : %.4f\n', MCC);
fprintf('F1(Yes)      : %.4f  (Prec=%.4f, Rec=%.4f)\n', F1_Yes, PrecY, RecY);
fprintf('F1(No)       : %.4f  (Prec=%.4f, Rec=%.4f)\n', F1_No, PrecN, RecN);
disp('Confusion Matrix [rows: True No/Yes ; cols: Pred No/Yes]:');
disp(CM);
%% — Helper functions —
function [trainMask, testMask] = local_build_folds(cvobj)
```

```
    K = cvobj.NumTestSets;
    trainMask = cell(K,1); testMask  = cell(K,1);
    for k = 1:K
        trainMask{k} = cvobj.training(k); testMask{k}  = cvobj.test(k);
    end
end
function f = local_nsga_obj_fun(maskReal, X, y, trainMask, testMask, netOpts, typ)
    % Returns [1 - F1_Yes, p/P]. Thresholds maskReal > 0.5 to select features.
    m = maskReal > 0.5; p = sum(m); P = size(X,2);
     if p == 0
        f = [1.0, 1.0];
        return;
    end
    Xsel = X(:, m); all_y_true = []; all_y_pred = [];
    K = numel(trainMask);
    for k = 1:K
        tr = trainMask{k}; te = testMask{k}; Xtr = zscore(Xsel(tr, :)); ytr = y(tr);
       Xte = zscore(Xsel(te, :)); yte = y(te); mdl  = fitcnet(Xtr, ytr, netOpts{:});
       yhat = predict(mdl, Xte);
       all_y_true = [all_y_true; yte]; all_y_pred = [all_y_pred; string(yhat)];
    end
    CM = confusionmat(all_y_true, all_y_pred, 'Order', {'No','Yes'});
    TN = CM(1,1); FP = CM(1,2);
    FN = CM(2,1); TP = CM(2,2);
    precY = TP / max(TP + FP, eps);
    recY  = TP / max(TP + FN, eps);
    F1Y   = 2 * precY * recY / max(precY + recY, eps);
    f = [1 - F1Y, p / P];  % both objectives to be minimised
end
function [c, ceq] = local_nsga_type_constraints(maskReal, typeA_idx, typeB_idx, minA, minB)
    m = maskReal > 0.5;
    c = [];
```

```
    if minA > 0
        c(end+1) = (minA - sum(m(typeA_idx)));
    end
    if minB > 0
        c(end+1) = (minB - sum(m(typeB_idx)));
    end
    ceq = [];
end
```

E.4 NSGA-II Feature Selection for Scour Locating-ML Models

```
% Requires in workspace:
%   x_frs (N x P)      : feature matrix (Type A + Markov parameters)
%   y_frs (N x 1)      : labels ('Yes'/'No')
%   Scr_frs            : scour percentages (used only for logging/export)
%   input_name (1 x P) : feature names
%   num_typeA          : number of Type A (DWT-based) features
%   typ                : ''RILA-Leading'' or ''RILA-Trailing''
%   BridgeDataAllfin.(br).x/.y/.Scr/.Snm : pier-level data per bridge
rng(42,'twister');  % reproducibility
%% — Mode toggle and stopping criteria —
fastMode = true;   % true: 3-fold, 40 generations; false: 5-fold, 60 generations
if fastMode
    n_foldext      = 3; MaxGenerations = 40; MaxStallGens   = 10;
else
    n_foldext      = 5; MaxGenerations = 60; MaxStallGens   = 15;
    end
%% — Feature typing (Type A = first num_typeA, Markov = remaining) —
P         = size(x_frs, 2);
typeA_idx = 1:num_typeA;
typeB_idx = (num_typeA+1):P;
minTypeA  = 0;
minTypeB  = 0;
```

```
%% — GA variable setup —
nVars = P;
lb    = zeros(1, nVars);
ub    = ones(1, nVars);
% Fix CV folds once for all evaluations
cvext = cvpartition(y_frs, 'KFold', n_foldext);
[trainMask, testMask] = local_build_folds(cvext);
%% — Fixed ANN template (local to NSGA-II run) —
% Standardize is disabled in fitcnet; z-score is applied manually.
p = struct('LayerSizes',[50 20], ...
           'Lambda',1e-4, ...
           'Standardize', false, ...
           'WeightInit','glorot', ...
           'BiasInit','zeros', ...
           'Activation','tanh');
netOpts = {'LayerSizes',  p.LayerSizes, 'Lambda', p.Lambda,. 'Standardize', p.Standardize, ...
   'LayerWeightsInitializer', p.WeightInit, 'LayerBiasesInitializer',.BiasInit, 'Activations',
p.Activation, 'ClassNames', {'No','Yes'}};
%% — Objective and constraints —
% Objectives to minimise: [1 - F1_Yes, (#features / total)]
objFun = @(mask) local_nsga_obj_fun_pier(mask, x_frs, y_frs, trainMask, testMask, netOpts);
if (minTypeA > 0) || (minTypeB > 0)
   nonlcon = @(mask) local_nsga_type_constraints(mask, typeA_idx, typeB_idx, minTypeA,
minTypeB);
else
   nonlcon = [];
end
%% — NSGA-II options —
PopSize = 100;
opts = optimoptions('gamultiobj', ...
```

```
    'PopulationSize', PopSize, 'MaxGenerations', MaxGenerations, 'MaxStallGenerations', MaxStallGens, 'FunctionTolerance', 1e-3, 'CrossoverFraction', 0.8, 'UseParallel', true, 'Display', 'iter', 'PlotFcn', @gaplotpareto);
    logBase = sprintf('NSGAII_%s_PIER', char(typ));
    opts = optimoptions(opts, ...
        'OutputFcn', @(options,state,flag) nsga_log_outfcn(options,state,flag,logBase));
    %% — Run NSGA-II —
    [pop_masks, pop_fvals, exitflag, output] = gamultiobj( ...
        objFun, nVars, [], [], [], [], lb, ub, nonlcon, opts);
    % pop_fvals: K x 2, col1 = (1 - F1_Yes), col2 = (#features / total)
    %% — Post-selection: F1_Yes >= 0.85, then largest subset —
    minF1         = 0.85;
    F1_from_pop   = 1 - pop_fvals(:, 1);
    nfeat_from_pop = sum(pop_masks > 0.5, 2);
    ok = F1_from_pop >= minF1;
    if any(ok)
        cand = find(ok);
        [~, mx] = max(nfeat_from_pop(ok));
        best_mask_real = pop_masks(cand(mx), :);
    else
        bestF1 = max(F1_from_pop);
        tie    = abs(F1_from_pop - bestF1) <= 1e-12;
        cand   = find(tie);
        [~, mx] = max(nfeat_from_pop(tie));
        best_mask_real = pop_masks(cand(mx), :);
        warning('No solution reached F1_Yes >= %.2f. Selected best available Pareto solution.', minF1);
    end
    best_mask            = best_mask_real > 0.5;
    important_idx        = find(best_mask);
    important_input_names = input_name(important_idx);
    best_n_feat          = numel(important_idx);
    fprintf('\nNSGA-II (pier-level) selection: n_feat = %d (rule: largest n_feat with F1_Yes >= %.2f)\n', ...
```

```
        best_n_feat, minF1);
%% — Build selected pier-based dataset per bridge (preserve Snm) —
BridgeDataSelected = struct();
bridgeListFS = fieldnames(BridgeDataAllfin);
for b = 1:numel(bridgeListFS)
    br   = bridgeListFS{b}; Xb   = BridgeDataAllfin.(br).x;
    Yb   = BridgeDataAllfin.(br).y; Scrb = BridgeDataAllfin.(br).Scr;
    BridgeDataSelected.(br).x   = Xb(:, important_idx); BridgeDataSelected.(br).y   = Yb;
    BridgeDataSelected.(br).Scr = Scrb;
    if isfield(BridgeDataAllfin.(br), 'Snm')
        BridgeDataSelected.(br).Snm = BridgeDataAllfin.(br).Snm;
    end
end
BridgeDataSelected.meta.selected_idx   = important_idx;
BridgeDataSelected.meta.selected_names = important_input_names(:);
%% — Post-hoc metrics for the selected NSGA-II mask (pier-level) —
% Uses same CV folds and ANN template, with training statistics for z-scoring.
if exist('max','var'); clear max; end
if exist('eps','var'); clear eps; end
bmax = @(a,b) builtin('max', a, b);
Xsel = x_frs(:, important_idx);
K    = numel(trainMask);
all_true = [];
all_pred = [];
for k = 1:K
    tr = trainMask{k};
    te = testMask{k};
    [Ztr, mu, sg] = zscore(Xsel(tr,:));
    sg(sg == 0)   = 1;
    Zte = bsxfun(@rdivide, bsxfun(@minus, Xsel(te,:), mu), sg);
    ytr = y_frs(tr); yte = y_frs(te);
    mdl  = fitcnet(Ztr, ytr, netOpts{:});
```

```
    yhat = predict(mdl, Zte);
    all_true = [all_true; yte]; all_pred = [all_pred; string(yhat)];
    end
order    = {'No','Yes'};
all_true = categorical(all_true, order);
all_pred = categorical(all_pred, order);
CM = confusionmat(all_true, all_pred, 'Order', order);
if ~isequal(size(CM), [2 2])
    tmp = zeros(2,2);
    tmp(1:size(CM,1), 1:size(CM,2)) = CM;
    CM = tmp;
end
TN = CM(1,1); FP = CM(1,2);
FN = CM(2,1); TP = CM(2,2);
Accuracy = (TP + TN) / bmax(sum(CM(:)), 1);
PrecY  = TP / bmax(TP + FP, 1e-12);
RecY   = TP / bmax(TP + FN, 1e-12);
F1_Yes = 2 * PrecY * RecY / bmax(PrecY + RecY, 1e-12);
PrecN = TN / bmax(TN + FN, 1e-12);
RecN  = TN / bmax(TN + FP, 1e-12);
F1_No = 2 * PrecN * RecN / bmax(PrecN + RecN, 1e-12);
BalancedAcc = 0.5 * (RecY + RecN);
den = sqrt( bmax(TP+FP,1e-12) * bmax(TP+FN,1e-12) * ...
           bmax(TN+FP,1e-12) * bmax(TN+FN,1e-12) );
MCC = ((TP*TN) - (FP*FN)) / bmax(den, 1e-12);
fprintf('\n=== Metrics for selected NSGA-II mask (%d features, pier-level) ===\n', ...
        numel(important_idx));
fprintf('Accuracy     : %.4f\n', Accuracy);
fprintf('BalancedAcc  : %.4f\n', BalancedAcc);
fprintf('MCC          : %.4f\n', MCC);
fprintf('F1(Yes)      : %.4f  (Prec=%.4f, Rec=%.4f)\n', F1_Yes, PrecY, RecY);
fprintf('F1(No)       : %.4f  (Prec=%.4f, Rec=%.4f)\n', F1_No, PrecN, RecN);
```

```
disp('Confusion Matrix [rows: True No/Yes ; cols: Pred No/Yes]:');
disp(CM);
%% — Local helper functions —
function [trainMask, testMask] = local_build_folds(cvobj)
    K = cvobj.NumTestSets;
    trainMask = cell(K,1);
    testMask  = cell(K,1);
    for k = 1:K
        trainMask{k} = cvobj.training(k);
        testMask{k}  = cvobj.test(k);
    end
end
function f = local_nsga_obj_fun_pier(maskReal, X, y, trainMask, testMask, netOpts)
    % Returns [1 - F1_Yes, p/P] for a given feature mask.
    m = maskReal > 0.5;
    p = sum(m); P = size(X,2);
        if p == 0
        f = [1.0, 1.0];
        return;
    end
    Xsel = X(:, m); all_y_true = [];  all_y_pred = [];
          K = numel(trainMask);
    for k = 1:K
        tr = trainMask{k}; te = testMask{k}; Xtr = zscore(Xsel(tr, :));
        ytr = y(tr); Xte = zscore(Xsel(te, :)); yte = y(te);
        mdl = fitcnet(Xtr, ytr, netOpts{:}); yhat = predict(mdl, Xte);
        all_y_true = [all_y_true; yte]; all_y_pred = [all_y_pred; string(yhat)];
    end
    CM = confusionmat(all_y_true, all_y_pred, 'Order', {'No','Yes'});
    TN = CM(1,1); FP = CM(1,2); FN = CM(2,1); TP = CM(2,2);
    precY = TP / max(TP + FP, eps); recY  = TP / max(TP + FN, eps);
    F1Y   = 2 * precY * recY / max(precY + recY, eps); f = [1 - F1Y, p / P];
```

```
end
function [c, ceq] = local_nsga_type_constraints(maskReal, typeA_idx, typeB_idx, minA, minB)
    m = maskReal > 0.5;
    c = [];
    if minA > 0
        c(end+1) = (minA - sum(m(typeA_idx)));
    end
    if minB > 0
        c(end+1) = (minB - sum(m(typeB_idx)));
    end
    ceq = [];
end
```

ANNEX F: MATLAB Code for Sliding-Window Batch Construction

This annex contains the MATLAB code of the sliding windows method calculations in section 5.5 that are used for constructing batches of scour detection and scour localization ML models.

F.1 Sliding-Window Construction for Scour Detection ML models

```
%% ==== Sliding-window construction for scour detection (bridge-level) ====
% Assumes:
% - bridgeNames2          : cell array of bridge names (used in detection model)
% - BridgeDataAllSorted   : struct with fields per bridge:
%       .x   -> [Ns x pr_all] per-crossing feature matrix
%       .y   -> [Ns x 1] labels (''Yes''/''No'') per crossing (bridge-level)
%       .Scr -> [Ns x nSupp] scour % per support per crossing
% - ma                    : sliding-window length (number of crossings per batch, ws)
% - num_pr, num_pr_arx    : number of Type-A and Type-B features per crossing
% Sliding window parameters
pr_all = num_pr + num_pr_arx;        % pr_all : total number of input features per crossing (n_feat)
windowResults = struct();
for i = 1:numel(bridgeNames2)
    bridgeName = bridgeNames2{i};
    % — Extract data for this bridge —
    xC   = BridgeDataAllSorted.(bridgeName).x;   % xC : [Ns x pr_all] per-crossing features
    yC   = BridgeDataAllSorted.(bridgeName).y;   % yC : [''Yes''/''No''] per crossing
    ScrC = BridgeDataAllSorted.(bridgeName).Scr; % ScrC : [Ns x nSupp] scour % per support per crossing
    Ns = length(yC);                % Ns : number of crossings (simulations) for this bridge
    F  = Ns - ma + 1;               % F : number of sliding windows F_w = N_sim - ws + 1
```

```
% X   : [F x (ma + 4)*pr_all]
%       Batch feature matrix. For each window z:
%       - first 4*pr_all columns: mean, std, min, max over the ws crossings
%       - next ma*pr_all columns: raw features of all ws crossings stacked.
% SCR : [F x nSupp]
%       Batch scour matrix. Each row is the scour % vector over supports
%       from the last crossing in the window.
% B_Y : [F x 1]
%       Numeric batch labels (1 = "No", 2 = "Yes") taken from the last crossing.
% window_x   : [ma x pr_all x F]
%       Raw features stored per window (for inspection/debug).
% window_y   : [ma x 1 x F]
%       Raw labels per crossing inside each window (numeric).
% window_Scr : [ma x nSupp x F]
%       Raw scour% per support per crossing inside each window.
X          = zeros(F, (ma + 4) * pr_all);              % X : batch-level input features
SCR        = zeros(F, size(ScrC, 2));                  % SCR : last-day scour% vector per batch
B_Y        = zeros(F, 1);                              % B_Y : numeric batch labels
window_x   = zeros(ma, pr_all, F);
window_y   = zeros(ma, 1, F);
window_Scr = zeros(ma, size(ScrC, 2), F);
% — Convert "Yes"/"No" to numeric (1 = healthy, 2 = scoured) —
A_y = zeros(length(yC), 1);
A_y(strcmp(yC, "Yes")) = 2;
A_y(strcmp(yC, "No"))  = 1;
% — Build sliding windows along the simulation axis —
for z = 1:F
    % indices of crossings in this window: j = z, ..., z+ma-1
    window_y(:,1,z)   = A_y(z:z+ma-1);      % labels of crossings in window z
    window_Scr(:,:,z) = ScrC(z:z+ma-1,:);   % scour% per support in window z
    % Batch-level scour%: vector from the last crossing in the window
    SCR(z,:) = window_Scr(end,:,z);         % SCR(z,:) = s(z+ma-1,:)
```

```
% Batch label: label of the last crossing in the window
if window_y(end,1,z) == 2
    B_Y(z,1) = 2;                    % scoured
else
    B_Y(z,1) = 1;                    % healthy
end
% Optional safety fix (used during debugging): if any B_Y==0, derive from last-day Scr%.
% In the cleaned data used in the thesis this branch should never be triggered.
if any(B_Y == 0)
    lastScr = squeeze(window_Scr(end,1,:));   % [F x 1], first support's last-day scour%
    lastScr = lastScr(:);
    B_Y(B_Y == 0 & lastScr >= 5) = 2;
    B_Y(B_Y == 0 & lastScr <  5) = 1;
    fprintf('Fixed %d window labels that were 0 by last-day Scr%%.\n', nnz(B_Y == 0));
end
% Store raw features for this window
for o = 1:pr_all
    window_x(:,o,z) = xC(z:z+ma-1,o);   % feature o over ws crossings
end
% — Aggregated statistics: mean, std, min, max over ws crossings —
for jStat = 1:4                      % 1:mean, 2:std, 3:min, 4:max
    for o = 1:pr_all
        colIdx = (jStat - 1)*pr_all + o;
        vals   = window_x(:,o,z);
        if     jStat == 1
            X(z,colIdx) = mean(vals);
        elseif jStat == 2
            X(z,colIdx) = std(vals);
        elseif jStat == 3
            X(z,colIdx) = min(vals);
        else
            X(z,colIdx) = max(vals);
```

```
            end
        end
    end
    % — Raw features stacked after the statistics —
    % r(z) = [x(z)^T, x(z+1)^T, ..., x(z+ws-1)^T]^T
    for p = 1:ma
        for k = 1:pr_all
            idx      = 4*pr_all + (p-1)*pr_all + k;
            X(z,idx) = xC(z + p - 1, k);
        end
    end
end
% — Convert numeric labels back to "Yes"/"No" —
Y = strings(F,1);               % Y : [F x 1] batch labels ("Yes"/"No")
Y(B_Y == 2) = "Yes"; Y(B_Y == 1) = "No";
% — Store per-bridge windowed data —
windowResults.(bridgeName).XW         = X;          % XW : windowed features for this bridge
windowResults.(bridgeName).YW         = Y;          % YW : windowed labels
windowResults.(bridgeName).SCRW       = SCR;        % SCRW : last-day scour% vectors
windowResults.(bridgeName).B_YW       = B_Y;        % B_YW : numeric labels
windowResults.(bridgeName).window_x   = window_x;   % optional: raw window features
windowResults.(bridgeName).window_y   = window_y;   % optional: raw window labels
windowResults.(bridgeName).window_Scr = window_Scr; % optional: raw window scour%
end
%% — Combine all sliding windows from all bridges (for detection classifier) —
x_combined   = [];   % combined X across bridges
y_combined   = [];   % combined Y across bridges
Scr_combined = [];   % combined SCR across bridges
for i = 1:numel(bridgeNames2)
    bridge = bridgeNames2{i};
    x_combined   = [x_combined; windowResults.(bridge).XW];
    y_combined   = [y_combined; windowResults.(bridge).YW];
```

```
    Scr_combined = [Scr_combined; windowResults.(bridge).SCRW];
end
% Final bridge-level detection inputs (assigned to base workspace):
assignin('base', 'X',   x_combined);   % X : [N_w x (ma+4)*pr_all] batch feature matrix
assignin('base', 'Y',   y_combined);   % Y : [N_w x 1] batch labels (''No''/''Yes'')
assignin('base', 'SCR', Scr_combined); % SCR : [N_w x nSupp] last-day scour% per support
```

F.2 Sliding-Window Construction for Scour Localisation ML models

```
% Assumes:
% - bridgeNames           : cell array of bridge names (used in localisation model)
% - BridgeDataAllfin      : struct with fields per bridge:
%       .x   -> [N_sim * nSupp x pr_all] stacked per-support features
%       .y   -> [N_sim * nSupp x 1] labels (''Yes''/''No'') per support per crossing
%       .Scr -> [N_sim * nSupp x 1] scour% per support per crossing
%       .Snm -> [N_sim * nSupp x 1] support names (same for all crossings of a given support)
% - rowsPerSimMap         : struct mapping bridge name -> number of supports (nSupp)
% - ma                    : sliding-window length (number of crossings per batch, ws)
% - num_pr, num_pr_arx    : number of Type-A and Type-B features per support
% Sliding window parameters
pr_all = num_pr + num_pr_arx;         % pr_all : total number of input features per support (n_feat)
windowResults = struct();
for i = 1:numel(bridgeNames)
    bridgeName = bridgeNames{i};
    % — Extract stacked support-level data for this bridge —
    xC   = BridgeDataAllfin.(bridgeName).x;    % xC : [N_sim*nSupp x pr_all]
    yC   = BridgeDataAllfin.(bridgeName).y;    % yC : [''Yes''/''No''] per support per crossing
    ScrC = BridgeDataAllfin.(bridgeName).Scr;  % ScrC : [N_sim*nSupp x 1] scour% per support per
crossing
    SnmC = BridgeDataAllfin.(bridgeName).Snm;  % SnmC : [N_sim*nSupp x 1] support names
    rowsPerSim = rowsPerSimMap.(bridgeName);   % rowsPerSim : number of supports nSupp
    nSim       = floor(size(xC, 1) / rowsPerSim);  % nSim : total number of crossings
    F          = nSim - ma + 1;                % F : number of sliding windows along j
```

```
numPiers = rowsPerSim;                  % numPiers : nSupp for this bridge
% ——————————————————————————
% Each sample corresponds to (window z, support p).
% Total samples per bridge: F * numPiers.
% X   : [F*numPiers x (ma + 4)*pr_all]
%       For each (z,p) we store:
%       - 4*pr_all stats (mean, std, min, max) over ws crossings of that support
%       - ma*pr_all raw feature vectors of that support across ws crossings.
% SCR : [F*numPiers x 1]
%       Scour% of that support at the last crossing in the window.
% B_Y : [F*numPiers x 1]
%       Numeric batch labels (1 = "No", 2 = "Yes") for that support.
% SNM : [F*numPiers x 1]
%       Support name for each sample (window z, support p).
% window_x   : [ma x pr_all x (F*numPiers)]
%       Raw features for that support over ws crossings (per sample).
% window_y   : [ma x 1 x (F*numPiers)]
%       Raw labels for that support over ws crossings.
% window_Scr : [ma x 1 x (F*numPiers)]
%       Raw scour% for that support over ws crossings.
X          = zeros(F * numPiers, (ma + 4) * pr_all); % X : batch features per (z,p)
SCR        = zeros(F * numPiers, 1);                % SCR : last-day scour% per (z,p)
B_Y        = zeros(F * numPiers, 1);                % B_Y : numeric labels per (z,p)
SNM        = strings(F * numPiers, 1);              % SNM : support names per (z,p)
window_x   = zeros(ma, pr_all, F * numPiers);
window_y   = zeros(ma, 1, F * numPiers);
window_Scr = zeros(ma, 1, F * numPiers);
% — Convert "Yes"/"No" to numeric —
A_y = zeros(size(yC));
A_y(strcmp(yC, "Yes")) = 2;     % 2 = scoured
A_y(strcmp(yC, "No"))  = 1;     % 1 = healthy
sampleIdx = 1;                  % global index for (window z, support p)
```

```
% Each sampleIdx corresponds to a specific support p and window starting at sim z.
for z = 1:F                    % loop over window positions
    for p = 1:rowsPerSim       % loop over supports
        % Collect ws crossings for this support p inside window z
        for w = 1:ma           % w = position inside window (b in text)
            simIdx = z + w - 1;                  % simulation index j
            rowIdx = (simIdx - 1)*rowsPerSim + p; % row in stacked matrices xC,yC,ScrC
            window_x(w,:,sampleIdx)   = xC(rowIdx, :);   % features of support p at crossing j
            window_y(w,1,sampleIdx)   = A_y(rowIdx);     % label of support p at crossing j
            window_Scr(w,1,sampleIdx) = ScrC(rowIdx);    % scour% of support p at crossing j
        end
        % Batch-level scour% and label from last crossing in the window
        SCR(sampleIdx) = window_Scr(end,1,sampleIdx);    % s(z+ma-1,p)
        SNM(sampleIdx) = SnmC(rowIdx);                   % support name snm(p)
        if window_y(end,1,sampleIdx) == 2
            B_Y(sampleIdx) = 2;
        else
            B_Y(sampleIdx) = 1;
        end
        % — Aggregated stats over ws crossings for this support —
        for jStat = 1:4          % 1:mean, 2:std, 3:min, 4:max
            for o = 1:pr_all
                colIdx = (jStat - 1)*pr_all + o;
                vals   = window_x(:, o, sampleIdx);
                if     jStat == 1
                    X(sampleIdx, colIdx) = mean(vals);
                elseif jStat == 2
                    X(sampleIdx, colIdx) = std(vals);
                elseif jStat == 3
                    X(sampleIdx, colIdx) = min(vals);
                else
                    X(sampleIdx, colIdx) = max(vals);
```

```
                end
            end
        end
        % — Raw features stacked after the statistics —
        % r(z,p) = [x(z,p)^T, ..., x(z+ws-1,p)^T]^T
        for w = 1:ma
            for o = 1:pr_all
                idx             = 4*pr_all + (w - 1)*pr_all + o;
                X(sampleIdx,idx) = window_x(w, o, sampleIdx);
            end
        end
        sampleIdx = sampleIdx + 1;   % move to next (z,p) pair
    end
end
% — Convert numeric labels back to ''Yes''/''No'' —
Y = strings(F * numPiers, 1);      % Y : [F*nSupp x 1] batch labels (''Yes''/''No'')
Y(B_Y == 2) = ''Yes'';
Y(B_Y == 1) = ''No'';
% — Store per-bridge windowed localisation data —
windowResults.(bridgeName).XW         = X;          % XW : features per (window, support)
windowResults.(bridgeName).YW         = Y;          % YW : labels per (window, support)
windowResults.(bridgeName).SCRW       = SCR;        % SCRW : scour% per (window, support)
windowResults.(bridgeName).B_YW       = B_Y;        % B_YW : numeric labels per (window, support)
windowResults.(bridgeName).SNMW       = SNM;        % SNMW : support names per (window,
support)
windowResults.(bridgeName).window_x   = window_x;   % optional: raw window features
windowResults.(bridgeName).window_y   = window_y;   % optional: raw window labels
windowResults.(bridgeName).window_Scr = window_Scr; % optional: raw window scour%
end
%% — Combine all sliding windows from all bridges (for localisation classifier) —
x_combined   = [];   % combined X_b across bridges
y_combined   = [];   % combined Y_b across bridges
```

```
Scr_combined = [];   % combined SCR_b across bridges
Snm_combined = [];   % combined SNM_b across bridges
for i = 1:numel(bridgeNames)
    bridge = bridgeNames{i};
    x_combined   = [x_combined;   windowResults.(bridge).XW];
    y_combined   = [y_combined;   windowResults.(bridge).YW];
    Scr_combined = [Scr_combined; windowResults.(bridge).SCRW];
    Snm_combined = [Snm_combined; windowResults.(bridge).SNMW];
end
% Final support-level localisation inputs:
X_b   = x_combined;             % X_b   : [N_w*nSupp x (ma+4)*pr_all] batch features
Y_b   = string(y_combined);     % Y_b   : [N_w*nSupp x 1] labels (''No''/''Yes'')
SCR_b = Scr_combined;           % SCR_b : [N_w*nSupp x 1] last-day scour% per support
SNM_b = Snm_combined;           % SNM_b : [N_w*nSupp x 1] support names per sample
```

ANNEX G: MATLAB code for SMOTE-based Oversampling

This annex summarises the MATLAB implementation of SMOTE-based oversampling used to balance the “healthy” and “scoured” classes in the scour detection and scour localisation models. In both cases, the minority class is oversampled using a SMOTE helper function called smote6 for detection, and smote8 for localisation, with k = 5 nearest neighbours and Hamming distance in feature space.

G.1 Scour Detection Models: SMOTE Balancing

For the scour detection models, each row of the training data corresponds to a full crossing, with feature matrix X, class labels Y (''No''/''Yes''), where “No” denotes healthy and “Yes” denotes scoured classes, and scour percentages SCR:

```
Num_Yes = sum(Y == ''Yes'');
Num_No  = sum(Y == ''No'');
if Num_No / Num_Yes > 1.05
    disp('Imbalanced data - Generating ''Yes'' samples using SMOTE');
    % Minority class: ''Yes''
    X_minority   = X(Y == ''Yes'', :);
    Scr_minority = SCR(Y == ''Yes'', :);
    % No support names needed at bridge level
    Z_minority = strings(size(Scr_minority, 1), 1);  % Dummy
    num_synthetic = Num_No - Num_Yes;
    k = 5;
    [X_synthetic, y_synthetic, Scr_synthetic, ~] = smote6(X_minority, Scr_minority(:, 1), [], num_synthetic, k, ''Yes'');
    % Append synthetic samples
    X   = [X;  X_synthetic]; Y   = [Y;  y_synthetic];
    SCR = [SCR; [Scr_synthetic, 0.9 * Scr_synthetic, 1.1 * Scr_synthetic]];
elseif Num_Yes / Num_No > 1.05
```

```
    disp('Imbalanced data - Generating "No" samples using SMOTE, no of minority samples:');
    % Minority class: "No"
    X_minority   = X(Y == "No", :);
    disp(size(X_minority, 1));
    Scr_minority = SCR(Y == "No", :);
    Z_minority   = strings(size(Scr_minority, 1), 1);  % Dummy
    num_synthetic = Num_Yes - Num_No;
    k = 5;
    [X_synthetic, y_synthetic, Scr_synthetic, ~] = smote6( X_minority, Scr_minority(:, 1), [], num_synthetic, k, "No");
    % Append synthetic samples
    X   = [X;  X_synthetic];
    Y   = [Y;  y_synthetic];
    SCR = [SCR; [Scr_synthetic, 0.9 * Scr_synthetic, 1.1 * Scr_synthetic]];
else
    disp('Yes / No data are already balanced. No SMOTE applied.');
end
% Re-check class balance
fprintf('After balancing: Yes = %d, No = %d\n', sum(Y == "Yes"), sum(Y == "No"));
disp('Frequency table of SMOTE-balanced Y:');
tabulate
```

G.2 Scour Detection Models: SMOTE Balancing

For the localisation models, each row of the training data corresponds to a specific support on a specific crossing, with feature matrix X_b, labels Y_b, scour percentages SCR_b, and support names SNM_b. The SMOTE step is:

```
Num_Yes = sum(Y_b == "Yes"); Num_No  = sum(Y_b == "No");
if Num_No / max(Num_Yes, 1) > 1.05
    disp('Imbalanced data - Generating "Yes" samples using SMOTE');
    % Minority class: "Yes"
    X_minority = X_b(Y_b == "Yes", :); Scr_min    = SCR_b(Y_b == "Yes", :);
       Z_min      = SNM_b(Y_b == "Yes", :);   % support names
```

```
    num_synth = Num_No - Num_Yes;
    k = 5;
    [X_syn, y_syn, Scr_syn, Z_syn] = smote8( X_minority, Scr_min(:, 1), Z_min, num_synth, k,''Yes'');
    X   = [X_b;   X_syn]; Y   = [Y_b;   y_syn];
    SCR = [SCR_b; Scr_syn]; SNM = [SNM_b; Z_syn];
elseif Num_Yes / max(Num_No, 1) > 1.05
    disp('Imbalanced data - Generating ''No'' samples using SMOTE');
    % Minority class: ''No''
    X_minority = X_b(Y_b == ''No'', :);  Scr_min    = SCR_b(Y_b == ''No'', :);
    Z_min      = SNM_b(Y_b == ''No'', :);   % support names
    num_synth = Num_Yes - Num_No;
    k = 5;
    [X_syn, y_syn, Scr_syn, Z_syn] = smote8(X_minority, Scr_min(:, 1), Z_min, num_synth, k, ''No'');
    X   = [X_b;   X_syn]; Y   = [Y_b;   y_syn];
    SCR = [SCR_b; Scr_syn]; SNM = [SNM_b; Z_syn];
else
    disp('Yes / No data are already balanced. No SMOTE applied.');
    % Carry through original data if no balancing is needed
    X   = X_b; Y   = Y_b; SCR = SCR_b; SNM = SNM_b;
  end
fprintf('After balancing: Yes = %d, No = %d\n', sum(Y == ''Yes''), sum(Y == ''No''));
```

G.3 SMOTE Helper functions

Both helper functions implement the same SMOTE logic: for each minority sample, one of its k nearest minority neighbours is chosen using Hamming distance, and a synthetic sample is generated by interpolation in feature space. Scour percentages are drawn uniformly from three pre-defined bins that depend on the target class, and (for localisation) support names are inherited from either the base or neighbour sample.

G.3.1 Bridge-level SMOTE (no support names)

```
function [X_synthetic, y_synthetic, Scr_synthetic, Z_synthetic] = smote6(X_minority, Scr_minority, Z_minority, num_synthetic, k, targetClass)
    [n_samples, n_features] = size(X_minority);
```

```
% Handle optional Z_minority (unused at bridge level)
use_Z = exist('Z_minority', 'var') && ~isempty(Z_minority);
% Find k-nearest neighbours in feature space (Hamming distance)
[indices, ~] = knnsearch(X_minority, X_minority, ...
    'K', min(k, n_samples), 'Distance', 'hamming');
% Initialise synthetic arrays
X_synthetic  = zeros(num_synthetic, n_features);
y_synthetic  = strings(num_synthetic, 1);   % class labels
Scr_synthetic = zeros(num_synthetic, 1);
if use_Z
    Z_synthetic = strings(num_synthetic, 1);  % optional identifiers
else
    Z_synthetic = [];
end
% Set up scour bins based on target class
if strcmp(targetClass, "Yes")
    % Synthetic "Yes" cases
    bins = [5   10; 10 15; 15 20];
elseif strcmp(targetClass, "No")
    % Synthetic "No" cases
    bins = [0    1.5; 1.5 3.5; 3.5 5];
else
    error('targetClass must be either "Yes" or "No".');
end
n_per_bin  = floor(num_synthetic / 3);
bin_counts = [n_per_bin, n_per_bin, num_synthetic - 2 * n_per_bin];
current_idx = 1;
for b = 1:3
    min_val = bins(b, 1); max_val = bins(b, 2);
    for j = 1:bin_counts(b)
        % Randomly choose a base sample and one of its neighbours
        sample_idx  = randi(n_samples);
```

```
            neighbor_idx = indices(sample_idx, randi(min(k, n_samples)));
            % Synthetic feature by interpolation
            diff = X_minority(neighbor_idx, :) - X_minority(sample_idx, :);
            gap  = rand(1, n_features);
            X_synthetic(current_idx, :) = X_minority(sample_idx, :) + gap .* diff;
            % Class label for the synthetic point
            y_synthetic(current_idx) = targetClass;
            % Scour percentage drawn uniformly in the current bin
            Scr_synthetic(current_idx) = min_val + (max_val - min_val) * rand;
            % Optional identifier
            if use_Z
                if rand < 0.5
                    Z_synthetic(current_idx) = Z_minority(sample_idx);
                else
                    Z_synthetic(current_idx) = Z_minority(neighbor_idx);
                end
            end
            current_idx = current_idx + 1;
        end
    end
end
```

G.3.2 Support-level SMOTE

```
function [X_synthetic, y_synthetic, Scr_synthetic, Z_synthetic] = smote8(X_minority, Scr_minority, Z_minority, num_synthetic, k, targetClass)
    [n_samples, n_features] = size(X_minority);
    % Handle optional Z_minority (support names at localisation level)
    use_Z = exist('Z_minority', 'var') && ~isempty(Z_minority);
    % Find k-nearest neighbours in feature space (Hamming distance)
    [indices, ~] = knnsearch(X_minority, X_minority, ...
        'K', min(k, n_samples), ...
        'Distance', 'hamming');
```

```
% Initialise synthetic arrays
X_synthetic   = zeros(num_synthetic, n_features);
y_synthetic   = strings(num_synthetic, 1);   % class labels
Scr_synthetic = zeros(num_synthetic, 1);
if use_Z
    Z_synthetic = strings(num_synthetic, 1); % support locations
else
    Z_synthetic = [];
end
% Set up scour bins based on target class
if strcmp(targetClass, "Yes")
    bins = [5   10; 10 15; 5 20];      % Synthetic "Yes" (scoured) cases
elseif strcmp(targetClass, "No")
    bins = [0    1.5; 1.5 3.5; 3.5  5];      % Synthetic "No" (healthy) cases
else
    error('targetClass must be either "Yes" or "No".');
end
n_per_bin  = floor(num_synthetic / 3);
bin_counts = [n_per_bin, n_per_bin, num_synthetic - 2 * n_per_bin];
current_idx = 1;
for b = 1:3
    min_val = bins(b, 1); max_val = bins(b, 2);
    for j = 1:bin_counts(b)
        % Randomly choose a base sample and one of its neighbours
        sample_idx  = randi(n_samples);
        neighbor_idx = indices(sample_idx, randi(min(k, n_samples)));
        % Synthetic feature by interpolation
        diff = X_minority(neighbor_idx, :) - X_minority(sample_idx, :);
        gap  = rand(1, n_features);
        X_synthetic(current_idx, :) = X_minority(sample_idx, :) + gap .* diff;
        % Class label
        y_synthetic(current_idx) = targetClass;
```

```
            % Scour percentage drawn uniformly in the current bin
            Scr_synthetic(current_idx) = min_val + (max_val - min_val) * rand;
            % Support name inherited from base or neighbour
            if use_Z
                if rand < 0.5
                    Z_synthetic(current_idx) = Z_minority(sample_idx);
                else
                    Z_synthetic(current_idx) = Z_minority(neighbor_idx);
                end
            end
            current_idx = current_idx + 1;
        end
    end
end
```

ANNEX H: Histograms of input features for the RILA rear-mounted case

This annex complements Section 5.7 by reporting the histogram analysis of the Type-A and Type-B features for the RILA rear-mounted sensor, for both the scour detection and scour localisation models. The same feature definitions, preprocessing steps, sliding-window configuration ws = 7 and $\log_{10}$-transform as in Section 5.7 are used.

H.1 Histograms For Scour Detection Models

Figure H.1 to Figure H.3 show the histograms of the individual features. For the RILA rear-mounted case, the individual Type-A histograms show that the $\log_{10}$-transformed skewness and kurtosis descriptors remain concentrated within limited ranges, with clear differences in spread between the approximation and detail levels. The $\log_{10}$-transformed total energy and minimum amplitude of approximation coefficient form a small number of dominant value bands. The Type-B parameters for the rear-mounted sensor displays broad and often multi-peaked histograms, with several Markov features spanning many orders of magnitude on the $\log_{10}$ axis. g_1 shows large variability with a few spikes, while g_2–g_{18} exhibit more regular multi-modal patterns. Overall, the rear-mounted individual features behave in line with the front-mounted case: Type-B parameters show larger variability and more distinct value groups across windows than the DWT-based Type-A features.

Figure H.4 to Figure H.6 illustrate the corresponding aggregated Type-A features. The $\log_{10}$-transformed skewness and kurtosis descriptors are mainly concentrated in narrow intervals, with clear differences between the approximation level and the detail levels. As in the front-mounted case, the mean total energy and mean local minima of approximation coefficient display a few dominant value bands, showing that windows tend to fall into several groups with similar average energy or minimum approximation coefficients.

Type-B aggregated features are shown from Figure H.7 to Figure H.11. The histograms of the mean values are generally broad and often multi-peaked. Mean *g(1)* behaves somewhat differently from the other parameters, with a skewed distribution and a few very spikes. This pattern indicates that *g(1)* is

close to zero in most windows for the rear-mounted case, which is consistent with the sensor being located at the back of the axles. The main dynamic response is captured by later Markov coefficients *g(k)* with *k≥2*, so the first coefficient mostly carries low energy. Many $\log_{10}$-transofmed standard deviation are concentrated near small values, suggesting limited within-window fluctuation for most Markov coefficients, but some also display heavier tails with occasional high-variability windows. The minimum and maximum histograms remain clearly multi-peaked for many *g(k)* showing that the aggregated Markov features capture distinct response levels across windows.

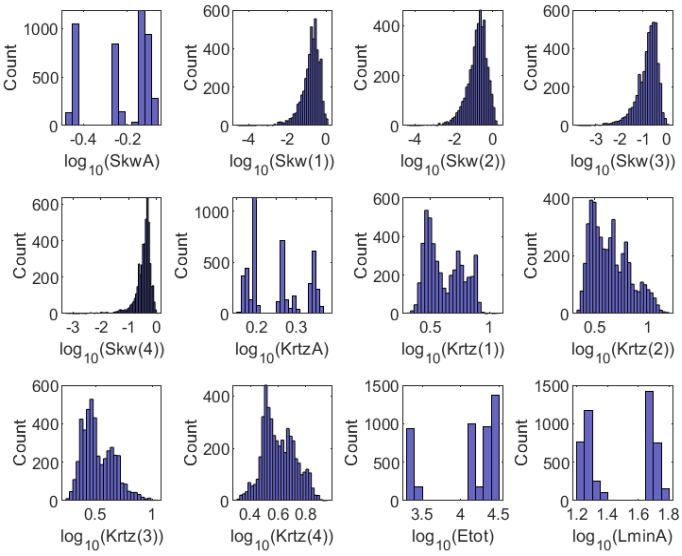


Figure H.1. Histograms of $\log_{10}$-transformed Type-A features from the first crossing in each 7-day window of the scour detection dataset (RILA rear-mounted case).

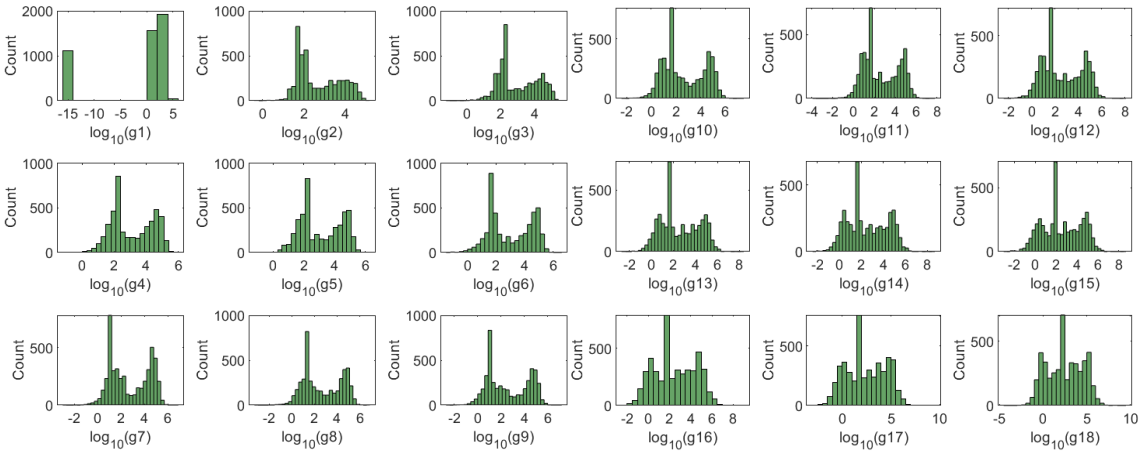


Figure H.2. Histograms of $\log_{10}$-transformed Type-B features from the first crossing in each 7-day window of the bridge-level scour detection dataset part 1 (RILA rear-mounted case).

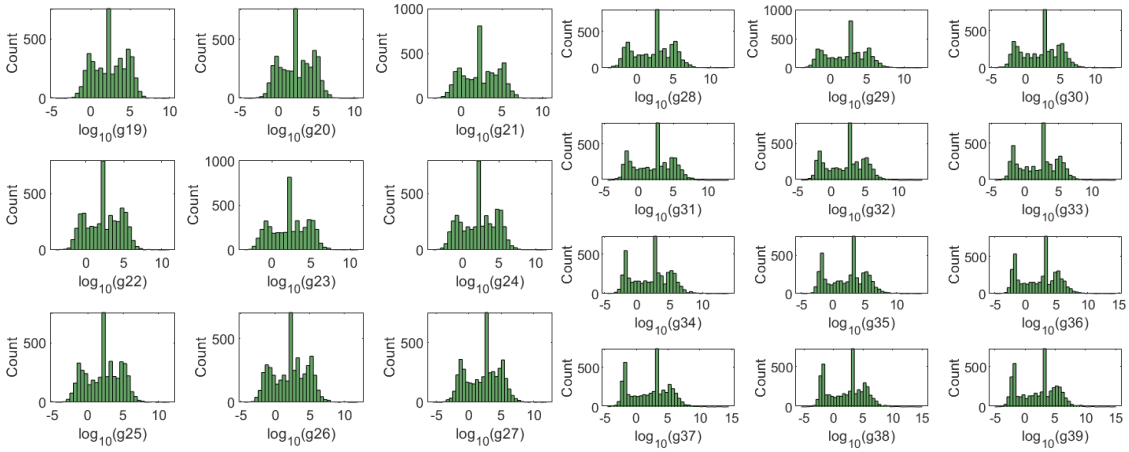


Figure H.3. Histograms of $\log_{10}$-transformed Type-B features from the first crossing in each 7-day window of the bridge-level scour detection dataset part 2 (RILA rear-mounted case).

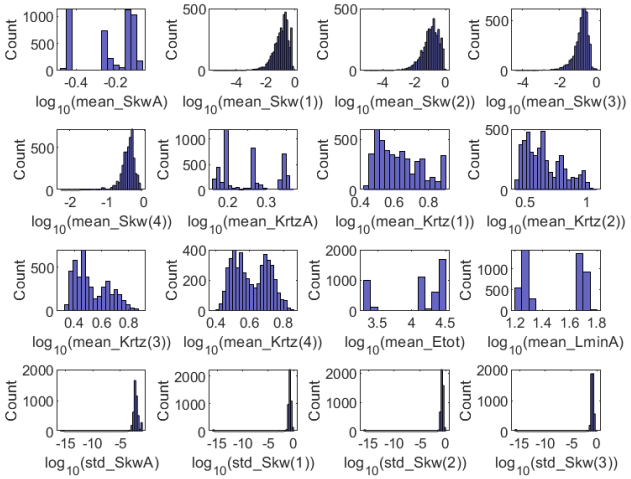


Figure H.4. Histograms of $\log_{10}$-transformed aggregated Type-A features for batch size 7, part 1 (RILA rear-mounted case).

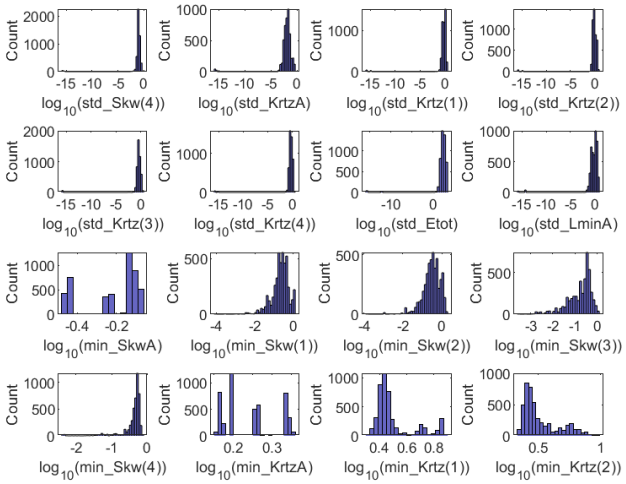


Figure H.5. Histograms of $\log_{10}$-transformed aggregated Type-A features for batch size 7, part 2 (RILA rear-mounted case).

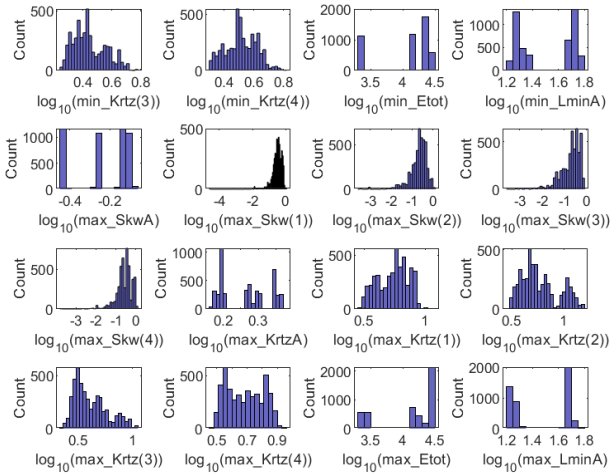


Figure H.6. Histograms of $\log_{10}$-transformed aggregated Type-A features for batch size 7, part 3 (RILA rear-mounted case).

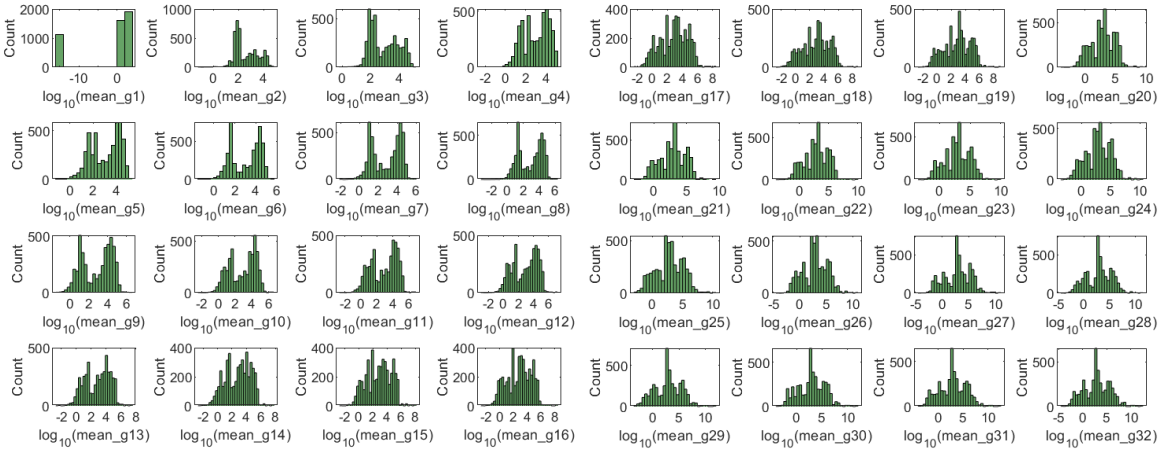


Figure H.7. Histograms of $\log_{10}$-transformed aggregated Type-B features for batch size 7, part 1 (RILA rear-mounted case).

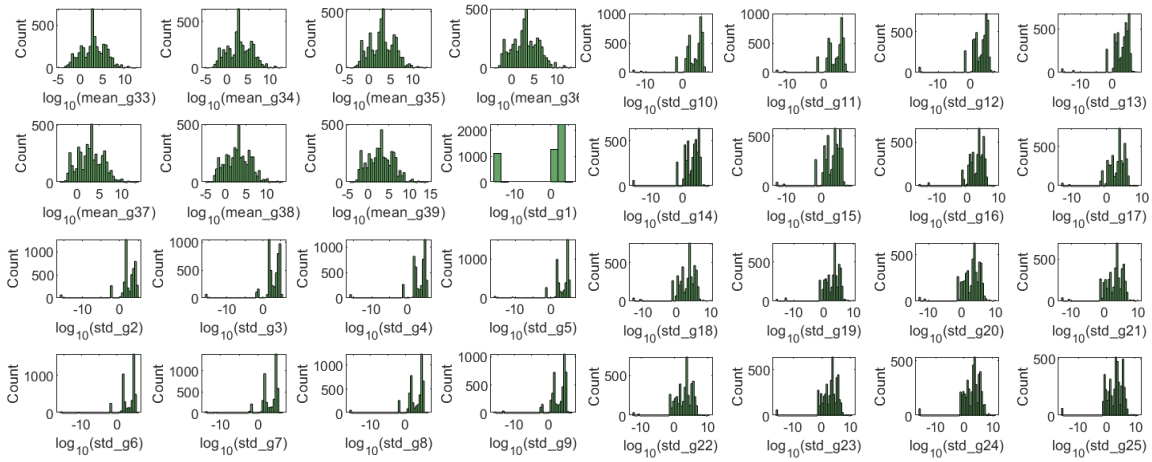


Figure H.8. Histograms of $\log_{10}$-transformed aggregated Type-B features for batch size 7, part 2 (RILA rear-mounted case).

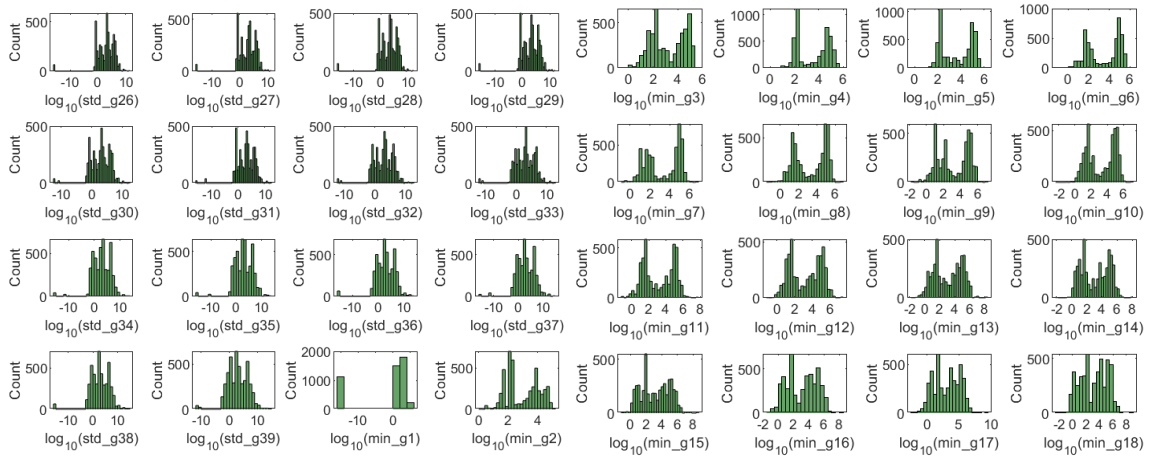


Figure H.9. Histograms of $\log_{10}$-transformed aggregated Type-B features for batch size 7, part 3 (RILA rear-mounted case).

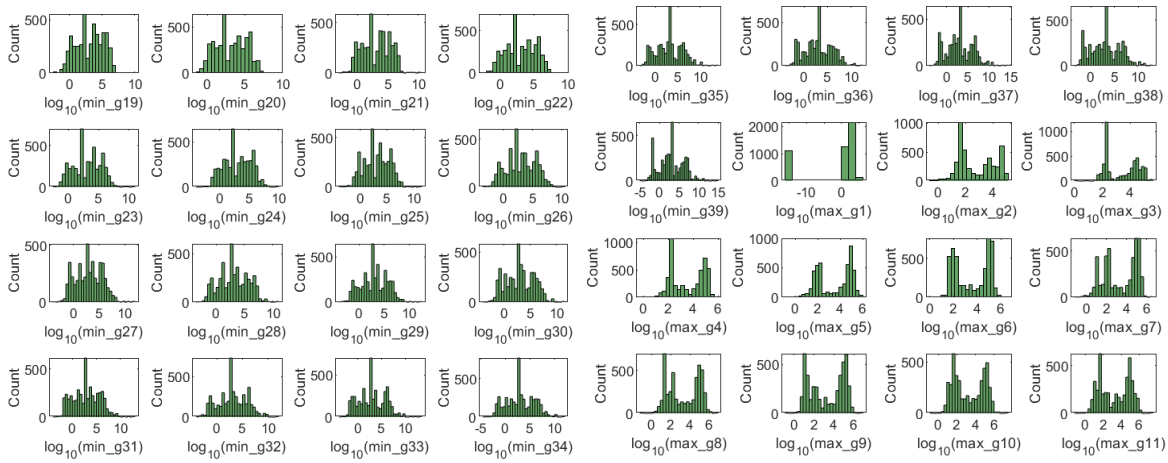


Figure H.10. Histograms of $\log_{10}$-transformed aggregated Type-B features for batch size 7, part 4 (RILA rear-mounted case).

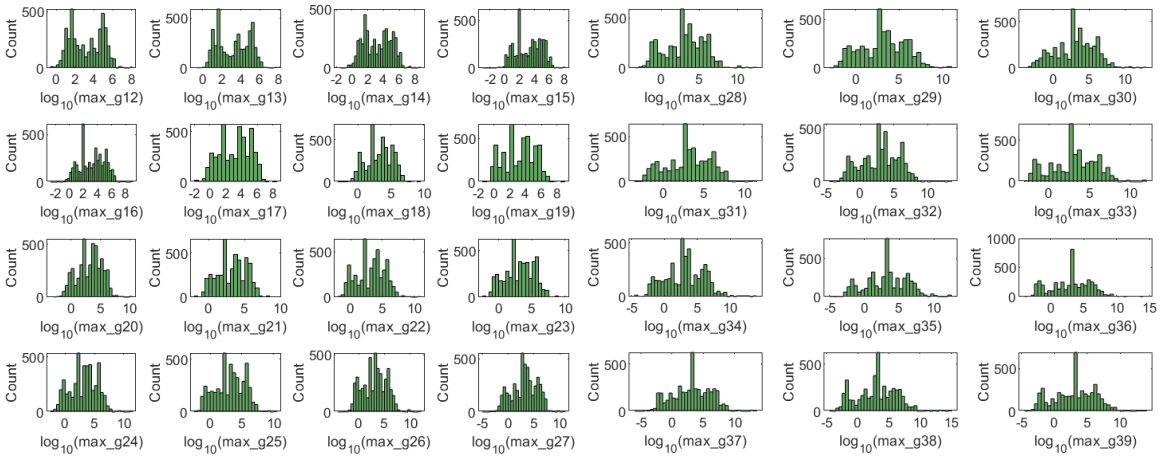


Figure H.11. Histograms of $\log_{10}$-transformed aggregated Type-B features for batch size 7, part 5 (RILA rear-mounted case).

H.2 Histograms For Scour Localisation Models

Figure H.12 andFigure H.13 show the histograms of Type-A and Type-B individual features of the RILA rear-mounted scour localisation model, respectively. The skewness descriptors have narrow and unimodal histograms, indicating limited variation in wavelet asymmetry across windows and supports. *Krtz(A)* and *Krtz(1–2)* have broader and often multi-peaked distributions, while *Krtz(3)* and *Krtz(4)* exhibit tall spikes at high $\log_{10}$ levels with smaller counts at lower levels. This indicates that large kurtosis values occur in a ununiform substantial fraction of batches. The energy and $L_{min}{}^{A}$ features cover a wider range on the $\log_{10}$ axis and are multi-modal. This reflects distinct regimes in average wavelet energy and characteristic length scale across the dataset.

For the individual Type-B features *g(2)–g(4)* and *g(6)–g(10)*, the histograms are narrow and almost symmetric around a single peak, indicating that these Markov coefficients stay within a limited range across crossing–support windows. In contrast, *g(5), g(11)* and *g(12)* have wider, often multi-modal distributions with heavier tails, so these orders carry the largest variability. *g(1)* and *g(13)* have tall spikes at very negative $\log_{10}$ values with only small bars near and above zero, showing that the first and last coefficients are zero in most windows and become non-zero only in a small subset of cases.

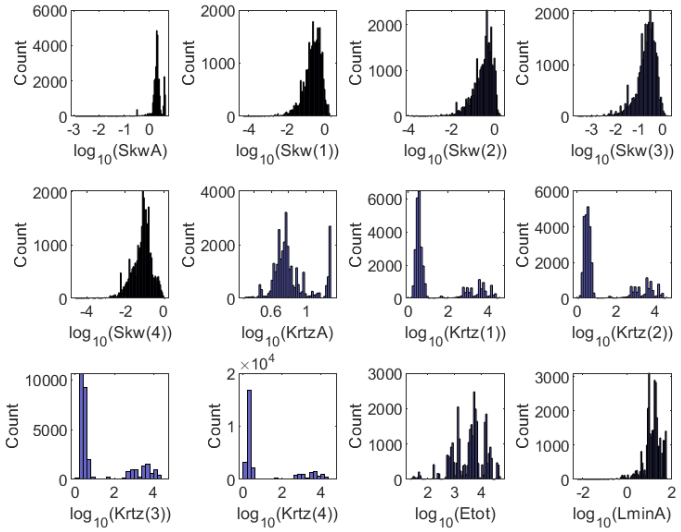


Figure H.12. Histograms of $\log_{10}$-transformed Type-A features from the first crossing in each 7-day window of the scour localization dataset (RILA rear-mounted case).

Figure H.14 and Figure H.15 show the $\log_{10}$-transformed aggregated Type-A batch features of the RILA rear-mounted scour localisation model. The mean aggregates of *Skw(A)* and *Skw(1–4)* form narrow and mostly unimodal histograms. In contrast, *Krtz(1–4)*, and mean E_{tot} span a wider range, multi-modal distribution, which points to distinct regimes in wavelet peak and energy. The standard-deviation

aggregates are concentrated at very low $\log_{10}$ values with a few separate modes. This shows within each window at a given support the corresponding Type-A descriptor generally varies weakly across the *ws = 7* crossings, with occasional groups of windows where fluctuations are larger.

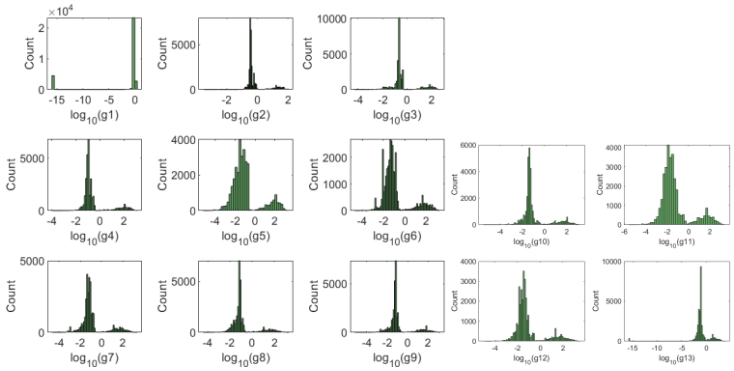


Figure H.13. Histograms of $\log_{10}$-transformed Type-B features from the first crossing in each 7-day window of the scour localization dataset (RILA rear mounted case).

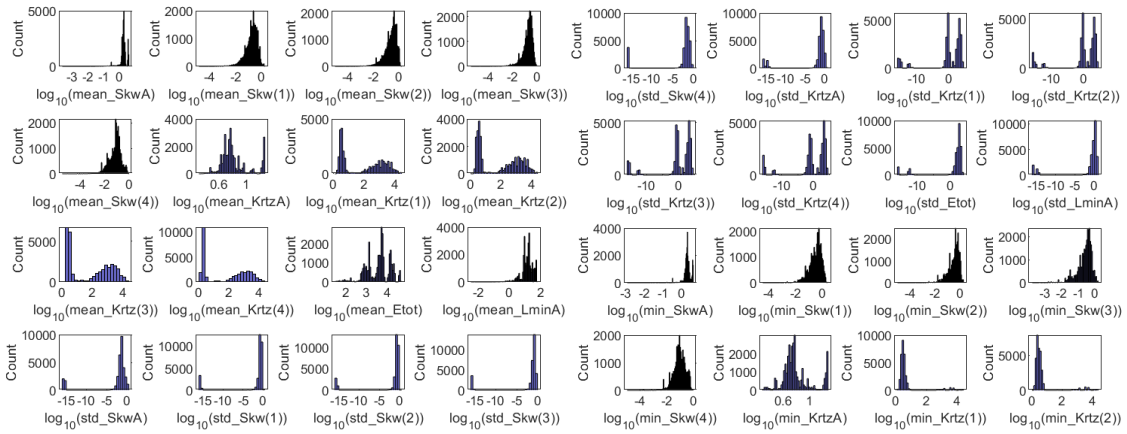


Figure H.14. Histograms of $\log_{10}$-transformed aggregated Type-A features for batch size 7, part 1 of the scour localization dataset (RILA rear-mounted case).

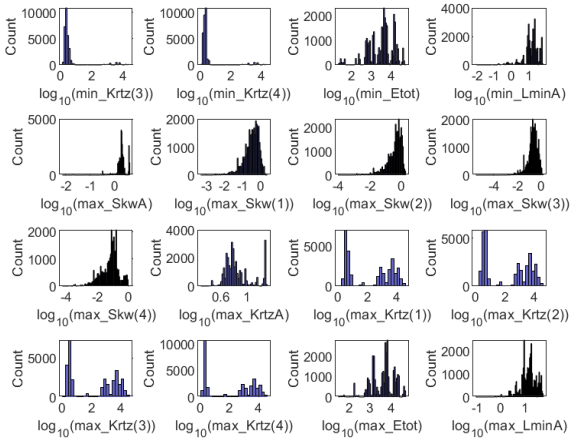


Figure H.15. Histograms of $\log_{10}$-transformed aggregated Type-A features for batch size 7, part 2 of the scour localization dataset (RILA rear-mounted case).

Figure H.16 to Figure H.18 show the $\log_{10}$-transformed aggregated Type-B features of the RILA rear-mounted scour localisation model. The mean aggregates of *g(2)–g(4)* and *g(6)–g(10)* form narrow, almost symmetric peaks, indicating that the window-averaged Markov coefficients at these orders remain in a limited range across batches. In contrast, the mean histograms of *g(5), g(11) and g(12)* are wider and often multi-modal, so these orders exhibit distinct levels in different groups of batches. *g(1)* and *g(13)* have tall spikes at very negative $\log_{10}$ values with only small bars at higher levels, showing that the first and last coefficients are close to zero in most batches. The $\log_{10}$-transformed standard-deviation aggregates are concentrated at low values for all orders, with only a few secondary modes. This indicates that within each window at a given support the Markov coefficients do not change much across the $w_s = 7$ crossings. The minimum and maximum aggregates of *g(2)–g(10)* lie in a relatively narrow band, while *g(1)* and *g(13)* are close to zero. For *g(5), g(11)* and *g(12)*, the minimum and maximum histograms span a wider range of values, confirming that these orders carry the largest variation.

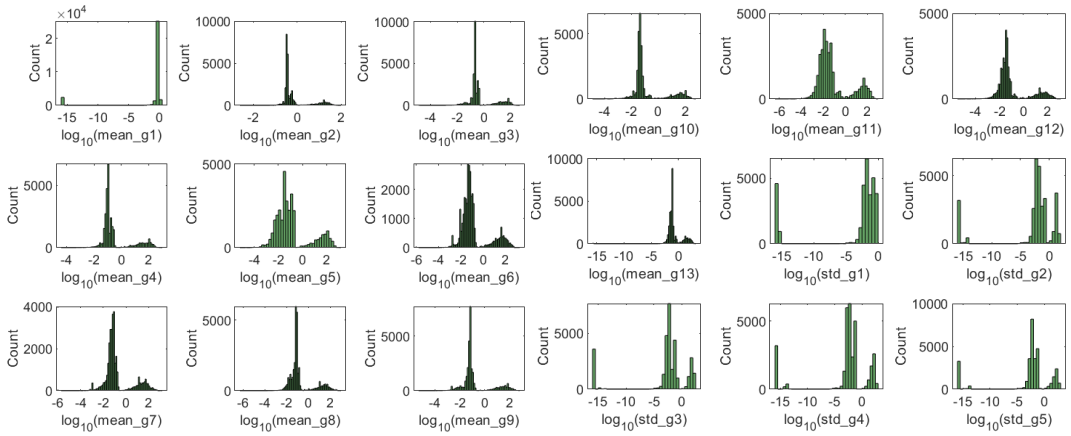


Figure H.16. Histograms of $\log_{10}$-transformed aggregated Type-B features for batch size 7, part 1 of the scour localization dataset (RILA rear-mounted case).

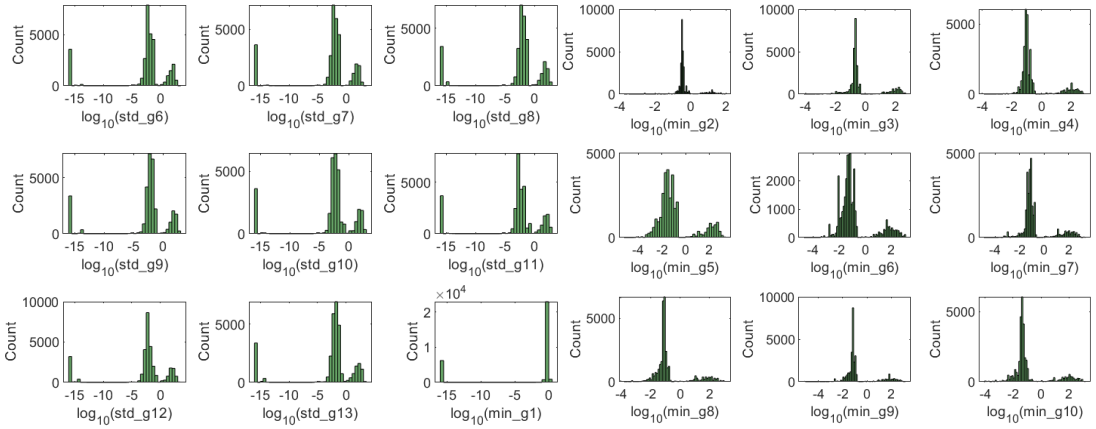


Figure H.17. Histograms of $\log_{10}$-transformed aggregated Type-B features for batch size 7, part 2 of the scour localization dataset (RILA rear-mounted case).

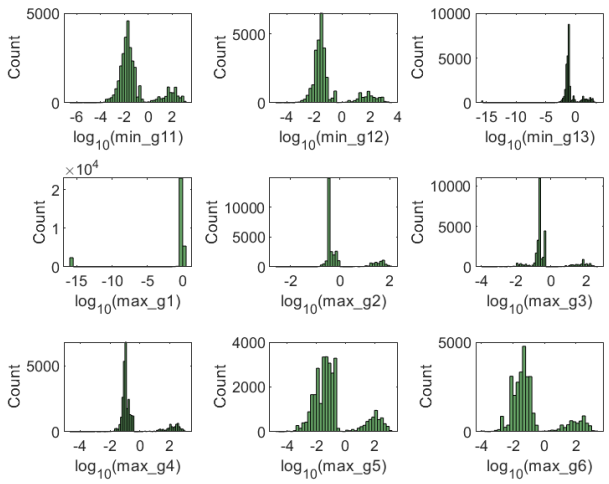


Figure H.18. Histograms of $\log_{10}$-transformed aggregated Type-B features for batch size 7, part 3 of the scour localization dataset (RILA rear-mounted case)

ANNEX I: Correctly Predicted Scoured Supports

This annex belongs to section 6.2.3. The full list of correctly predicted scoured supports for each bridge and each scour localisation algorithm is provided here. The number of true positive counts, i.e., how many times that support was correctly predicted as scoured, and its fraction over all correctly predicted supports tabulated in Table I.1, and Table I.2, respectively for front and rear mounted cases.

Table I.1. RILA front mounted case correctly predicted scoured supports.

Model	w_s	Support Name	TP count	TP total	TP fraction	Model	w_s	Support Name	TP count	TP total	TP fraction	Model	w_s	Support Name	TP count	TP total	TP fraction
ANN	7	Pier 2_Colon	1282	14807	0.09	RF	7	Pier 2_Colon	1331	15280	0.09	SVM	7	Pier 2_Colon	1308	14489	0.09
ANN	7	Pier 1_Colon	1074	14807	0.07	RF	7	Pier 1_Colon	1098	15280	0.07	SVM	7	Pier 1_Colon	1073	14489	0.07
ANN	7	Abutment 1_RTeign	999	14807	0.07	RF	7	Abutment 1_RTeign	1028	15280	0.07	SVM	7	Abutment 1_RTeign	1000	14489	0.07
ANN	7	Pier 2_RTeign	899	14807	0.06	RF	7	Pier 2_RTeign	923	15280	0.06	SVM	7	Pier 2_RTeign	968	14489	0.07
ANN	7	Pier 3_RTeign	803	14807	0.05	RF	7	Pier 3_RTeign	844	15280	0.06	SVM	7	Pier 3_RTeign	853	14489	0.06
ANN	7	Pier 1_WCM15	694	14807	0.05	RF	7	Pier 1_WCM15	732	15280	0.05	SVM	7	Pier 1_WCM15	743	14489	0.05
ANN	7	Abutment 2_RTeign	649	14807	0.04	RF	7	Abutment 2_RTeign	711	15280	0.05	SVM	7	Abutment 2_RTeign	670	14489	0.05
ANN	7	Abutment 2_Colon	624	14807	0.04	RF	7	Pier 1_RTeign	645	15280	0.04	SVM	7	Pier 1_RTeign	599	14489	0.04
ANN	7	Pier 1_RTeign	607	14807	0.04	RF	7	Abutment 2_Colon	605	15280	0.04	SVM	7	Pier 7_WCM15A	551	14489	0.04
ANN	7	Abutment 1_Colon	538	14807	0.04	RF	7	Abutment 1_Colon	562	15280	0.04	SVM	7	Pier 4_WCM15A	520	14489	0.04
ANN	7	Pier 7_WCM15A	517	14807	0.03	RF	7	Pier 7_WCM15A	548	15280	0.04	SVM	7	Abutment 2_Colon	463	14489	0.03
ANN	7	Pier 4_WCM15A	487	14807	0.03	RF	7	Pier 3_WCM15A	504	15280	0.03	SVM	7	Pier 3_WCM15A	454	14489	0.03
ANN	7	Abutment 1_WCM15	465	14807	0.03	RF	7	Abutment 1_WCM15	490	15280	0.03	SVM	7	Pier 6_WCM15A	453	14489	0.03
ANN	7	Pier 3_WCM15A	455	14807	0.03	RF	7	Pier 4_WCM15A	490	15280	0.03	SVM	7	Pier 2_WCM15A	436	14489	0.03
ANN	7	Pier 6_WCM15	445	14807	0.03	RF	7	Pier 6_WCM15	449	15280	0.03	SVM	7	Pier 5_WCM15	435	14489	0.03
ANN	7	Pier 6_WCM15A	444	14807	0.03	RF	7	Pier 2_WCM15	442	15280	0.03	SVM	7	Pier 6_WCM15	421	14489	0.03
ANN	7	Pier 2_WCM15A	425	14807	0.03	RF	7	Pier 6_WCM15A	436	15280	0.03	SVM	7	Pier 2_WCM15	418	14489	0.03
ANN	7	Pier 5_WCM15A	408	14807	0.03	RF	7	Pier 5_WCM15	422	15280	0.03	SVM	7	Abutment 1_Colon	412	14489	0.03
ANN	7	Pier 5_WCM15	388	14807	0.03	RF	7	Abutment 2_WCM15	408	15280	0.03	SVM	7	Pier 4_WCM15	406	14489	0.03
ANN	7	Pier 2_WCM15	384	14807	0.03	RF	7	Pier 2_WCM15A	407	15280	0.03	SVM	7	Pier 5_WCM15A	390	14489	0.03
ANN	7	Pier 8_WCM15A	380	14807	0.03	RF	7	Pier 5_WCM15A	404	15280	0.03	SVM	7	Pier 8_WCM15A	367	14489	0.03

ANN	7	Abutment 2_WCM15	377	14807	0.03	RF	7	Pier 4_WCM15	387	15280	0.03	SVM	7	Abutment 2_WCM15	363	14489	0.03
ANN	7	Pier 4_WCM15	363	14807	0.02	RF	7	Pier 8_WCM15A	347	15280	0.02	SVM	7	Pier 3_WCM15	322	14489	0.02
ANN	7	Pier 3_WCM15	339	14807	0.02	RF	7	Pier 3_WCM15	337	15280	0.02	SVM	7	Pier 1_WCM15A	300	14489	0.02
ANN	7	Pier 1_WCM15A	276	14807	0.02	RF	7	Pier 1_WCM15A	287	15280	0.02	SVM	7	Abutment 1_WCM15	257	14489	0.02
ANN	7	Abutment 2_WCM15A	257	14807	0.02	RF	7	Abutment 2_WCM15A	240	15280	0.02	SVM	7	Abutment 2_WCM15A	221	14489	0.02
ANN	7	Abutment 1_WCM15A	228	14807	0.02	RF	7	Abutment 1_WCM15A	203	15280	0.01	SVM	7	Abutment 1_WCM15A	86	14489	0.01
ANN	15	Pier 2_Colon	1277	14792	0.09	RF	15	Pier 2_Colon	1316	15089	0.09	SVM	15	Pier 2_Colon	1305	14367	0.09
ANN	15	Pier 1_Colon	1026	14792	0.07	RF	15	Pier 1_Colon	1135	15089	0.08	SVM	15	Pier 1_Colon	1074	14367	0.07
ANN	15	Abutment 1_RTeign	1017	14792	0.07	RF	15	Abutment 1_RTeign	1001	15089	0.07	SVM	15	Abutment 1_RTeign	981	14367	0.07
ANN	15	Pier 2_RTeign	948	14792	0.06	RF	15	Pier 2_RTeign	907	15089	0.06	SVM	15	Pier 2_RTeign	955	14367	0.07
ANN	15	Pier 3_RTeign	841	14792	0.06	RF	15	Pier 3_RTeign	838	15089	0.06	SVM	15	Pier 3_RTeign	785	14367	0.05
ANN	15	Pier 1_WCM15	783	14792	0.05	RF	15	Pier 1_WCM15	736	15089	0.05	SVM	15	Pier 1_WCM15	725	14367	0.05
ANN	15	Pier 1_RTeign	657	14792	0.04	RF	15	Abutment 2_RTeign	716	15089	0.05	SVM	15	Abutment 2_RTeign	701	14367	0.05
ANN	15	Abutment 2_RTeign	644	14792	0.04	RF	15	Pier 1_RTeign	660	15089	0.04	SVM	15	Pier 1_RTeign	629	14367	0.04
ANN	15	Abutment 2_Colon	549	14792	0.04	RF	15	Abutment 2_Colon	639	15089	0.04	SVM	15	Pier 7_WCM15A	575	14367	0.04
ANN	15	Pier 4_WCM15A	514	14792	0.03	RF	15	Abutment 1_Colon	566	15089	0.04	SVM	15	Pier 4_WCM15A	508	14367	0.04
ANN	15	Abutment 1_Colon	513	14792	0.03	RF	15	Pier 7_WCM15A	535	15089	0.04	SVM	15	Pier 6_WCM15A	501	14367	0.03
ANN	15	Pier 7_WCM15A	511	14792	0.03	RF	15	Pier 6_WCM15	483	15089	0.03	SVM	15	Pier 3_WCM15A	488	14367	0.03
ANN	15	Pier 6_WCM15	475	14792	0.03	RF	15	Pier 4_WCM15A	479	15089	0.03	SVM	15	Pier 6_WCM15	478	14367	0.03
ANN	15	Pier 3_WCM15A	461	14792	0.03	RF	15	Pier 3_WCM15A	469	15089	0.03	SVM	15	Abutment 2_Colon	459	14367	0.03
ANN	15	Pier 6_WCM15A	456	14792	0.03	RF	15	Pier 2_WCM15	454	15089	0.03	SVM	15	Pier 5_WCM15	430	14367	0.03
ANN	15	Pier 2_WCM15	445	14792	0.03	RF	15	Pier 6_WCM15A	443	15089	0.03	SVM	15	Pier 2_WCM15	411	14367	0.03
ANN	15	Abutment 1_WCM15	408	14792	0.03	RF	15	Pier 5_WCM15	411	15089	0.03	SVM	15	Pier 2_WCM15A	394	14367	0.03

ANN	15	Pier 2_WCM15A	407	14792	0.03	RF	15	Abutment 1_WCM15	402	15089	0.03	SVM	15	Pier 5_WCM15A	383	14367	0.03
ANN	15	Pier 5_WCM15A	406	14792	0.03	RF	15	Abutment 2_WCM15	389	15089	0.03	SVM	15	Pier 4_WCM15	376	14367	0.03
ANN	15	Pier 5_WCM15	388	14792	0.03	RF	15	Pier 5_WCM15A	387	15089	0.03	SVM	15	Pier 8_WCM15A	372	14367	0.03
ANN	15	Abutment 2_WCM15	375	14792	0.03	RF	15	Pier 8_WCM15A	372	15089	0.02	SVM	15	Abutment 2_WCM15	354	14367	0.02
ANN	15	Pier 4_WCM15	318	14792	0.02	RF	15	Pier 4_WCM15	369	15089	0.02	SVM	15	Abutment 1_Colon	351	14367	0.02
ANN	15	Pier 8_WCM15A	318	14792	0.02	RF	15	Pier 2_WCM15A	353	15089	0.02	SVM	15	Pier 3_WCM15	319	14367	0.02
ANN	15	Pier 1_WCM15A	288	14792	0.02	RF	15	Pier 3_WCM15	294	15089	0.02	SVM	15	Pier 1_WCM15A	285	14367	0.02
ANN	15	Pier 3_WCM15	288	14792	0.02	RF	15	Pier 1_WCM15A	266	15089	0.02	SVM	15	Abutment 1_WCM15	226	14367	0.02
ANN	15	Abutment 2_WCM15A	255	14792	0.02	RF	15	Abutment 2_WCM15A	241	15089	0.02	SVM	15	Abutment 2_WCM15A	212	14367	0.01
ANN	15	Abutment 1_WCM15A	224	14792	0.02	RF	15	Abutment 1_WCM15A	228	15089	0.02	SVM	15	Abutment 1_WCM15A	90	14367	0.01
ANN	31	Pier 2_Colon	1243	14693	0.08	RF	31	Pier 2_Colon	1299	14740	0.09	SVM	31	Pier 2_Colon	1265	14270	0.09
ANN	31	Pier 1_Colon	1063	14693	0.07	RF	31	Pier 1_Colon	1115	14740	0.08	SVM	31	Pier 1_Colon	1050	14270	0.07
ANN	31	Abutment 1_RTeign	1033	14693	0.07	RF	31	Abutment 1_RTeign	980	14740	0.07	SVM	31	Abutment 1_RTeign	988	14270	0.07
ANN	31	Pier 2_RTeign	927	14693	0.06	RF	31	Pier 2_RTeign	896	14740	0.06	SVM	31	Pier 2_RTeign	958	14270	0.07
ANN	31	Pier 3_RTeign	826	14693	0.06	RF	31	Pier 3_RTeign	849	14740	0.06	SVM	31	Pier 3_RTeign	816	14270	0.06
ANN	31	Pier 1_WCM15	714	14693	0.05	RF	31	Pier 1_WCM15	766	14740	0.05	SVM	31	Pier 1_WCM15	780	14270	0.05
ANN	31	Abutment 2_RTeign	667	14693	0.05	RF	31	Abutment 2_RTeign	718	14740	0.05	SVM	31	Abutment 2_RTeign	708	14270	0.05
ANN	31	Pier 1_RTeign	660	14693	0.04	RF	31	Pier 1_RTeign	643	14740	0.04	SVM	31	Pier 1_RTeign	663	14270	0.05
ANN	31	Abutment 2_Colon	565	14693	0.04	RF	31	Abutment 2_Colon	559	14740	0.04	SVM	31	Pier 7_WCM15A	498	14270	0.03
ANN	31	Abutment 1_Colon	563	14693	0.04	RF	31	Abutment 1_Colon	520	14740	0.04	SVM	31	Pier 4_WCM15A	492	14270	0.03
ANN	31	Pier 7_WCM15A	491	14693	0.03	RF	31	Pier 7_WCM15A	516	14740	0.04	SVM	31	Pier 2_WCM15	477	14270	0.03
ANN	31	Pier 4_WCM15A	475	14693	0.03	RF	31	Pier 4_WCM15A	478	14740	0.03	SVM	31	Abutment 2_Colon	472	14270	0.03
ANN	31	Pier 3_WCM15A	463	14693	0.03	RF	31	Pier 3_WCM15A	477	14740	0.03	SVM	31	Pier 3_WCM15A	441	14270	0.03

ANN	31	Pier 2_WCM15	446	14693	0.03	RF	31	Pier 2_WCM15	469	14740	0.03	SVM	31	Pier 6_WCM15	436	14270	0.03
ANN	31	Pier 6_WCM15A	439	14693	0.03	RF	31	Pier 6_WCM15A	465	14740	0.03	SVM	31	Abutment 1_Colon	435	14270	0.03
ANN	31	Pier 6_WCM15	431	14693	0.03	RF	31	Pier 6_WCM15	448	14740	0.03	SVM	31	Pier 5_WCM15A	414	14270	0.03
ANN	31	Abutment 2_WCM15	423	14693	0.03	RF	31	Abutment 1_WCM15	385	14740	0.03	SVM	31	Pier 6_WCM15A	413	14270	0.03
ANN	31	Pier 2_WCM15A	408	14693	0.03	RF	31	Pier 2_WCM15A	384	14740	0.03	SVM	31	Abutment 2_WCM15	371	14270	0.03
ANN	31	Abutment 1_WCM15	401	14693	0.03	RF	31	Abutment 2_WCM15	381	14740	0.03	SVM	31	Pier 2_WCM15A	347	14270	0.02
ANN	31	Pier 5_WCM15A	398	14693	0.03	RF	31	Pier 5_WCM15	361	14740	0.02	SVM	31	Pier 5_WCM15	346	14270	0.02
ANN	31	Pier 5_WCM15	352	14693	0.02	RF	31	Pier 5_WCM15A	359	14740	0.02	SVM	31	Pier 4_WCM15	339	14270	0.02
ANN	31	Pier 3_WCM15	335	14693	0.02	RF	31	Pier 4_WCM15	349	14740	0.02	SVM	31	Pier 8_WCM15A	333	14270	0.02
ANN	31	Pier 8_WCM15A	334	14693	0.02	RF	31	Pier 8_WCM15A	344	14740	0.02	SVM	31	Pier 3_WCM15	303	14270	0.02
ANN	31	Pier 4_WCM15	328	14693	0.02	RF	31	Pier 3_WCM15	307	14740	0.02	SVM	31	Abutment 1_WCM15	292	14270	0.02
ANN	31	Abutment 2_WCM15A	243	14693	0.02	RF	31	Pier 1_WCM15A	256	14740	0.02	SVM	31	Pier 1_WCM15A	277	14270	0.02
ANN	31	Pier 1_WCM15A	237	14693	0.02	RF	31	Abutment 2_WCM15A	211	14740	0.01	SVM	31	Abutment 2_WCM15A	225	14270	0.02
ANN	31	Abutment 1_WCM15A	228	14693	0.02	RF	31	Abutment 1_WCM15A	205	14740	0.01	SVM	31	Abutment 1_WCM15A	131	14270	0.01
ANN	61	Pier 2_Colon	1245	14245	0.09	RF	61	Pier 2_Colon	1229	14025	0.09	SVM	61	Pier 2_Colon	1301	13724	0.09
ANN	61	Pier 1_Colon	1034	14245	0.07	RF	61	Abutment 1_RTeign	984	14025	0.07	SVM	61	Pier 1_Colon	1040	13724	0.08
ANN	61	Abutment 1_RTeign	975	14245	0.07	RF	61	Pier 1_Colon	964	14025	0.07	SVM	61	Abutment 1_RTeign	1005	13724	0.07
ANN	61	Pier 2_RTeign	909	14245	0.06	RF	61	Pier 2_RTeign	963	14025	0.07	SVM	61	Pier 2_RTeign	911	13724	0.07
ANN	61	Pier 1_WCM15	765	14245	0.05	RF	61	Pier 3_RTeign	815	14025	0.06	SVM	61	Pier 3_RTeign	790	13724	0.06
ANN	61	Pier 3_RTeign	759	14245	0.05	RF	61	Pier 1_WCM15	749	14025	0.05	SVM	61	Pier 1_WCM15	774	13724	0.06
ANN	61	Abutment 2_RTeign	692	14245	0.05	RF	61	Abutment 2_RTeign	746	14025	0.05	SVM	61	Abutment 2_RTeign	686	13724	0.05
ANN	61	Pier 1_RTeign	606	14245	0.04	RF	61	Abutment 2_Colon	605	14025	0.04	SVM	61	Pier 1_RTeign	595	13724	0.04
ANN	61	Abutment 2_Colon	523	14245	0.04	RF	61	Pier 1_RTeign	576	14025	0.04	SVM	61	Pier 4_WCM15A	512	13724	0.04

ANN	61	Pier 3_WCM15A	507	14245	0.04	RF	61	Abutment 1_Colon	544	14025	0.04	SVM	61	Abutment 2_Colon	508	13724	0.04
ANN	61	Pier 4_WCM15A	483	14245	0.03	RF	61	Pier 4_WCM15A	485	14025	0.03	SVM	61	Pier 7_WCM15A	471	13724	0.03
ANN	61	Pier 7_WCM15A	453	14245	0.03	RF	61	Pier 3_WCM15A	468	14025	0.03	SVM	61	Pier 3_WCM15A	452	13724	0.03
ANN	61	Abutment 1_Colon	447	14245	0.03	RF	61	Pier 7_WCM15A	427	14025	0.03	SVM	61	Pier 5_WCM15A	424	13724	0.03
ANN	61	Abutment 1_WCM15	427	14245	0.03	RF	61	Pier 2_WCM15	408	14025	0.03	SVM	61	Pier 6_WCM15A	411	13724	0.03
ANN	61	Pier 6_WCM15A	414	14245	0.03	RF	61	Pier 5_WCM15A	404	14025	0.03	SVM	61	Pier 2_WCM15	402	13724	0.03
ANN	61	Pier 5_WCM15A	411	14245	0.03	RF	61	Pier 6_WCM15A	382	14025	0.03	SVM	61	Pier 6_WCM15	365	13724	0.03
ANN	61	Pier 2_WCM15	401	14245	0.03	RF	61	Abutment 1_WCM15	381	14025	0.03	SVM	61	Abutment 1_Colon	359	13724	0.03
ANN	61	Pier 5_WCM15	382	14245	0.03	RF	61	Pier 6_WCM15	356	14025	0.03	SVM	61	Pier 5_WCM15	352	13724	0.03
ANN	61	Pier 6_WCM15	369	14245	0.03	RF	61	Pier 5_WCM15	354	14025	0.03	SVM	61	Pier 8_WCM15A	317	13724	0.02
ANN	61	Pier 4_WCM15	365	14245	0.03	RF	61	Pier 3_WCM15	323	14025	0.02	SVM	61	Pier 4_WCM15	312	13724	0.02
ANN	61	Pier 3_WCM15	360	14245	0.03	RF	61	Abutment 2_WCM15	320	14025	0.02	SVM	61	Pier 3_WCM15	308	13724	0.02
ANN	61	Abutment 2_WCM15	345	14245	0.02	RF	61	Pier 4_WCM15	318	14025	0.02	SVM	61	Abutment 2_WCM15	305	13724	0.02
ANN	61	Pier 2_WCM15A	343	14245	0.02	RF	61	Pier 2_WCM15A	293	14025	0.02	SVM	61	Pier 2_WCM15A	296	13724	0.02
ANN	61	Pier 8_WCM15A	320	14245	0.02	RF	61	Pier 8_WCM15A	266	14025	0.02	SVM	61	Pier 1_WCM15A	276	13724	0.02
ANN	61	Pier 1_WCM15A	247	14245	0.02	RF	61	Pier 1_WCM15A	254	14025	0.02	SVM	61	Abutment 1_WCM15	266	13724	0.02
ANN	61	Abutment 1_WCM15A	242	14245	0.02	RF	61	Abutment 1_WCM15A	217	14025	0.02	SVM	61	Abutment 2_WCM15A	211	13724	0.02
ANN	61	Abutment 2_WCM15A	221	14245	0.02	RF	61	Abutment 2_WCM15A	194	14025	0.01	SVM	61	Abutment 1_WCM15A	75	13724	0.01

Table I.2. RILA rear mounted case correctly predicted scoured supports.

Model	w_s	Support Name	TP count	TP total	TP fraction	Model	w_s	Support Name	TP count	TP total	TP fraction	Model	w_s	Support Name	TP count	TP total	TP fraction
ANN	7	Pier 2_Colon	1105	14378	0.08	RF	7	Pier 2_Colon	1163	14993	0.08	SVM	7	Pier 2_Colon	1070	13879	0.08
ANN	7	Abutment 2_Colon	1016	14378	0.07	RF	7	Abutment 2_Colon	1147	14993	0.08	SVM	7	Abutment 2_Colon	1043	13879	0.08
ANN	7	Pier 1_Colon	940	14378	0.07	RF	7	Pier 1_Colon	903	14993	0.06	SVM	7	Pier 2_RTeign	907	13879	0.07
ANN	7	Pier 2_RTeign	903	14378	0.06	RF	7	Pier 2_RTeign	902	14993	0.06	SVM	7	Pier 1_Colon	893	13879	0.06
ANN	7	Abutment 1_RTeign	793	14378	0.06	RF	7	Abutment 1_RTeign	844	14993	0.06	SVM	7	Abutment 1_RTeign	793	13879	0.06
ANN	7	Pier 3_RTeign	692	14378	0.05	RF	7	Pier 3_RTeign	764	14993	0.05	SVM	7	Pier 5_WCM15	767	13879	0.06
ANN	7	Pier 5_WCM15	689	14378	0.05	RF	7	Pier 5_WCM15	744	14993	0.05	SVM	7	Pier 3_RTeign	696	13879	0.05
ANN	7	Abutment 1_Colon	688	14378	0.05	RF	7	Abutment 2_RTeign	727	14993	0.05	SVM	7	Abutment 2_RTeign	691	13879	0.05
ANN	7	Abutment 2_RTeign	682	14378	0.05	RF	7	Abutment 1_Colon	709	14993	0.05	SVM	7	Pier 1_RTeign	561	13879	0.04
ANN	7	Pier 2_WCM15A	563	14378	0.04	RF	7	Pier 1_RTeign	559	14993	0.04	SVM	7	Pier 2_WCM15A	537	13879	0.04
ANN	7	Pier 1_WCM15A	518	14378	0.04	RF	7	Pier 2_WCM15A	550	14993	0.04	SVM	7	Pier 1_WCM15A	532	13879	0.04
ANN	7	Pier 4_WCM15	504	14378	0.04	RF	7	Pier 4_WCM15	513	14993	0.03	SVM	7	Pier 8_WCM15A	507	13879	0.04
ANN	7	Pier 8_WCM15A	499	14378	0.03	RF	7	Pier 8_WCM15A	506	14993	0.03	SVM	7	Pier 4_WCM15	505	13879	0.04
ANN	7	Pier 1_RTeign	477	14378	0.03	RF	7	Pier 1_WCM15A	498	14993	0.03	SVM	7	Pier 6_WCM15	488	13879	0.04
ANN	7	Pier 6_WCM15	476	14378	0.03	RF	7	Pier 6_WCM15	496	14993	0.03	SVM	7	Abutment 1_Colon	458	13879	0.03
ANN	7	Abutment 2_WCM15	423	14378	0.03	RF	7	Abutment 2_WCM15	450	14993	0.03	SVM	7	Pier 6_WCM15A	406	13879	0.03
ANN	7	Abutment 1_WCM15A	367	14378	0.03	RF	7	Pier 6_WCM15A	420	14993	0.03	SVM	7	Pier 1_WCM15	363	13879	0.03
ANN	7	Pier 6_WCM15A	356	14378	0.02	RF	7	Pier 3_WCM15	366	14993	0.02	SVM	7	Abutment 2_WCM15	349	13879	0.03
ANN	7	Pier 1_WCM15	350	14378	0.02	RF	7	Pier 1_WCM15	345	14993	0.02	SVM	7	Pier 5_WCM15A	330	13879	0.02
ANN	7	Pier 5_WCM15A	331	14378	0.02	RF	7	Abutment 1_WCM15A	343	14993	0.02	SVM	7	Pier 4_WCM15A	329	13879	0.02
ANN	7	Pier 4_WCM15A	328	14378	0.02	RF	7	Pier 5_WCM15A	330	14993	0.02	SVM	7	Pier 2_WCM15	319	13879	0.02
ANN	7	Pier 3_WCM15	324	14378	0.02	RF	7	Pier 2_WCM15	325	14993	0.02	SVM	7	Pier 3_WCM15	299	13879	0.02
ANN	7	Pier 2_WCM15	317	14378	0.02	RF	7	Pier 4_WCM15A	325	14993	0.02	SVM	7	Pier 7_WCM15A	246	13879	0.02

ANN	7	Abutment 2_WCM15A	297	14378	0.02	RF	7	Pier 7_WCM15A	298	14993	0.02	SVM	7	Pier 3_WCM15A	222	13879	0.02
ANN	7	Pier 7_WCM15A	284	14378	0.02	RF	7	Abutment 2_WCM15A	280	14993	0.02	SVM	7	Abutment 1_WCM15A	212	13879	0.02
ANN	7	Abutment 1_WCM15	231	14378	0.02	RF	7	Abutment 1_WCM15	276	14993	0.02	SVM	7	Abutment 1_WCM15	185	13879	0.01
ANN	7	Pier 3_WCM15A	225	14378	0.02	RF	7	Pier 3_WCM15A	210	14993	0.01	SVM	7	Abutment 2_WCM15A	171	13879	0.01
ANN	15	Pier 2_Colon	1111	14377	0.08	RF	15	Pier 2_Colon	1149	14752	0.08	SVM	15	Abutment 2_Colon	1096	13792	0.08
ANN	15	Abutment 2_Colon	1082	14377	0.08	RF	15	Abutment 2_Colon	1108	14752	0.08	SVM	15	Pier 2_Colon	1087	13792	0.08
ANN	15	Pier 2_RTeign	884	14377	0.06	RF	15	Pier 2_RTeign	893	14752	0.06	SVM	15	Pier 2_RTeign	883	13792	0.06
ANN	15	Pier 1_Colon	866	14377	0.06	RF	15	Pier 1_Colon	889	14752	0.06	SVM	15	Pier 1_Colon	861	13792	0.06
ANN	15	Abutment 1_RTeign	794	14377	0.06	RF	15	Abutment 1_RTeign	806	14752	0.05	SVM	15	Abutment 1_RTeign	792	13792	0.06
ANN	15	Pier 5_WCM15	726	14377	0.05	RF	15	Pier 3_RTeign	754	14752	0.05	SVM	15	Pier 5_WCM15	748	13792	0.05
ANN	15	Pier 3_RTeign	682	14377	0.05	RF	15	Pier 5_WCM15	749	14752	0.05	SVM	15	Pier 3_RTeign	691	13792	0.05
ANN	15	Abutment 1_Colon	659	14377	0.05	RF	15	Abutment 2_RTeign	715	14752	0.05	SVM	15	Abutment 2_RTeign	664	13792	0.05
ANN	15	Abutment 2_RTeign	646	14377	0.04	RF	15	Abutment 1_Colon	671	14752	0.05	SVM	15	Pier 2_WCM15A	567	13792	0.04
ANN	15	Pier 8_WCM15A	555	14377	0.04	RF	15	Pier 2_WCM15A	564	14752	0.04	SVM	15	Pier 1_RTeign	534	13792	0.04
ANN	15	Pier 2_WCM15A	542	14377	0.04	RF	15	Pier 6_WCM15	527	14752	0.04	SVM	15	Pier 1_WCM15A	514	13792	0.04
ANN	15	Pier 1_RTeign	517	14377	0.04	RF	15	Pier 1_RTeign	522	14752	0.04	SVM	15	Pier 8_WCM15A	514	13792	0.04
ANN	15	Pier 1_WCM15A	493	14377	0.03	RF	15	Pier 8_WCM15A	516	14752	0.03	SVM	15	Pier 4_WCM15	469	13792	0.03
ANN	15	Pier 6_WCM15	491	14377	0.03	RF	15	Pier 1_WCM15A	503	14752	0.03	SVM	15	Pier 6_WCM15	463	13792	0.03
ANN	15	Pier 4_WCM15	485	14377	0.03	RF	15	Abutment 2_WCM15	472	14752	0.03	SVM	15	Abutment 1_Colon	416	13792	0.03
ANN	15	Abutment 2_WCM15	432	14377	0.03	RF	15	Pier 4_WCM15	452	14752	0.03	SVM	15	Abutment 2_WCM15	401	13792	0.03
ANN	15	Pier 6_WCM15A	393	14377	0.03	RF	15	Pier 6_WCM15A	407	14752	0.03	SVM	15	Pier 2_WCM15	374	13792	0.03
ANN	15	Abutment 1_WCM15A	361	14377	0.03	RF	15	Abutment 1_WCM15A	362	14752	0.02	SVM	15	Pier 5_WCM15A	367	13792	0.03
ANN	15	Pier 5_WCM15A	345	14377	0.02	RF	15	Pier 3_WCM15	338	14752	0.02	SVM	15	Pier 1_WCM15	357	13792	0.03
ANN	15	Pier 4_WCM15A	339	14377	0.02	RF	15	Pier 4_WCM15A	332	14752	0.02	SVM	15	Pier 6_WCM15A	350	13792	0.03
ANN	15	Pier 3_WCM15	320	14377	0.02	RF	15	Pier 5_WCM15A	332	14752	0.02	SVM	15	Pier 3_WCM15	340	13792	0.02
ANN	15	Pier 1_WCM15	318	14377	0.02	RF	15	Pier 1_WCM15	328	14752	0.02	SVM	15	Pier 4_WCM15A	339	13792	0.02

ANN	15	Pier 2_WCM15	304	14377	0.02	RF	15	Pier 2_WCM15	328	14752	0.02	SVM	15	Pier 7_WCM15A	257	13792	0.02
ANN	15	Abutment 2_WCM15A	289	14377	0.02	RF	15	Pier 7_WCM15A	285	14752	0.02	SVM	15	Pier 3_WCM15A	238	13792	0.02
ANN	15	Pier 7_WCM15A	260	14377	0.02	RF	15	Abutment 1_WCM15	270	14752	0.02	SVM	15	Abutment 1_WCM15	189	13792	0.01
ANN	15	Abutment 1_WCM15	251	14377	0.02	RF	15	Abutment 2_WCM15A	262	14752	0.02	SVM	15	Abutment 2_WCM15A	154	13792	0.01
ANN	15	Pier 3_WCM15A	232	14377	0.02	RF	15	Pier 3_WCM15A	218	14752	0.01	SVM	15	Abutment 1_WCM15A	127	13792	0.01
ANN	31	Pier 2_Colon	1070	14313	0.07	RF	31	Pier 2_Colon	1151	14359	0.08	SVM	31	Pier 2_Colon	1204	13396	0.09
ANN	31	Abutment 2_Colon	979	14313	0.07	RF	31	Abutment 2_Colon	1077	14359	0.08	SVM	31	Abutment 2_Colon	1024	13396	0.08
ANN	31	Pier 1_Colon	889	14313	0.06	RF	31	Pier 2_RTeign	914	14359	0.06	SVM	31	Pier 2_RTeign	840	13396	0.06
ANN	31	Pier 2_RTeign	876	14313	0.06	RF	31	Pier 1_Colon	872	14359	0.06	SVM	31	Pier 1_Colon	833	13396	0.06
ANN	31	Abutment 1_RTeign	856	14313	0.06	RF	31	Abutment 1_RTeign	835	14359	0.06	SVM	31	Abutment 1_RTeign	816	13396	0.06
ANN	31	Pier 5_WCM15	725	14313	0.05	RF	31	Pier 3_RTeign	731	14359	0.05	SVM	31	Pier 5_WCM15	713	13396	0.05
ANN	31	Pier 3_RTeign	692	14313	0.05	RF	31	Abutment 2_RTeign	696	14359	0.05	SVM	31	Pier 3_RTeign	689	13396	0.05
ANN	31	Abutment 2_RTeign	649	14313	0.05	RF	31	Abutment 1_Colon	674	14359	0.05	SVM	31	Abutment 2_RTeign	664	13396	0.05
ANN	31	Abutment 1_Colon	643	14313	0.04	RF	31	Pier 5_WCM15	655	14359	0.05	SVM	31	Pier 1_RTeign	533	13396	0.04
ANN	31	Pier 8_WCM15A	533	14313	0.04	RF	31	Pier 2_WCM15A	542	14359	0.04	SVM	31	Pier 8_WCM15A	533	13396	0.04
ANN	31	Pier 1_RTeign	522	14313	0.04	RF	31	Pier 1_WCM15A	529	14359	0.04	SVM	31	Pier 1_WCM15A	513	13396	0.04
ANN	31	Pier 2_WCM15A	521	14313	0.04	RF	31	Pier 1_RTeign	496	14359	0.03	SVM	31	Pier 2_WCM15A	506	13396	0.04
ANN	31	Pier 1_WCM15A	511	14313	0.04	RF	31	Pier 6_WCM15	485	14359	0.03	SVM	31	Pier 4_WCM15	494	13396	0.04
ANN	31	Pier 4_WCM15	475	14313	0.03	RF	31	Pier 4_WCM15	481	14359	0.03	SVM	31	Pier 6_WCM15	479	13396	0.04
ANN	31	Pier 6_WCM15	462	14313	0.03	RF	31	Pier 8_WCM15A	480	14359	0.03	SVM	31	Pier 6_WCM15A	368	13396	0.03
ANN	31	Abutment 2_WCM15	443	14313	0.03	RF	31	Abutment 2_WCM15	401	14359	0.03	SVM	31	Pier 1_WCM15	363	13396	0.03
ANN	31	Pier 6_WCM15A	384	14313	0.03	RF	31	Pier 6_WCM15A	384	14359	0.03	SVM	31	Pier 5_WCM15A	346	13396	0.03
ANN	31	Pier 5_WCM15A	354	14313	0.02	RF	31	Pier 1_WCM15	343	14359	0.02	SVM	31	Abutment 2_WCM15	338	13396	0.03
ANN	31	Pier 1_WCM15	345	14313	0.02	RF	31	Pier 3_WCM15	338	14359	0.02	SVM	31	Pier 3_WCM15	332	13396	0.02
ANN	31	Pier 4_WCM15A	343	14313	0.02	RF	31	Abutment 1_WCM15A	331	14359	0.02	SVM	31	Pier 2_WCM15	315	13396	0.02
ANN	31	Pier 3_WCM15	331	14313	0.02	RF	31	Pier 5_WCM15A	309	14359	0.02	SVM	31	Pier 4_WCM15A	294	13396	0.02

ANN	31	Pier 2_WCM15	329	14313	0.02	RF	31	Pier 2_WCM15	302	14359	0.02	SVM	31	Abutment 1_Colon	278	13396	0.02
ANN	31	Abutment 1_WCM15A	319	14313	0.02	RF	31	Pier 4_WCM15A	288	14359	0.02	SVM	31	Pier 7_WCM15A	253	13396	0.02
ANN	31	Pier 7_WCM15A	295	14313	0.02	RF	31	Pier 7_WCM15A	284	14359	0.02	SVM	31	Pier 3_WCM15A	223	13396	0.02
ANN	31	Abutment 2_WCM15A	274	14313	0.02	RF	31	Abutment 2_WCM15A	278	14359	0.02	SVM	31	Abutment 1_WCM15	161	13396	0.01
ANN	31	Abutment 1_WCM15	256	14313	0.02	RF	31	Abutment 1_WCM15	247	14359	0.02	SVM	31	Abutment 1_WCM15A	148	13396	0.01
ANN	31	Pier 3_WCM15A	237	14313	0.02	RF	31	Pier 3_WCM15A	236	14359	0.02	SVM	31	Abutment 2_WCM15A	136	13396	0.01
ANN	61	Pier 2_Colon	1124	13883	0.08	RF	61	Pier 2_Colon	1124	13734	0.08	SVM	61	Pier 2_Colon	1074	12897	0.08
ANN	61	Abutment 2_Colon	1046	13883	0.08	RF	61	Abutment 2_Colon	1027	13734	0.07	SVM	61	Abutment 2_Colon	930	12897	0.07
ANN	61	Pier 1_Colon	843	13883	0.06	RF	61	Pier 1_Colon	846	13734	0.06	SVM	61	Pier 1_Colon	858	12897	0.07
ANN	61	Pier 2_RTeign	827	13883	0.06	RF	61	Pier 2_RTeign	814	13734	0.06	SVM	61	Pier 2_RTeign	845	12897	0.07
ANN	61	Abutment 1_RTeign	799	13883	0.06	RF	61	Abutment 1_RTeign	770	13734	0.06	SVM	61	Abutment 1_RTeign	842	12897	0.07
ANN	61	Pier 5_WCM15	727	13883	0.05	RF	61	Pier 5_WCM15	722	13734	0.05	SVM	61	Pier 5_WCM15	721	12897	0.06
ANN	61	Abutment 1_Colon	666	13883	0.05	RF	61	Pier 3_RTeign	693	13734	0.05	SVM	61	Pier 3_RTeign	657	12897	0.05
ANN	61	Abutment 2_RTeign	657	13883	0.05	RF	61	Abutment 2_RTeign	645	13734	0.05	SVM	61	Abutment 2_RTeign	616	12897	0.05
ANN	61	Pier 3_RTeign	649	13883	0.05	RF	61	Abutment 1_Colon	597	13734	0.04	SVM	61	Pier 1_RTeign	513	12897	0.04
ANN	61	Pier 8_WCM15A	550	13883	0.04	RF	61	Pier 8_WCM15A	548	13734	0.04	SVM	61	Pier 8_WCM15A	510	12897	0.04
ANN	61	Pier 1_RTeign	496	13883	0.04	RF	61	Pier 1_RTeign	506	13734	0.04	SVM	61	Pier 2_WCM15A	488	12897	0.04
ANN	61	Pier 2_WCM15A	465	13883	0.03	RF	61	Pier 4_WCM15	493	13734	0.04	SVM	61	Pier 4_WCM15	454	12897	0.04
ANN	61	Pier 4_WCM15	459	13883	0.03	RF	61	Pier 2_WCM15A	491	13734	0.04	SVM	61	Pier 1_WCM15A	452	12897	0.04
ANN	61	Pier 1_WCM15A	416	13883	0.03	RF	61	Pier 1_WCM15A	438	13734	0.03	SVM	61	Pier 6_WCM15	413	12897	0.03
ANN	61	Pier 6_WCM15A	407	13883	0.03	RF	61	Pier 6_WCM15	410	13734	0.03	SVM	61	Pier 6_WCM15A	389	12897	0.03
ANN	61	Pier 6_WCM15	396	13883	0.03	RF	61	Pier 6_WCM15A	394	13734	0.03	SVM	61	Pier 1_WCM15	368	12897	0.03
ANN	61	Abutment 2_WCM15	370	13883	0.03	RF	61	Abutment 2_WCM15	356	13734	0.03	SVM	61	Pier 3_WCM15	346	12897	0.03
ANN	61	Pier 1_WCM15	364	13883	0.03	RF	61	Pier 3_WCM15	342	13734	0.02	SVM	61	Pier 4_WCM15A	338	12897	0.03
ANN	61	Pier 5_WCM15A	359	13883	0.03	RF	61	Pier 2_WCM15	339	13734	0.02	SVM	61	Pier 5_WCM15A	337	12897	0.03
ANN	61	Pier 4_WCM15A	329	13883	0.02	RF	61	Pier 1_WCM15	326	13734	0.02	SVM	61	Pier 2_WCM15	324	12897	0.03

ANN	61	Pier 2_WCM15	321	13883	0.02	RF	61	Pier 5_WCM15A	305	13734	0.02	SVM	61	Pier 7_WCM15A	280	12897	0.02
ANN	61	Pier 7_WCM15A	301	13883	0.02	RF	61	Pier 7_WCM15A	302	13734	0.02	SVM	61	Pier 3_WCM15A	269	12897	0.02
ANN	61	Pier 3_WCM15	292	13883	0.02	RF	61	Pier 4_WCM15A	296	13734	0.02	SVM	61	Abutment 2_WCM15	265	12897	0.02
ANN	61	Abutment 1_WCM15A	275	13883	0.02	RF	61	Abutment 1_WCM15A	269	13734	0.02	SVM	61	Abutment 1_Colon	228	12897	0.02
ANN	61	Abutment 1_WCM15	272	13883	0.02	RF	61	Abutment 2_WCM15A	238	13734	0.02	SVM	61	Abutment 1_WCM15	146	12897	0.01
ANN	61	Abutment 2_WCM15A	261	13883	0.02	RF	61	Abutment 1_WCM15	223	13734	0.02	SVM	61	Abutment 1_WCM15A	117	12897	0.01
ANN	61	Pier 3_WCM15A	212	13883	0.02	RF	61	Pier 3_WCM15A	220	13734	0.02	SVM	61	Abutment 2_WCM15A	117	12897	0.01