RIVETED STEEL RAILWAY BRIDGE HEALTH MONITORING AND DAMAGE DETECTION

Ahmed Rageh
University of Nebraska-Lincoln, a.eissa.rageh@huskers.unl.edu

RIVETED STEEL RAILWAY BRIDGE HEALTH MONITORING AND DAMAGE DETECTION

by

Ahmed E. Rageh

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Doctor of Philosophy

Major: Civil Engineering

(Structural Engineering)

Under the Supervision of Professor Daniel G. Linzell

Lincoln, Nebraska

May 2020

RIVETED STEEL RAILWAY BRIDGE HEALTH MONITORING AND DAMAGE DETECTION

Ahmed E. Rageh, Ph.D.

University of Nebraska, 2020

Advisor: Daniel G. Linzell

Visual inspection is often used to assess the condition of railway bridges at discrete points in time, an approach that can be subjective and possibly unsafe. Alternatively, certain bridges have their condition assessed via the installation of a large number of sensors. These sensors can be costly to place, power and maintain. Therefore, reducing their numbers and maximizing the extracted information is of utmost importance. In addition, evaluating bridge condition from measured response can be quite challenging due to loading and environmental variations, especially when a limited number of sensors are used.

The focus of this research is to develop an automated hybrid experimental-numerical framework to detect and locate damage and estimate its intensity. The framework was developed analytically, based on Proper Orthogonal Modes (POMs) and Artificial Neural Networks (ANNs), and validated experimentally using 1 and 8 weeks of measured strains collected from a monitoring system placed onto an in-service, multi-span, railway bridge. The analytical work involved using three sensor instrumentation sets and investigated structural response for two bridge spans of different type and size. To generate training data for the ANNs, Modeling uncertainties that could lead to erroneous indication or omission of damage are incorporated into framework development via a systematic analyses. The procedure was based on synergizing POMs extracted from measured structural response and POMs calculated from the numerical model with a robust damage feature independent of level and location of modeling uncertainty. A hybrid experimental-numerical approach was developed and implemented to estimate damage scenario POMs from field measured strains. ANNs were trained and tested using these POMs with DL and DI being detected. These results show the promise of the POD-ANN method as a robust, real-time fatigue damage detection tool for steel railway bridges.

ACKNOWLEDGEMENTS

I would like to acknowledge and dedicate the dissertation to my wife Khadija Zogheib, my son Owais Rageh and my daughter Abrar Rageh who were very patient, supportive and showed me unfailing love while I pursued my master degree.

I would like to acknowledge and dedicate this work to my advisor, Daniel G. Linzell, who has been incredibly supportive and inspiring. I would like also to gratefully acknowledge the assistance, access, data and expertise provided by Union Pacific in association with this project. I would like also to acknowledge support provided by NSF Award #1636805 BD Spokes, Planning, Midwest: Big Data Innovations for Bridge Health.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 : Introduction

1.1. Background

Accurate and readily available bridge health assessment is essential to ensure safety, irrespective of their use. Many railway bridges in the U.S. are over 100 years old and are subjected to increased load demands and frequencies. As a result, accurate monitoring of their health is important to, at the least, prevent costly traffic interruption, and, more importantly, eliminate potentially catastrophic safety issues.

Current condition evaluation (i.e., health monitoring) processes for both highway and railway bridges are largely based on visual inspections at prescribed frequencies, which are subject to human error and, in the limit, may fail to reveal significant damage in a timely manner (1). Furthermore, mandating that these inspections occur at prescribed frequencies, frequently approaching 24 months, could be an inefficient use of human capital depending on current health (2).

In many cases, a structural deficiency is caused by the degradation of material properties or changes in geometry and; therefore, a change in response typically takes place. Detecting and evaluating these deficiencies can occur via deployment of a structural health monitoring (SHM) system. A SHM needs to detect damage, determine its location, assess its severity and estimate residual useful life. While extensive research has been completed in the SHM domain, a method that effectively integrates these components does not yet exist (3). Vibration-based methods are commonly implemented for SHM systems with frequencies, mode-shapes, or strain mode-shapes being investigated. However, modal parameters are generally sensitive to global damage and localized damage location and severity is, subsequently, difficult to detect (3-6).

In certain circles, SHM systems are classified as being either model- or field data-based (3). Model-based methods quantify bridge health by comparing results from a physical model, often a finite element model (FEM), against measured responses. A model updating strategy may be used to reduce discrepancies between measured and predicted responses (3). Model-based methods are very common when assessing bridge health; however, they may not be suitable for real-time applications. Developing the FE model can: be complicated and time-consuming; require expert knowledge when assessing performance and updating parameters; include effects of modeling uncertainty; and pose a challenge when environmental variations need to be incorporated. Field data-based bridge performance is usually solely assessed via examination of measured responses without the need for a physical model, which makes them suitable to real-time monitoring and quickly assessing bridge health (3).

Major challenges associated with SHM damage detection exist. These can include: optimizing measurement systems, including sensor locations, types and quantities; cost; measurement noise; non-stationary loads; environmental variations; modeling uncertainties; addressing existing deterioration effects; maintaining data continuity; and site and structures access. To address these issues, researchers have investigated: optimal vibration sensor placement (7-9); measurement noise effects on damage detection (10-12); non-stationary loading effects on vibration-based health monitoring (13); environmental variation effects on vibration-based damage detection

(13-15); uncertainties associated with common damage detection methods (16); and field applications (13).

To address two important challenges in the context of railway bridges, the effects of load uncertainty and environmental variability on SHM system effectiveness, researchers have investigating the use of statistical methods including Principal Component Analysis (PCA) (10, 17) and Singular Value Decomposition (SVD) (18) to eliminate or reduce those effects. However, those studies involved FE models and/or small-scale beam testing in a controlled environment subjected to known, dynamic inputs.

One approach to detect damage and address environmental and load variations is the utilization of Proper Orthogonal Decomposition (POD). POD is a statistical method that reduces the dimensionality of a given data set where Proper Orthogonal Modes (POMs), created from POD, are graphical means to identify damage, load, and environmental variations. POD has been used by Shane et al., Eftekhar Azam et al. and Lanata et al. to detect damage existence and severity when environmental and load variations were included (19-22). Those studies involved FE models to validate proposed frameworks and methodologies were not applied to transportation infrastructure.

Artificial Neural Networks (ANNs), which recognize patterns by building a non-linear parameterized mapping between a set of inputs and a set of outputs, have been applied to bridge damage detection by Zang et al., Jin et al., Dworakowski et al., Gu et al. and Sbarufatti (23-26). They have often been utilized with various statistical methods and filtering techniques, such as Independent Component Analysis (ICA), Kalman filters and variance analysis, to: detect damage (23-25); reduce SHM false positives; (26) and identify fatigue prone locations (27). They have not been used to detect damage location and severity in bridges using measured response due to a lack of training and testing data.

Modeling uncertainties have been shown to affect damage accumulation prediction and future damage estimates. Moaveni et al. performed shake table tests of a full-scale, seven story reinforced concrete building to study uncertainties in common damage detection methods using modal identification and FE model updating (16). It was concluded that modeling errors (e.g., mesh size) influenced the level of confidence in damage identification results (28, 29). To provide a systematic analysis and damage identification scheme, Papadimitriou et al. adopted transitional Markov Chain Monte Carlo for Bayesian damage identification (30). Model updating performance was shown to rely profoundly on the number of unknown parameters in the system (20), model class selection (i.e., complexity) (31) and user expertise.

Environmental effects have been shown to increase uncertainties associated with damage identification. Moaveni et al. (32) studied a pedestrian bridge where measured ambient temperature variations were shown to significantly influence field-estimated frequencies, which then introduced uncertainties into FEM updating (32). To eliminate these effects, a static polynomial regression technique that represented relationships between identified natural frequencies and measured temperatures was adopted.

Field data-based SHM appears more suitable for real-time monitoring to avoid disadvantages associated model-based methods and detect damage quickly (3). Additional research is needed to (1) ensure that damage existence, location and intensity are accurately identified when loading, environmental variations and modeling uncertainties are included and (2) develop reduced sensors set schemes for various types of bridges.

1.2. Problem statement

Ensuring the safety of railway bridges and avoiding closure or collapse is becoming more challenging because of the size of the network, infrastructure age and the absence of accurate, real-time assessment tools. Assessing the health of railway bridges via visual inspection at prescribed frequencies provides intermittent data that is subject to human interpretation and may fail to reveal significant damage.

1.3. Objectives

Even though model-based SHM is the most common damage detection framework, the current research relies on the use of measured field response to eliminate model-based disadvantages that include:

1. The need for sophisticated finite element models, which can be expensive and time consuming to develop; and
2. The inability to incorporate environmental variations.

The proposed framework is based on supervised machine learning methods to allow for detecting damage location and intensity. Unsupervised learning methods detect changes in bridge behavior, however; further investigation is needed to evaluate their causes.

The current research focuses on developing an autonomous, hybrid experimental-numerical SHM framework that utilizes reduced sensors sets and:

1. Accurately determines damage intensity and location;
2. Is robust against loading uncertainties;
3. Is robust against the effects of modeling uncertainties; and
4. Is robust against measurement noise.

1.4. Scope

The objectives were addressed by:

1. Performing a detailed literature review on model- and field data-based SHM systems.
2. Creating an analytical framework that:
    - Used POMs and ANNs to automate damage feature detection.
    - Allowed for damage location and intensity to be automatically varied using MATLAB and the SAP2000 Open Application Programming Interface (OAPI) to create ANN training and testing scenarios.
3. Applying the developed analytical framework to one week of measured strains from a truss span to validate applicability in a hybrid experimental-numerical framework SHM that used:
    - Measured data preprocessed to reduce loading variation effects with data windowing and cleansing automatically performed; and

    - POMs calculated from preprocessed, measured strains using ANNs trained and tested with data from a field deployed SHM system.
4. Generalizing the proposed framework to include various bridge spans, instrumentation plans and structural response.
5. Incorporating modeling uncertainties to allow for training the developed framework using damage scenarios developed from measured responses with:
    - Analytical models having various structural configurations and connection conditions being utilized for both base and uncertain models; and
    - Investigating development of a robust damage feature that is independent of modeling uncertainty.

Chapter 2 : Literature Review

Over 60% of railway bridges were constructed before 1950, with 50% of those structures being steel bridges that commonly used riveted connections (33). In general, steel bridges are subjected to a wide range of deficiencies associated with fatigue and corrosion (34). Specifically, common structural deficiencies in steel railway bridges include deterioration of stringer-to-floor beam and stringer-to-lower lateral bracing connections and frozen bearings (35). Bridge owners are certainly concerned about all these deficiencies; however, stringer-to-floor beam connection deterioration is of significant importance for these types of bridges as connection failure could cause extensive damage, traffic disruptions and possibly failure given that many are non-redundant systems (34, 36, 37). As a result, the current study focuses on identification of these types of deficiencies.

2.1. Riveted railway bridge connections

The behavior and integrity of riveted steel railway bridges have been examined by multiple investigators, with a number of studies focused on stringer to floor beam connections. A railway deck truss was examined and interaction between various bridge elements was shown to induce additional internal effects, such as stringer axial forces and floor beam lateral bending, due to main truss longitudinal deformations (38). Consequently, the fatigue life of those components and their connections might be compromised. Published numerical and laboratory research concluded that stringer-to-floor beam connections are more rigid than they initially assumed and, as a result, large bending stress cycles and corresponding fatigue degradation in connecting angles and rivets may develop (34, 36-42). An analytical study focused on stringer end fixity ratio effects on fatigue damage accumulation found that stringer-to-floor beam connections experienced more severe damage than floor beam-to-main girder connections in steel riveted plate girder railway bridges (40). Another study showed that, for double-track steel railway bridges, increased connection fatigue damage was observed when both tracks were loaded simultaneously (36). Stringer-to-floor beam end-fixity ratios vary widely and, as a result, no clearly defined relationship exists. A full-scale laboratory test on three panels from a demolished riveted steel railway bridge concluded that stringer ends could be subjected to negative end moments that were 67% of continuous stringer bending moments and vertical fatigue cracks developed in connecting angles under cyclic loading (37). Laboratory and field tests indicated a high amount of variations in end fixity ratios (34, 37-40, 42). Stringer-to-floor beam connection end fixity ratio was shown to influence stress time histories significantly and, in-turn, fatigue life predictions for connecting elements with fatigue crack development mainly attributed to out-of-plane deformation of connecting angles and stress concentrations at rivets heads (40).

2.2. Bridge condition assessment and SHM

Current bridge monitoring practice is based on visual inspections at prescribed frequencies, a process that is subject to human error and, in the limit, may fail to reveal significant damage in a timely manner (1). Additionally, performing these inspections at prescribed frequencies, often approaching 24 months regardless of the state of health of the structure, could be inefficient (2). These concerns have motivated extensive research on the smart evaluation of structural condition. The process of implementing a continuous and autonomous damage identification scheme for civil structures is commonly referred to as structural health monitoring (SHM). In many cases, a structural deficiency is caused by the degradation of material properties or changes in its geometry and, therefore, variations in the structure's dynamic properties. It is known that structural deficiencies oftentimes cannot be measured directly (43) and that modal properties including curvature (44, 45), Eigenmodes (28, 46, 47) and modal strain energy (48, 49) are sensitive to these types of deficiencies.

In certain circles, SHM systems are classified as being either model- or field data-based (3). Model-based methods quantify bridge health by comparing results from a physical model, often a finite element model (FEM), against measured responses. Model-based damage detection is executed in two stages: modal identification and model updating. The model updating strategy used to reduce discrepancies between measured and predicted responses (3). Model-based methods are very common when assessing bridge health; however, they may not be suitable for real-time applications since developing the FE model can be complicated, time-consuming and require expert knowledge when selecting updating parameters. One of the major disadvantages of model-based methods is the existence of modeling uncertainties that might affect predicted model responses significantly.

Issues associated with model-based SHM, parameter identification, and stochasticity of excitation sources have motivated SHM research focused on statistical response analysis. These methods have focused on identifying deviations from normal structural conditions from statistical analyses of acquired sensor data (50).

SHM systems, which extract information via processing measured responses and apply damage identification methods to the data to extract important information, are usually data-driven and incorporate sparse sensor networks to collect desired response quantities such as strains, accelerations or displacements (51). In field data-based methods, bridge performance is usually solely assessed via examination of measured responses without the need for a physical model, which makes them suitable to real-time monitoring and assessing the health of a bridge quickly (3).

In data-based SHM, detection methods focus on extracting damage features that are "hidden" in recorded sensor data. One of the major recent research objectives in the SHM community has been the development of automated feature extraction algorithms that enable autonomous damage identification methods when structures are subjected to various demands. A considerable amount of research effort has been dedicated to investigating key aspects associated with the development of data-driven SHM systems that can detect damage.

O'Connor et al. implemented a continuous monitoring system on a highway bridge and used coupled Statistical Process Control (SPC) and Gaussian Process Regression (GPR) for centralized damage identification based on novelty detection (52). GPR was employed to eliminate statistical variations caused by vehicle-bridge interaction and environmental effects and relatively long duration time series were used to determine the SPC threshold. Research has focused on using numerical techniques to identify the damage, such as one study that implemented Principle Component Analysis (PCA) to utilize frequency variations as damage indicators independent of site conditions (10). Laboratory tests of a short, base-excited cantilevered beam were completed at twelve temperatures with “damage” represented as changes in mass. PCA was shown to detect these “damage” levels successfully. Analytical studies were also performed and showed that frequencies were influenced by moving load mass and speed, damage location and intensity (10).

Recently, some researchers have focused on developing a damage localization framework utilizing Proper Orthogonal Decomposition (POD) (53). In one study, an algorithm that produced PODs based on a structure’s Frequency Response Function (FRF) was used to detect simulated beam damage, with that damage being effectively detected over a specific frequency range (54). Variations in Proper Orthogonal Modes (POMs) from an array of sensors were adopted as damage features. POMs are directly extracted from the structural response and are theoretically sensitive to both changes in system parameters and external excitation. Ruotolo et al. completed one of the first investigations of POD-based damage detection (55). They proposed a damage index based on structure accelerations and successfully detected stiffness reductions in a truss member under different operational conditions.

Vanlanduit et al. proposed using robust Singular Value Decomposition (SVD) to detect damage from changes in velocity spectra subspaces when the damaged structure was subjected to different non-structural surface treatments (18). Galvanetto et al. performed a similar study, but adopted variations in Proper Orthogonal Values (POVs) as damage indices (56). More recently, Shane et al. employed POD to detect damage severity and location in a composite beam numerical model (19). Eftekhar Azam et al. studied damage detection in a building subjected to base shear excitations using variations in an SVD subspace (20, 21). Bellino et al. proposed a PCA-based method for detecting damage in linear time-varying systems (10). Lanata et al. developed a method for static damage detection in beams via simulated experiments using POD (22). Xia et al. completed an investigation of damage detection using Principal Component Analysis (PCA) of dynamic strain data from a simulated bridge (57).

To date, all damage detection efforts utilizing POD-based feature extraction have focused on either broadband or stationary structure excitations. Those assumptions are often violated in real-life situations. To evaluate the effects of non-stationarity input on POD-based reduced-order models subject to various seismic excitations, Eftekhar Azam et al. trained a POD subspace of structural response of the Pirelli Tower in Milan subjected to El Centro earthquake time-histories (58). They verified the accuracy of the trained model when Kobe and Friuli time-histories were used to excite the structure.

Research focusing on using multiple numerical methods, including Artificial Neural Networks (ANNs), PCA and Radial Basis Functions, to detect wind turbine blade fatigue damage has shown promising results when compared against experimental studies that induced fatigue cracks (59). Another study on a prototype wind turbine compared SHM schemes based on statistical damage features against those based on modal parameters. It was concluded that statistically-based methods outperformed model-based methods and succeeded in identifying induced damage (60).

ANNs furnish a generic, non-linear parameterized mapping between an ensemble of inputs and a set of outputs. Once an ANN is trained on available sample data, it can recognize patterns; therefore, ANNs are suitable tools for signature analysis (24). Recently Jin et al. used an extended Kalman filter for estimating weights of a regression neural network for damage detection of a highway bridge under severe temperature changes (23). Dworakowski et al. proposed a classification neural network for fatigue damage detection in aircraft (25). Gu et al. proposed a framework based on ANNs for removing false-positive SHM alarms stemming from temperature variations (26). Sbarufatti proposed a framework for optimizing ANN hyperparameters based on analysis of variance and used the optimized ANN for fatigue damage identification (27).

2.3. Modeling Uncertainties (MU)

While data-driven methods have been shown to be an efficient tool to detect damage, current literature lacks data from actual, in-service structures needed to further train associated SHM frameworks. Even if a bridge owner allows for the exploration of several damage scenarios in-situ, sufficient data would not be produced to train the framework. To address this issue, the work presented herein implements a hybrid experimental-numerical, output-only approach to develop damage training scenarios under non-stationary excitations. Classical output-only damage detection methods are usually based on the operational modal analysis under stationary excitations and require low noise-to-signal ratios. To include non-stationary loading conditions, a damage detection methodology based on POD and ANNs that integrated analytical and experimental data was developed (61, 62). Supervised learning was used to classify output-only response, reduce Proper Orthogonal Mode (POM) variations from load variability and directly associate detected changes in Damage Intensity (DI) and Damage Location (DL). Several aforementioned damage identification methods rely on computational models for identification of structural damage; therefore, MU can affect their damage identification precision and robustness.

MUs that influence damage detection effectiveness have been addressed using three SHM frameworks: offline; online; and machine learning (16, 29, 63-67). Offline framework applications have been examined by multiple researchers, with one study investigating modeling error effects on a model-based framework using a numerical process that involved optimal selection of modes and modal residual weights to define multiple model updating classes (63). Bayesian model class selection and model averaging techniques were implemented to detect damage for selected classes and structural damage was detected with high accuracy when performed analytically. Another study involved examined modeling error effects on a damage detection framework based on wavelet coefficients developed using finite element (FE) analyses of two-dimensional (2D) frames under sinusoidal excitation (64). Simulated errors included excitation force, mass, support stiffness, member bending stiffness and damping ratios and the framework was minimally influenced by the errors.

Dynamic tests on a full-scale, seven-story, reinforced concrete building were conducted to further examine uncertainty effects on the accuracy of well-established damage detection methods in an offline framework (16, 29). Certain modal parameter uncertainties, including modeling errors stemming by mesh density, influenced the level of confidence associated with detected damage. Effects arising from modeling errors associated with nonlinear FE model updating have also been examined, with an Unscented Kalman Filter (UKF) used to estimate examined model parameters and data from dynamically tested buildings, one a two-dimensional steel frame and the other a three-dimensional concrete frame, investigating uncertainty influence on damage detection accuracy (65). Results showed that MUs had a significant impact on damage detection using updated FE models.

A companion study incorporated a UKF to estimate FE model parameters and a Linear Kalman Filter (LKF) to estimate simulation errors (68). A 2D steel frame model was employed to validate the approach and MUs included gravity loads, structure geometry, damping ratios and member inertia and results illustrated that using the proposed framework for model updating allowed for more accurate damage detection. The accuracy of online damage detection frameworks of nonlinear systems having unknown parameters and ill-defined numerical models have been investigated using a Bayesian approach and UKFs (66). Model complexity and parameterization effects that introduced uncertainties were examined via comparison against laboratory tests of a nonlinear joint setup and the effectiveness of the proposed method was demonstrated.

One study of machine learning damage detection frameworks looked at modal property based damage detection and implemented neural-networks to investigate modeling errors (67). The study found that mode-shapes component ratios or differences between those ratios, which are considered to be robust damage features, showed minimal sensitivity to modeling errors. Applicability of the framework was validated via numerical analyses, laboratory tests of a simple beam and field tests of a multi-beam bridge. It is recognized that using an integrated approach introduces inherent MUs largely related to model complexity.

2.4. Environmental variations and SHM

Environmental effects have been shown to increase uncertainties associated with damage identification and may drastically affect modal identification results. In some cases, variations in modal properties from environmental conditions could overshadow changes from structural deficiencies. Moaveni et al. (32) studied a pedestrian bridge where measured ambient temperature variations were shown to significantly influence field-estimated frequencies, which then introduced uncertainties into FEM updating (32). To eliminate these effects, a static polynomial regression technique that represented relationships between identified natural frequencies and measured temperatures was adopted.

Hu et al. (69) compared the effectiveness of different statistical approaches to remove environmental and/or operational effects using a large, continuously collected dynamic and environmental response data set from two pedestrian bridges. The study concluded that measured temperature variations had a significant impact on frequencies with a nonlinear relationship being observed. Statistical approaches that were examined included multiple linear regression and principal component regression, with both successfully removed the influence of environmental variations where damage scenarios being clearly detected (69).

Kim et al. (13) developed a Bayesian framework to account for temperature and traffic loads for a long-term monitoring system that examined a highway simply supported bridge and focused on its dynamic responses. Three combinations of recorded data were used in conjunction with the proposed Bayesian framework to examine the effect of temperature and vehicle loads as environmental and operational factors. The combinations included: (i) frequencies, temperature and traffic loads; (ii) frequencies and temperature; and (iii) frequencies only. The study concluded that considering temperature and traffic load variations yielded more accurate bridge health assessments when compared with the other two scenarios (13).

Silva et al. (70) used Deep Principal Component Analysis (DPCA), which was developed to address PCA limitations, to account for environmental variations on estimated frequencies from acceleration measured continuously from the Z-24 and Tamara bridges. DPCA was shown to effectively reduce environmental effects on modal properties and improved the accuracy of damage detection. The relationship between measured frequencies and environmental variations was shown to be nonlinear for the Z-24 bridge and linear for Tamara bridge.

In many cases, traffic loads or seismic excitations are classified as environmental variations and/or operational factors (13). Most operational modal analysis (OMA) algorithms equate ambient excitations to stationary, white noise. In many cases, such assumptions could be violated and consequently, modal properties would not be consistently identified. Research to address issues caused by non-stationary external inputs on the modal identification process was completed by Ghahari et al. and Abazarsa et al. (71, 72). Ghahari et al. developed a modal identification method to account for non-stationary (i.e., seismic) excitations that was analytically validated against a 5-story shear building. The proposed method, which utilized blind modal identification, accurately detected frequencies and mode-shapes. Abazarsa et al. verified the method using data extracted from shake table testing.

2.5. POD-ANN method for damage identification

2.5.1. POD for feature extraction

POD of a data set is accomplished by obtaining a set of ordered, orthonormal bases and collecting detailed information concerning relevant energy contents. As a result, POD addresses feature extraction by discovering underlying, hidden information in the data and dimensionality reduction by appropriately capturing dynamic system features in the smallest corresponding subspace. The concept of POD was central to the development of various techniques, such as Principal Component Analysis (PCA); Karhunen–Loève decomposition (KLD), and Singular Value Decomposition (SVD) (73-75). A detailed discussion of PCA, KLD and SVD commonalities can be found elsewhere (76).

2.5.1.1. Physical interpretations of POD

Close connections between POMs and natural Eigenmodes of mechanistic systems have been established (77, 78). Theoretical and experimental research has also attempted to link POMs to linear and nonlinear mechanical system Eigenmodes (79, 80). Free vibrations of an undamped linear system having a mass matrix proportional to the identity matrix were shown to provide POMs that asymptotically converged to its Eigenmodes (78). POMs of a lightly damped similar system were, in turn, shown to closely estimate its Eigenmodes, except for forced harmonic vibrations. Findings also showed that independent of system mass, POMs could coincide with Eigenmodes for certain resonant frequencies (81). North established a general criterion relating POMs to Eigenmodes for mechanical systems excited by noise (82).

2.5.1.2. PCA and SVD as POD methods

PCA looks for the subspace that features maximum data series variability when attempting to discover core dependency structures within that data. This can be interpreted as, in a state-space domain, directions in which data variability is important. When dealing with a structural system, those directions are akin to its Eigenvectors, which depend only on system properties. In this case, the principal subspace is purely statistical and is a function of external excitations and mechanistic properties of the system.

The current study uses data sets comprised of samples taken from time histories of in-situ bridge response to a train passage, with data from each train passage being stored in snapshot matrices. SVD of the snapshot matrix helps to extract damage features (75):

$$\mathbf{U} = \mathbf{L}\,\boldsymbol{\Sigma}\,\mathbf{R}^{\mathrm{T}}. \tag{1}$$

Where: $\mathbf{U} \in \mathbb{R}^{n_m \times n_s}$ is the snapshot matrix from n_m measurements and n_s samples; $\mathbf{L} \in \mathbb{R}^{n_m \times n_m}$ is an orthonormal matrix whose columns are the left singular vectors of $\mathbf{U}$; $\boldsymbol{\Sigma} \in \mathbb{R}^{n_m \times n_s}$ is a diagonal semi-matrix whose components Σ_{ii} are singular values of $\mathbf{U}$; and $\mathbf{R} \in \mathbb{R}^{n_s \times n_s}$ is an orthonormal matrix whose columns are the right singular vectors of $\mathbf{U}$.

It is known that the left singular vectors are POMs of the snapshot matrix. It was shown that first bridge response POMs to train passages (i.e., the first left singular vector of the corresponding snapshot matrices) contain information on intensity and location of damage at the stringer to floor-beam connections. It should be noted that, unlike linear vibration modes, structural response POMs could vary as excitation source changes. Therefore, when POMs are used as damage features, a machine learning algorithm is needed to differentiate variations induced by damage from ones caused by load variations.

2.5.2. ANN for damage identification

Feedforward ANNs have been extensively studied for structural damage identification. To facilitate autonomous damage identification, a two-layer, feedforward ANN was adopted by several authors for creating nonlinear mapping between damage indices and features extracted from the structural response (25, 26, 83). Based on these studies, ANN feedforward damage identification models have been constructed based on a linear combination of predetermined nonlinear basis functions $\xi_j(\boldsymbol{\varphi})$ (84):

$$\boldsymbol{d}(\boldsymbol{\varphi}, \mathbf{W}) = f\left(\sum_{j=1}^{M} w_j \, \xi_j(\boldsymbol{\varphi})\right). \tag{2}$$

Where: $\boldsymbol{d}$ is the vector of damage indices; $\boldsymbol{\varphi}$ is the damage feature; $\mathbf{W}$ is the matrix of ANN weights; and $f(\blacksquare)$ is the identity for regression problems and is a nonlinear activation function (i.e., *softmax*). It was proven that this architecture approximated arbitrary nonlinear functions well (85, 86). The relationship between input and the j^{th} component of the output for this type of ANN is given by (84):

$$d_k(\boldsymbol{\varphi}, \mathbf{W}) = \sigma\left(\sum_{j=1}^{M} \mathrm{W}_{kj}^{(2)} \, h\left(\sum_{i=1}^{D} \mathrm{W}_{ji}^{(1)} \, \varphi_i + \mathrm{W}_{j0}^{(1)}\right) + \mathrm{W}_{k0}^{(2)}\right). \tag{3}$$

Where: $\boldsymbol{d} \in \mathbb{R}^{n_d}$ is the vector whose rows feature damage indices; $\boldsymbol{\varphi} \in \mathbb{R}^{n_s}$ are damage feature vectors; M denotes the number of neurons in the hidden layer; $\mathrm{W}_{kj}^{(2)}$ and $\mathrm{W}_{k0}^{(2)}$

represent weights and biases of the output layer; and $W_{ji}^{(1)}$ and $W_{j0}^{(1)}$ represent the weights and biases of the hidden layer. In this study a hyperbolic tangent sigmoid activation function, $h(\blacksquare)$, is employed for the hidden layer. The activation function for the regression output layer $\sigma(\blacksquare)$ is represented by the identity matrix. In a supervised learning-based damage identification scheme, ANN weights $\mathbf{W}$ need to be obtained using a set of damage features and corresponding damage indices.

2.5.3. Overview of POD-ANN for damage identification

The authors developed a supervised Machine Learning scheme for detecting, locating, and quantifying the intensity of fatigue-induced damage in railway bridges using POMs and ANNs (62). A neural classifier was trained to categorize response to different load patterns, and a regression ANN was subsequently trained using an ensemble of applied loads to detect possible damage from resulting, categorized POMs. In doing so, the average strain time-history root mean square (RMS) for each train passage was used as a feature weight classification and the first snapshot matrix POM was used as the damage feature. Since damage feature and, subsequently, damage detection accuracy could be sensitive to variations in train load under operational conditions, the ANN needed to be robust to address damage scenarios and train axle loads not used for training.

It is known that Bayesian regularized ANNs are more robust than standard back-propagation ANNs and can decrease the need for cross-validation. Bayesian regularized ANNs are difficult to over-train since evidence procedures provide an objective Bayesian criterion for stopping training using early cessation. Moreover, the regularization term added to their objective function also makes them robust to overfitting. To improve ANN generalization capabilities for damage identification under operational conditions, Bayesian regularization was adopted for optimally finding ANN weights (85, 87). The following objective function needed to be minimized to find the optimal weights:

$$\mathrm{E} = \beta\ \mathrm{E}^{d} + \alpha\ \mathrm{E}^{\mathrm{w}} \tag{4}$$

in which:

$$\mathrm{E}^{\mathrm{w}} = \sum_{j=1}^{N_{\mathrm{w}}} \mathrm{w}_j^2 \tag{5}$$

and:

$$\mathrm{E}^{d} = \sum_{\mathrm{k}=1}^{\mathrm{N}} \lVert \mathbf{t}_{\mathrm{k}} - \mathrm{y}_{\mathrm{k}}(\mathbf{x}_{\mathrm{k}}, \mathbf{w}) \rVert^2. \tag{6}$$

In Equations 4-5, $\mathbf{w}$ is a vector that includes all network weights; α and β are objective function parameters; and $\mathbf{x}_{\mathrm{k}}$ and $\mathbf{t}_{\mathrm{k}}$ respectively denote the training input vectors and their corresponding target values for $k = 1,2,\dots,N$. The ratio of objective function parameters determines training emphasis, with larger α/β pushing the network towards generalization and smaller ratios driving the network towards error minimization (88).

For the previous study, the first POM $\boldsymbol{\varphi} \in \mathbb{R}^{n_m}$ of bridge strain response to train passage was used as a damage feature (61, 62). Therefore, when using N_t train passage scenarios and N_d damage scenarios, the following input matrix can be used for training the ANN:

$$[[\boldsymbol{\varphi}^{1,1} \quad \cdots \quad \boldsymbol{\varphi}^{N_d,1}] \quad \cdots \quad [\boldsymbol{\varphi}^{1,N_t} \quad \cdots \quad \boldsymbol{\varphi}^{N_d,N_t}]]_{n_m \times (N_t \times N_d)} \tag{7}$$

Where superscripts for $\boldsymbol{\varphi}$ respectively denote the damage and the training scenario.

The current research work proposed a hybrid experimental-numerical damage detection framework based on POD-ANN to detect damages in a real-world bridge system. MU effects on the developed framework were also examined and addressed. Model-based SHM systems were not adopted for the current research because they can be time-consuming to construct and environmental variations can be challenging to incorporate. It is also important to mention that supervised learning was used to allow for direct detection of damage location and intensity as opposed to unsupervised learning methods that detect general behavioral changes only.

Chapter 3 : Investigated Bridge and Analytical Models

3.1. Bridge description

The bridge under study is located in central Nebraska and is an open deck, double-track structure consisting of five, single span, steel Warren trusses and six, single span, steel, through girder systems (Fig. 1). The total length of the bridge is 381.0 m. The bridge spans are simply supported and composed of rolled and riveted built-up steel elements. Each span supports two tracks spaced laterally at 3.95 m center-to-center. The rails rest on wood ties that are supported by stringers spaced laterally at 2.15 m on center.

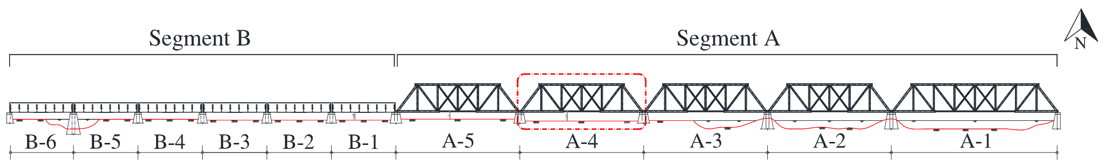


Fig. 1 Bridge elevation looking north, studied span circled.

The truss span is 44.7 m long and contains six panels with stringers connected to floor beams that are spaced 7.45 m on-center longitudinally. A lateral wind bracing system is provided using top and bottom laterals. Truss diagonals, end-posts, verticals, top chords and end bottom chords are riveted built-up members while midspan diagonals and bottom chords are eyebars of varying thickness. Floor beams and stringers are riveted, built-up I-sections with the floor beams composed of a web plate, flange angles and cover plates while the stringers use web plate and flange angles. Lower lateral bracing members are single angles of varying dimensions, while upper lateral bracing is laced angles. Stringer lateral bracing is also provided using single angles located close to the top flange. An elevation and plan view of the examined truss span is shown in Fig. 2.

Through girder spans are 22.0 m long and divided into 7 panels with floor beams longitudinally spaced at 3.14 m. Through girders and floor beams are riveted, built-up-I-sections having a web plate and flanges constructed using angles and cover plates of varying number and thickness. Stringers are rolled, S 24x80, I-beams. Bottom lateral bracing is composed of single angles of varying dimensions. A girder elevation and span framing plan are shown in Fig. 3. More details about cross-sectional dimensions can be found in (35).

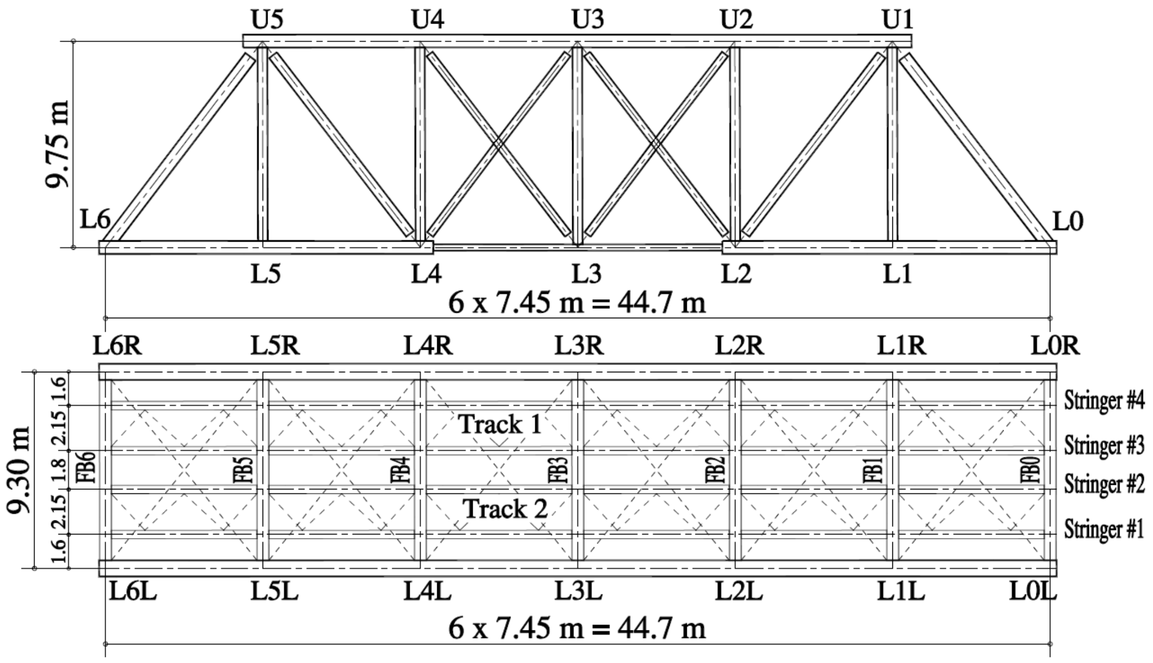


Fig. 2 Truss span elevation and plan view.

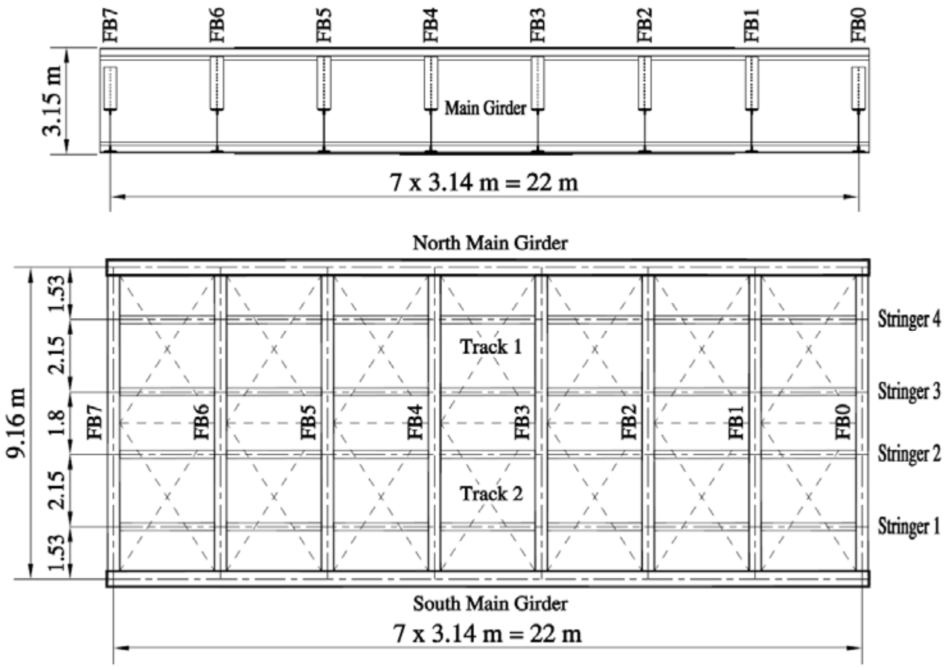


Fig. 3 Through girder span elevation and plan view.

3.2. Analytical models

Three-dimensional frame finite-element models were developed in SAP2000 for the bridge spans under study. The constructed models contained the trusses and plate girders, stringers, floor beams and bracing systems. Developed models were calibrated against measured response under routine train passages. Acceptable agreement existed between measured and predicted responses (35). Train loads were applied via a set of moving point loads. Riveted truss elements and floor beams were modeled as rigidly connected at their ends while truss eyebars and bracing members were pinned. Stiff rotational springs were used at the ends of each stringer span to facilitate simulating connection damage via reduction of their stiffness. Isometric views of the developed models are shown in Fig. 4. More details about constructed models are provided in (35).

The bridge models were excited using actual train loads that traversed the structures. Weigh-In-Motion (WIM) recorded loading configurations for 81 trains having varying axle loads, spacings and overall lengths were selected. The numerical analyses and extraction of significant results for each of the studied damage scenarios were automatically performed using MATLAB (89) and SAP2000 Open Application Programming Interface (OAPI) for the 81 trains with strains extracted at the stringer ends (62). Extracted strains at the instrumented locations were placed into matrices with each matrix contained time histories for one train passage “snapshot.” Means of the RMSs of the snapshot matrices were used to sort train events.

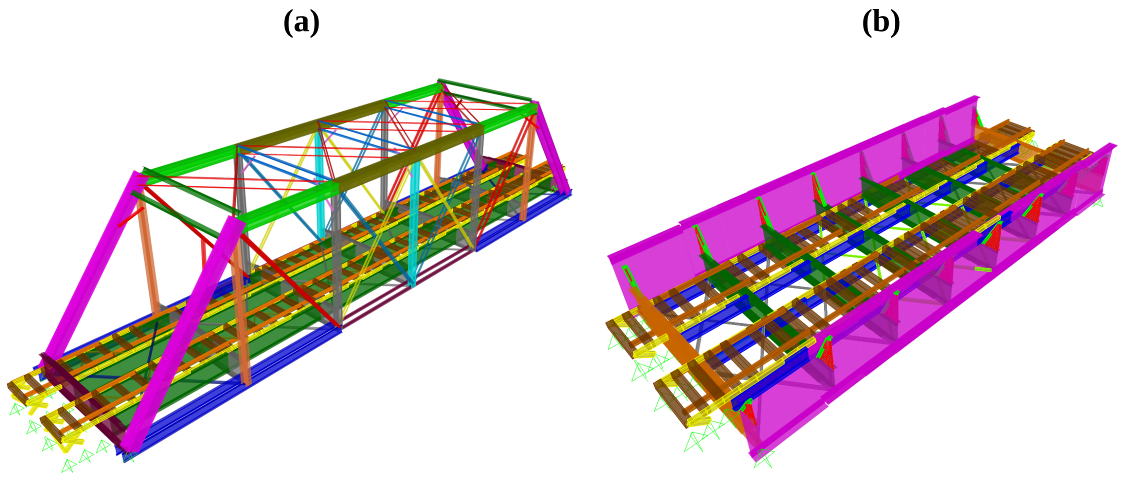


Fig. 4 SAP2000 isometric view: (a) truss span; and (b) plate girder span.

3.3. Analytical stringer-to-floor beam connection damage simulation

As mentioned earlier, stringer-to-floor beam connections were modeled using rotational springs to facilitate simulating damage via reduction in rotational stiffness.

Published fatigue laboratory tests of two panels systems containing stringers with riveted end connections showed a gradual decrease in rotational stiffness associated with the crack growth (37). As a result, a semi-rigid connection would ultimately become a pinned connection. It was reported that the developed crack vertical projection depth at which pinned behavior was observed was at approximately 40% of the angle leg length. Fig. 5 depicts typically developed fatigue cracks in the angle legs and Fig. 6 shows the corresponding decreases in rotational stiffness (37).

Continuously reducing connection rotational stiffness could be used to simulate crack propagation and associated reduction in stringer end strains. For the current research work, the damage was simulated at the 20 stringer ends by reducing end spring rotational stiffness between 0% and 100% (i.e., DI = 0% to DI = 100%) in increments of 10%. Stiffness reductions were applied to one stringer end at a time, with other connections being undamaged. Connection damage was simulated at one side of the floor-beam because either side would affect stringer continuity, resulting bending moments and stresses.

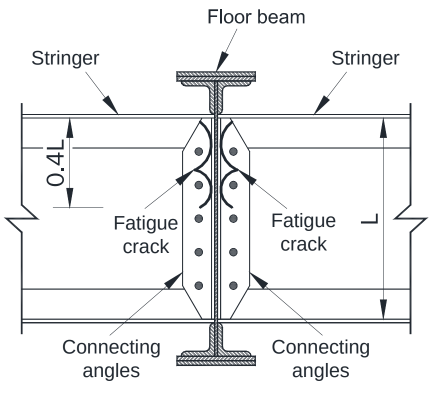


Fig. 5 Depiction of fatigue cracks in stringer-to-floor beam connecting angles (37).

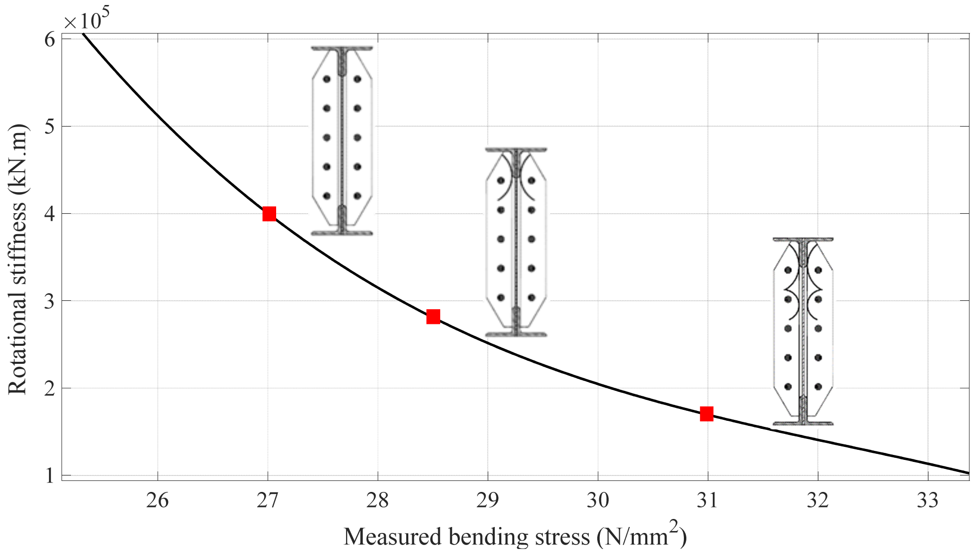


Fig. 6 Rotational stiffness reductions associated with crack propagation (37).

Chapter 4 : POD-ANN Strain Based Damage Detection Framework Development

Published: the Journal of Structural Control and Health Monitoring, https://doi.org/10.1002/stc.2288

Eftekhar Azam, S., Rageh, A., & Linzell, D.

4.1. Abstract

In this Chapter, a supervised learning scheme is proposed for detecting, locating and quantifying the intensity of damage in structures using Artificial Neural Networks (ANNs) and Proper Orthogonal Decomposition (POD). For structural systems, such as buildings and bridges, Proper Orthogonal Modes (POMs), denoted as $\boldsymbol{\varphi}$, associated with their response are functions of:

(i) Applied external loads.

(ii) Mechanistic properties.

To detect damage location and intensity, a supervised learning strategy was adopted to help discriminate POM ($\boldsymbol{\varphi}$) variations due to damage from those that were caused by applied load variations. A neural classifier was trained to categorize response to different load patterns and a regression ANN was subsequently trained using an ensemble of applied loads to detect possible damage from the categorized $\boldsymbol{\varphi}$. To demonstrate the effectiveness of the proposed approach, simulated experiments were performed with the intent of identifying damage indices in the truss span of the investigated bridge, see Fig. 2. The validated, three-dimensional (3D) finite element (FE)

model of the span shown in Fig. 4 (a) was used to generate strain time histories under train loads measured from Weigh-In-Motion (WIM) stations near the bridge.

Due to the relatively short duration of each loading event, it is assumed that environmental variabilities are negligible within the time interval of each train passage. However, for different time intervals or under different loading conditions, it is understood that environmental variability may need to be included.

4.2. ANNs for snapshot matrix classification, POM regression and damage identification

As stated earlier, ANNs have been extensively studied in association with structural damage identification. The current study explores the use of two-stage supervised ANNs for structural damage identification, as outlined in Fig. 7. In the first stage, a classification ANN is employed to assign the snapshot matrix to a certain response category. In the second stage, an ANN is trained for nonlinear regression to a set of POMs and their associated damage indexes. The resulting $\boldsymbol{\varphi}$ are subsequently used to identify damage. MATLAB Neural Network Toolbox was used for data analysis and constructing the classification and regression networks (90).

4.3. Coupled POM and ANN output-only damage detection

As presented in **Chapter 3**Chapter 3, the proposed, coupled POD-ANN framework was validated using simulated experiments analyzing response of the truss span using FE models subjected to recorded train loads from WIM stations near the bridge site. Measured train axle loads traversed the model and structural response to those loads was examined by studying strain time-histories.

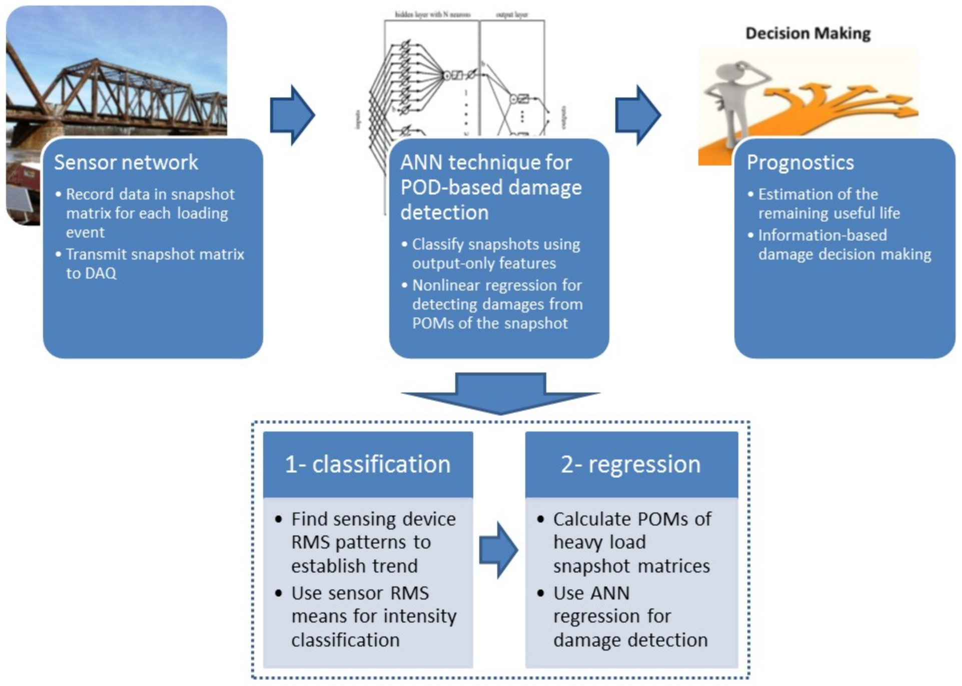

Decision Making
Sensor network
• Record data in snapshot matrix for each loading event
• Transmit snapshot matrix to DAQ
ANN technique for POD-based damage detection
• Classify snapshots using output-only features
• Nonlinear regression for detecting damages from POMs of the snapshot
Prognostics
• Estimation of the remaining useful life
• Information-based damage decision making
1- classification
• Find sensing device RMS patterns to establish trend
• Use sensor RMS means for intensity classification
2- regression
• Calculate POMs of heavy load snapshot matrices
• Use ANN regression for damage detection

Fig. 7 Proposed Nonlinear, Two-Layer, ANN technique for POD-based damage detection.

A numerical sensitivity analysis was completed by Rageh (35) on the bridge considered span (Fig. 2) to quantify change in strains time histories due to deficiencies at a set of instrumented locations. The study concluded that using 20 sensors at locations selected for this study would be the smallest set of sensors needed for detecting stringer-to-floor beam connection damage. See Section 3.1 for further details on the bridge structure.

A parametric study using the analytical model described in Section 3.2 was completed to examine $\boldsymbol{\varphi}$ variations as a function of changes in:

(i) Non-stationary excitations caused by moving trains.

(ii) Structural damage locations (DL) and Damage Intensity (DI).

WIM data for 81 trains was used for the FE simulations, with this data being assumed to statistically represent the distribution of train loads under operational conditions. Strain sensors were used because they provide a more direct means of monitoring certain deficiencies, such as fatigue damage. Moreover, strain measurements can provide more sensitivity to local changes in structural response. Strains were extracted from the analytical models at the instrumented locations (IL) shown in Fig. 8. It is important to reiterate that damage locations (DLs) were at instrument locations that could detect damage on either side of the floor-beam.

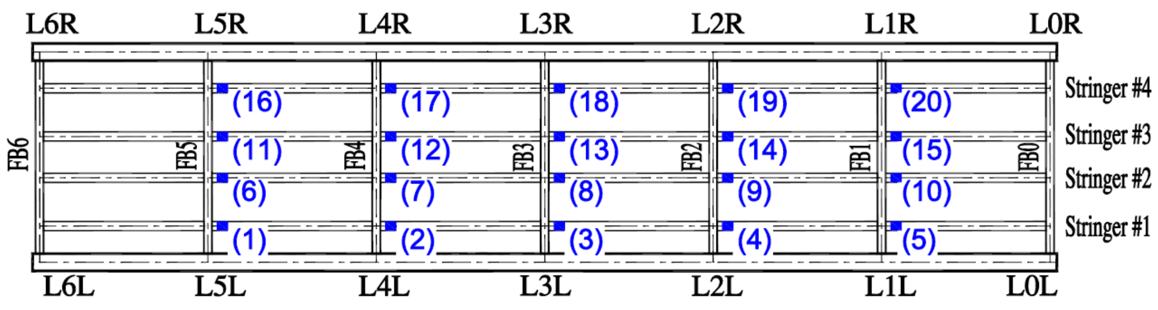


Fig. 8 Instrument locations.

4.3.1. Non-stationary excitation influence on POMs

Non-stationary excitation sensitivity was initially examined by looking at the effect of train speed on POMs ($\boldsymbol{\varphi}$), with $\boldsymbol{\varphi}$ developed based on strain time-history information as detailed below ("output"). A representative surface plot, one that shows changes in values for components of the first POM of each snapshot matrix at different ILs (FEM nodes reporting strain time-histories) for a healthy bridge and reference train speed is shown in Fig. 9. For this case, 50 mph was chosen as the reference train speed and surface plots are presented as the algebraic difference between $\boldsymbol{\varphi}$ components at 25 mph and 75 mph speeds. The figure shows surface plots for $\boldsymbol{\varphi_{25}} - \boldsymbol{\varphi_{50}}$ and $\boldsymbol{\varphi_{25}} - \boldsymbol{\varphi_{75}}$ intervals with observed values that deviate from zero indicating $\boldsymbol{\varphi}$ changes as a function of train speed.

To reduce changes in $\boldsymbol{\varphi}$ when dealing with snapshot matrices associated with trains with different speeds, samples in the snapshot matrix were chosen so that they featured the same number of IL strain peaks during train passage which resulted in unequal-length of snapshot matrices. This was accomplished using an automated peak-picking algorithm in MATLAB (89). As evidenced by plots in Fig. 10, including the same number of peaks could visibly reduce the variability of $\boldsymbol{\varphi}$ caused by changes in train speed.

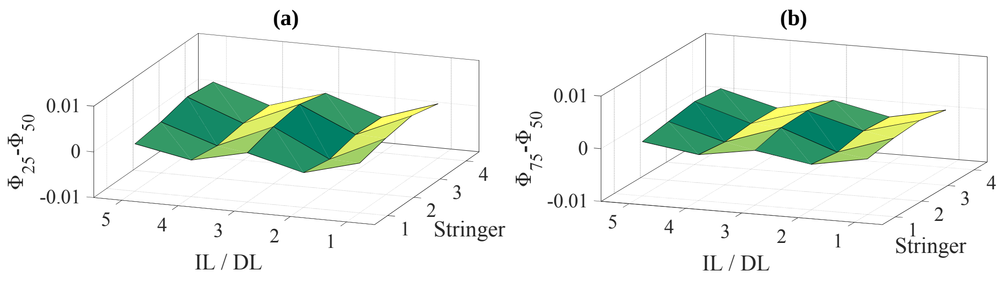


Fig. 9 Train speed influence on $\boldsymbol{\varphi}$, equal-length snapshot matrices: (a) $\boldsymbol{\varphi_{25}} - \boldsymbol{\varphi_{50}}$; and (b) $\boldsymbol{\varphi_{25}} - \boldsymbol{\varphi_{75}}$.

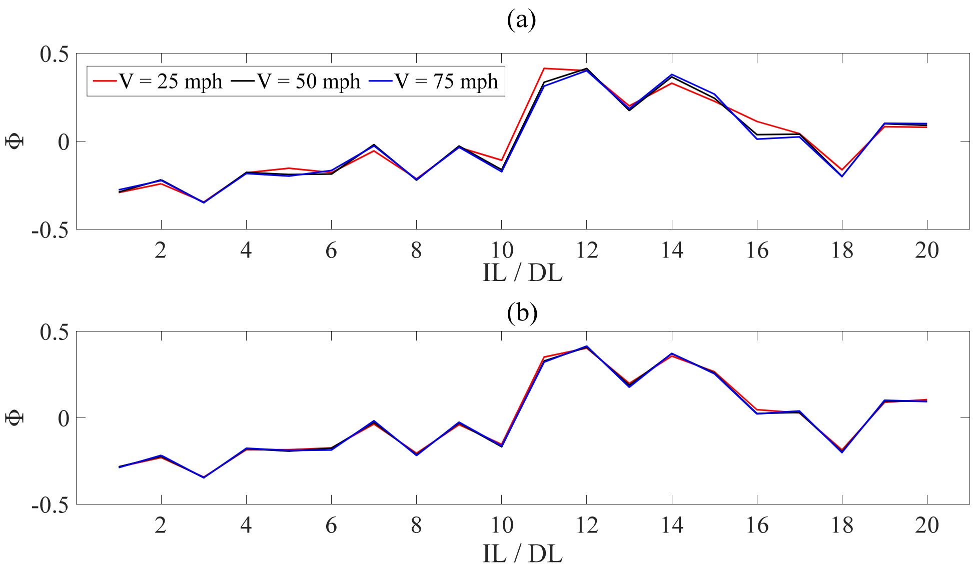


Fig. 10 Train speed influence on : (a) equal; and (b) unequal-length snapshot matrices.

After peak picking, the 81 trains involved in the study were sorted using strain time history root mean squares (RMS) extracted at the ILs shown in Fig. 2. The resulting, sorted, train RMS data is shown in numerical order by sensor number in Fig. 11 along with accompanying statically equivalent uniform loads. Results are nondimensionalized with respect to maximum RMS and statically equivalent loads for the 81 trains.

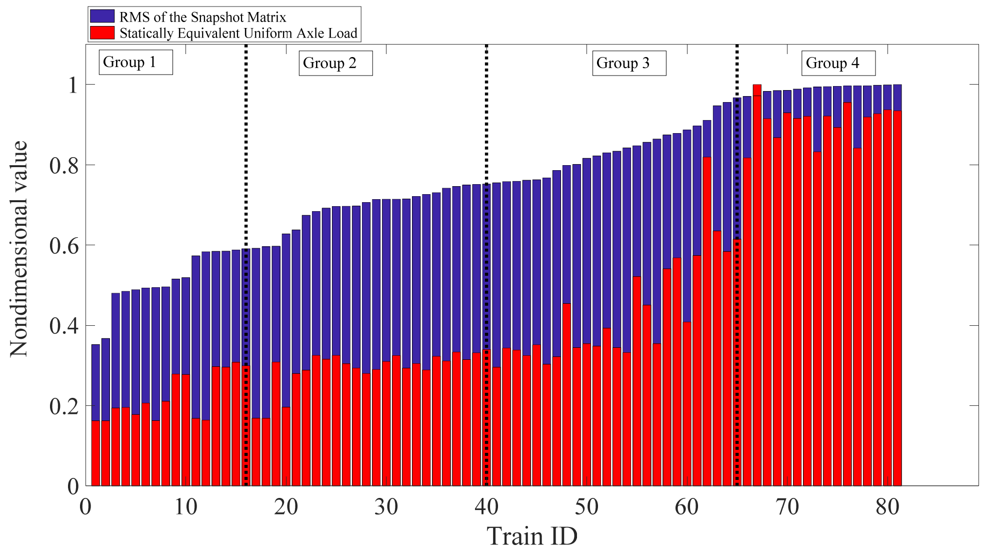


Fig. 11 Trains sorted by nondimensionalized strain time-history RMS with corresponding nondimensionalized statically equivalent uniform load.

To examine $\boldsymbol{\varphi}$ variability in snapshot matrices having relatively close RMS values, trains were arbitrarily divided into four groups based on RMS as denoted by the vertical black dashed lines in Fig. 11. Group 1 (lowest RMS) included 16 trains, Group 2 24 trains, Group 3 25 trains and Group 4 16 trains. $\boldsymbol{\varphi}$ for each group are shown in the 2D plots in Fig. 12, with Group 4 having the best correlation.

Group 4 allowed for efficient training of a regression-based ANN, with the ANN being used to link the first $\boldsymbol{\varphi}$ of the snapshot matrices to damage indices. ANN training used 70% of the available samples, 15% for validation, and 15% for testing.

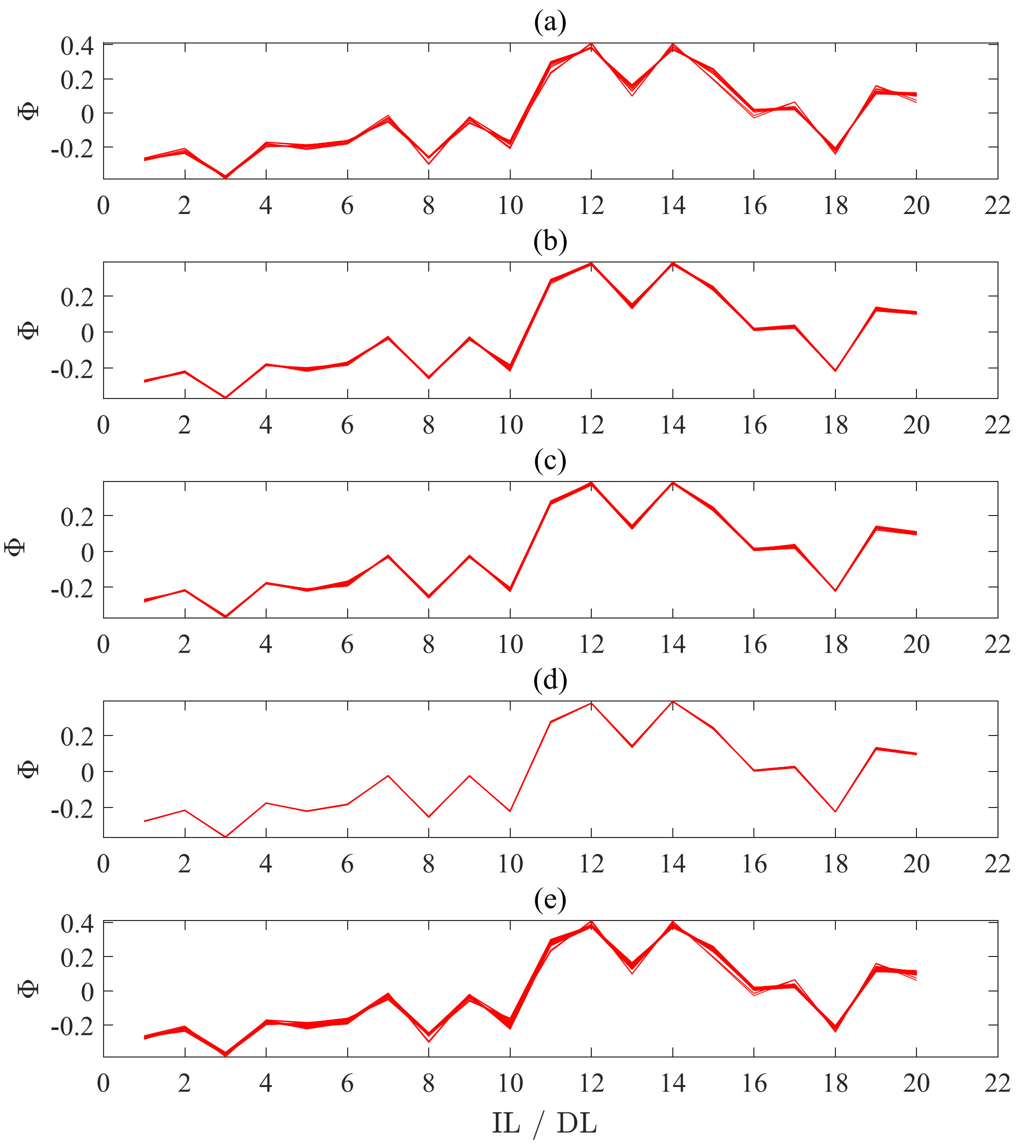


Fig. 12 $\boldsymbol{\varphi}$ plotted by grouped train passages: (a) Group 1; (b) Group 2; (c) Group 3; (d) Group 4; and (e) all groups.

4.3.2. Measurement noise influence on POMs

One of the most important features in any SHM framework is robustness to noise. In this section, the effect of measurement noise intensity and spatial correlation on $\boldsymbol{\varphi}$ (i.e., correlation between $\boldsymbol{\varphi}$s in a spatial domain) was studied. Fig. 13 depicts strain snapshot matrices obtained from 4 loading scenarios. Uncorrelated noise was considered using 10% and 15% RMS noise to signal ratios and extracted $\boldsymbol{\varphi}$ were compared against reference signals. In Fig. 14, the noise was spatially correlated. As seen in Fig. 13 and Fig. 14, POMs were shown to be robust against both elevated noise levels and spatial correlation.

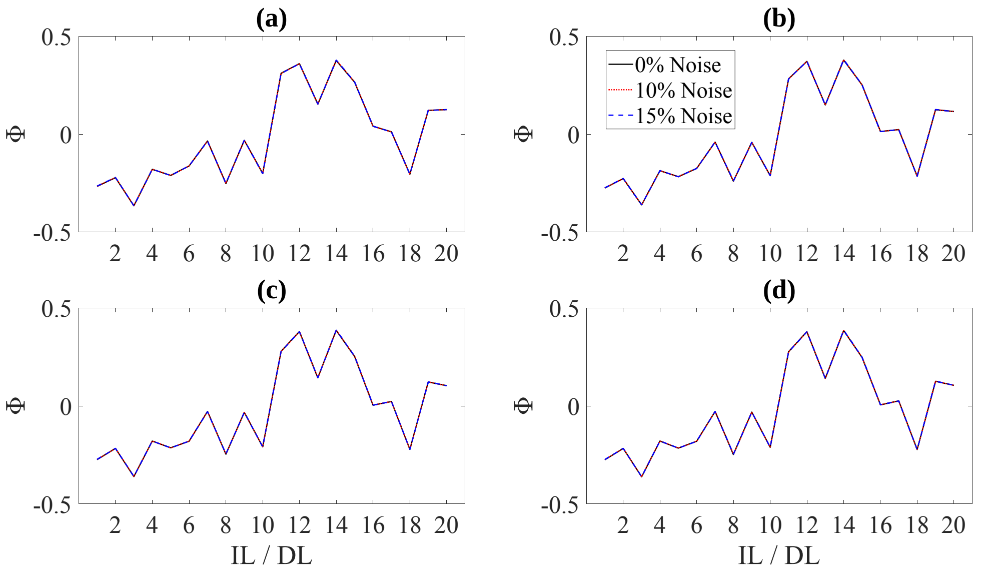


Fig. 13 Spatially uncorrelated $\boldsymbol{\varphi}$ at various noise intensities: (a) Train 67; (b) Train 70; (c) Train 73; and (d) Train 76.

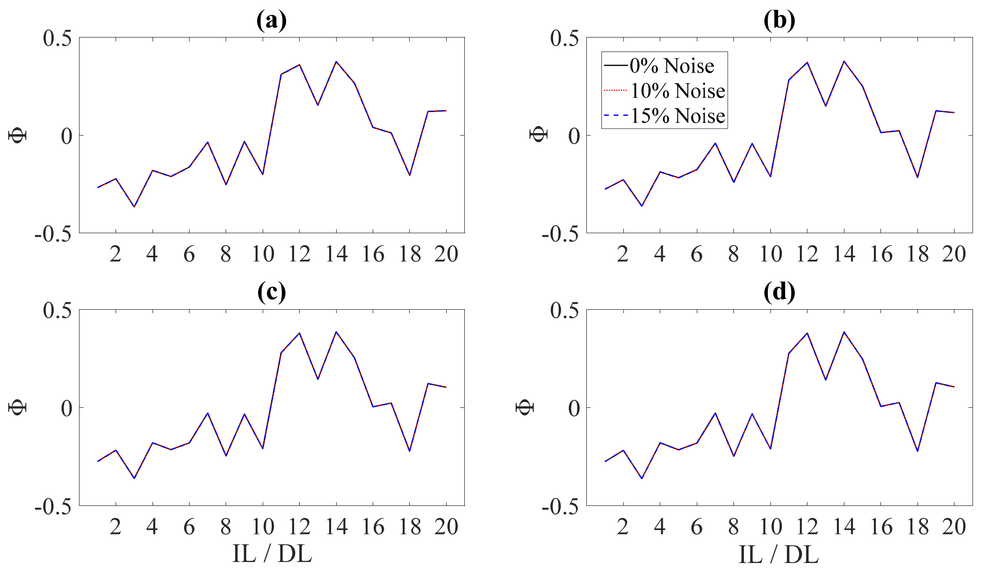


Fig. 14 Spatially correlated $\boldsymbol{\varphi}$ at various noise intensities: (a) Train 67; (b) Train 70; (c) Train 73; and (d) Train 76.

4.3.3. Structural deficiency influence on POMs

Structural deficiency influence on $\boldsymbol{\varphi}$ was initially investigated by reducing stringer-to-floor-beam rotational stiffness to mimic development of fatigue cracks at their connections. POMs for the healthy bridge subjected to ten heavy train passes were compared to those for a bridge having deficient connections under the same train loads (Fig. 15). For stiffness reductions of 20, 40 and 60%, minimal changes to resulting $\boldsymbol{\varphi}$ were observed. At 80% stiffness reduction, a significant change was observed for $\boldsymbol{\varphi}$ at the simulated damage locations. This indicates that critical levels of damage could be intuitively detected from $\boldsymbol{\varphi}$; however, small deficiencies could be overlooked.

4.4. ANNs for damage detection

To further investigate $\boldsymbol{\varphi}$ sensitivity to damage, particularly for damage intensities lower than 20%, ANN-based regression of damage indices to $\boldsymbol{\varphi}$ was completed. A set of 9 heavy trains were randomly selected from Group #4 (see Fig. 11) and were used to generate strain time-histories for 10% to 100% reductions in rotational stiffness in the vicinity of all individual DLs shown in Fig. 2. This approach produced 200 damage scenario POMs for a given train load, with $\boldsymbol{\varphi}$ for the healthy structure also included in the training dataset. To determine a suitable number of internal training neurons, various numbers were explored (25, 50, 100 and 200). Training with 100 internal neurons provided optimal results and were used throughout the study. A flowchart of the procedure is shown in Fig. 16.

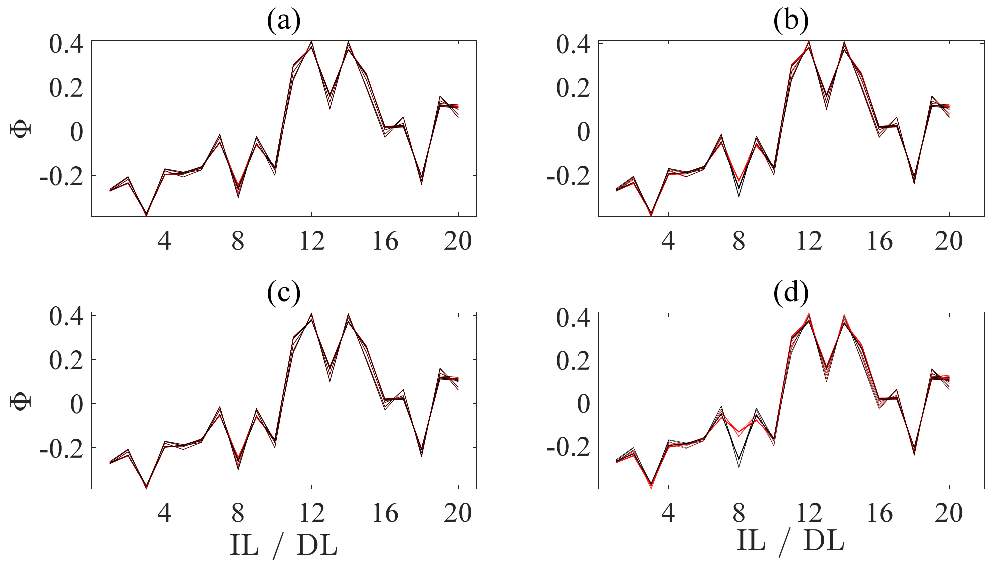
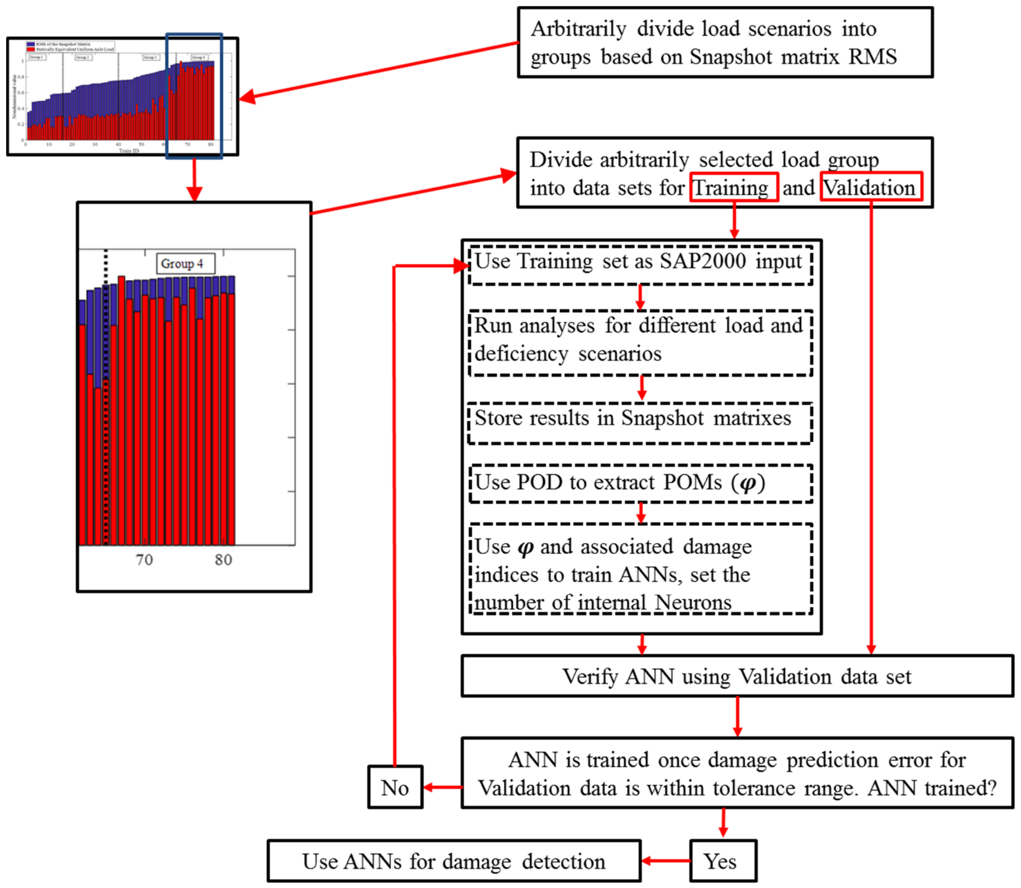


Fig. 15 $\boldsymbol{\varphi}$ of a healthy under Train 72 to Train 81 passages compared to (a) DI 20% at DL 8; (b) DI 40% at DL 8; (c) DI 60% at DL 8; and (d) DI 80% at DL 8. Red lines represent deficient cases; black lines healthy cases.



Arbitrarily divide load scenarios into groups based on Snapshot matrix RMS
Group 4
70
80
Divide arbitrarily selected load group into data sets for Training and Validation
Use Training set as SAP2000 input
Run analyses for different load and deficiency scenarios
Store results in Snapshot matrixes
Use POD to extract POMs (φ)
Use φ and associated damage indices to train ANNs, set the number of internal Neurons
Verify ANN using Validation data set
ANN is trained once damage prediction error for Validation data is within tolerance range. ANN trained?
No
Yes
Use ANNs for damage detection

Fig. 16 Process for selecting train loads and training and testing ANNs.

Remaining train loads in Group #4 were not used to develop the regression function and were adopted for further testing the accuracy of the proposed damage detection technique. Fig. 17 depicts results for this case, plotting nondimensionalized damage estimates for stiffness reductions of 10, 40 and 80% at DLs 3 and 15 in Fig. 2. The figure shows that, for the three damage intensities, negligible estimation error existed with error levels decreasing at higher damage levels.

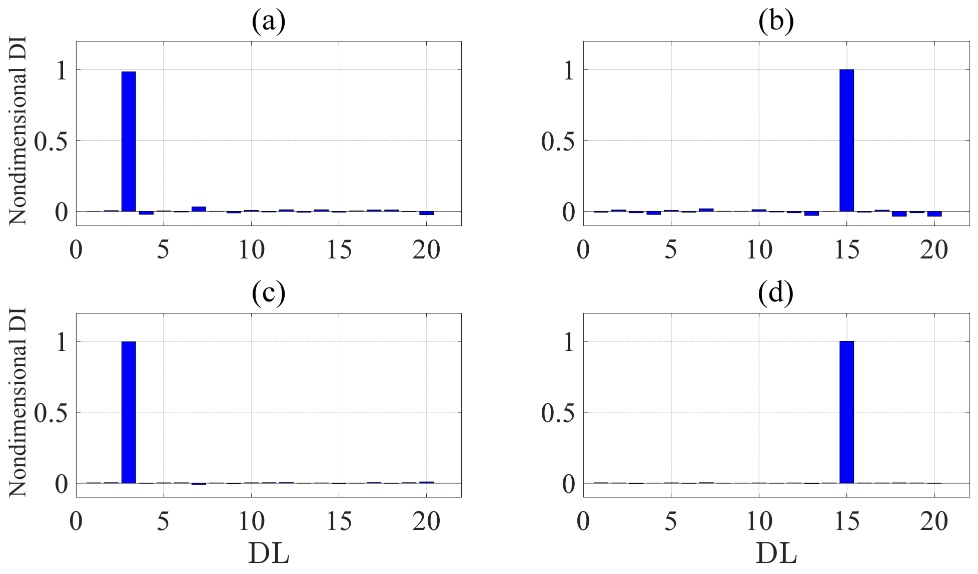


Fig. 17 ANN-regression Nondimensionalized DI: (a) DI 10% at DL 3; (b) DI 10% at DL 15; (c) DI 40% at DL 3; and (d) DI 80% at DL 15.

A more intuitive tool was developed to help end-users determine where damage was predicted on the bridge floor system. Examples are shown in Fig. 18 for DIs ranging from 50 to 100%. This “heat map” overlays identified damage index vectors onto a plane representing the floor system and, ultimately, DLs, and provided another way to confirm if ANN regression accurately identified damage for various DLs and DIs.

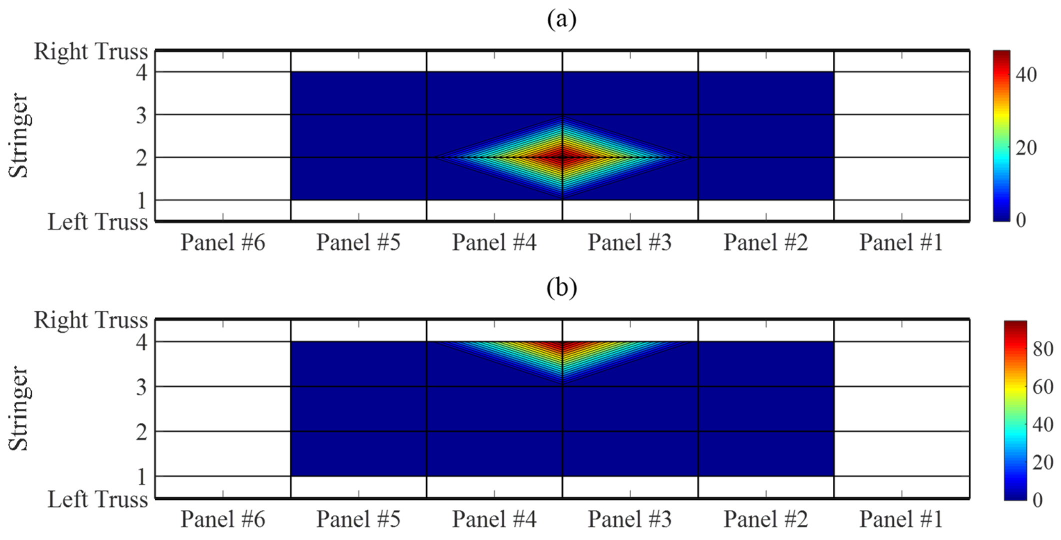


Fig. 18 Damage contour maps obtained by ANN-regression: (a) DI 50% at DL 8; and (b) DI 100% at DL 18.

Results from a study investigating the effect of a number various loading scenarios on the accuracy of ANN damage identification are shown in Fig. 19, Fig. 20 and Fig. 21. Three cases were considered. The first case, shown in Fig. 19, used six randomly selected train loads in Group #4 for training and the remaining loads for testing ANN. The second and the third cases, reported in Fig. 20 and Fig. 21, used 8 and 9 trains for training and remaining loads for testing. It was observed that when the number of training loading scenarios reduced, damage identification error increased. These observations confirmed the expectation that, given a sufficient number of train load scenarios ANNs could be generalized to address future unknown loadings.

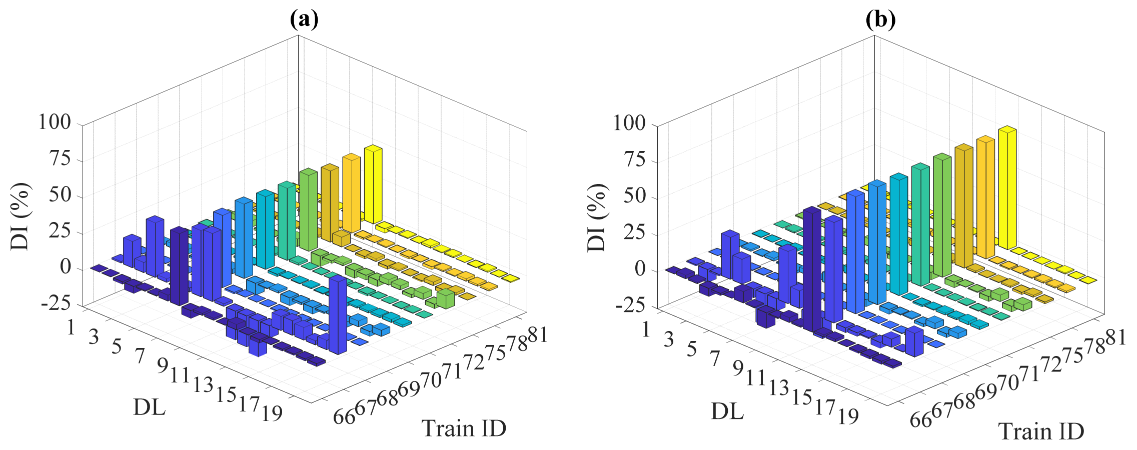


Fig. 19 Detected DI from ANN-regression for (a) DI 50% at DL 8; and (b) DI 80% at DL 13. 6 train loads for training, 10 for testing.

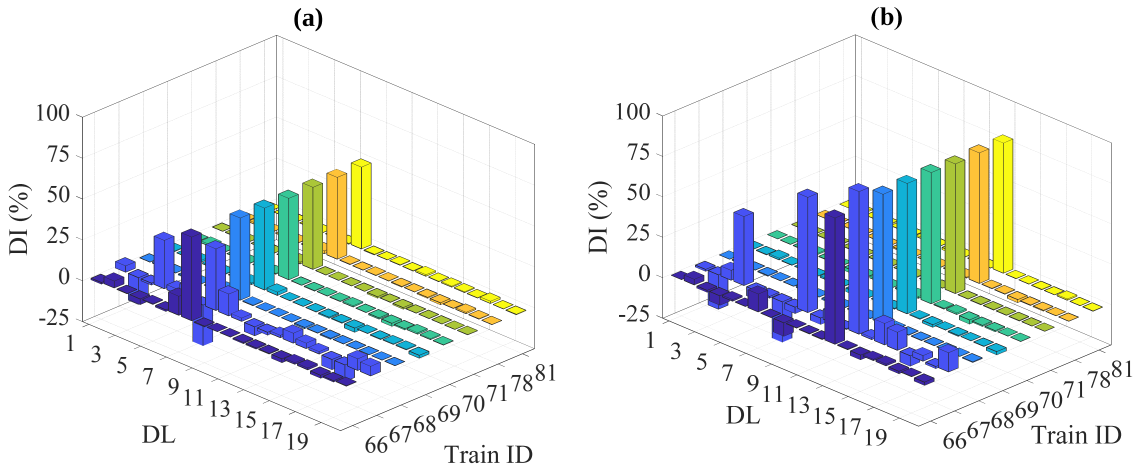
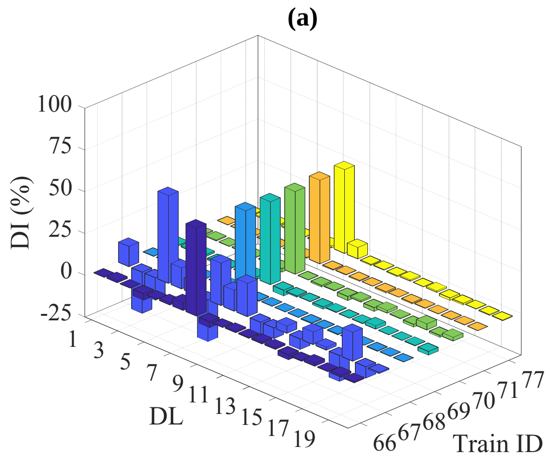


Fig. 20 Detected DI from ANN-regression for (a) DI 50% at DL 8; and (b) DI 80% at DL 13. 8 train loads for training, 8 for testing.



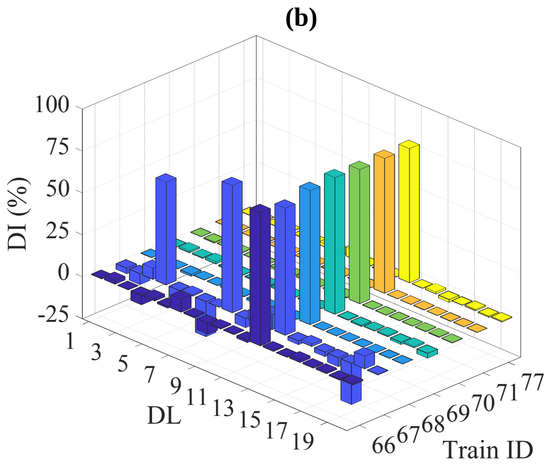


Fig. 21 Detected DI from ANN-regression for (a) DI 50% at DL 8; and (b) DI 80% at DL 13. 9 train loads for training, and 6 for testing.

Finally, to study the effect of categorizing loading scenarios on the damage detection performance of ANNs trained using Group #4 loads, $\boldsymbol{\varphi}$ of bridge responses to trains in Groups #1, #2, and #3, which not used for training, were used as ANN input. $\boldsymbol{\varphi}$ of the healthy structure was considered to assess the potential of ANNs to provide false positives. In Fig. 22 DIs for the healthy structure are shown when loading scenarios were selected from Groups #1 to #4. It was observed that, as the train axle loads become lighter and unlike the category of trains used for ANN training, DI prediction error increased.

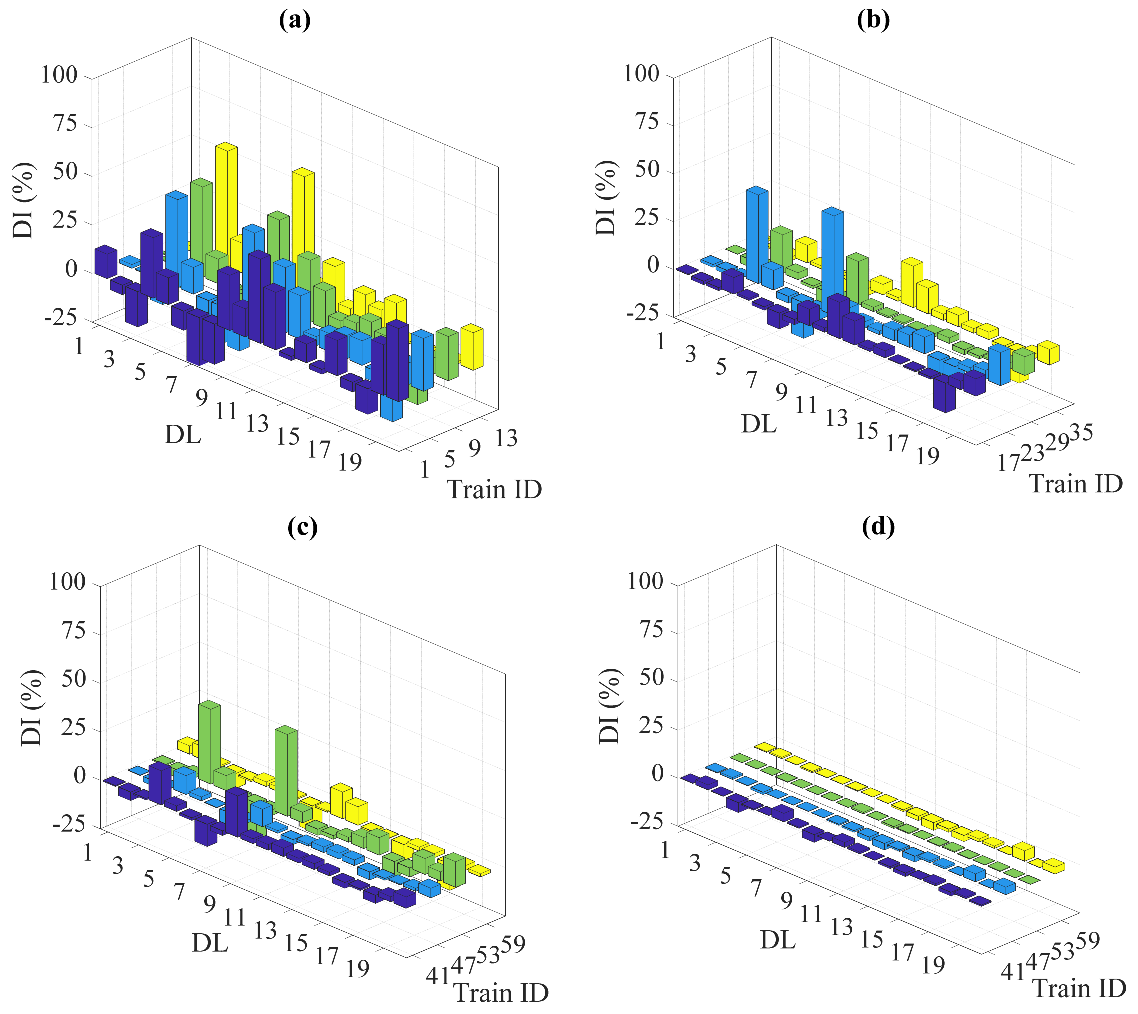


Fig. 22 ANN testing for DI of 0% with Group #4 trains were used for training and ANN testing trains from: (a) Group #1; (b) Group #2; (c) Group #3; (d) Group #4.

4.5. Conclusions

This chapter presents a framework for automated structural damage detection using Proper Orthogonal Decomposition and Artificial Neural Networks (POD-ANNs). Structural response POMs ($\boldsymbol{\varphi}$) were adopted as damage features since structures subjected to non-stationary excitations rendered $\boldsymbol{\varphi}$ dependent on more than damage. The proposed framework was applied to damage detection of a truss span in the considered, in-service, railway bridge, where inputs are highly non-stationary for each train passage loading scenario.

To complete the study, the 3D FE model from **Chapter 3**Chapter 3 was used to generate pseudo-experimental data (i.e. damage scenarios) and measured WIM data obtained was used to supply the input loads. Subsequently, ANN was used for training classification and to develop a regression function that categorized structural response and damage indexes (DIs) using $\boldsymbol{\varphi}$ developed from strain snapshot matrixes of each train passage.

It was concluded that:

- variations in train speed and train axle load can have marginal effects on $\boldsymbol{\varphi}$ but relatively small changes could affect damage detection accuracy when other sources of $\boldsymbol{\varphi}$ variability are present;
- automated peak picking mitigated discrepancies associated with speed variations and was the preferred method for selecting snapshot matrix data used to calculate $\boldsymbol{\varphi}$;
- categorizing the $\boldsymbol{\varphi}$ based on average RMS for all sensors could decrease variability due to train axle loads;
- numerical analyses verified that the regression network could accurately generalize damage detection under train load scenarios not used for training; and
- the method can effectively identify damage at relatively low intensities, (approaching a 10% reduction in connection stiffness).

Chapter 5 : POD-ANN Framework Application to Field Measured Truss Response

Published: The Journal of Civil Structural Health Monitoring, **https://doi.org/10.1007/s13349-018-0311-6**

Rageh, A., Linzell, D. G., & Azam, S. E.

5.1. Abstract

This chapter presents a framework for automated damage detection using a continuous stream of structural health monitoring data. The study utilized measured strains from an optimized sensor set deployed on the truss span of the investigated bridge described in Chapter 3. Stringer-to-floor beam connection deterioration was the focus of this chapter; however, the proposed methodology could be used to assess the condition of a wide range of structural elements and details. This chapter utilized the POD-ANN framework developed in Chapter 4 with POMs ($\boldsymbol{\varphi}$) used as a damage features and the Artificial Neural Networks (ANNs) used as an automated approach to infer Damage Location (DL) and Damage Intensity (DI) from $\boldsymbol{\varphi}$. $\boldsymbol{\varphi}$ variations, which are traditionally input (load) dependent, were ultimately utilized as damage indicators. Input variability necessitated implementing ANNs to help decouple POM changes due to load variations from those caused by deficiencies, which render the proposed framework input independent. To develop an automated and efficient output only damage detection framework, data cleansing and preparation was conducted prior to ANN training. Damage "scenarios" were artificially introduced into select output (strain) datasets recorded while monitoring train passes across the selected bridge. This information, in

turn, was used to train ANNs using the MATLAB Neural Net Toolbox. Trained ANNs were tested against monitored loading events and artificial damage scenarios.

The applicability of the proposed, output-only framework was investigated via studies of the bridge under operational conditions. To account for the effects of potential deficiencies at the stringer-to-floor beam connections, measured signal amplitudes were artificially decreased at select locations. It was concluded that the proposed framework could successfully detect artificial connection deficiencies imposed on measured signals under operational conditions.

5.2. Monitoring system

To explore the efficacy of the coupled POD-ANN methodology, the truss span of the investigated bridge detailed in Fig. 2 was instrumented using strain transducers. Measured responses were continuously transferred to a data acquisition system that was accessed remotely.

In general, steel bridges can be subjected to a wide variety of deficiencies caused by corrosion, fatigue cracks, scour and other items (34). For the bridge under study, main structural deficiencies included stringer-to-floor beam connection deterioration, deterioration of the stringer and bottom lateral connections and members, and frozen roller supports (35). While all of these deficiencies can be of concern to the bridge owner, as stated earlier, fatigue of stringer-to-floor beam connections is of primary concern for many riveted, steel, truss railway bridges as connection failure can lead to collapse, potential safety concerns for employees and citizens and expensive traffic disruptions

(34, 36, 37). Therefore, while other SHM configurations could be utilized on the selected bridge, the SHM system reported herein was designed to focus on potential deficiencies at the stringer-to-floor beam connections. It should be noted that the proposed method can be used to detect other deficiencies using differing sensor configurations (35).

A total of 24 strain transducers, manufactured by Bridge Diagnostics, Inc. (BDI), were deployed on the bridge to measure structural response under train passage and 20 of those instruments were installed at stringer ends, as shown in Fig. 8. Sensors were installed on the stringer bottom flanges. Selected instrument locations were based on recommendations by the bridge owner and preliminary FEM models.

The monitoring system consists of a BDI data logger, a wireless base station and wireless nodes, with each node being connected to 4 strain sensors using cables, as shown in Fig. 23. The system was powered using six 24-volt batteries that were charged by two solar panels, also shown in Fig. 23. Example sensor installations on stringer bottom flanges near floor beam connections are shown in Fig. 24. Data was collected remotely with the system set to activate and record strains at a sampling rate of 50 Hz when a train crossed the bridge.

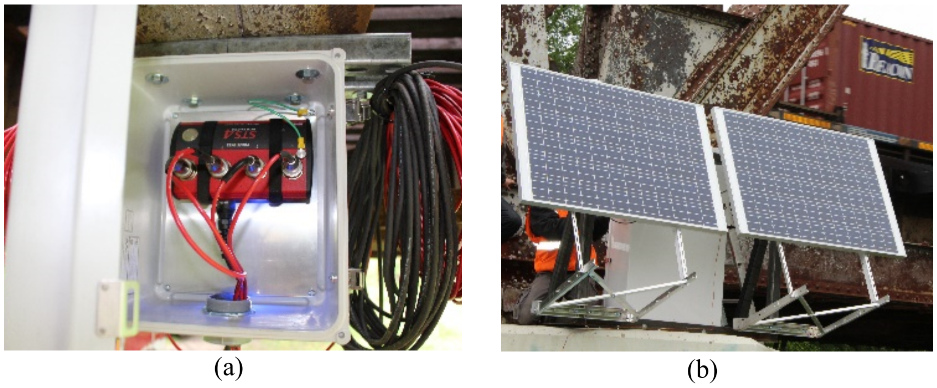


Fig. 23 Deployed monitoring system: (a) BDI node; (b) solar panels.

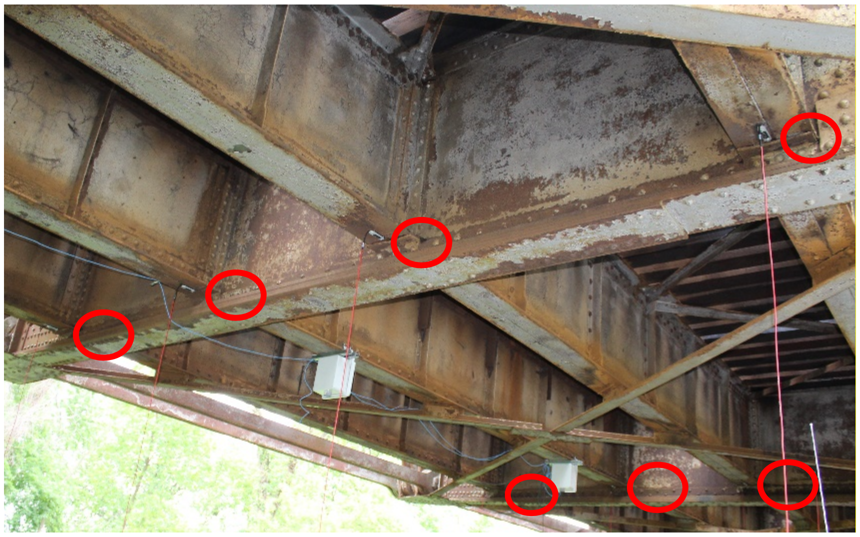


Fig. 24 Installed strain sensors (circled).

5.3. Feature extraction and data cleansing

The current study considered one week of train “events,” which totaled 363 recorded passages. Live loads associated with these datasets were unknown and varied with respect to speed, number of axles, and axle load magnitudes. The proposed methodology is described in the flow chart in Fig. 25.

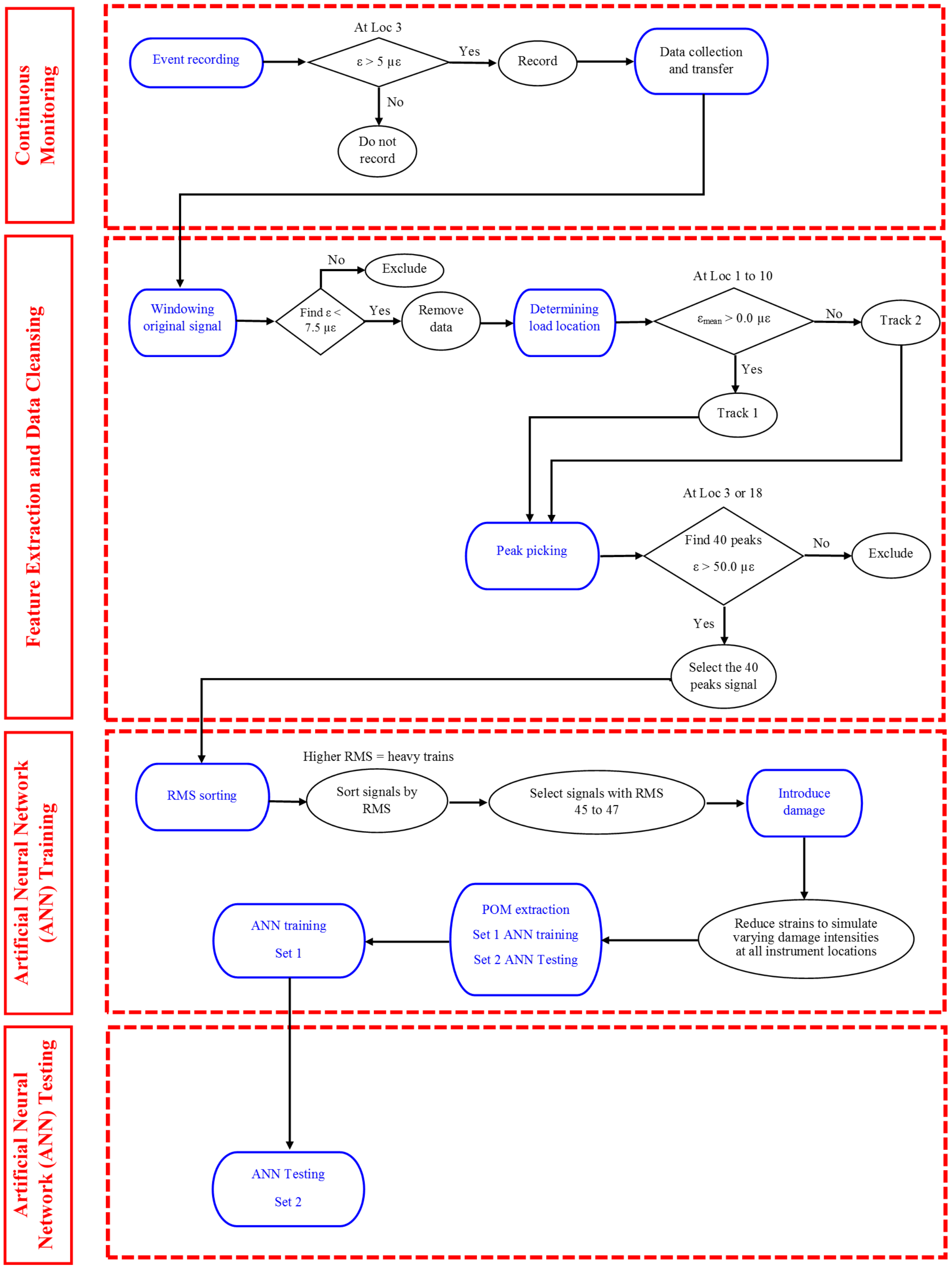

Continuous Monitoring
Event recording
At Loc 3
ε > 5 με
Yes
No
Record
Do not record
Data collection and transfer
Feature Extraction and Data Cleansing
Windowing original signal
Find ε < 7.5 με
No
Exclude
Yes
Remove data
Determining load location
At Loc 1 to 10
ε_mean > 0.0 με
No
Track 2
Yes
Track 1
Peak picking
At Loc 3 or 18
Find 40 peaks ε > 50.0 με
No
Exclude
Yes
Select the 40 peaks signal
Artificial Neural Network (ANN) Training
RMS sorting
Higher RMS = heavy trains
Sort signals by RMS
Select signals with RMS 45 to 47
Introduce damage
Reduce strains to simulate varying damage intensities at all instrument locations
POM extraction
Set 1 ANN training
Set 2 ANN Testing
ANN training
Set 1
Artificial Neural Network (ANN) Testing
ANN Testing
Set 2

Fig. 25 Feature extraction and data cleansing methodology.

$\boldsymbol{\varphi}$ are known to contain information on structural deficiencies and have been widely used for model reduction (91), impact localization (92), and damage detection in mechanical systems (20, 93). $\boldsymbol{\varphi}$ are used to graphically highlight data having the most variation for a given number of events. Therefore, for the current study, $\boldsymbol{\varphi}$ were dependent on recorded strain magnitude and duration, which are a function of train configuration and speed. To minimize $\boldsymbol{\varphi}$ variations, strain signals included in snapshot matrices used for $\boldsymbol{\varphi}$ extraction needed to be of similar magnitude and features with data reduction and cleansing being necessary prior to mode extraction. Therefore, data windowing, load location identification and peak picking were performed. MATLAB algorithms were implemented or developed to render this process autonomous and involved steps are described in the sections that follow.

5.3.1. Windowing recorded events

The first step was the elimination of time intervals having negligible live load strains, typically before and after train passage. The MATLAB *find* algorithm was applied to window the recorded strains at IL 3 in Fig. 8, which was selected since that sensor was positioned at midspan under a train rail and, as a result, would be quite sensitive to live load effects. Time steps involving strain changes greater than 7.5 $\mu\varepsilon$ were selected as the first filter through an offline trial and error process. Subsequently, all time steps having magnitudes less than 7.5 $\mu\varepsilon$ at the start and end of each event were eliminated. Two representative recorded signals at IL 3 and 18 are shown in Fig. 26 before and after windowing.

5.3.2. Determining load location

The second step focused on developing an automated classification of recorded signals based on the track upon which the train crossed the bridge. Initial field testing and model results indicated that stringer end bottom flange strains would be in compression if that stringer was underneath the loaded track and in tension if the other track was loaded (35). Means for recorded strains at ILs 1 to 10, underneath Track 2, were calculated and if values exceeded zero the train was classified as being located on Track 1, with the opposite sign indicating the train was located on Track 2. This classification showed that 187 of 363 trains traversed the bridge on Track 1. Windowed strain signals at ILs 3 and 18 for a train on Track 1 and Track 2 are shown in Fig. 27.

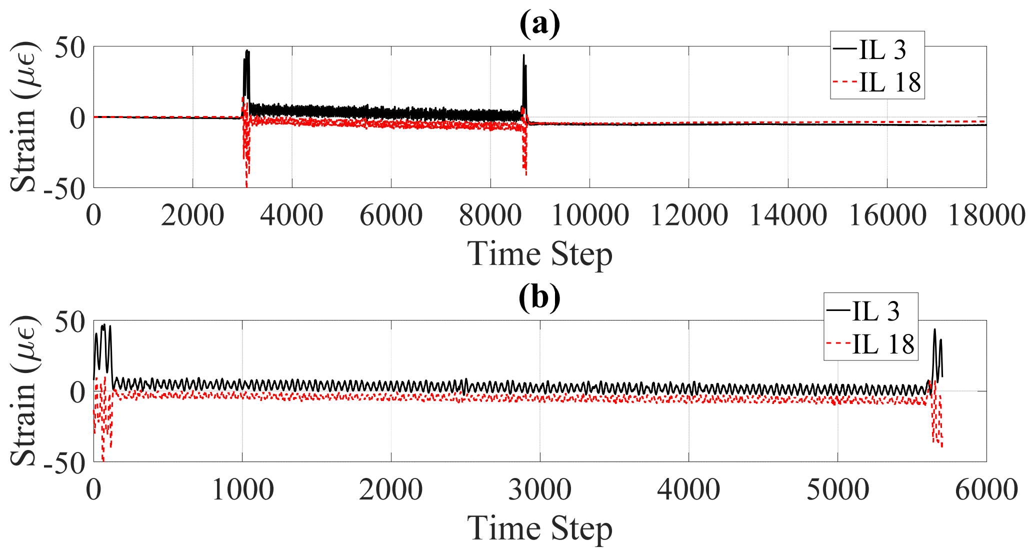


Fig. 26 Signal windowing: (a) Original; (b) Windowed.

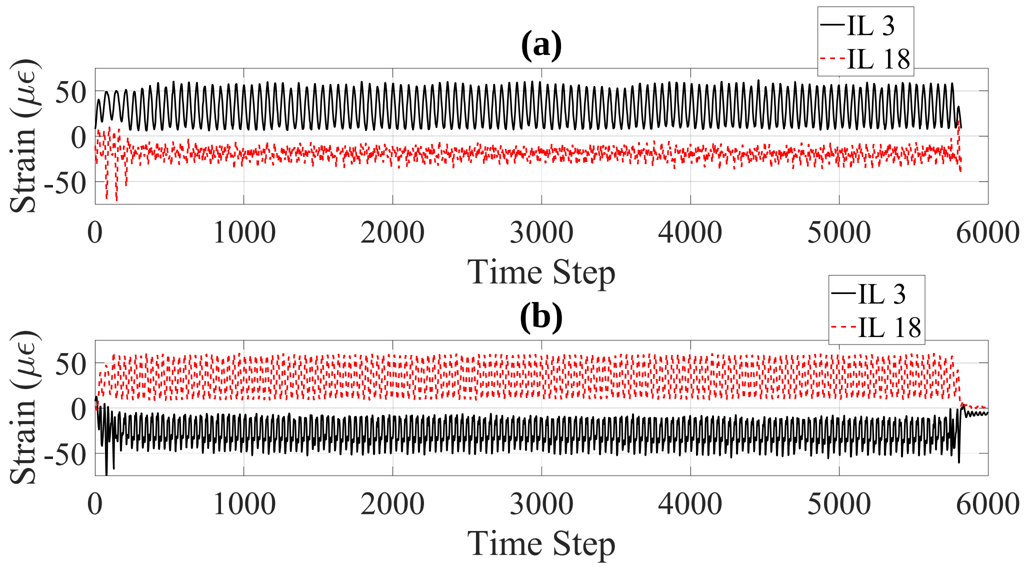


Fig. 27 Load location: (a) Track 1; (b) Track 2.

5.3.3. Automated peak picking

The third step focused on automatically selecting a constant number of peaks in each recorded event dataset so that $\boldsymbol{\varphi}$ variations due to train load disparities were minimized. A lower bound threshold of 50 με for recorded strain peaks was established based on average strains recorded at IL 3 when Track 1 was loaded or the corresponding location (IL 18) when Track 2 was loaded. The MATLAB *findpeaks* function was used to select the first 40 peaks having strains greater than 50 με at IL 3 or 18, with 40 peaks being selected to ensure that the snapshot matrices included enough samples for stable $\boldsymbol{\varphi}$ calculation.

The developed code excluded the first five peaks, corresponding to 4 train cars, to eliminate transient response developed from the locomotives. After automated peak peaking was performed, 74 events were filtered, 15 for trains on Track 1 and the remainder for trains on Track 2. Representative final strain events used to develop $\boldsymbol{\varphi}$ for trains on Tracks 1 and 2 at IL 3 and 18 are shown in Fig. 28 (a) is for a case where the train is located on Track 1, while Fig. 28 (b) is for the train located on Track 2.

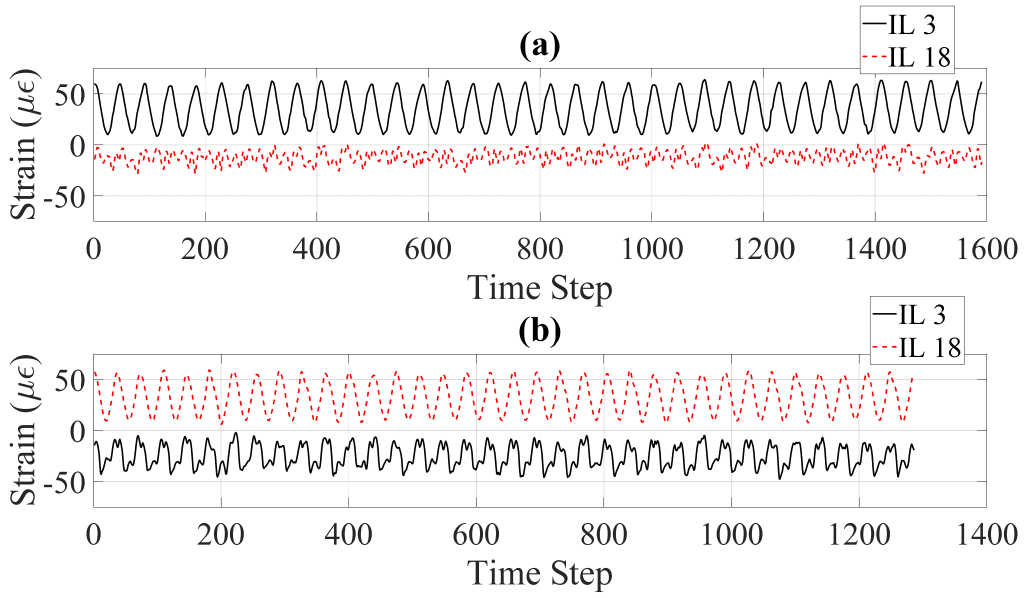


Fig. 28 Peak picking: (a) Track 1; (b) Track 2.

5.4. ANN training

To mitigate the influence of variable, non-stationary, external inputs on field measured strain $\boldsymbol{\varphi}$ variability, snapshot matrices for both tracks were sorted based on average RMS, with matrices having RMS averages between 45.4 and 47.1, a range of minimal variation, being selected for damage detection. In Fig. 29, average RMS snapshot matrices for Track 1 and 2 are shown. Track 2 matrices were selected for training and testing the developed damage detection ANNs, with matrices for trains 29 to 46 specifically being selected due to similar RMS averages. It is important to note that testing snapshot "events" were not included in the ANN training process. They were used to investigate how accurately the proposed method detected damage under various loading scenarios in the same average RMS range. Snapshots for trains 29, 35, 41 and 46 were randomly selected for ANN testing, while snapshots for trains 30 to 34, 36 to 40 and 42 to 45 were used for ANN training.

Since imposing actual damage on the studied bridge was not permissible, measured strains were reduced by multiplying a reduction factor to account for varying levels of damage at stringer-to-floor beam connections. This "damage" was simulated via the reduction of field-measured strains at selected locations, with reductions being proportional to assumed DIs. These reductions were based on the assumption that connection deterioration would reduce rotational stiffness, and in the limit would convert a semi-rigid ("healthy") connection to a pinned connection having limited to no moment restraint (37). Potential crack propagation through the connection depth was modeled via continuous decrease in connection rotational stiffness (37), resulting in smaller moments at the connection and, accordingly, smaller stringer end, bottom flange strains. In this study, induced damage at any connection was simulated using a strain reduction factor at that connection, with other connections being undamaged.

5.4.1. ANN nonlinear regression for damage detection

To generate training data, 10 DIs, varying between 10% and 90% in 10% increments, were examined at each IL (see Fig. 8). The DIs were sequentially varied at the 20 ILs for 14 train events, which produced 2800 damage scenarios. These damage scenarios trained the ANNs using MATLAB *Neural Net Fitting* function, where various numbers of internal neurons were explored to ensure ANNs were accurately generalized for damage identification. A nonlinear regression ANN was used to establish damage detection from $\boldsymbol{\varphi}$ of examined scenarios. It was decided that 70% of the input $\boldsymbol{\varphi}$ would be used for training, 15% for validation and 15% for testing during ANN regression analysis.

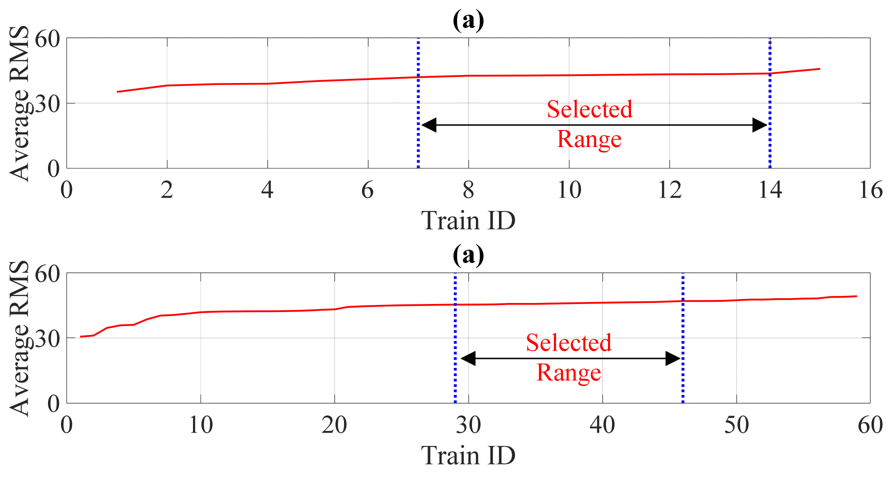


Fig. 29 RMS averages: (a) Track 1; and (b) Track 2.

5.5. ANN testing

As stated earlier, Trains 29, 35, 41 and 46 were randomly selected to test ANN effectiveness, with tested ANNs featuring 25, 50, 100 and 200 internal neurons, with the number of neurons being selected via trial and error to determine the appropriate number of internal neurons. A representative comparison between ANNs using 100 and 200 neurons for a DI of 90% at DL 8 when loaded by Train 29 is shown in Fig. 30. The figure indicated that both ANNs predicted damage location and intensity very well; however, the network with 200 neurons appeared to be marginally affected by overfitting, as evidenced by false positives and negatives shown in Fig. 30 (b). Using 100 neurons was shown to be the most robust for predicting damage location and intensity.

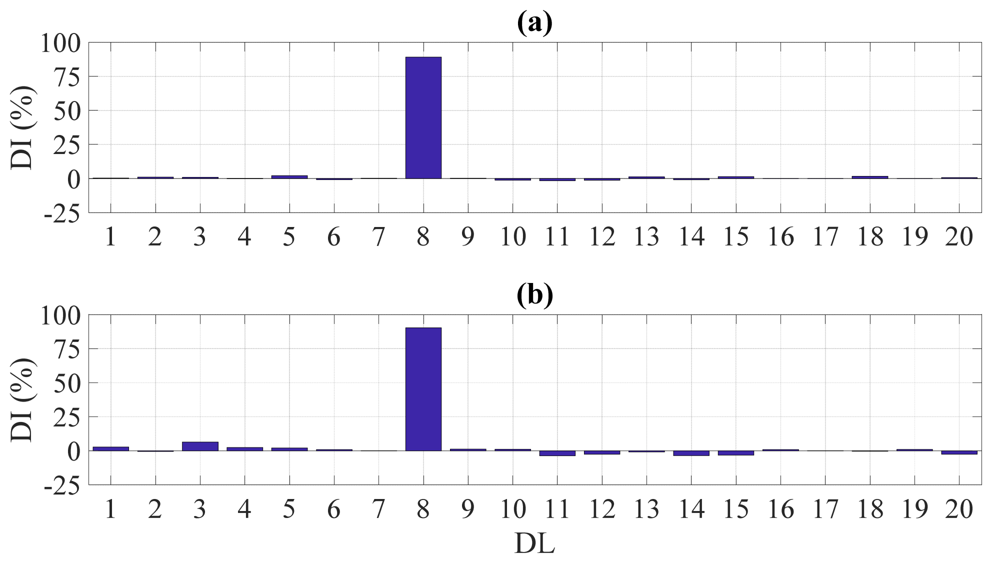


Fig. 30 ANN testing, Train 29: (a) 100 neurons; and (b) 200 neurons.

To ensure the developed method was robust against false positive damage signals, $\boldsymbol{\varphi}$ for a healthy bridge subjected to various loading events was also used to test trained ANNs. As shown in Fig. 31, it was observed that, for events associated with the four trains selected for ANN testing, the maximum false-positive DI was approximately 6%, which was deemed to be low when compared against the actual DI of 0%. These results supported the premise that the method would successfully detect damage with the caveat that an acceptable threshold should be established via long-term monitoring and corresponding statistical analyses.

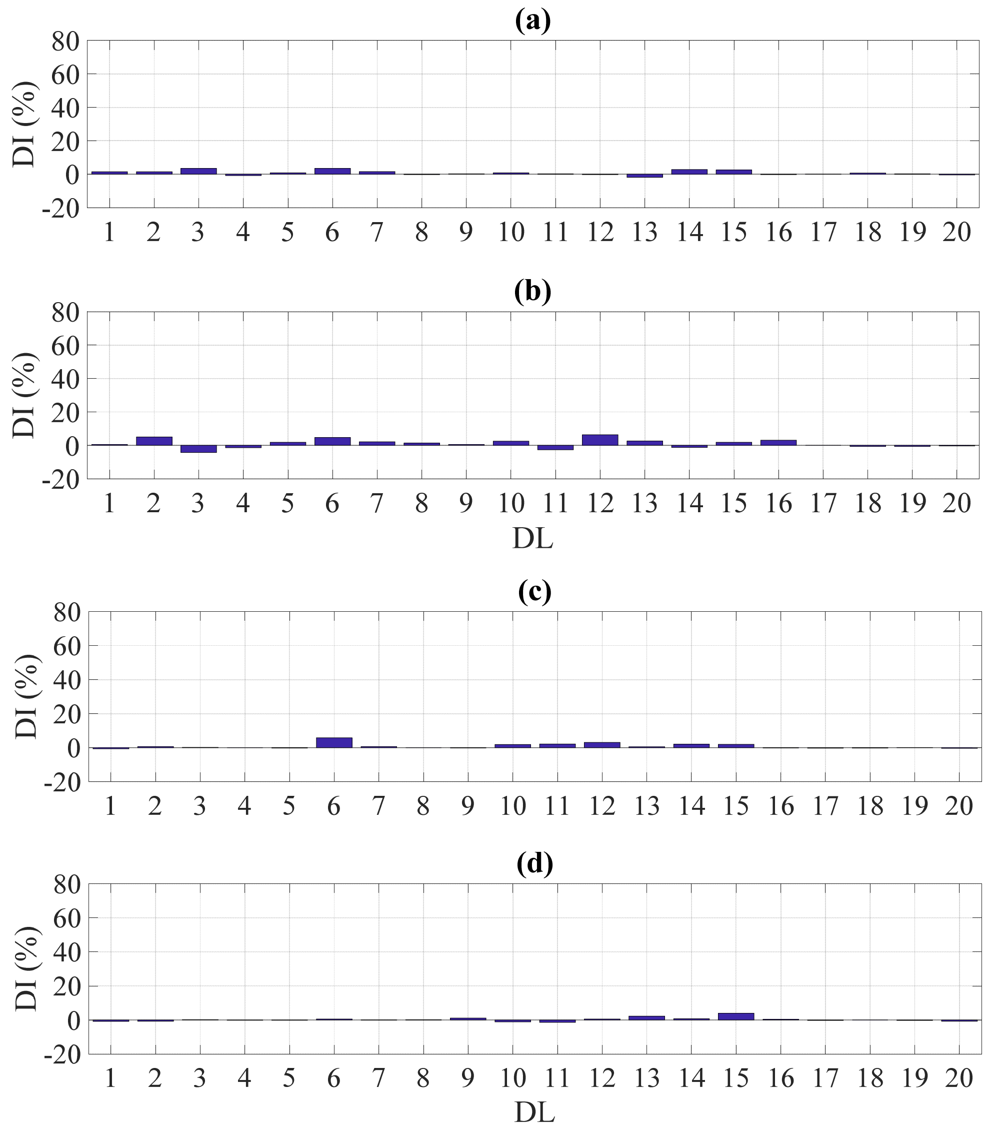


Fig. 31 Healthy bridge ANN testing: (a) Train 29; (b) Train 35; (c) Train 41; and (d) Train 46.

To further ascertain the ability of the proposed methodology to detect DL and DI, ANN DI predictions at various DLs were studied. For the sake of brevity, the results of the analysis for some representative damage scenarios are reported herein. The choice of DL and DI were arbitrary; however, the DIs ranged from low to high and the DLs were chosen so that it would cover various locations on the bridge. Fig. 32 demonstrates the ability of the proposed methodology for capturing the studied range of DIs at DL 8. Predicted DIs for Train 29 were 17, 37, 58, 79 and 89% for imposed DIs of 20, 40, 60, 80 and 90%, respectively. Results of ANN testing at DL 11 are shown in Fig. 33 and indicated that DI predictions might be affected by recorded signals from certain trains, especially for low DIs. A DI of 20% was captured well for all testing sets except for Train 35, where false-positive DIs, approaching 5%, existed.

Conversely, Fig. 34 presents damage identification capabilities of trained ANNs for a DI of 60% at DL 13 under multiple train loads. Trains 29, 35, 41 and 46 were used with DL and DI were, again, captured accurately with predicted DIs ranging between 57 and 60%.

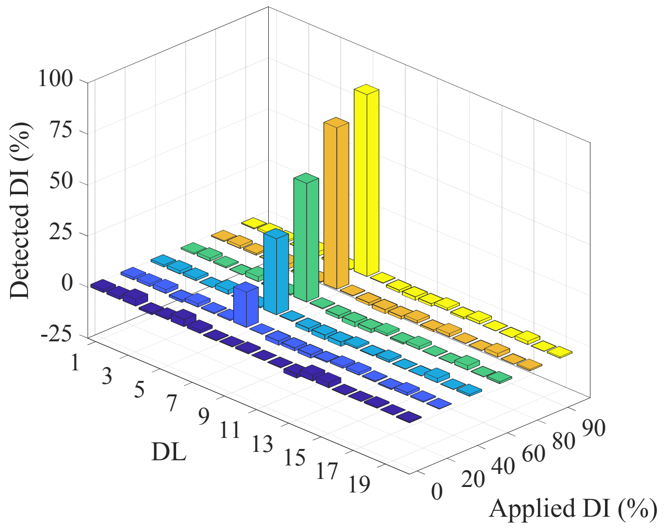


Fig. 32 ANN testing for DL 8 under Train 29, all DIs.

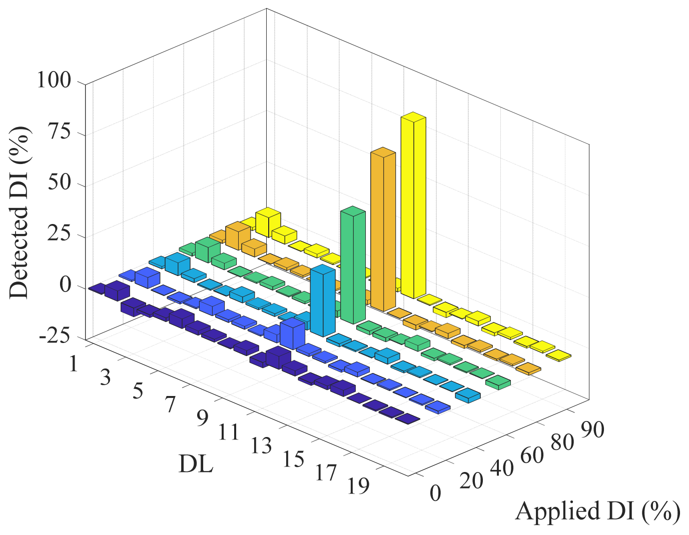


Fig. 33 ANN testing for DL 11 under Train 35, all DIs.

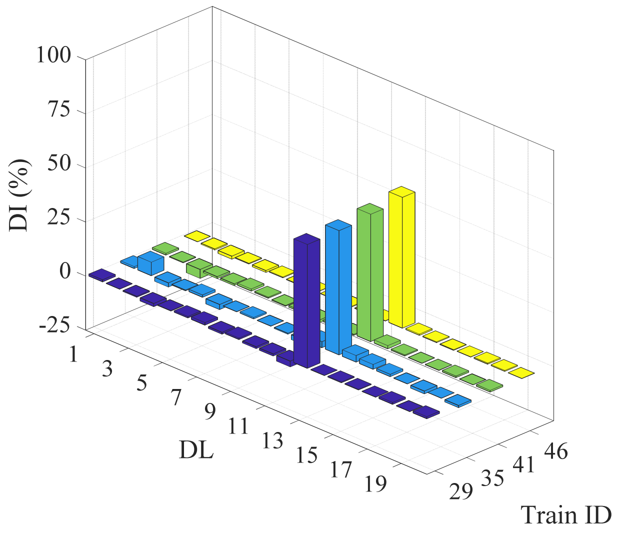


Fig. 34 ANN testing for DL 13 with DI 60% under Trains 29, 35, 41 and 46.

To estimate the importance of classifying trains based on RMS, as shown in Fig. 29, the trained ANN was tested using healthy $\boldsymbol{\varphi}$ for 4 trains whose average RMS was located outside the selected range. Trains 10, 20, 50 and 55 were selected for this exercise. A moderate increase in false-positives was evident for three of the four selected trains (Trains 10, 50 and 55), with false positives predicting DI ranges between 8 and 25% as shown in Fig. 35. For Train 20, the predicted DI showed a significant increase in false-positives, with maximum value of 70% as shown in Fig. 35. These results showed that selecting ANN training sets based on the associated average RMS is necessary to reduce $\boldsymbol{\varphi}$ variations associated with change in non-stationary loading configurations.

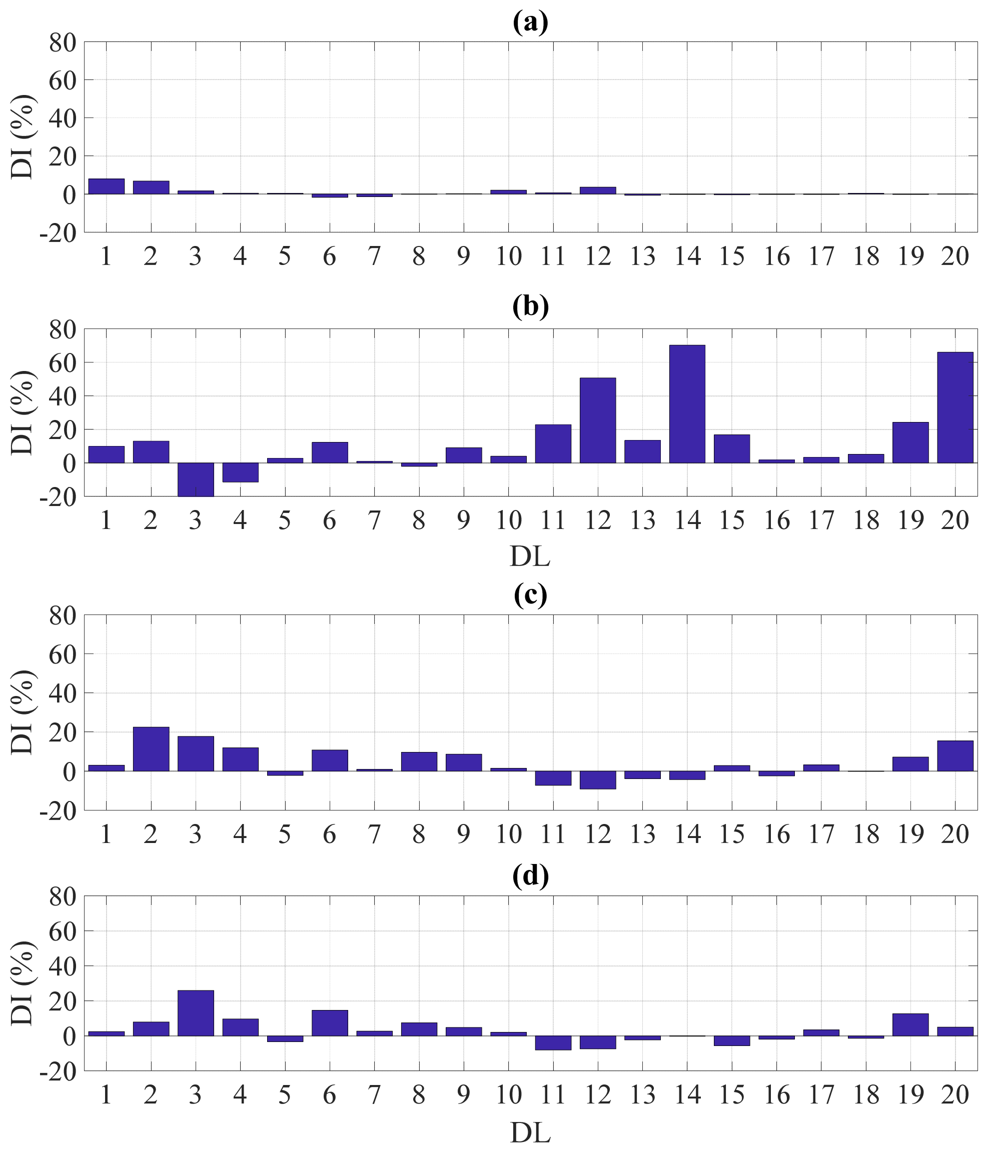


Fig. 35 Healthy bridge ANN testing: (a) Train 10; (b) Train 20; (c) Train 50; and (d) Train 55.

To examine the effectiveness of the proposed methodology for detecting damage using noisy strain signals, as is often the case with low-cost sensors, zero-mean, white Gaussian noise was added to measured strain time-histories. A representative example of noisy strain signals for Train 41 is shown in Fig. 36. ANNs were retrained and retested using the “noisy” data and results showed that the proposed methodology was capable of capturing damage from noisy signals with acceptable accuracy. “Noisy” strains at DL 15 with a DI 80% under Trains 29, 35, 41 and 46 are shown in Fig. 37. Predicted DIs ranged from 78 to 81%, which showed a good match with simulated damage.

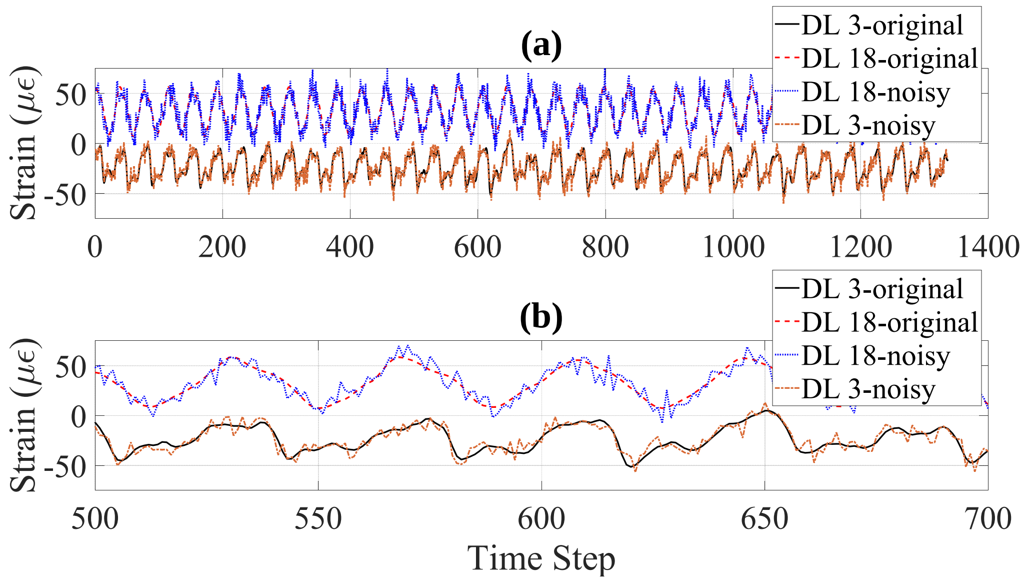


Fig. 36 Signal with 20% simulated noise, Locations 3 and 18, Train 41; (a) peak picking; and (b) magnified view.

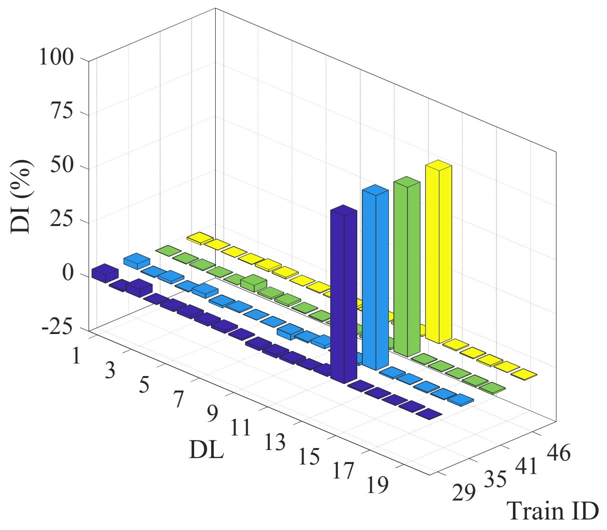


Fig. 37 ANN testing at DL 15 with simulated DI 80% and simulated noise 20% under Trains 29, 35, 41 and 46.

5.6. Conclusions

In this chapter, the proposed POD-ANN damage detection framework discussed in Chapter 4 was applied to measured strain responses from the monitoring system deployed on the truss span shown in Chapter 3. Results demonstrated its efficacy for detecting deficiencies in stringer-to-floor beam connections and the approach can be extended to include damage detection for other structural systems, details and types of data. The following conclusions were drawn from the study:

- The proposed method successfully detects damage using strain outputs induced by unknown, nonstationary external inputs;
- Automated data cleansing prior to POM ($\boldsymbol{\varphi}$) extraction was necessary to reduce discrepancies caused by nonstationary inputs;

- The developed approach could accurately capture damage represented by DIs greater than 20%, with clearly improved accuracy for DIs higher than 40%; and
- The method is robust enough to predict damage supplied from highly noisy signals accurately.

It should be reiterated that the current study was performed neglecting environmental variability; however, due to the short amount of time required for a train to traverse the bridge it was assumed that environmental effects were negligible. It is noteworthy that existing filtering methods in the literature, even in the absence of modeling uncertainties and environmental effects, lead to large estimation errors when measurement noise is large and a relatively large number of DIs have to be identified. For example, hybrid particle filters systematically account for modeling and measurement errors but are prone to bias as the noise-to-signal ratio increases (62).

Chapter 6 : POD-ANN Framework Application to Varying Spans, Responses and Instrumentations

6.1. Abstract

An extended application of the POD-ANNs framework discussed in Chapter 4 and Chapter 5 (61, 62) is presented herein. The study presented in this chapter utilized the finite element models of the truss and through girder spans discussed in Chapter 3. The current study expanded the framework to include strains, rotations about member major axes, vertical displacements and accelerations for three different proposed instrumentation plans per span (35) to investigate applicability of the framework to other instrumentation schemes, response measurements and span types. Connection damage was again simulated by reducing rotational spring stiffness at member ends and various responses were extracted for each damage scenario. Instrumentation plan 1 (IP-1) was located close to stringer ends, while plans 2 and 3 (IP-2 and IP-3) had instruments located at midspan of the stringers (35).

As shown in the previous chapters, the methodology utilizes POMs ($\boldsymbol{\varphi}$) as the damage feature identification method and ANNs to automatically determine damage location (DL) and intensity (DI) and decouples $\boldsymbol{\varphi}$ variations from variations in train loads. To train and test developed ANNs, 10 DIs were investigated at stringer connections to simulate stringer-to-floor beam deterioration under selected train passages. ANN testing results showed that DLs and DIs were detected accurately for both spans using different instrumentation plans that measure various response quantities.

6.2. Analyses and instrumentation

As discussed in Chapter 3, a truss and through plate girder span from the investigated bridge were selected as testbeds. For the description and sketches of both spans see Section 3.1. The current analytical study focused on applying the POD-ANN framework with strains, member strong-axis rotations, vertical displacements or accelerations measured at locations that were part of 3 analytically modeled "instrumentation" plans (IP). Listed responses were selected because they best reflect damage at stringer-to-floor beam connections, which were reported by the bridge owner to be prevalent locations for damage. Instrumentation and analysis results are summarized in the following sections.

6.2.1. Analytical "instrumentation" plans and simulated damage

Commonly reported deficiencies for these types of railway bridges include (35):

(i) Stringer-to-floor beam connection deterioration and damage.

(ii) Stringers and bottom laterals connection deterioration and member failure.

(iii) Frozen supports.

As stated earlier, stringer-to-floor beam connection deterioration was one of the more prevalent and important deficiencies to identify and track and, as a result, "instruments" in the models were placed to detect changes of various responses due to damage at or near those locations.

Three instrumentation schemes were examined analytically and are shown in Fig. 38 for the truss span and Fig. 39 for the through plate girder span. For the truss span, IP-1 in Fig. 38 (a) had 20 sensors at stringer ends, IP-2 in Fig. 38 (b) had 20 sensors at midspan of the stringers and IP-3 in Fig. 38 (c) had 12 sensors at midspan of the stringers. For the through plate girder span, IP-1 in Fig. 39 (a) had 24 sensors at stringer ends, IP-2 in Fig. 39 (b) had 24 sensors at midspan and IP-3 in Fig. 39 (c) had 12 sensors at midspan. Selected instrument locations were based on sensitivity analyses and are reported elsewhere (35).

For each of the examined IP, the damage was simulated at 20 stringer ends in the truss span and 24 in the through girder span. Damage was simulated on one side of the floor-beam because it was assumed that damage to the connection on either side would have the same influence on stringer continuity. Differences between the number of simulated damage locations were due to the number of panels and stringers in each span (i.e., the truss span has 24 stringers while the through plate girder has 28 stringers). As discussed in Chapter 3, connection deterioration would reduce rotational stiffness and, as a result, a "rigid" connection would revert to semi-rigid connection and, ultimately, a pinned connection (37). Continuous reduction in rotational stiffness was used to simulate crack propagation through connecting clip angles. Simulated Damage Locations (DL) are shown in Fig. 40.

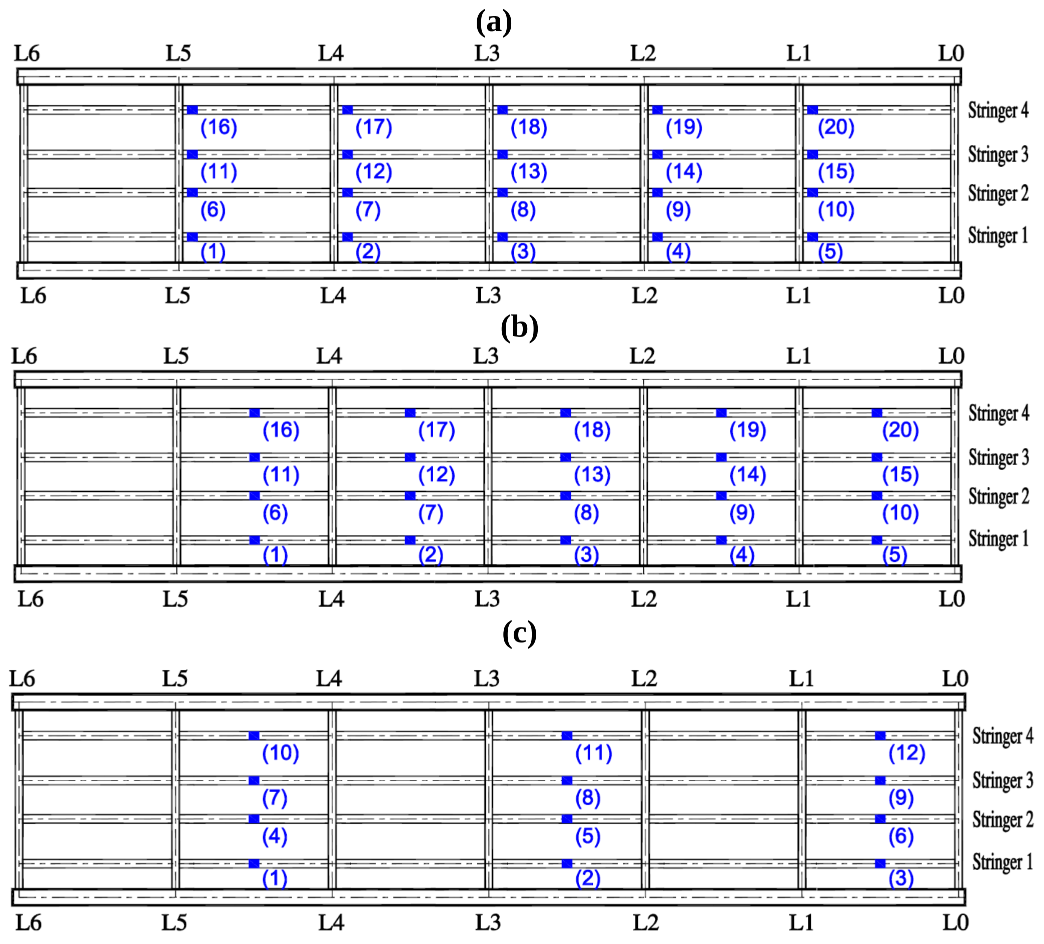


Fig. 38 Truss span instrumentation plans: (a) Instrumentation Plan 1 (IP-1); (b) Instrumentation Plan 2 (IP-2); and (c) Instrumentation Plan 3 (IP-3).

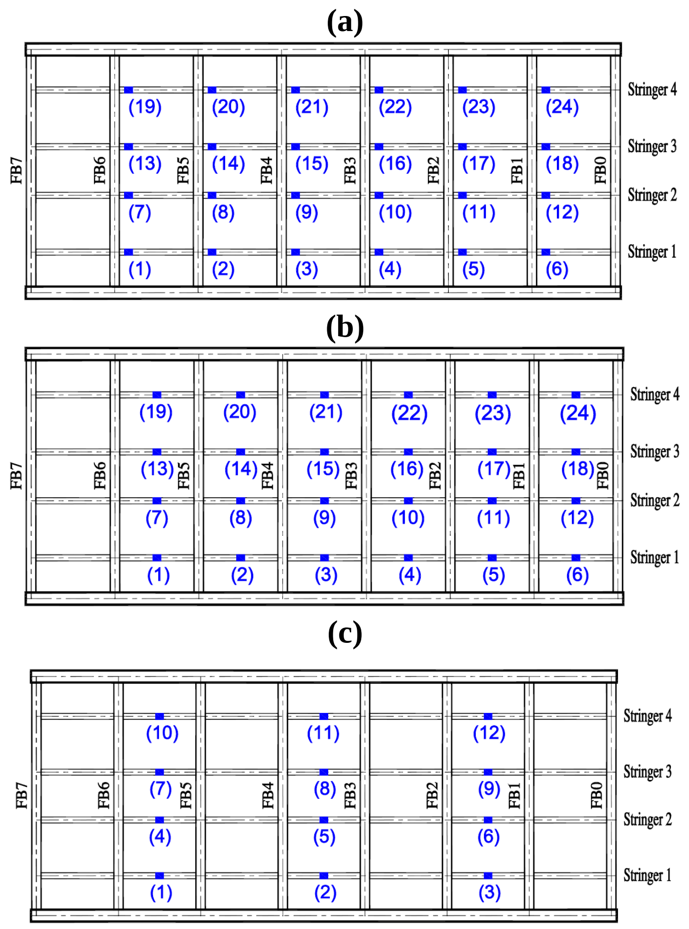


Fig. 39 Through girder span instrumentation plans: (a) Instrumentation Plan 1 (IP-1); (b) Instrumentation Plan 2 (IP-2); and (c) Instrumentation Plan 3 (IP-3).

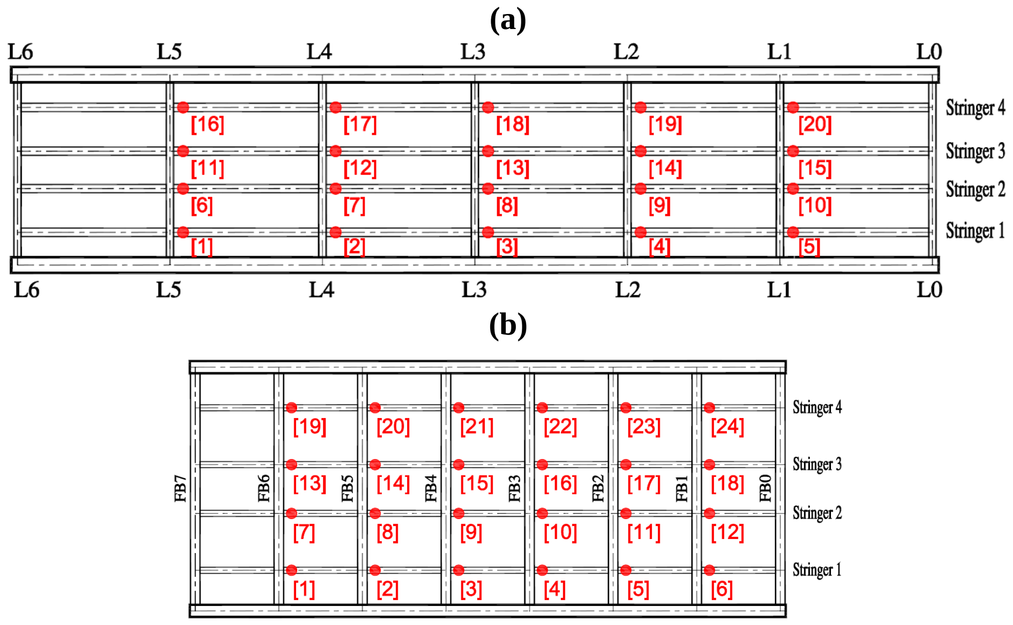


Fig. 40 Simulated damage locations (DL): (a) truss span; and (b) through girder span.

As mentioned in Chapter 3, the bridge owner provided Weigh-In-Motion (WIM) data for 81 trains of varying loadings, axle spacing, lengths and travel speeds. Preliminary analyses, which assumed that the bridges were undamaged, subjected to the 81 trains were completed using MATLAB and the SAP2000 Open Application Programming Interface (OAPI) to extract structural responses at IP-1 locations in Fig. 38 (a). Of the 81 trains, 24 were selected having the highest strain RMS (62) were selected for the current study, see Fig. 11.

6.2.2. Analyses

Three-dimensional frame finite-element models were developed in SAP2000 for the bridge spans under study. Developed analytical models are described and shown in Section 3.2. Isometric views of developed models are shown in Fig. 4. More details about constructed models are provided in (35). Train loads were applied via a set of point moving loads.

Multi-step static and time history analyses were used to obtain static and dynamic response for each train passage and damage scenario. Strains, strong-axis rotations and vertical displacements were extracted from the multi-step static analyses while vertical accelerations were extracted from the time-history analyses. The maximum number of modes defined in the modal analyses were 50 and 25 for the truss and the plate girder spans, respectively, and were selected based on comparing strain and displacement results from static analyses.

MATLAB and SAP2000 OAPI Open Application Programming Interface (OAPI) were used to automate each type of analysis. Automation was implemented to:

(i) Model selected train loads;

(ii) Simulate damage locations and intensities for each train passage, of which there were 201 scenarios for the truss and 241 for the through plate girder; and

(iii) Extract internal effects that included strains, strong-axis rotations, vertical displacements and accelerations.

6.3. ANN training using POMs

ANN training data was generated for 10 DIs between 10% and 100% in 10% increments at each DL, with each increment representing progressive damage at a stringer-to-floor beam connection. A total of 4800 (truss span) and 5760 (through girder span) damage scenarios were analytically studied using this approach for the 24 selected train passages. These damage scenarios helped train ANNs using the MATLAB *Neural Net*

Fitting function, where various numbers of internal neurons were explored to ensure ANNs were accurately generalized for damage identification. ANNs could detect DL and DI once trained with the available data. A nonlinear regression ANN was selected to establish damage detection with $\boldsymbol{\varphi}$ for each DI/DL scenario. As mentioned earlier, 100 internal training neurons were adopted for ANN training. It was decided that 70% of the input $\boldsymbol{\varphi}$ would be used for training, 15% for ANN validation and 15% for testing with 18 trains strain $\boldsymbol{\varphi}$ being used to train the ANNs. The number of trains selected for ANN training was based on a trial and error approach examining 6, 12 or 18 trains.

Representative comparisons under 6 train training passages and 6, 12 or 18 testing passages are shown in Fig. 41 for the truss span with a DI 50% at DL 8. ANNs were trained and tested with $\boldsymbol{\varphi}$ from stringer end bending rotations. Fig. 41 (a) shows testing of ANNs trained using six train passages, Fig. 41 (b) shows testing of ANNs trained using 12 passages and Fig. 41 (b) shows the testing of ANNs trained using 18 passages. As expected, the figure indicates that increasing the number of events in the training process increased DI/DL prediction accuracy and reduced false positives and negatives. More detail on selected trains can be found in (62).

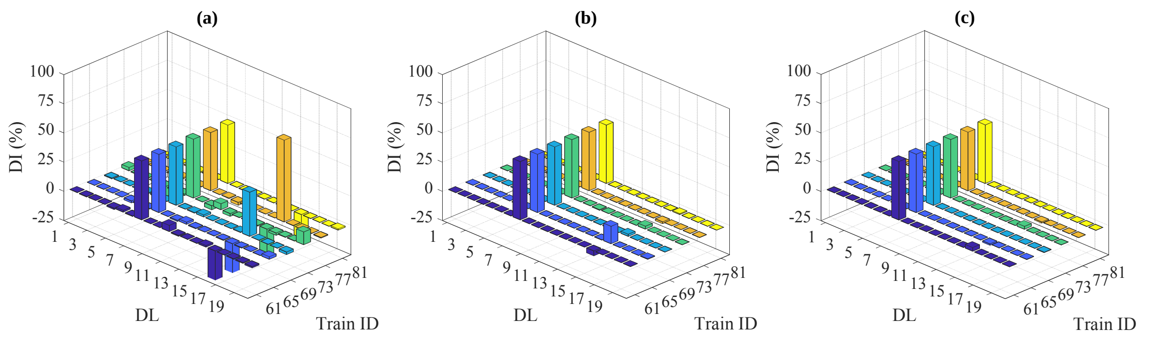


Fig. 41 Influence of number of train events used in ANN training process, DI=50% at DL 3, rotation ***φ***, truss span: (a) Training with six trains; (b) Training with 12 trains; and (c) Training with 18 trains.

6.4. ANN testing, truss span

As stated earlier, 75% of DI/DL scenarios were used for ANN training, with training and testing proportions being selected based on trial and error. This provided 3600 damage scenarios for training and 1200 for testing using extracted strain, rotation, displacement and acceleration ***φ***. Representative results that detail the effectiveness with which each response ***φ*** detects damage are presented for IP-1 in the following section. More results are shown in Appendix 1**.**

6.4.1. IP-1

Representative results for a single train (Train 69) at DL 3 for a DI of 40% are shown in Fig. 42. The figure details detected DI/DL for ANNs tested and trained with ***φ*** extracted from various structural responses. As shown in the figures, DI/DL were detected with very good accuracy with ANNs trained with various response ***φ*** except for acceleration, which showed very low DIs at all locations. Fifteen acceleration DLs indicated false positives and 4 DLs indicated false negatives with a maximum predicted DI = 7.5%, Fig. 42 (d).

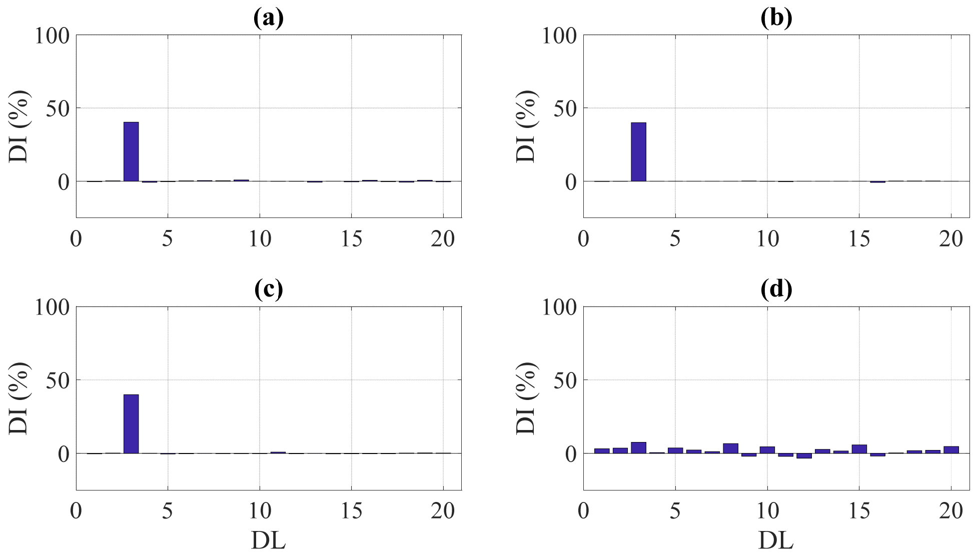


Fig. 42 Truss span ANNs testing, IP 1, Train 69, DI=40% at DL 3: (a) strain; (b) rotation; (c) displacement; and (d) acceleration.

To investigate the efficiency and effectiveness of using POD-ANN to detect damage under different train loads, results from six train events at DL 13 with a DI of 80% are shown in Fig. 43. The results indicated that strain $\boldsymbol{\varphi}$ accurately predicted DI and DL for all trains. Predicted DI ranged from 79.2 % to 81.0 % with maximum false positive and negative being 1.0 and 1.2 %, respectively as shown in Fig. 43 (a). Fig. 43 (b) shows rotation $\boldsymbol{\varphi}$s also demonstrating reasonable accuracy. Displacements did not demonstrate similar accuracy levels, but were still reasonable, as shown in Fig. 43 (c). Fig. 43 (d) shows acceleration $\boldsymbol{\varphi}$s were not reliable for determining DLs or DIs for any of the considered trains. As a result of studies that were completed, $\boldsymbol{\varphi}$s extracted from strain or rotation measurements were recommended for the framework using IP-1.

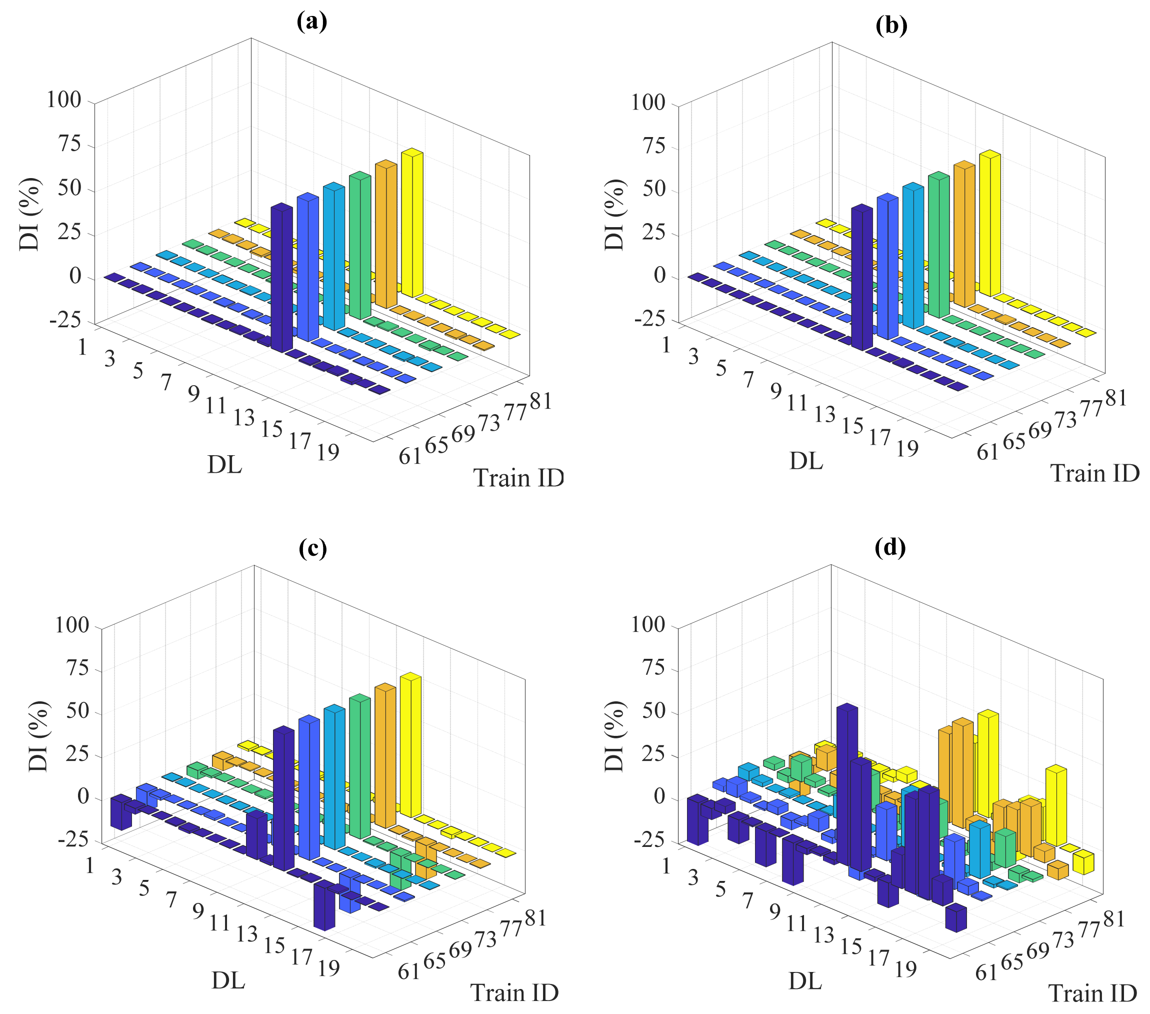


Fig. 43 Truss span ANNs testing, IP 1, all testing trains, DI=80% at DL 13: (a) strain; (b) rotation; (c) displacement; and (d) acceleration.

6.5. ANN testing, through girder span

As stated earlier, 75% of DI/DL scenarios were used for ANN training, with training and testing distributions selected based on trial and error. This provided 4320 damage scenarios for training and 1440 for testing using extracted strain, rotation, displacement and acceleration ***φ***s. Representative results that detail the effectiveness with which each response ***φ*** detected damage are presented in the following section for IP-3. More results for ANNs testing are shown in Appendix 1**.**

6.5.1. IP-3

Representative results for a single train (Train 69) at DL 3 with a DI of 40% are shown in Fig. 44. The figure details detected DI/DL with $\boldsymbol{\varphi}$s extracted from structural responses discussed previously. As shown in the figures, DI/DL were detected with very good accuracy except for acceleration, which again showed very low DIs at all locations (Fig. 44 (d)). Predicted damage for strain, rotation and displacement $\boldsymbol{\varphi}$ at DL 3 varied between 37 and 40% with maximum false positives and negatives of 9 and 4.0%, respectively, Fig. 44 (a-c).

To investigate the efficiency and effectiveness of using POD-ANN to detect damage under different train loads, results from 6 train events at DL 15 with a DI of 80% are shown in Fig. 45. The results indicate that using strain, rotation and displacement $\boldsymbol{\varphi}$s predicted DI and DL accurately for all trains. Predicted DI at DL 15 for strain, rotation and displacement $\boldsymbol{\varphi}$s ranged between 77 and 83% with false positives and negatives being less than 5%, Fig. 45 (a-c). ANN testing for acceleration $\boldsymbol{\varphi}$ detected neither DL nor DI accurately, Fig. 45 (d).

As a result of the studies that were completed, POMs extracted from strain, rotation and displacement were recommended for through girder span IP-3.

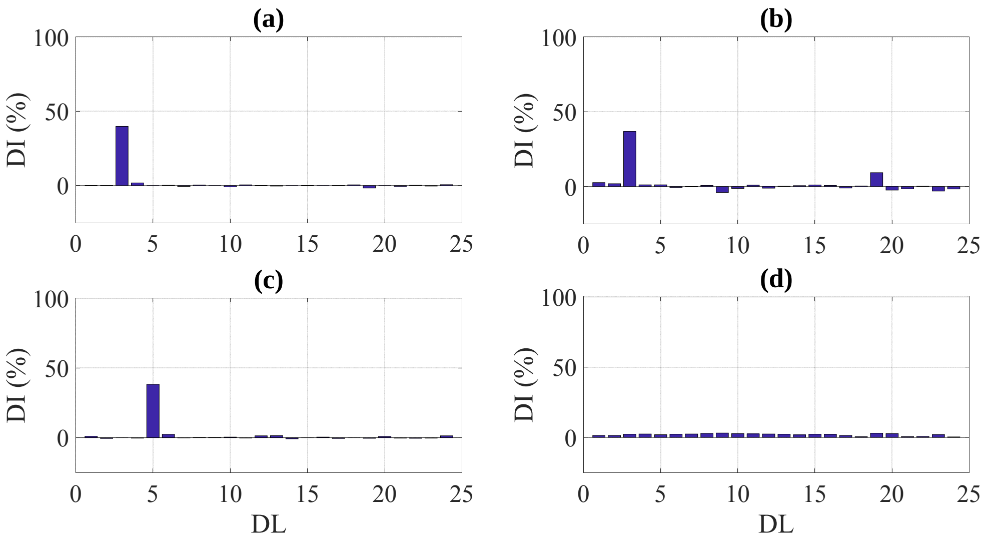


Fig. 44 Through girder span ANNs testing, IP 3, testing Train 69, DI=40% at DL 3: (a) strain; (b) rotation; (c) displacement; and (d) acceleration.

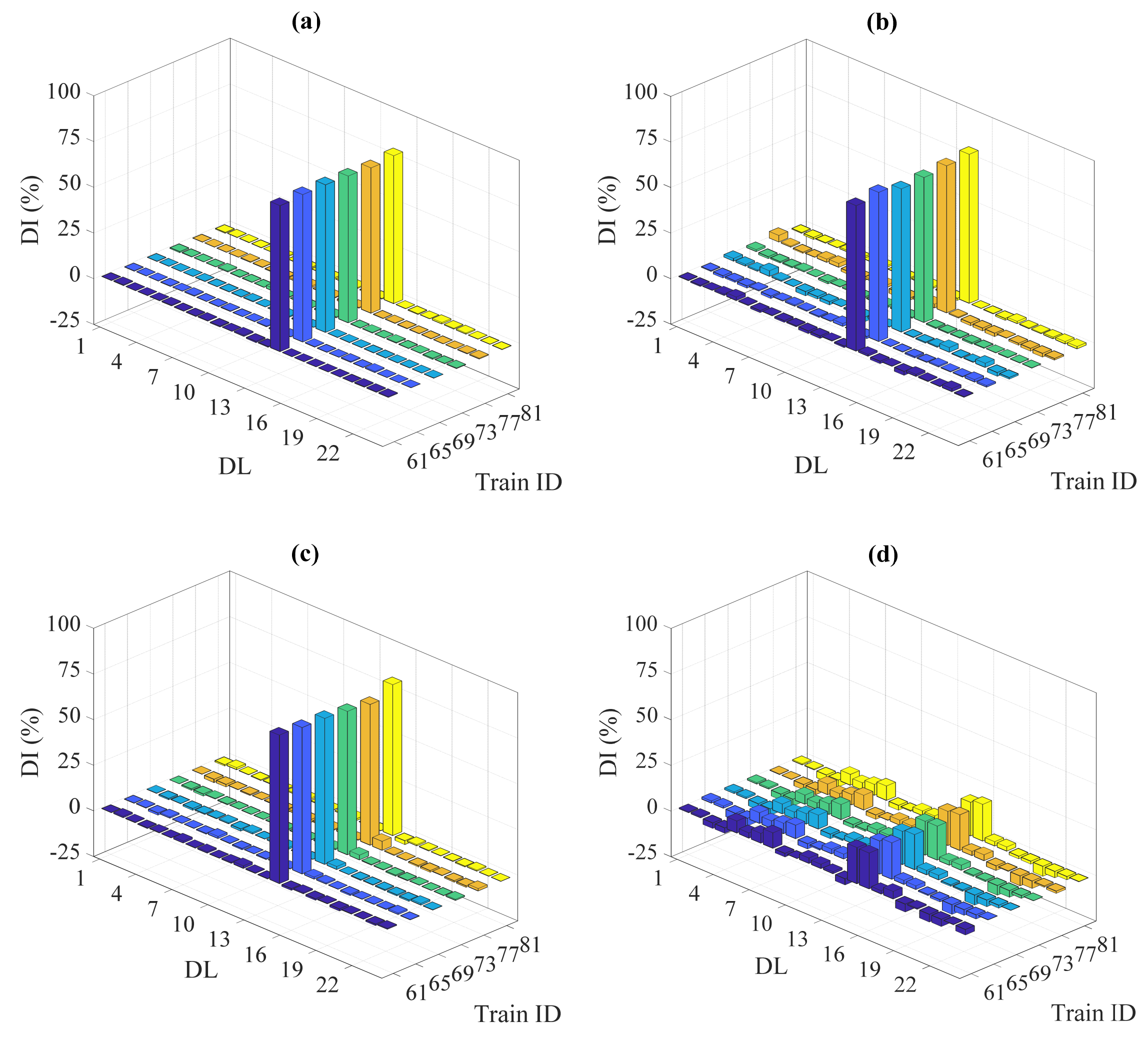


Fig. 45 Through girder span ANNs testing, IP 3, all testing trains, DI=80% at DL 15: (a) strain; (b) rotation; (c) displacement; and (d) acceleration.

6.6. Conclusions

The viability of expanding an output-only, automated, strain-based damage detection framework that utilized the POD-ANN framework described earlier to other response parameters (rotations, displacements and accelerations) was studied. The expanded framework was examined via its application to the truss and plate girder spans discussed in Chapter 3. Optimal instrumentation schemes were concurrently studied; with three proposed sensors schemes being examined for each bridge span (35).

A set of 24 routine train loading events and combinations of DLs and DIs were simulated using MATLAB and the SAP2000 OAPI. Results demonstrated the efficiency and applicability of the proposed framework for detecting stringer-to-floor beam damage for a wide variety of responses, train events and instrumentation plans. The proposed methodology could be extended further to include other structural systems or deficiencies.

It was concluded that:

- DL and DI were captured accurately for studied bridge spans.
- Damage could be accurately detected using as few as 20 sensors in the truss span and 12 sensors in the through girder span.
- Strain, rotation or displacement sensors could be used to detect damage locations in the investigated bridge spans.

Chapter 7 : Mitigating Influence of Modeling Uncertainty on POD-ANN Framework

Published: International Journal of Fatigue, **https://doi.org/10.1016/j.ijfatigue.2019.105458**

Rageh, A., Eftekhar Azam, S., & Linzell, D.

7.1. Abstract

Damage detection efficiency and accuracy were shown to be significantly influenced by modeling uncertainties (MUs) by overshadowing damage effects. To investigate the applicability of the framework detailed in Chapter 4 and Chapter 5 to in-service bridges, a systematic analysis of the effect of MUs on the proposed POD-ANN framework was necessary. MU influence on the performance of the POD-ANN damage detection method was investigated and a new procedure for generating training data for ANNs was proposed. The procedure was founded on synchronizing $\boldsymbol{\varphi}$s extracted from measured structural response with those calculated from a numerical model to allow for generating ANN training data sets based on measured responses. This chapter discusses how numerical and field investigations were synchronized.

The main objective of the numerical investigation was to identify a robust damage feature independent of level and location of assumed MUs. Results provided in this chapter showed that DL and DI were detected with high accuracy for studied uncertainty cases; however, as expected, damage detection accuracy reduced as MU increased. A hybrid experimental-numerical approach was then implemented for the field investigation studies. This approach applied identified damage features from the numerical investigation to measurements of the truss span response of the studied bridge with damage scenarios used to train the framework being produced. MATLAB algorithms

were developed that preprocessed field data and eliminated $\boldsymbol{\varphi}$ variations resulted from loading uncertainties. ANNs were trained and tested using the field strain estimated $\boldsymbol{\varphi}$ from the hybrid approach and DL and DI results were obtained under non-stationary, unknown train loads.

7.2. Novel damage feature for POD-ANN damage identification in the presence of MU

Based on MU existence and assumptions made on the availability of train axle loads, four possible damage identification scenarios are possible:

1. No MU, known loading configurations;
2. No MU, unknown loading configurations;
3. MU, known loading configurations; and
4. MU, unknown loading configurations.

The first two scenarios could be effectively addressed using the framework described in Chapter 4 and Chapter 5 (61, 62). The two latter scenarios feature new MU and POD-based damage features. Proposed methodologies to address these cases are presented in the following two subsections.

7.2.1. ΔPOMs for training ANNs in the presence of MU, absence of load uncertainty

In general, it is known that structural response POMs ($\boldsymbol{\varphi}$) are a function of sensor network topology; sensor types; structure mechanistic and geometric properties; and external loads. These parameters can be represented as:

$$\boldsymbol{\varphi} = \begin{bmatrix} \varphi_1 \\ \vdots \\ \varphi_j \\ \vdots \\ \varphi_{n_m} \end{bmatrix} = g(\boldsymbol{\vartheta}^l, \boldsymbol{\vartheta}^g, \Xi, \boldsymbol{\mathcal{F}}) \quad (8)$$

where: $\boldsymbol{\vartheta}^l \in \mathbb{R}^{p_l}$ is the vector that includes p_l parameters that affect local structural response; $\boldsymbol{\vartheta}^g \in \mathbb{R}^{p_g}$ denotes the vector that includes p_g parameters that affect global structural response; $\Xi \in \mathbb{R}^{n_m \times \boldsymbol{n_d}}$ defines coordinates of the sensor network so that φ_j corresponds to $\Xi_{j,1:3}$; and $\boldsymbol{\mathcal{F}} \in \mathbb{R}^{\boldsymbol{m}}$ denotes the vector of external loads.

If the change in mechanical or geometric properties of a structural member affects the local structural response, any change in $\boldsymbol{\varphi}$ would be predominantly local. Assume that different models of a certain structure were available and denoted by $\mathcal{M}_i$, $i = 1,2, \dots, m$. Also, assume that $\boldsymbol{\vartheta}^g$ and Ξ for each $\mathcal{M}_i$ is constant. Finally, it will be assumed that the number of sensors is equal to the number of parameters that govern the local response, i.e. $n_m = p_l$, and that ϑ_j^l affects the structure's response at the sensor installed in $\Xi_{j,1:3}$. As a result:

$$\boldsymbol{\vartheta}^l = \begin{bmatrix} \vartheta_1^l \\ \vdots \\ \vartheta_j^l \\ \vdots \\ \vartheta_n^l \end{bmatrix}. \tag{9}$$

and, for all models $\mathcal{M}_i$:

$$g\left(\boldsymbol{\vartheta}^l + \Delta_j \boldsymbol{\vartheta}^l, \boldsymbol{\vartheta}^g, \Xi, \boldsymbol{\mathcal{F}}\right) - g(\boldsymbol{\vartheta}^l, \boldsymbol{\vartheta}^g, \Xi, \boldsymbol{\mathcal{F}}) \cong \begin{bmatrix} 0 \\ \vdots \\ \Delta\varphi_j \\ \vdots \\ 0 \end{bmatrix} \tag{10}$$

where: $\Delta_j \boldsymbol{\vartheta}^l$ denotes an increment of j^{th} row of the vector $\boldsymbol{\vartheta}^l$. It is shown that the change in the j^{th} component of POM ($\Delta\varphi_j$) is proportional to the increment of j^{th} row of the vector $\boldsymbol{\vartheta}^l$:

$$\Delta\varphi_j \propto \Delta_j \boldsymbol{\vartheta}^l. \tag{11}$$

It follows that, if $\Delta\boldsymbol{\varphi}$ is used as a damage feature within a POD-ANN damage identification framework, and $\boldsymbol{\vartheta}^g$, $\boldsymbol{\Xi}$ and $\boldsymbol{\mathcal{F}}$ are constant, any of $\mathcal{M}_i$ models could effectively generate damage scenarios required for training the ANNs. Extensive numerical investigations in Section 7.4 support this premise and show that, while $\boldsymbol{\varphi}$ globally varies by $\Delta_j \boldsymbol{\vartheta}^l$ for any $\mathcal{M}_i$ subjected to the same external load, $\Delta\boldsymbol{\varphi}$ varies locally. When the damage scenario is a local phenomenon, it will produce an increment in the value of the parameter that changes local structural response:

$$\Delta\boldsymbol{\varphi} = \boldsymbol{\varphi}^d - \boldsymbol{\varphi}^h \tag{12}$$

where: $\boldsymbol{\varphi}^d$ denotes the first POM of the structure in a damaged state; and $\boldsymbol{\varphi}^h$ stands for the first POM of healthy, baseline structure. To train the ANN for damage detection, the following features for N_d damage scenarios are proposed:

$$[\Delta\boldsymbol{\varphi}^1 \quad \cdots \quad \Delta\boldsymbol{\varphi}^{N_d}]_{n_m \times N_d} \tag{13}$$

where the superscript for $\Delta\boldsymbol{\varphi}$ indicates the damage scenario.

If the right model class is constructed for the actual structure, one can argue that ΔPOMs ($\Delta\boldsymbol{\varphi}$) obtained from response measured before and after a deficiency happens could be reasonably approximated by $\Delta\boldsymbol{\varphi}$ from models $\mathcal{M}_i$:

$$\Delta\boldsymbol{\varphi}_{exp} \cong \Delta\boldsymbol{\varphi}_{\mathcal{M}_i} \tag{14}$$

7.2.2. Approximating POMs for training ANNs in the presence of MU and load uncertainty: hybrid experimental-numerical approach

The premise from the previous section is used in herein for approximating damage scenario . Using $\boldsymbol{\varphi}$ of a healthy structure obtained from experimental data and $\Delta\boldsymbol{\varphi}$ of damage scenarios obtained from the numerical model $\mathcal{M}_i$ one can deduce the following when considering Equation 12 and Equation 14.:

$$\boldsymbol{\varphi}^d \cong \widehat{\boldsymbol{\varphi}}^d = \boldsymbol{\varphi}^h_{exp} + \Delta\boldsymbol{\varphi}_{\mathcal{M}_i} \tag{15}$$

where $\widehat{\boldsymbol{\varphi}}$ denotes an estimate of $\boldsymbol{\varphi}$. Estimated $\boldsymbol{\varphi}$ then could be used as input for training the ANN:

$$[[\widehat{\boldsymbol{\varphi}}^{1,1} \quad \cdots \quad \widehat{\boldsymbol{\varphi}}^{N_d,1}] \quad \cdots \quad [\widehat{\boldsymbol{\varphi}}^{1,N_t} \quad \cdots \quad \widehat{\boldsymbol{\varphi}}^{N_d,N_t}]]_{n_m\times(N_t\times N_d)}. \tag{16}$$

Extensive numerical analyses in Section 7.4 again support this premise.

7.3. Studied bridge, analytical model and instrumentations

The investigated bridge truss span was used in this chapter as a testbed for MUs study. For the truss details and analytical model, see Chapter 3. A total of 20 strain transducers installed at stringer bottom flanges close to stringer-to-floor beam connections were adopted, see Fig. 8. Strains were extracted from numerical investigation or measured from field tests at these locations.

7.4. Numerical validation

The objective of the numerical investigation was to identify a robust damage feature that is largely independent of MU and could be used to identify damage scenarios from field measured strains. Five numerical models helped identify the feature, one being the base model with no MU and the rest including simulated MUs. The most consistently recognized damage feature was selected and applied to data produced from the numerical models and field tests. A flowchart describing the numerical validation procedure is shown in Fig. 46.

For details about fatigue cracks and the associated reduction in the rotation stiffness, see Chapter 3. In the current study, the damage was simulated at the 20 stringer ends by reducing end spring rotational stiffness from 0% to 100% reduction (i.e., DI = 0% to DI = 100%) in increments of 10%. Stiffness reductions were applied to one stringer end at a time, with other connections being undamaged and, as stated earlier, Damage Locations (DLs) correspond to instrument locations (ILs) in Fig. 8. It is also important to mention that damage was simulated on one side of the floor-beam because it was assumed damage to connections on either side would have the same influence on stringer continuity.

The bridge models were excited using real train loads. Out of the 81 trains shown and RMS sorted in Fig. 11, 24 trains having RMS that caused higher strains and provided $\boldsymbol{\varphi}$ having minimal variation, were selected. This selected 24 trains being used in this chapter were Train 57 to Train 81 shown in Fig. 11.

The numerical study completed herein considered strain $\boldsymbol{\varphi}$ variations caused by:

(i) Non-stationary (i.e., moving) train loads.

(ii) DLs and DIs.

(iii) MUs.

Analyses for various train loadings and DL/DI combinations were completed automatically using MATLAB and SAP2000 OAPI. For each simulated MU, the number of the associated analyses was 4824 damage scenarios, which corresponded to 24 trains, 10 DIs and 20 DLs.

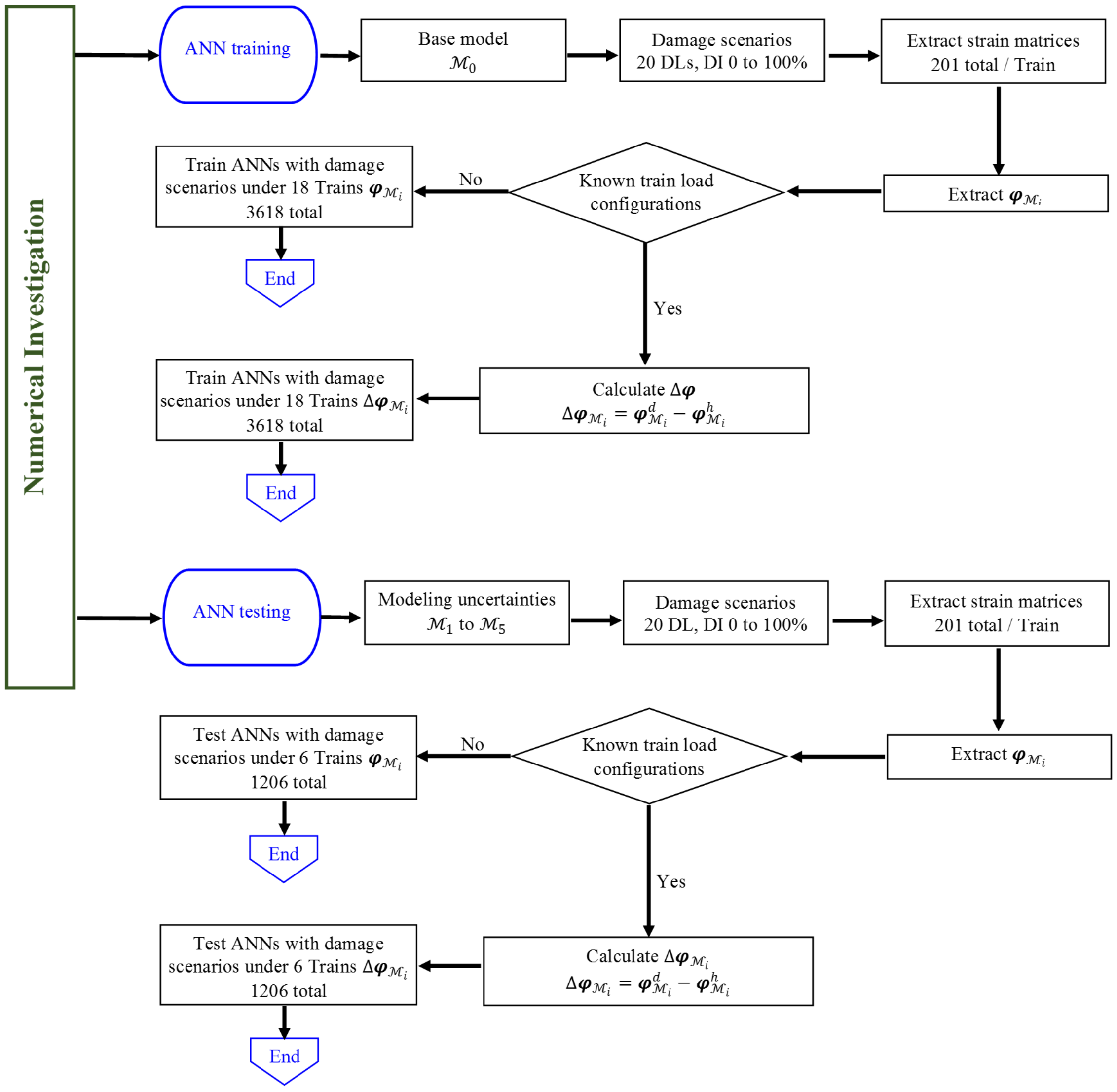

Numerical Investigation
ANN training
Base model $\mathcal{M}_0$
Damage scenarios 20 DLs, DI 0 to 100%
Extract strain matrices 201 total / Train
Extract $\boldsymbol{\varphi}_{\mathcal{M}_i}$
Known train load configurations
No
Yes
Train ANNs with damage scenarios under 18 Trains $\boldsymbol{\varphi}_{\mathcal{M}_i}$ 3618 total
End
Calculate $\Delta\boldsymbol{\varphi}$
$\Delta\boldsymbol{\varphi}_{\mathcal{M}_i} = \boldsymbol{\varphi}^d_{\mathcal{M}_i} - \boldsymbol{\varphi}^h_{\mathcal{M}_i}$
Train ANNs with damage scenarios under 18 Trains $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$ 3618 total
End
ANN testing
Modeling uncertainties $\mathcal{M}_1$ to $\mathcal{M}_5$
Damage scenarios 20 DL, DI 0 to 100%
Extract strain matrices 201 total / Train
Extract $\boldsymbol{\varphi}_{\mathcal{M}_i}$
Known train load configurations
No
Yes
Test ANNs with damage scenarios under 6 Trains $\boldsymbol{\varphi}_{\mathcal{M}_i}$ 1206 total
End
Calculate $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$
$\Delta\boldsymbol{\varphi}_{\mathcal{M}_i} = \boldsymbol{\varphi}^d_{\mathcal{M}_i} - \boldsymbol{\varphi}^h_{\mathcal{M}_i}$
Test ANNs with damage scenarios under 6 Trains $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$ 1206 total
End

Fig. 46 Numerical investigation flowchart.

The authors were unable to find resources that helped couple MATLAB code to the SAP2000 OAPI to complete automated analyses and output extraction. As a result, appendices for the processes developed for the current study are provided. 0Appendix 2 contains MATLAB code for running multi-step analyses and Appendix 3 details relationships between MATLAB coding and SAP2000 OAPI windows for some of the crucial functions.

7.4.1. MU cases

Stringer-to-floor beam end-fixity ratios vary widely and, as a result, no clearly defined relationship exists. Design practice assumes that these connections would transfer shear force only; however, laboratory and field tests indicated a high amount of end fixity (34, 37-40, 42). Stringer-to-floor beam connection end fixity ratio was shown to influence stress time histories significantly and, in-turn, fatigue life predictions for connecting elements with fatigue crack development mainly attributed to out-of-plane deformation of connecting angles and stress concentrations at rivets heads (40).

Because of significant variations of end fixity ratios observed from tests, higher discrepancies between actual and modeled ratios are expected. As a result, the end fixity ratio was selected as the major MU factor for the current study. As stated earlier, the base model had no MU ($\mathcal{M}_0$) and the end fixity ratio was assumed to be 67% of that for fully continuous stringer (37). This corresponded to a linear rotational spring coefficient of 803435 kN-m/rad applied uniformly to all stringer ends. The first 2 MU cases ($\mathcal{M}_1$and $\mathcal{M}_2$) uniformly varied all stringer end fixity ratios by + 80% and -50% of that assigned in $\mathcal{M}_0$, which corresponded to assigned spring coefficients of 1446183 and 401717 kN-m/rad for $\mathcal{M}_1$and $\mathcal{M}_2$, respectively. The 80% increase was based on results from model validation activities (35). The other 3 MU cases ($\mathcal{M}_3$ to $\mathcal{M}_5$) used randomly modified end fixity ratios at +/- 25, 50, and 100% of the $\mathcal{M}_0$ case, which produced coefficients varied randomly between the assigned values at each connection. Assigned spring coefficients are listed in Table 1 for each stringer. It is important to mention that $\mathcal{M}_1$ is a calibrated model for the investigated bridge. The model was validated against full-scale, measured responses (35).

To demonstrate influence of selected MU cases on strain response, results were compared for $\mathcal{M}_0$ to $\mathcal{M}_5$ cases where significant change in strain time-histories was observed, especially for Instrumented Locations (IL) located beneath the unloaded track (i.e., DL 11 to DL 20). Comparison of strains are shown in Fig. 47 for selected MU cases having 40, 60, 80 and 100% DI at DLs 3, 8, 13 and 18. Significant change in strain from $\mathcal{M}_0$ was observed for $\mathcal{M}_1$ to $\mathcal{M}_5$ strain time-histories.

Table 1 MU cases

Model	**Assignment**	**Normalized rotational spring coefficient ($\mathcal{M}_i/\mathcal{M}_0$)**
$\mathcal{M}_0$	Uniform	1.00
$\mathcal{M}_1$	Uniform, increase +80% of $\mathcal{M}_0$	1.80
$\mathcal{M}_2$	Uniform, decrease -50% of $\mathcal{M}_0$	0.50
$\mathcal{M}_3$	Random, +/- 25% of $\mathcal{M}_0$	Between 0.76 and 1.25
$\mathcal{M}_4$	Random, +/- 50% of $\mathcal{M}_0$	Between 0.53 and 1.45
$\mathcal{M}_5$	Random, +/- 100% of $\mathcal{M}_0$	Between 0.23 and 1.92

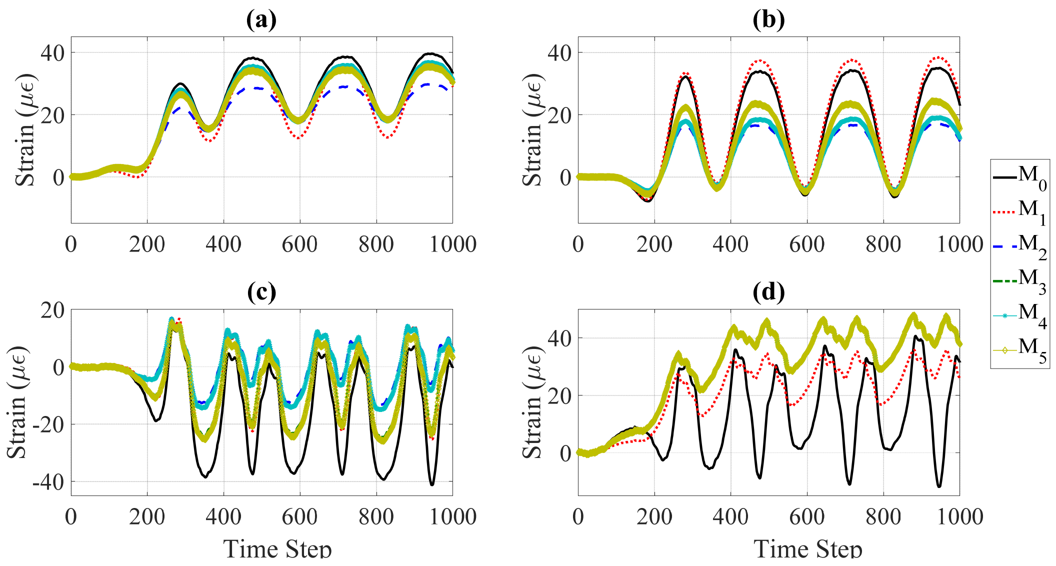


Fig. 47 Strain time-history comparisons for different MU scenarios: (a) DI 40% at DL 3; (b) DI 60% at DL 8; (c) DI 80% at DL 13; and (d) DI 100% at DL 18.

7.4.2. Robustness of ΔPOMs to MUs

Strains were extracted at the ends highlighted in Fig. 8 and corresponding $\boldsymbol{\varphi}_{\mathcal{M}_i}$ were calculated for each of the considered $\mathcal{M}_i$ cases and damage scenarios. Fig. 48 (a) shows a comparison of $\boldsymbol{\varphi}^h_{\mathcal{M}_i}$ for the 24 train events for the $\mathcal{M}_i$ cases where significant change was observed due to MUs and minimal effects observed from load variations. Fig. 48 (b) depicts the mean for $\boldsymbol{\varphi}^h_{\mathcal{M}_i}$ for considered $\mathcal{M}_i$ cases, which again highlights significant $\boldsymbol{\varphi}^h_{\mathcal{M}_i}$ changes associated with MUs. Since $\boldsymbol{\varphi}^h_{\mathcal{M}_i}$ varyies significantly for the MUs, it cannot be classified as a robust damage feature that could eliminate or reduce MU effects. To eliminate $\boldsymbol{\varphi}_{\mathcal{M}_i}$ variations attributed to examined MUs, $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$ was calculated for each damage scenario as:

$$\Delta\boldsymbol{\varphi}_{\mathcal{M}_i} = \boldsymbol{\varphi}^d_{\mathcal{M}_i} - \boldsymbol{\varphi}^h_{\mathcal{M}_i} \tag{17}$$

The resulted $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$ showed that variations due to MUs were significantly reduced when compared against $\boldsymbol{\varphi}_{\mathcal{M}_i}$ variations. As shown in Fig. 48, DL 18 experienced the highest variation in $\boldsymbol{\varphi}^{h}_{\mathcal{M}_i}$; however, when $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$ was used the variation was significantly lower and the DL accurately identified. Fig. 49 (a) shows that $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$at DL 18 for DI 80% and the considered train events showed minor variation observed for examined MUs and train loadings. Fig. 49 (b) compares the mean of $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$for the MUs and again shows minor variations. These minor variations indicate that, for the studied bridge, model and sceanarios, $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$could be considered as a robust damage feature that is largely independent of MUs and train loads.

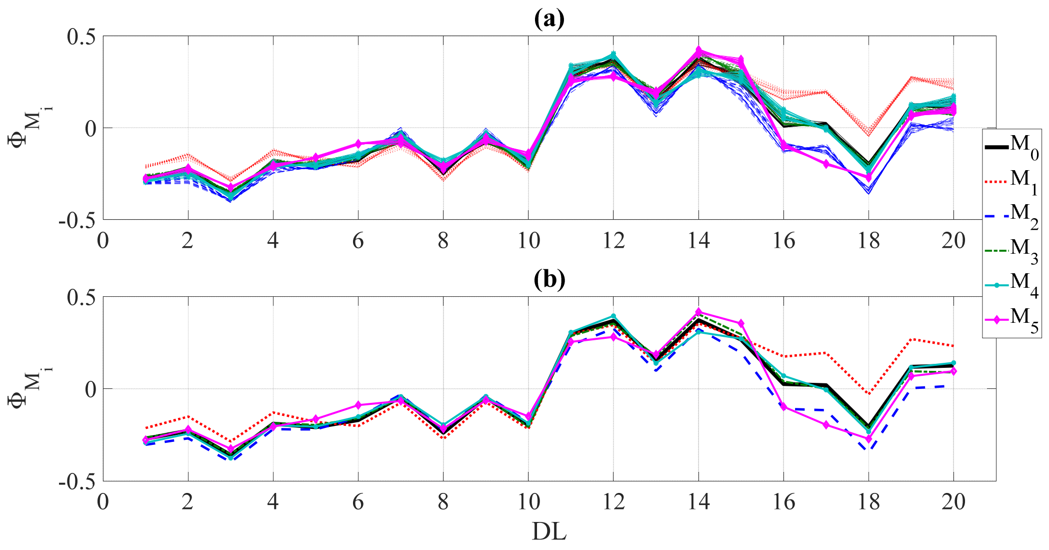


Fig. 48 Healthy POMs variations associated with MU scenarios $(\boldsymbol{\varphi}^{\boldsymbol{h}}_{\boldsymbol{\mathcal{M}_i}})$: (a) $\boldsymbol{\varphi}^{\boldsymbol{h}}_{\boldsymbol{\mathcal{M}_i}}$; and (b) $\boldsymbol{\varphi}^{\boldsymbol{h}}_{\boldsymbol{\mathcal{M}_i}}$ mean.

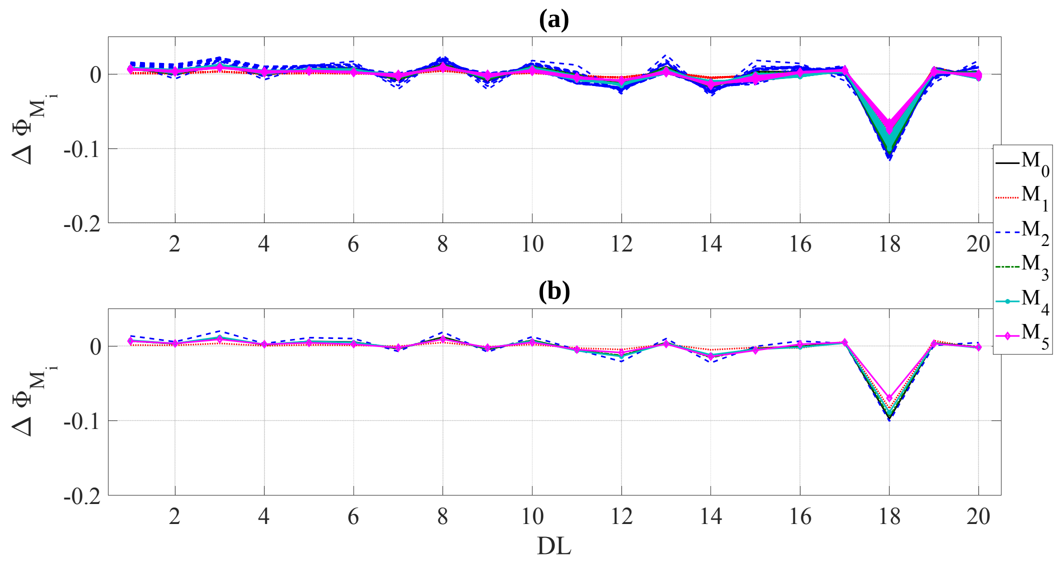


Fig. 49 $\Delta\boldsymbol{\varphi}_{\boldsymbol{\mathcal{M}_i}}$ at DL 18 with DI 80%: (a) $\Delta\boldsymbol{\varphi}_{\boldsymbol{\mathcal{M}_i}}$; and (b) $\Delta\boldsymbol{\varphi}_{\boldsymbol{\mathcal{M}_i}}$ mean.

Since generating damage scenarios for an in-situ bridge is largely impossible, an alternate way to develop damage scenarios is by implementing $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$ as a damage feature. To do so, $\widehat{\boldsymbol{\varphi}}^d$ was calculated to generate damage scenarios as a function of $\boldsymbol{\varphi}^h_{\mathcal{M}_1}$ to $\boldsymbol{\varphi}^h_{\mathcal{M}_5}$ and $\Delta\boldsymbol{\varphi}_{\mathcal{M}_0}$ as:

$$\widehat{\boldsymbol{\varphi}}^d_{\mathcal{M}_i} = \Delta\boldsymbol{\varphi}_{\mathcal{M}_0} + \boldsymbol{\varphi}^h_{\mathcal{M}_i} \tag{18}$$

Since $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$ variations due to MUs were observed to be minor, $\widehat{\boldsymbol{\varphi}}^d_{\mathcal{M}_i}$ is expected to match closely $\boldsymbol{\varphi}^d_{\mathcal{M}_i}$. To ensure the applicability of $\widehat{\boldsymbol{\varphi}}^d_{\mathcal{M}_i}$ for simulating actual damage scenarios, comparisons between $\boldsymbol{\varphi}^h_{\mathcal{M}_i}$, $\widehat{\boldsymbol{\varphi}}^d_{\mathcal{M}_i}$ and $\boldsymbol{\varphi}^d_{\mathcal{M}_i}$ where completed and a close agreement was observed between $\widehat{\boldsymbol{\varphi}}^d_{\mathcal{M}_i}$ and $\boldsymbol{\varphi}^d_{\mathcal{M}_i}$ at different MUs. One example of those comparisons shown in Fig. 50 for DIs of 40, 60, 80 and 100% simulated at DLs 3, 8, 13 and 18 for $\mathcal{M}_1$ to $\mathcal{M}_5$ cases. The comparison showed a very close match between $\widehat{\boldsymbol{\varphi}}^d_{\mathcal{M}_i}$

(i.e., estimated from $\Delta\boldsymbol{\varphi}_{\mathcal{M}_0}$ and $\boldsymbol{\varphi}^h_{\mathcal{M}_i}$) and $\boldsymbol{\varphi}^d_{\mathcal{M}_i}$ (i.e., calculated from deficient models strains $\mathcal{M}_1$ to $\mathcal{M}_5$) for various DIs, DLs and MUs. The results proved that generating damage scenario POMs ($\widehat{\boldsymbol{\varphi}}^d_{\mathcal{M}_i}$) from $\Delta\boldsymbol{\varphi}_{\mathcal{M}_0}$ of one model and $\boldsymbol{\varphi}^h_{\mathcal{M}_i}$ of another model is possible. As a result, a similar approach could be applied to generate $\widehat{\boldsymbol{\varphi}}^d_{exp}$ from $\boldsymbol{\varphi}^h_{exp}$ from measured strains. In other words, $\widehat{\boldsymbol{\varphi}}^d_{exp}$ can be generated from $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$ and $\boldsymbol{\varphi}^h_{exp}$.

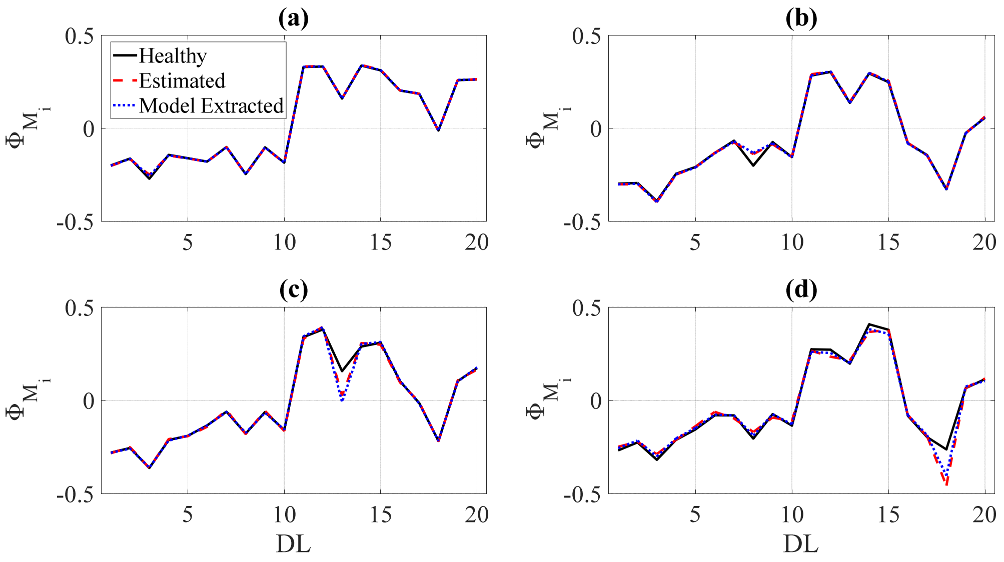


Fig. 50 $\widehat{\boldsymbol{\varphi}}^{\boldsymbol{d}}_{\boldsymbol{\mathcal{M}_i}}$ accuracy comparisons: (a) DI 40% at DL 3 with $\boldsymbol{\mathcal{M}_1}$; (b) DI 60% at DL 8 with $\boldsymbol{\mathcal{M}_2}$; (c) DI 80% at DL 13 with $\boldsymbol{\mathcal{M}_4}$; and (d) DI 100% at DL 18 with $\boldsymbol{\mathcal{M}_5}$.

It is important to note that, while inaccurately modeling stringer to floor beam connections is not the only source of MU, significant influence on predicted response accuracy is anticipated to occur when the connections are inaccurately represented. Other potential source of MU could include inaccurate representation of:

1. Material properties, represented using inaccurate elastic moduli (E)
2. Bridge support conditions.

To examine the influence of these MU sources on $\boldsymbol{\varphi}$, additional cases were studied. These included:

(i) uniformly increasing or reducing E of the entire bridge by 20% (1.2E and 0.8E)

(ii) uniformly increasing or reducing E of the stringers by 20% (1.2E Str and 0.8E Str)

(iii) using longitudinal springs at the free (roller supported) end to account for frozen bearings, with spring coefficients set to 50 and 100% of the bridge longitudinal stiffness (K) (0.5K Spr and 1.0K Spr)

The longitudinal bridge stiffness was estimated as the inverse of the longitudinal displacement at the roller ends of the bridge under longitudinal point loads.

In addition, the effects of multiple MU sources on $\boldsymbol{\varphi}$ were also examined as follows:

(i) $M_1 + 0.8E$

(ii) $M_2 + 1.2E\ Str$

(iii) $M_5 + 1.2E$

(iv) $M_1 + 0.8E\ Str + 0.5K\ Spr$

(v) $M_4 + 1.2E + 1.0K\ Spr$

Fig. 51 contains representative comparisons between $\boldsymbol{\varphi}_{\mathcal{M}_0}$ and $\boldsymbol{\varphi}$ for M_1 and M_5 and the sources of MU outlined in the preceding paragraph. As can be seen in the figure, including additional MU sources or their combinations slightly influenced $\boldsymbol{\varphi}$ when compared against the M_1 to M_5 cases, with agreement improving when a validated version of the base model was used. Fig. 52 shows additional comparisons between $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$ at DL 13 for DI 80%. Minor variations between the M_1 to M_5 cases and the additional cases were again observed. These findings support using MU cases M_1 to M_5 for the current study and also support the premise that $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$ is a robust damage feature largely independent of MUs.

Comparisons between $\boldsymbol{\varphi}^h_{\mathcal{M}_i}$, $\widehat{\boldsymbol{\varphi}}^d_{\mathcal{M}_i}$ and $\boldsymbol{\varphi}^d_{\mathcal{M}_i}$ for multiple MUs were also completed with excellent agreement observed. An example of those comparisons is shown in Fig. 53 for DIs of 40, 60, 80 and 100% at DL 3, 8, 13 and 18. The results prove that generating damage scenario POMs ($\widehat{\boldsymbol{\varphi}}^d_{\mathcal{M}_i}$) from $\Delta\boldsymbol{\varphi}_{\mathcal{M}_0}$ of one model and $\boldsymbol{\varphi}^h_{\mathcal{M}_i}$ from another model is possible when multiple MUs are included.

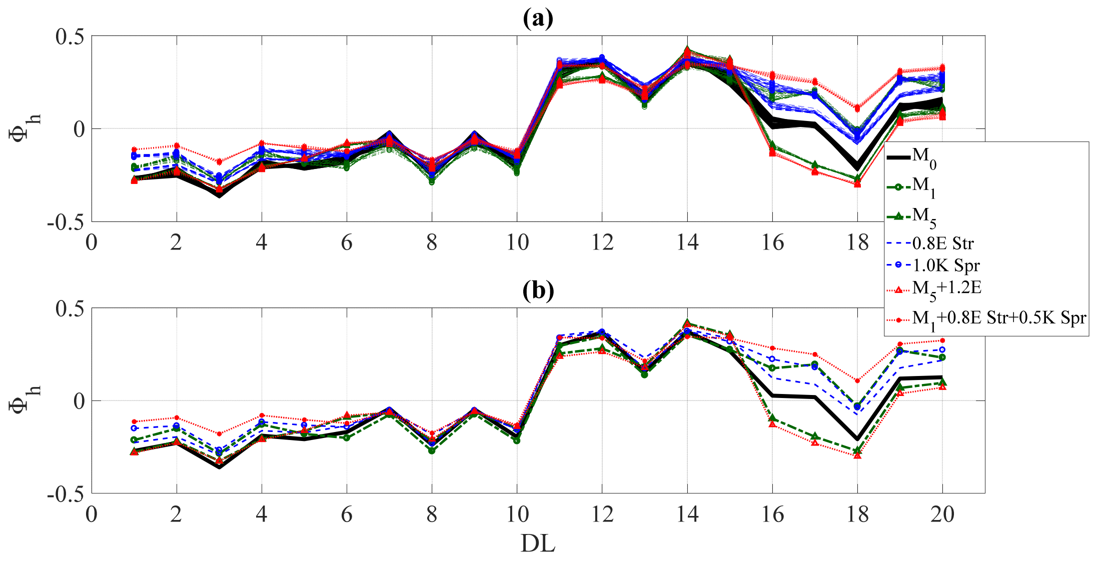


Fig. 51 Healthy POM variations for additional MU cases ($\boldsymbol{\varphi}^h_{\mathcal{M}_i}$): (a) $\boldsymbol{\varphi}^h_{\mathcal{M}_i}$; and (b) $\boldsymbol{\varphi}^h_{\mathcal{M}_i}$ mean.

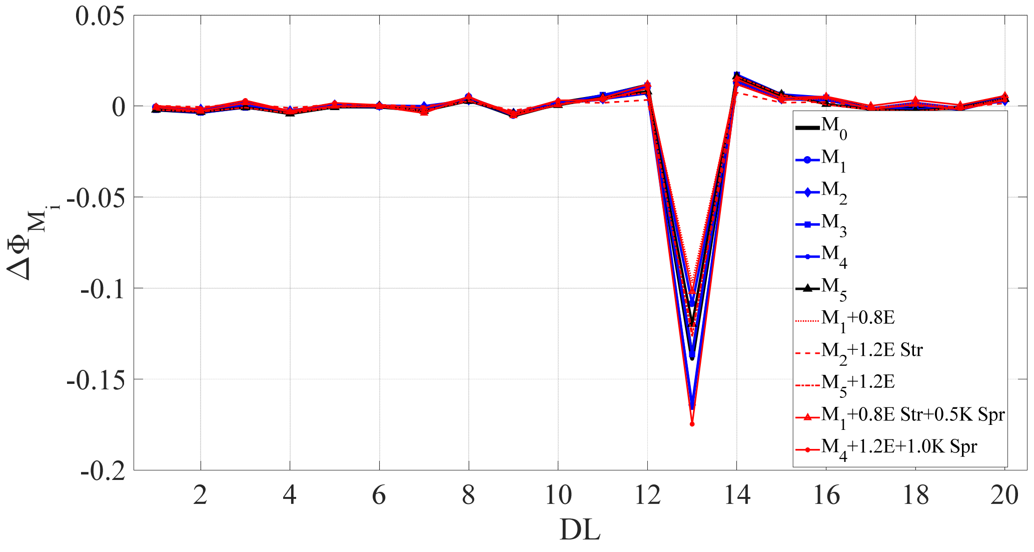


Fig. 52 $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$ at DL 13 with DI 80% for selected MU cases and cases with multiple MU sources.

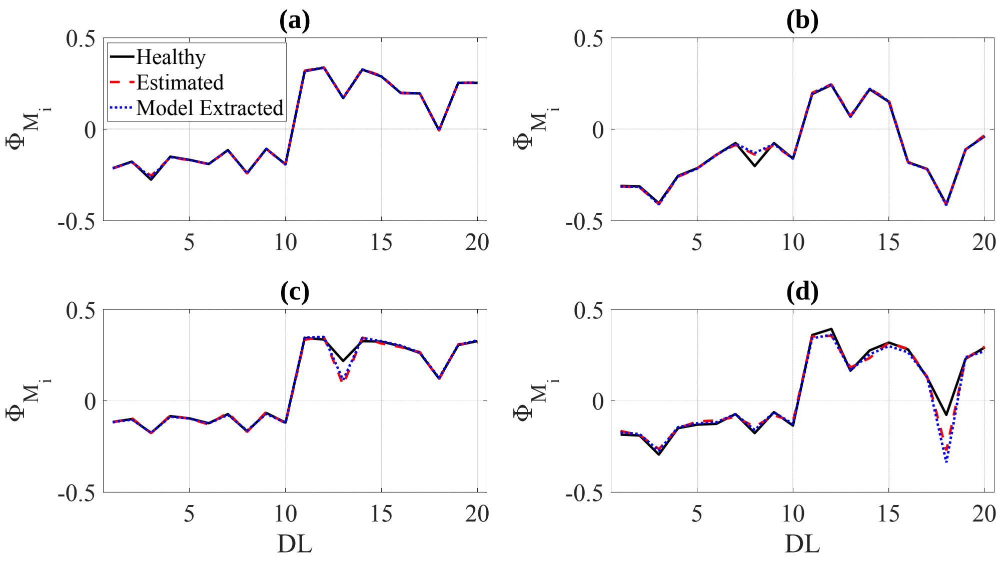


Fig. 53 $\widehat{\boldsymbol{\varphi}}^{d}_{\mathcal{M}_i}$ accuracy comparisons for cases with multiple MUs: (a) DI 40% at DL 3 with $\boldsymbol{M_1 + 0.8E}$; (b) DI 60% at DL 8 with $\boldsymbol{M_2 + 1.2E\ Str}$; (c) DI 80% at DL 13 with $\boldsymbol{M_1 + 0.8E\ Str + 0.5K\ Spr}$; and (d) DI 100% at DL 18 with $\boldsymbol{M_4 + 1.2E + 1.0K\ Spr}$.

7.4.3. ANN training and testing data sets

Training data generated for the 10 DIs at each DL shown in Fig. 8. A total of 4800 damage scenarios corresponding to each MU were developed by sequentially varying DIs at the designated DLs for the 24 train events. ANNs were trained using MATLAB *Neural Net Fitting* toolbox, which produced a nonlinear regression that, in turn, helped establish damage detection effectiveness. As mentioned earlier, 100 internal training neurons were adopted for the ANN training.

Extracted $\boldsymbol{\varphi}_{\mathcal{M}_i}^h$ and $\boldsymbol{\varphi}_{\mathcal{M}_i}^d$values were used to train and test the ANNs. Of the 24 trains, 18 used to train the ANNs and six to test their ability to detect DLs and DIs. More details about the process can be found elsewhere (62).

For a given MU and known loading scenario, $\Delta\boldsymbol{\varphi}_{\mathcal{M}_0}$ was used to train the ANNs and $\Delta\boldsymbol{\varphi}_{\mathcal{M}_1}$ to $\Delta\boldsymbol{\varphi}_{\mathcal{M}_5}$ were used to test them. The training and testing data sets differed based on:

(i) The presence of or absence of MUs.

(ii) Training load events were different from those used for testing.

For a real-world application; however, train loading configurations should be known to calculate $\Delta\boldsymbol{\varphi}$ (i.e., it is calculated as the difference between two $\boldsymbol{\varphi}$ for the same train event). For each MU and unknown loading scenario, $\widehat{\boldsymbol{\varphi}}_{\mathcal{M}_i}^d$ was then adopted to impose damage. Testing was completed for $\boldsymbol{\varphi}_{\mathcal{M}_i}^d$, which was directly calculated from deficient numerical models. These training and testing data sets differed because:

(i) $\widehat{\boldsymbol{\varphi}}_{\mathcal{M}_i}^d$ involved MUs used to train ANNs while $\boldsymbol{\varphi}_{\mathcal{M}_i}^d$ was used to test them.

(ii) Training train events loadings were different from testing loads.

A desktop computer having a multi-core architecture and a Windows 7 64-bit operating system was used to perform ANN training. The computer's Central Processing Unit was an Intel Xeon E5-2630 2.4 GHz processor with 8 Cores, 32 GB DDR4 of RAM and a 20 MB Smart Cash. Training time was between 1050 and 1300 minutes for each considered MU case.

7.4.4. Damage identification results

7.4.4.1. Training with MU and Known Loads

One of the most critical issues with any health monitoring system is avoiding false alarms, which are an erroneous report of damage detection. To ensure the developed method was robust against false alarms, $\boldsymbol{\varphi}$ for healthy scenarios subjected to testing train events were used to test trained ANNs for a given MU. Fig. 54 shows testing results when trained using MUs and known loads for healthy scenarios. It plots ANNs testing for healthy scenarios under selected testing trains for $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_4$ and $\mathcal{M}_5$ where the likelihood of false alarms is assumed to be low. It was observed that, for events associated with the six trains selected for ANN testing, the maximum false-positive DI was approximately less than 0.1%, which was deemed to be low when compared against the actual DI of 0%. These results support the premise that the method would successfully detect damage with the need for determining an acceptable threshold via long-term monitoring.

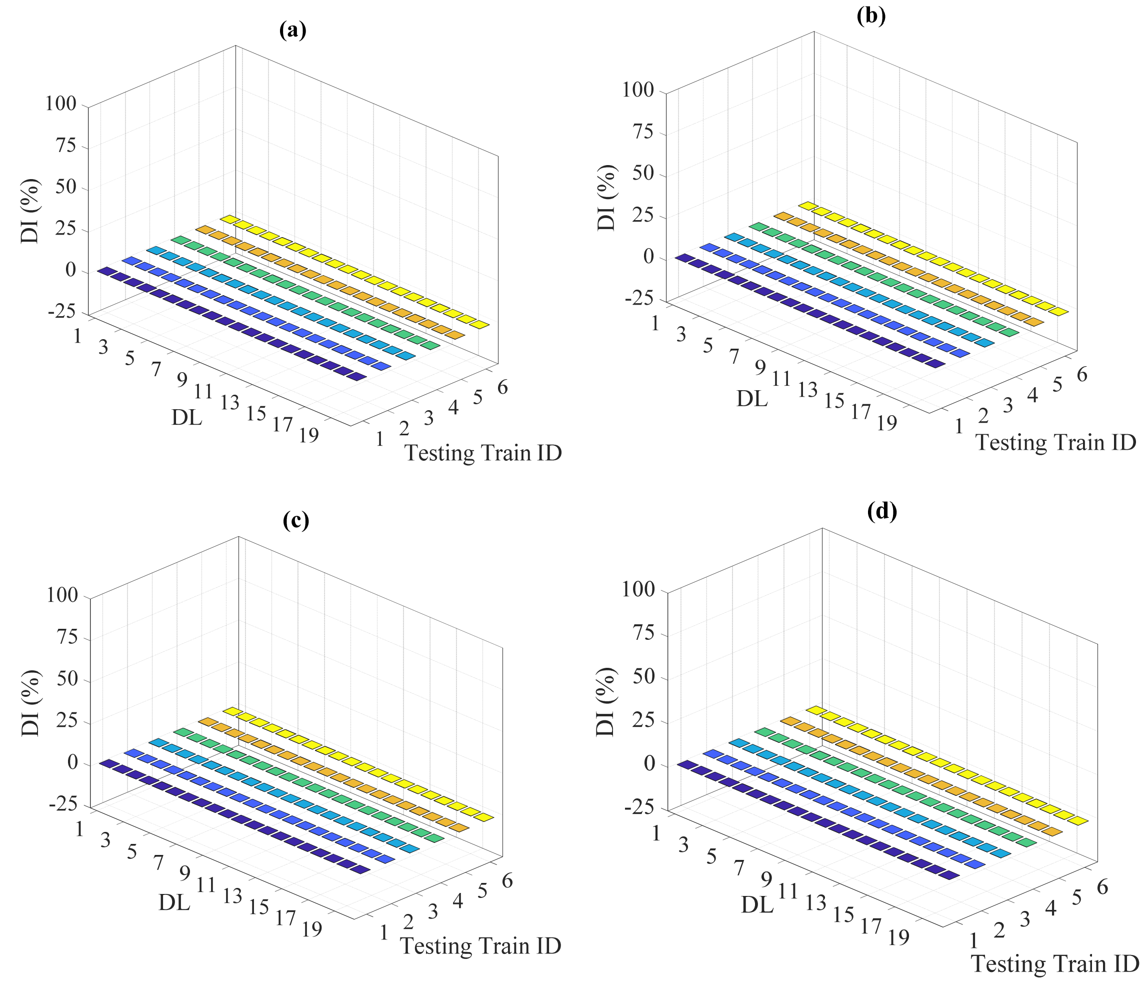


Fig. 54 ANN testing, all testing trains, DI 0%: (a) $\boldsymbol{\mathcal{M}_1}$; (b) $\boldsymbol{\mathcal{M}_2}$; (c) $\boldsymbol{\mathcal{M}_4}$; and (d) $\boldsymbol{\mathcal{M}_5}$.

To further evaluate the ability of the proposed methodology to detect DL and DI for various MUs, ANN damage detection effectiveness at instrumented locations was studied. The results of some of the damage scenarios are described herein for the sake of brevity. DLs and DIs were arbitrary with DLs of 3, 8, 13 and 18 chosen and DIs of 40, 60, 80 and 100% used for $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_4$ and $\mathcal{M}_5$. Results from ANNs for all train events are shown in Fig. 55. DIs and DLs were detected with reasonable accuracy for the considered MU cases. However, false positives and negatives were observed with varying magnitudes depending on the levels of MU and DL. Results presented in Fig. 55 are summarized in Table 2.

Table 2 MU and known loads testing results

Figure	**Subplot**	**MU case**	**Simulated DI and DL**	**Detected DI at simulated DL**	**False positives and negatives**
Fig. 55	**a**	$\mathcal{M}_1$	40% at DL 3	31 to 33%	3%
	b	$\mathcal{M}_2$	60% at DL 8	66 to 67%	15 and 25%
	c	$\mathcal{M}_4$	80% at DL 13	86 to 88%	9 and 9%
	d	$\mathcal{M}_5$	100% at DL 18	91 to 95%	4 and 10%
Fig. 56	**a**	$\mathcal{M}_1$	70% at DL 15	60 to 61%	4% and 8%
	b	$\mathcal{M}_2$	70% at DL 15	73 to 76%	10% and 12%
	c	$\mathcal{M}_4$	70% at DL 15	67 to 71%	15% and 18%
	d	$\mathcal{M}_5$	70% at DL 15	62 to 65%	10% and 18%

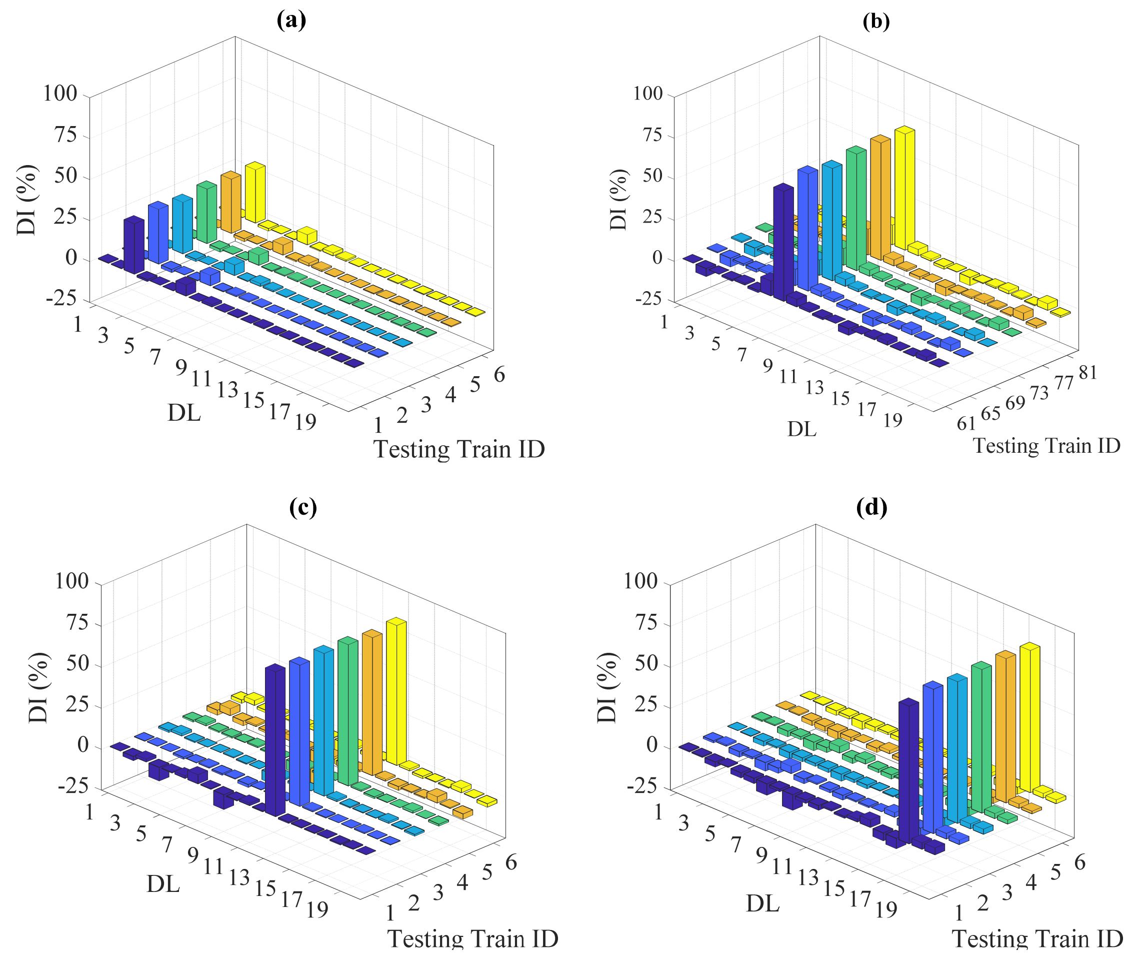


Fig. 55 ANNs testing, all testing trains: (a) DI 40% at DL 3, $\boldsymbol{\mathcal{M}_1}$; (b) DI 60% at DL 8, $\boldsymbol{\mathcal{M}_2}$; (c) DI 80% at DL 13, $\boldsymbol{\mathcal{M}_4}$; and (d) DI 100% at DL 18, $\boldsymbol{\mathcal{M}_5}$.

To illustrate the effect of MU level on DL and DI detection, another comparison between MUs at DL 15 with a DI of 70% under all testing trains was completed, as shown in Fig. 56. As demonstrated in the figure, the accuracy of detecting DL and DI varies based on the MU level, with both false-positives and negatives being observed. A summary of the testing results shown in Fig. 56 are listed in Table 2.

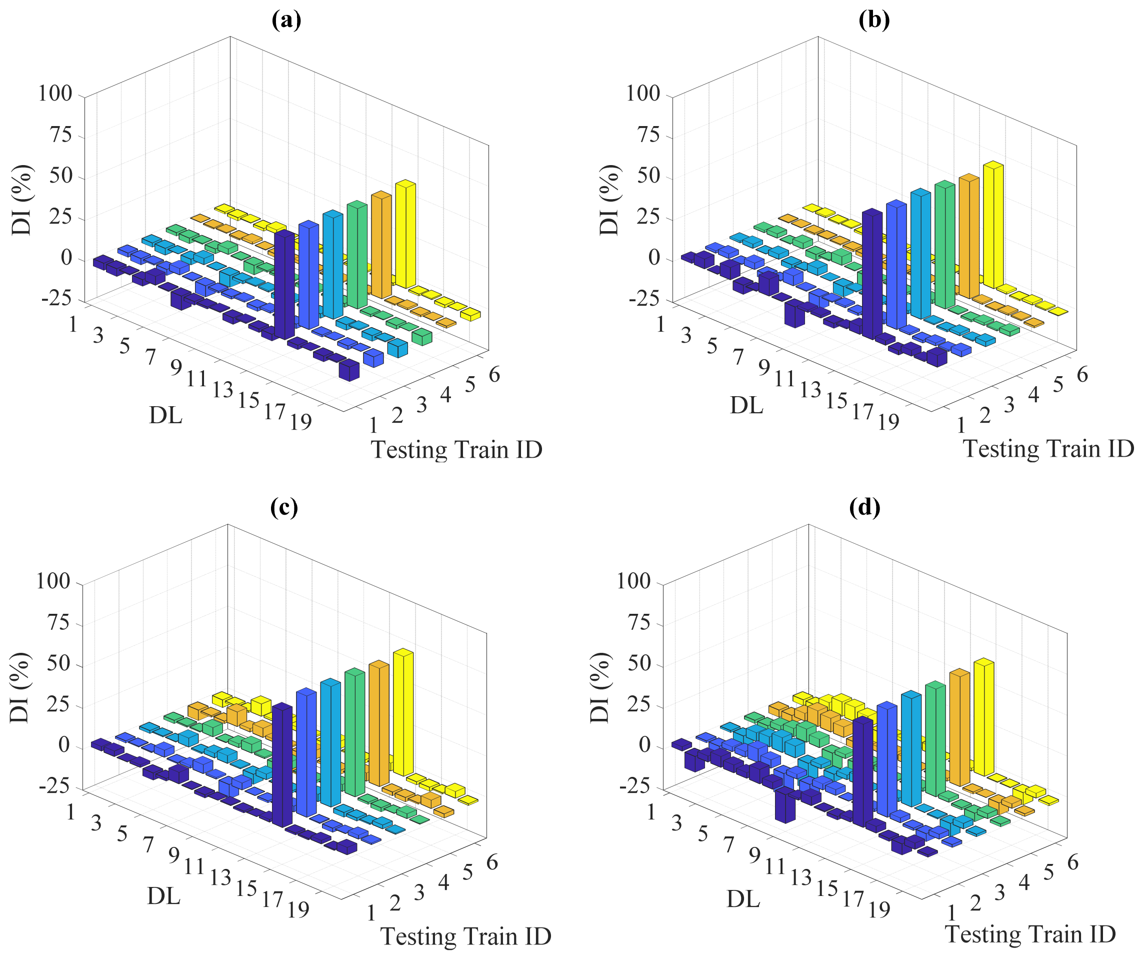


Fig. 56 ANNs testing with MU and known loading, DI 70% at DL 15, all testing trains: (a) $\boldsymbol{\mathcal{M}_1}$; (b) $\boldsymbol{\mathcal{M}_2}$; (c) $\boldsymbol{\mathcal{M}_4}$; and (d) $\boldsymbol{\mathcal{M}_5}$.

7.4.4.2. Training with MU and Unknown Loads

Fig. 57 shows testing results for MUs, unknown loads and healthy scenarios. It was observed that, for events associated with the six trains selected for ANN testing, the maximum false-positive DI was approximately less than 3%, which was considered low given the actual DI of 0%. As discussed previously, these results supported that the method would successfully detect damage with the need for determining an acceptable threshold via long-term monitoring since similar observations were made in the previous testing results.

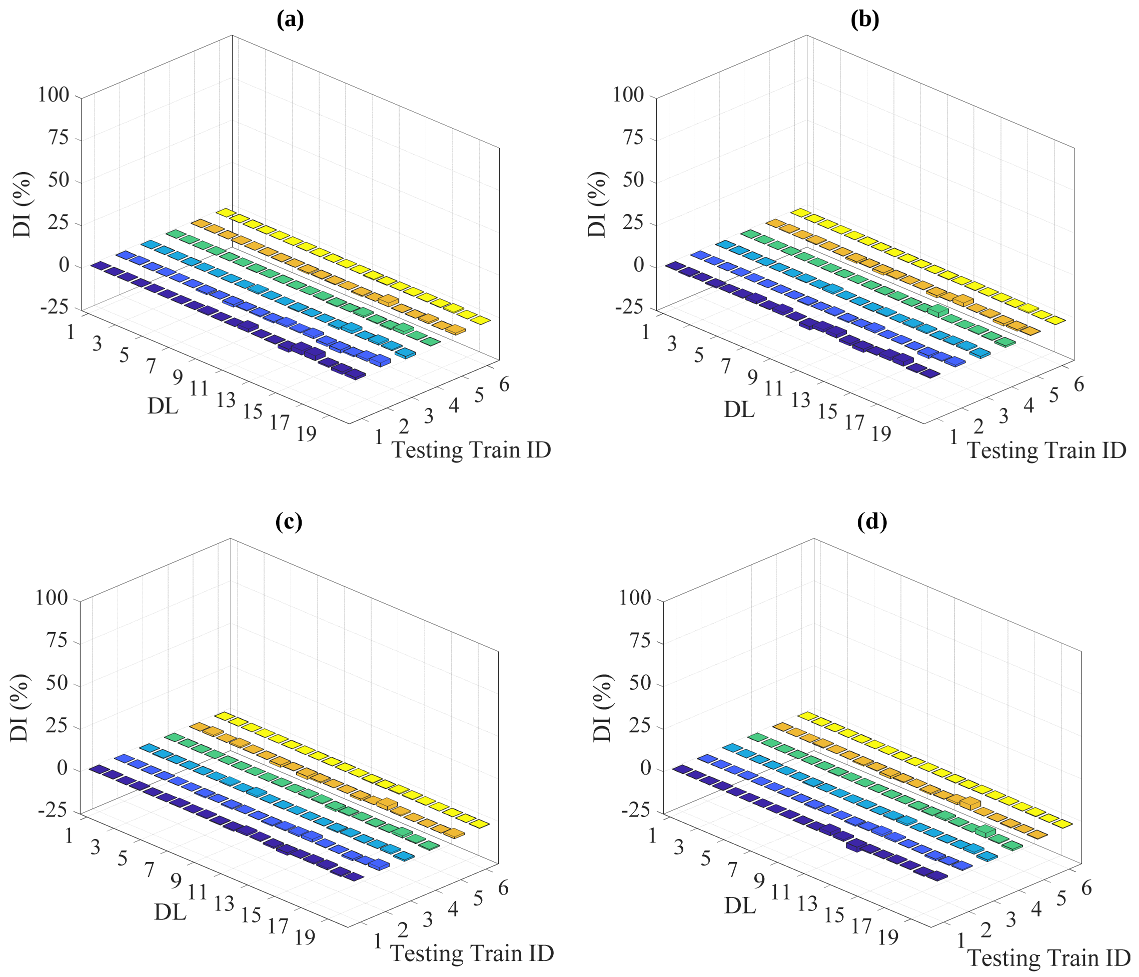


Fig. 57 ANNs testing with MU and unknown loading, all testing trains, DI 0%: (a) $\boldsymbol{\mathcal{M}_1}$; (b) $\boldsymbol{\mathcal{M}_2}$; (c) $\boldsymbol{\mathcal{M}_4}$; and (d) $\boldsymbol{\mathcal{M}_5}$.

Fig. 58 shows another example of ANN testing for all included train events and MUs for an unknown loading data set. As shown in the figure, DIs and DLs were detected with acceptable accuracy for considered MU cases. Observed false positives and negatives showed to have magnitudes varied with the level of the MU and DL. The testing results shown in Fig. 58 are summarized in Table 3.

Table 3 MU and unknown loads testing results

Figure	**Subplot**	**MU case**	**Simulated DI and DL**	**Detected DI at simulated DL**	**False positives and negatives**
Fig. 58	**a**	$\mathcal{M}_1$	40% at DL 3	30 to 33%	3%
	b	$\mathcal{M}_2$	60% at DL 8	71 to 78%	12 and 15%,
	c	$\mathcal{M}_4$	80% at DL 13	84 to 89%	3 and 5%
	d	$\mathcal{M}_5$	100% at DL 18	73 to 75%	21 and 24%
Fig. 59	**a**	$\mathcal{M}_1$	70% at DL 15	61 to 65%	7 and 10%
	b	$\mathcal{M}_2$	70% at DL 15	72 to 82%	6%
	c	$\mathcal{M}_4$	70% at DL 15	65 to 70%	17 and 16%
	d	$\mathcal{M}_5$	70% at DL 15	35 to 58%	26 and 15%

The influence of MU level on DL and DI detection is illustrated via comparisons for DL 15 and a DI of 70% for all testing trains shown in Fig. 59. As shown in the figure, DL and DI detection accuracy vary based on the MU level with false positives and negatives were observed. Summarized results of Fig. 59 are listed in Table 3.

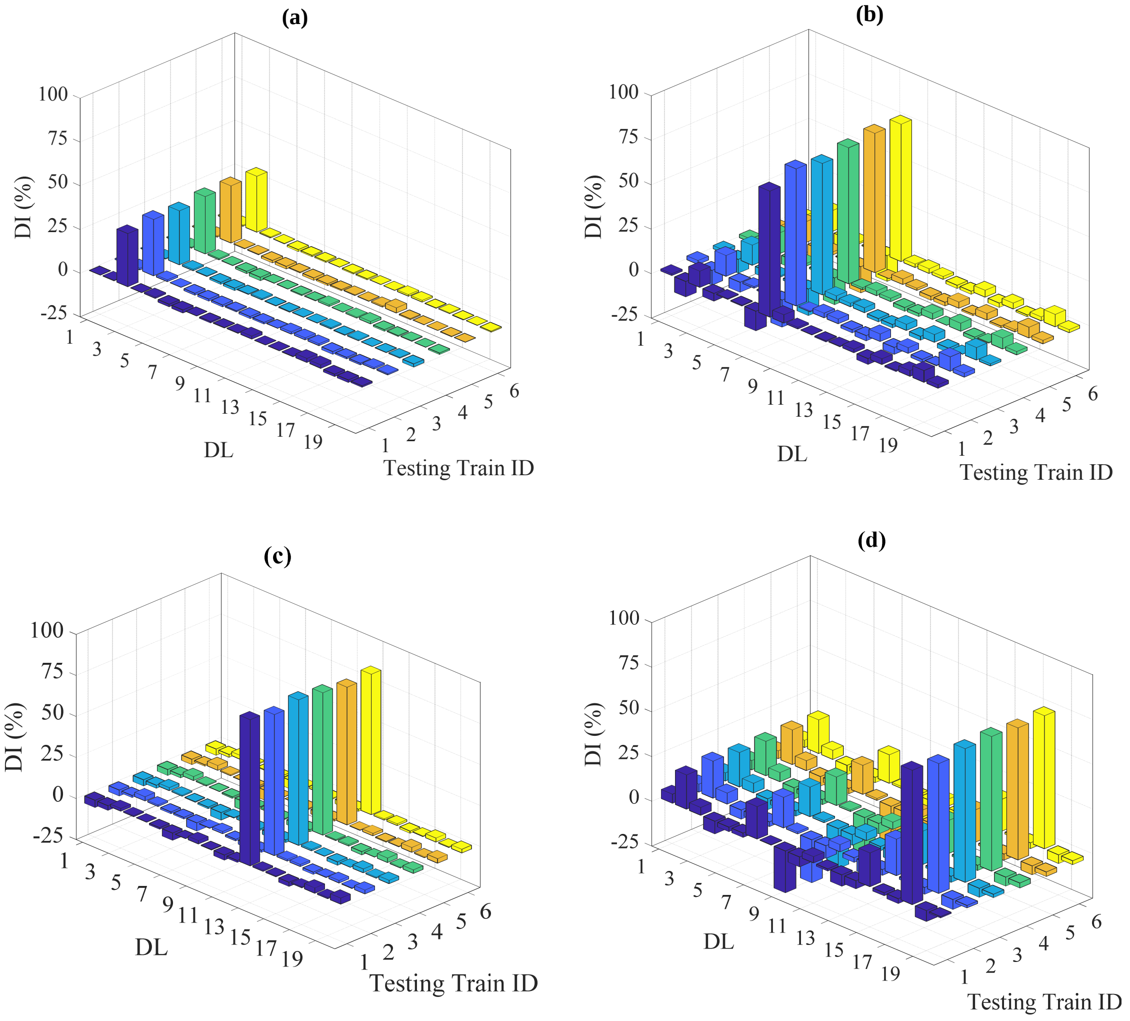


Fig. 58 ANNs testing MU and unknown loading, all testing trains: (a) DI 40% at DL 3 with $\boldsymbol{\mathcal{M}_1}$; (b) DI 60% at DL 8 with $\boldsymbol{\mathcal{M}_2}$; (c) DI 80% at DL 13 with $\boldsymbol{\mathcal{M}_4}$; and (d) DI 100% at DL 18 with $\boldsymbol{\mathcal{M}_5}$.

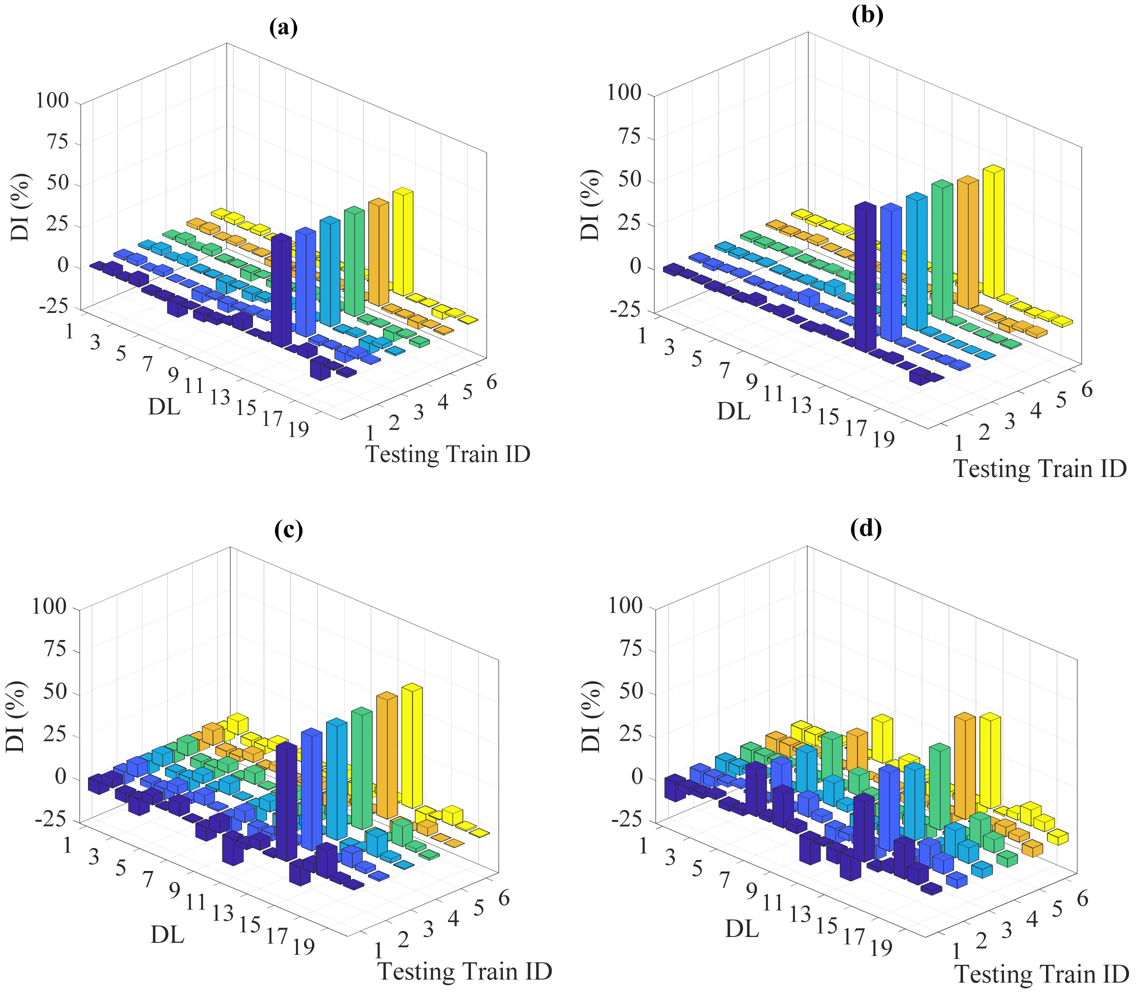


Fig. 59 ANNs testing with MU and unknown loading, DI 70% at DL 15, all testing trains: (a) $\boldsymbol{\mathcal{M}_1}$; (b) $\boldsymbol{\mathcal{M}_2}$; (c) $\boldsymbol{\mathcal{M}_4}$; and (d) $\boldsymbol{\mathcal{M}_5}$.

In this section, a numerical investigation was completed to identify a robust damage feature that is largely independent of MU to help generate damage scenarios from measured strains ($\boldsymbol{\varphi}_{exp}$). The identified damage feature was $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$, which is the difference between healthy and deficient POMs ($\boldsymbol{\varphi}$). ANNs were trained and tested based on $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$ being the damage feature for various MUs and known or unknown loadings. Testing results showed that DL and DI were reasonably detected for different MUs. However, false positives and negatives were observed with magnitudes and locations that varied based on MU and DL. In general, it can be concluded that MUs with stiffer stringer-to-floor beam end fixity ratios tended to underestimate DIs while MUs cases with more flexible stringer ends tended to overestimate the DI.

7.5. Field investigation

In this section, measured strains collected are used to develop $\boldsymbol{\varphi}_{exp}^{h}$ considering the studied bridge is currently in a healthy state (i.e., no observed stringer-to-floor beam connection damage). As a result, $\boldsymbol{\varphi}_{exp}^{h}$ is dependent on recorded strain signal magnitudes and durations, which are a function of train load, length and speed. To reduce $\boldsymbol{\varphi}_{exp}^{h}$ variations associated with train crossing events, data preprocessing was completed to ensure that snapshots used in $\boldsymbol{\varphi}_{exp}^{h}$ calculations were of similar magnitude and feature.

Since the identified damage feature $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$was shown to be independent of the MU level largely, a similar correlation was expected between model $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$ and field $\Delta\boldsymbol{\varphi}_{exp}$; meaning that both $\Delta\boldsymbol{\varphi}$ should be close to one another. As a result, $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$were used to generate damage scenarios as a function of $\boldsymbol{\varphi}_{exp}^{h}$ so that training and testing ANNs using damage scenarios developed from measured strains were possible. A flowchart describing the field investigation procedures is shown in Fig. 60.

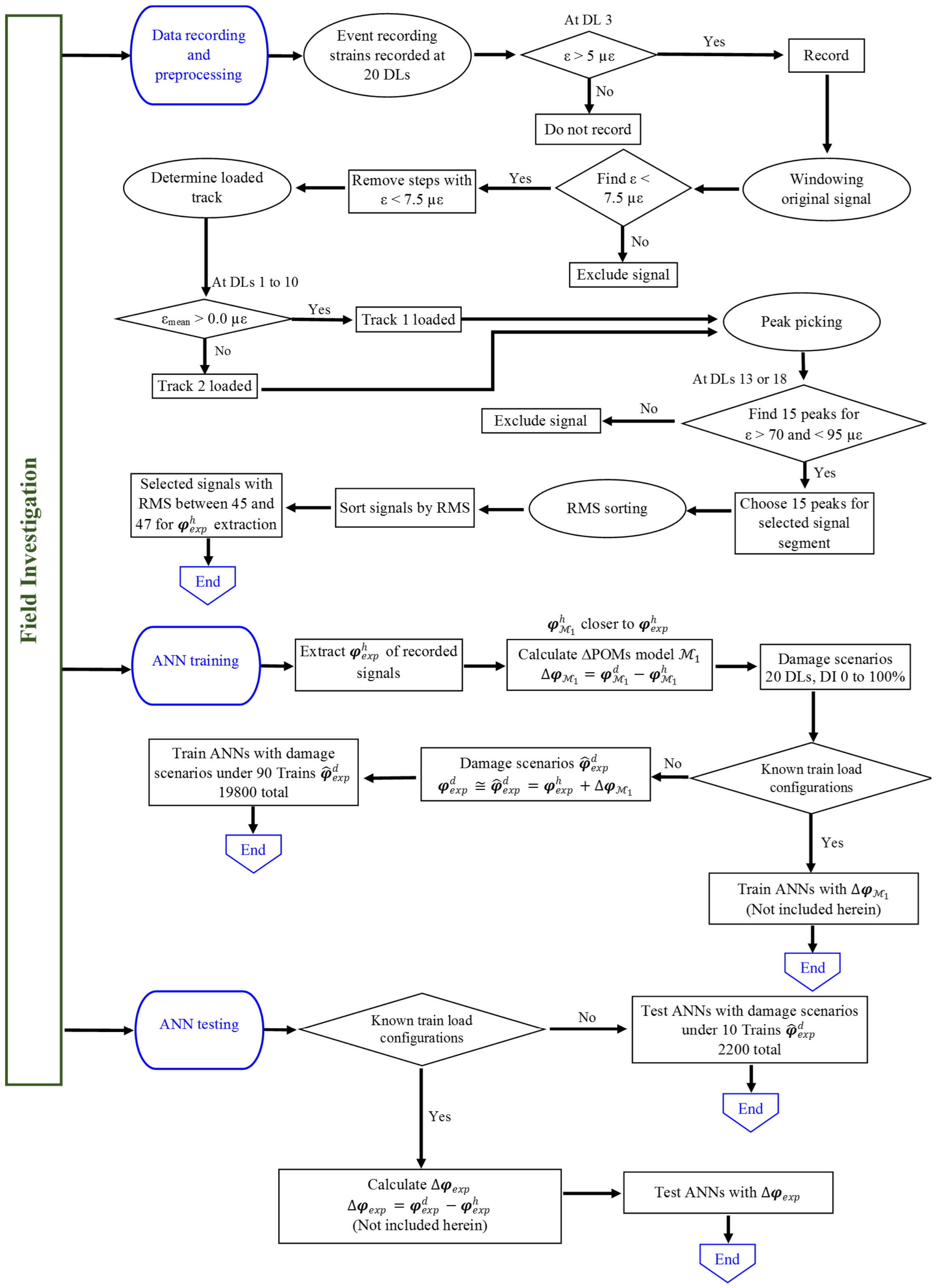

Field Investigation
Data recording and preprocessing
Event recording strains recorded at 20 DLs
At DL 3
$\varepsilon > 5\ \mu\varepsilon$
Yes
Record
No
Do not record
Windowing original signal
Find $\varepsilon <$ 7.5 $\mu\varepsilon$
Yes
No
Exclude signal
Remove steps with $\varepsilon < 7.5\ \mu\varepsilon$
Determine loaded track
At DLs 1 to 10
$\varepsilon_{mean} > 0.0\ \mu\varepsilon$
Yes
Track 1 loaded
No
Track 2 loaded
Peak picking
At DLs 13 or 18
Find 15 peaks for $\varepsilon > 70$ and $< 95\ \mu\varepsilon$
No
Exclude signal
Yes
Choose 15 peaks for selected signal segment
RMS sorting
Sort signals by RMS
Selected signals with RMS between 45 and 47 for $\boldsymbol{\varphi}^h_{exp}$ extraction
End
ANN training
Extract $\boldsymbol{\varphi}^h_{exp}$ of recorded signals
$\boldsymbol{\varphi}^h_{\mathcal{M}_1}$ closer to $\boldsymbol{\varphi}^h_{exp}$
Calculate ΔPOMs model $\mathcal{M}_1$
$\Delta\boldsymbol{\varphi}_{\mathcal{M}_1} = \boldsymbol{\varphi}^d_{\mathcal{M}_1} - \boldsymbol{\varphi}^h_{\mathcal{M}_1}$
Damage scenarios 20 DLs, DI 0 to 100%
Known train load configurations
No
Damage scenarios $\widehat{\boldsymbol{\varphi}}^d_{exp}$
$\boldsymbol{\varphi}^d_{exp} \cong \widehat{\boldsymbol{\varphi}}^d_{exp} = \boldsymbol{\varphi}^h_{exp} + \Delta\boldsymbol{\varphi}_{\mathcal{M}_1}$
Train ANNs with damage scenarios under 90 Trains $\widehat{\boldsymbol{\varphi}}^d_{exp}$ 19800 total
End
Yes
Train ANNs with $\Delta\boldsymbol{\varphi}_{\mathcal{M}_1}$ (Not included herein)
End
ANN testing
Known train load configurations
No
Test ANNs with damage scenarios under 10 Trains $\widehat{\boldsymbol{\varphi}}^d_{exp}$ 2200 total
End
Yes
Calculate $\Delta\boldsymbol{\varphi}_{exp}$
$\Delta\boldsymbol{\varphi}_{exp} = \boldsymbol{\varphi}^d_{exp} - \boldsymbol{\varphi}^h_{exp}$
(Not included herein)
Test ANNs with $\Delta\boldsymbol{\varphi}_{exp}$
End

Fig. 60 Field investigation flowchart.

7.5.1. Monitoring system and data collection

Stringer end strains were measured at instrumented locations depicted in Fig. 8 using 20 strain transducers manufactured by Bridge Diagnostics Inc. (BDI). Strains were recorded when a train crossed the bridge at a sampling rate of 50 Hz. Data preprocessing was completed without prior knowledge of train loads, lengths, location and direction of travel or speed. As a result, loads were non-stationary and unknown.

7.5.2. Data preprocessing and POM extraction

A total of 2951 train passage events were selected, with strains being collected between June 30 and August 31, 2017. The selection of data processing parameters such as strain magnitudes and signal length were based on trial and error approach to minimize $\boldsymbol{\varphi}_{exp}^{h}$ variations and maximize the remained number of training trains. The first preprocessing step involved eliminating time steps before and after the train passage. The second step classified signals based on which track was loaded. This yielded a total of 1471 out of the 2951 train events that crossed the bridge on Track 2. More details about the first two steps can be found in Chapter 5, see (61).

The third step retained signals for similar trains for $\boldsymbol{\varphi}_{exp}^{h}$ calculations to reduce loading configuration effects. To reduce $\boldsymbol{\varphi}_{exp}^{h}$ variations resulting from differing railroad car numbers and train loads, the first 15 strain peaks from a time-history were automatically selected using recorded strains at location 8 when Track 1 was loaded or location 13 when Track 2 loaded (see Fig. 8). Locations 13 and 18 were selected to be at the midspan of the bridge and to present both tracks. Selected locations located toward the midspan of the bridge to reduce the influence of data processing on sensors located toward the bridge ends because the selected signal time-window start, end and length should be the same for all sensors. Train events having strain magnitudes greater than 71 με and less than 95 με at locations 8 or 13 were retained, which resulted in 195 and 490 train events for Track 1 and Track 2, respectively. The MATLAB *findpeaks* function was used to select desired strain peaks within defined thresholds. The developed code excluded the first eight peaks out of the 15 that were eliminated to reduce additional transient variations from the locomotive passage. Train speed and axle spacing effects were reduced by selecting signals having a number of time steps greater than 210 and less than 225, which yielded 59 and 172 train events for Track 1 and Track 2, respectively. The thresholds outlined above were determined using a trial and error approach. Since the number of the retained train events for Track 2 was higher than those for Track 1, Track 2 events results were presented as a larger training set could be selected.

The fourth step ensured that the twenty stringer ends in Fig. 8 were subjected to appreciable strains from selected trains. To do so, retained Track 2 strain time-histories were sorted based on average RMS with strain snapshots for RMS between 44.5 and 47.0 selected that had minimal average RMS variation. After performing the fourth step, a total of 100 train events out of the original 1471 events for Track 2 were retained.

The progression of strain signal filtering steps at location 13 in Fig. 8 and resulting $\boldsymbol{\varphi}_{exp}^{h}$ for Track 2 are shown in Fig. 61. The description of the figure, involved train events and preprocessing steps are listed in Table 4.

Table 4 Measure strain time histories and corresponding $\boldsymbol{\varphi}_{exp}^{h}$

Figure	**Subplot**	**Number of train events**	**Preprocessing steps**
Fig. 61	**a-b**	1471	Trains classified to Track 2
	c-d	490	eliminating railroad car number,length and loading effects
	e-f	172	eliminating train speed and axle spacing effects
	g-h	100	Selected train events after sorting trains by RMS

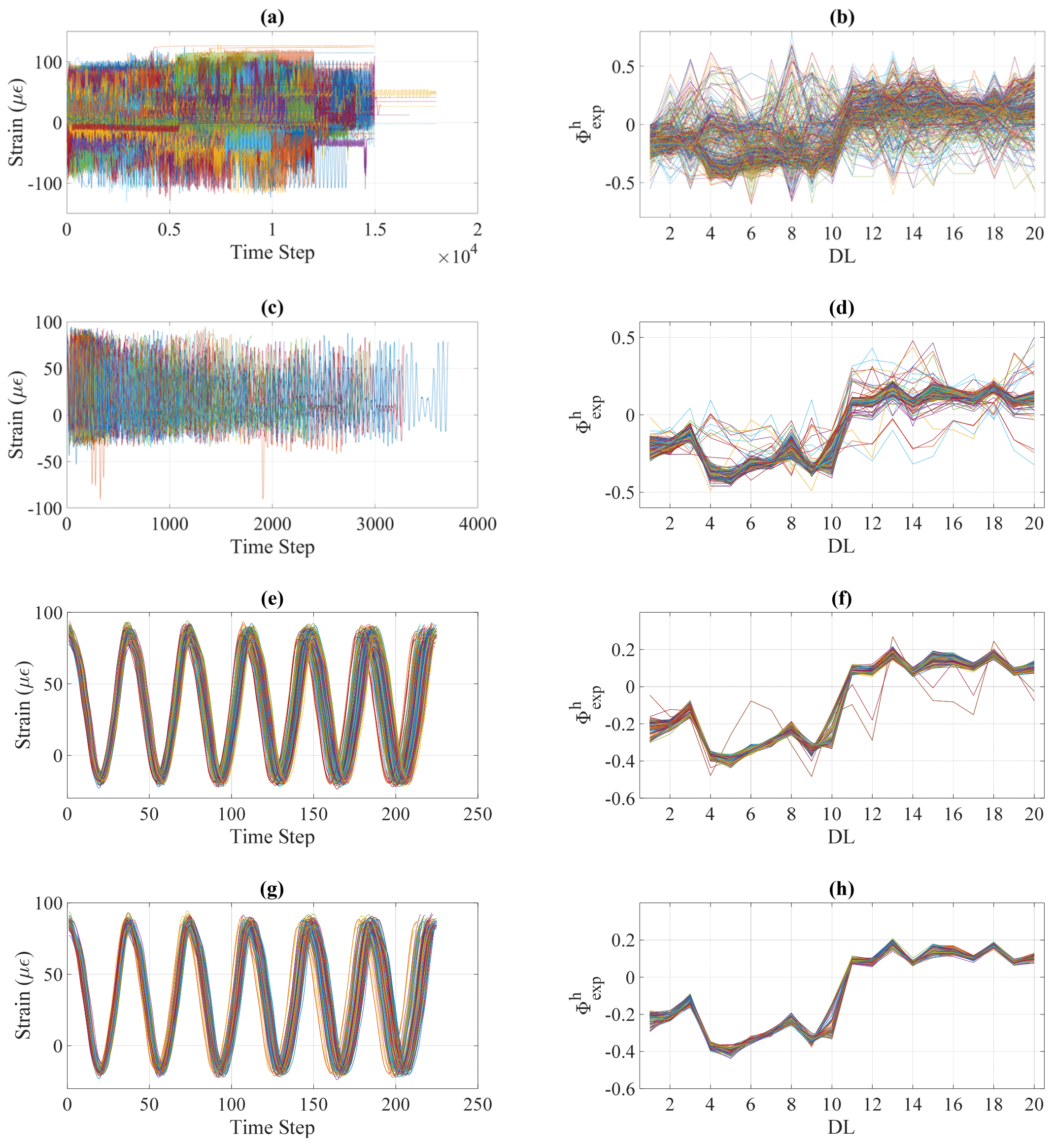


Fig. 61 Track 2 measure strain time-histories $\boldsymbol{\varphi^h_{exp}}$ values at location 13: (a-b) No filtering, 1471 train events; (c-d) railroad cars, overall length and load magnitude filters applied, 490 events; (e-f) speed and axle spacing filters applied, 172 events; and (g-h) average RMS range filter applied, 100 events.

It is observed that strains and $\boldsymbol{\varphi}_{exp}^{h}$ for the selected trains show close agreement. Average of Track 2 strain snapshot matrices RMS values after filtering out loading effects (172 trains events) are shown in Fig. 62 with selected trains highlighted.

It is important to note that changing data processing parameters would change the features and number of retained training and testing trains. However, the same criteria should be used for data processing when using the developed framework in assessing bridge health. For other bridges, new data processing parameters should be determined from measured responses.

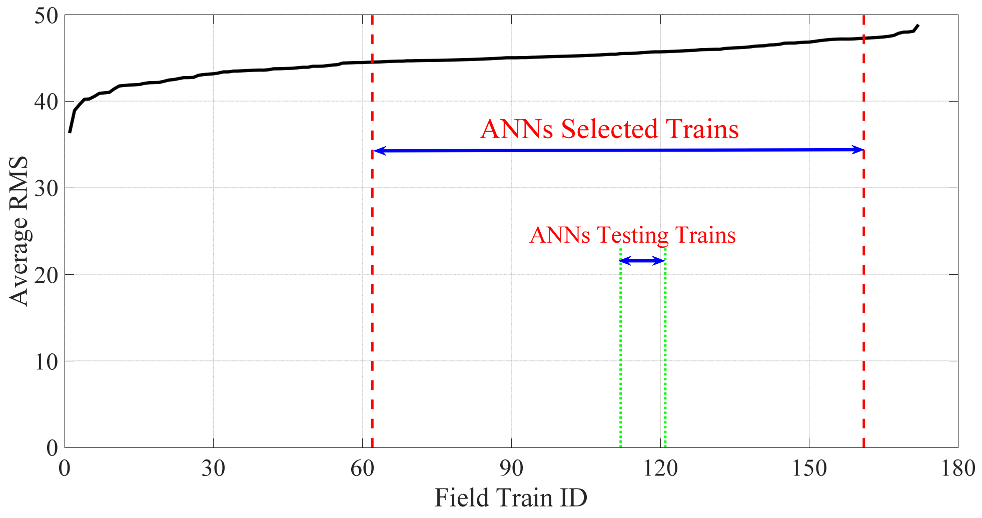


Fig. 62 Strain RMS for Track 2 Trains.

7.5.3. ANNs training and testing data sets

As shown in Fig. 62, the selected 100 train IDs were from 62 to 161, with 90 trains used for ANN training and ten trains used for testing. Testing IDs were from 112 to 121, with average RMS located in the middle of the selected range. Imposing damage to the actual structure was not possible and, as a result, $\boldsymbol{\varphi}_{exp}^{h}$ was calculated for the 100 train event strain snapshot matrices and used to develop estimated damage scenarios ($\widehat{\boldsymbol{\varphi}}_{exp}^{d}$) as a function of $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$. It is important to reiterate that $\Delta\boldsymbol{\varphi}_{\mathcal{M}_i}$ could be assumed to be very close to $\Delta\boldsymbol{\varphi}_{exp}$. Based on these findings, $\widehat{\boldsymbol{\varphi}}_{exp}^{d}$ was calculated as:

$$\boldsymbol{\varphi}_{exp}^{d} \cong \widehat{\boldsymbol{\varphi}}_{exp}^{d} = \boldsymbol{\varphi}_{exp}^{h} + \Delta\boldsymbol{\varphi}_{\mathcal{M}_1} \tag{19}$$

where $\Delta\boldsymbol{\varphi}_{\mathcal{M}_1}$was selected among considered MU cases because $\boldsymbol{\varphi}_{\mathcal{M}_1}^{h}$was closer to $\boldsymbol{\varphi}_{exp}^{h}$ when compared against the other MU cases, as shown in Fig. 63. A set of example comparisons between $\widehat{\boldsymbol{\varphi}}_{exp}^{d}$ and $\boldsymbol{\varphi}_{exp}^{h}$ is shown in Fig. 64 for DIs of 40, 60, 80 and 100% simulated at DL 3, 8, 13 and 18 for trains 61, 86, 111 and 136. The comparison showed significant variation for simulated DLs in $\widehat{\boldsymbol{\varphi}}_{exp}^{d}$ when compared against $\boldsymbol{\varphi}_{exp}^{h}$, with minor variations being observed at other healthy DLs. The figure also shows that the magnitude of the variation increases as DI increases.

For the 100 train events included in ANN testing and training, a total of 22000 damage scenarios were generated using $\Delta\boldsymbol{\varphi}_{\mathcal{M}_1}$, with 19800 used for training the ANNs. CPU time used for training increased to 4800 minutes when compared against ANNs training completed in the previous chapters.

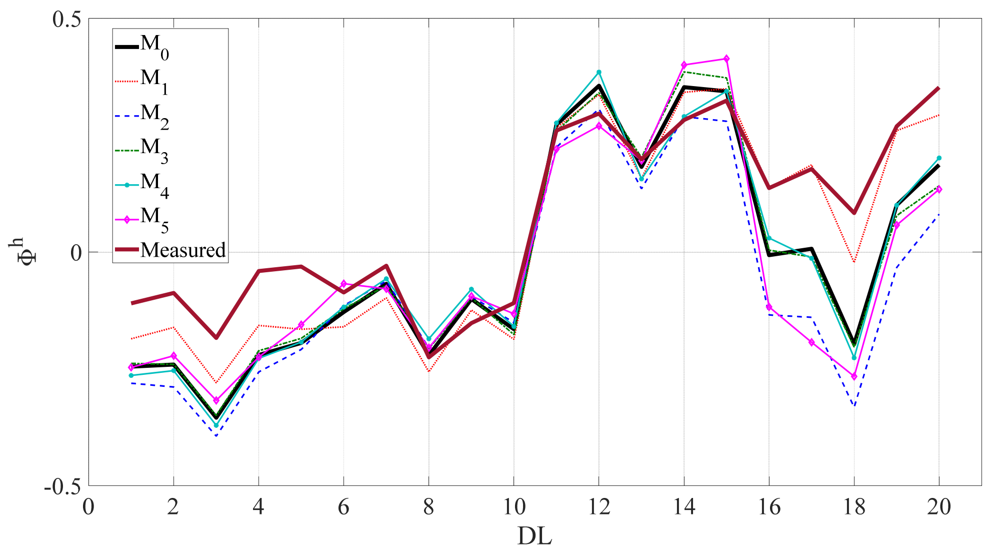


Fig. 63 $\boldsymbol{\varphi}^{\boldsymbol{h}}$ mean comparisons, measured and MU cases.

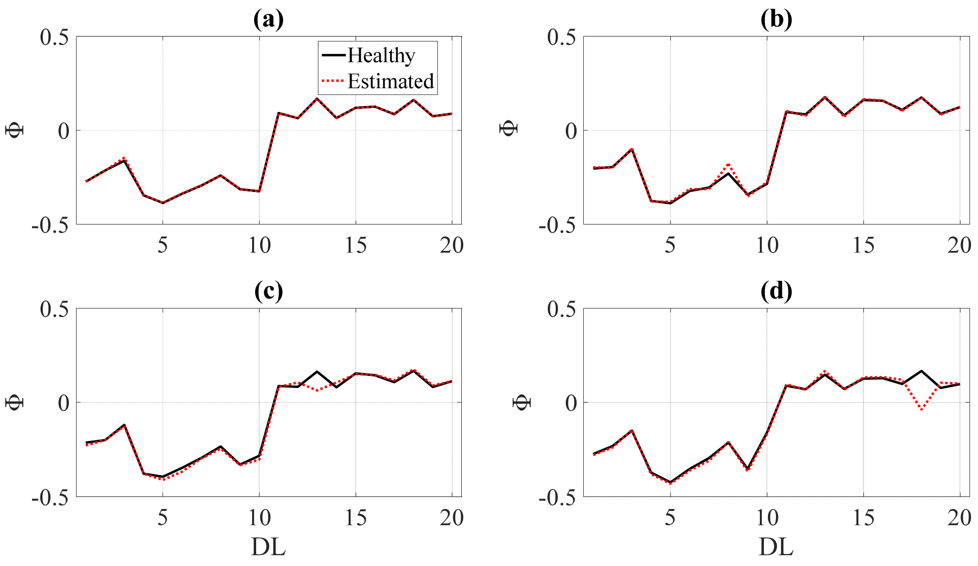


Fig. 64 $\widehat{\boldsymbol{\varphi}}_{\boldsymbol{exp}}^{\boldsymbol{d}}$ comparisons to $\boldsymbol{\varphi}_{\boldsymbol{exp}}^{\boldsymbol{h}}$ for: (a) DI 40% at DL 3, Train 61; (b) DI 60% at DL 8, Train 86; (c) DI 80% at DL 13, Train 111; and (d) DI 100% at DL 18, Train 136.

7.5.4. Damage identification results

As mentioned earlier, trains 112 to 121 were arbitrarily selected to validate ANN effectiveness. For the ten events observed, false positives were between 2 and 15%, which was considered low when compared to the DI of 0%. See Fig. 65. ANN testing for DIs of 40, 60, 80 and 100% at DL 3, 8, 13 and 18 are shown in Fig. 66 for all selected events. DI and DL were detected with acceptable accuracy for considered damage scenarios and testing events; however, detected DI accuracy and numbers of false positives varied based on DL and DI. It was observed that false-positive existence and magnitudes increased with lower DI and reduced as DI increased. The results shown in Fig. 66 are summarized in

Table 5.

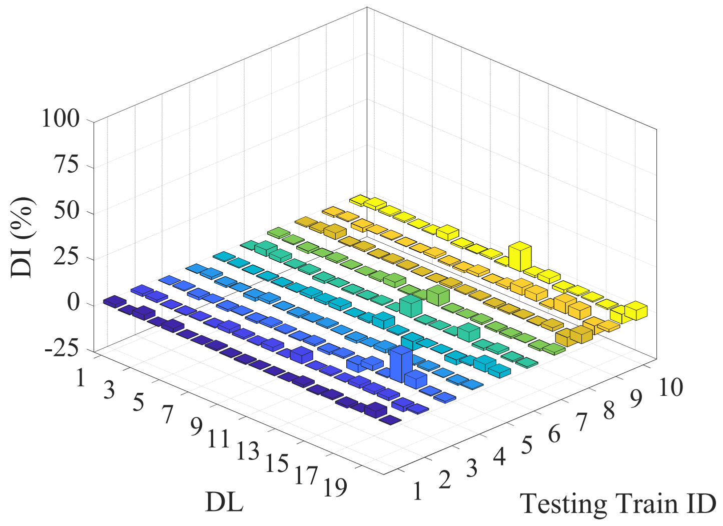


Fig. 65 ANNs field testing, all testing trains, DI 0%.

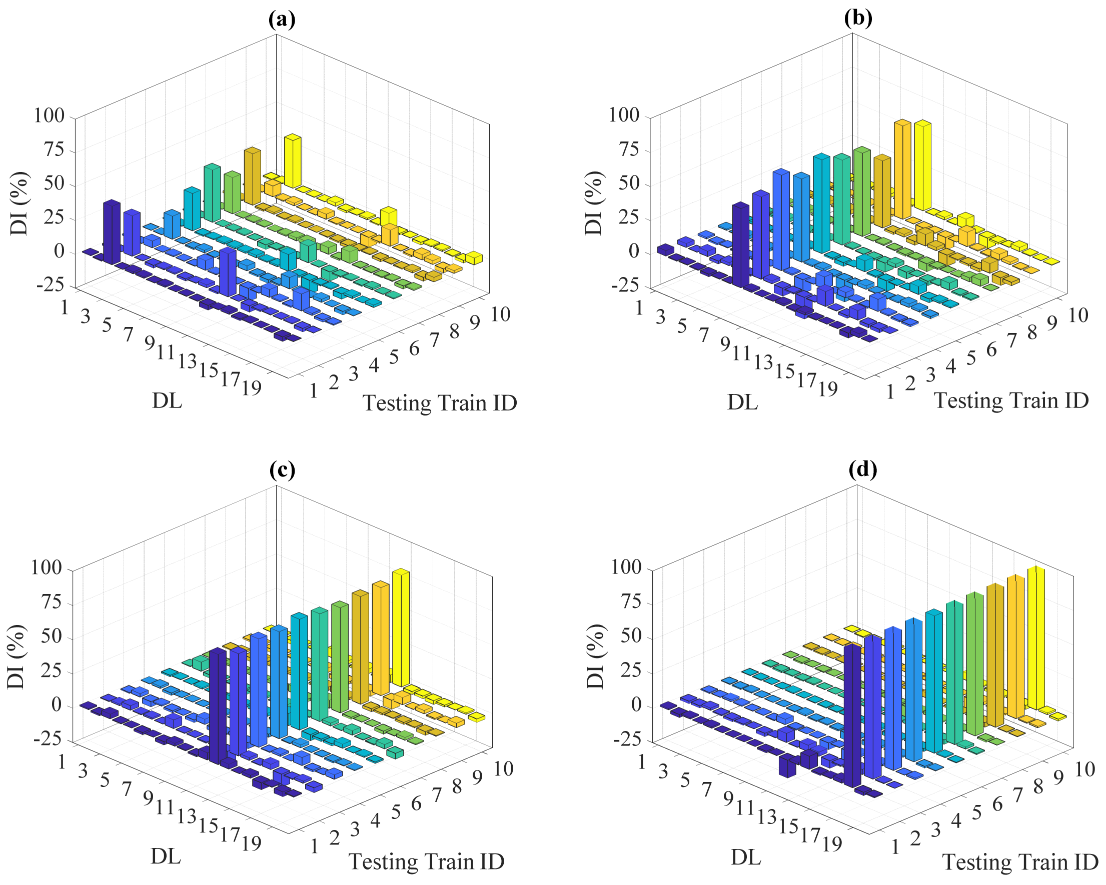


Fig. 66 ANNs field testing, all testing trains: (a) DI 40% at DL 3; (b) DI 60% at DL 8; (c) DI 80% at DL 13; and (d) DI 100% at DL 18.

Table 5 Field testing results

Figure	**Subplot**	**Simulated DI and DL**	**Detected DI at simulated DL**	**False positives and negatives**
Fig. 66	**a**	40% at DL 3	17 to 43%	32%
	b	60% at DL 8	49 to 70%	11 and 15%
	c	80% at DL 13	74 to 82%	12 and 10%
	d	100% at DL 18	100 to 102%	10 and 13%

In this section, a field investigation was completed that involved preprocessing measured strains to reduce $\boldsymbol{\varphi}_{exp}^{h}$ variations associated with variations in loading configurations, training/testing the ANNs with damage scenarios generated from $\boldsymbol{\varphi}_{exp}^{h}$

and $\Delta\boldsymbol{\varphi}_{\mathcal{M}_1}$. The results showed that DI and DL were detected with an acceptable accuracy under the included field testing train events.

The influence of environmental variations on the proposed framework was not investigated in the current study; however, strains were recorded for a short time-window under train passages, which was believed to minimize environmental variation effects. At the early stages of the SHM data recordings, strains were collected continuously without the use of Event Recording. Fig. 67 shows continuously measured strain responses at DLs 3, 8, 13, and 18 for 270 hours time-window with strain cycles showing weather heating and cooling effects identified. The continuous measurement was not implemented because these strain cycles are shown to present the sensors' material behavior, not bridge behavior. Sensors material deform with the weather heating and cooling cycles dependent from the structural elements, which resulted in such high strain cycles.

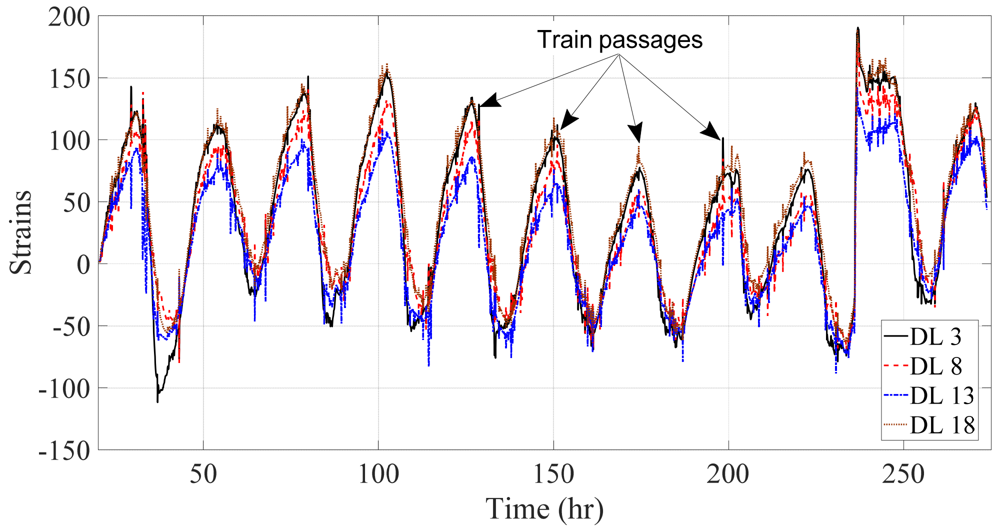


Fig. 67 Strain responses measured continuously at DLs 3, 8, 13 and 18.

7.6. Conclusions

This chapter used a hybrid experimental-numerical approach to further improve the POD-ANN framework so it can maintain damage detection accuracy in the presence of modeling uncertainties. The numerical portion of the study identified a robust damage feature, $\Delta\boldsymbol{\varphi}$, that is largely independent of MU level.

The experimental portion of the study generated damage scenarios based on $\Delta\boldsymbol{\varphi}_{\mathcal{M}_1}$, which represents a robust damage feature that is largely independent of MU level, to train ANNs under passage of 90 train events (Chapter 3) whose strain responses were recorded. ANNs were tested using an additional ten train passage events. Acceptable DI and DL detection was observed; however, DI detection accuracy increased as simulated DI also increased. In addition, magnitude and existence of false positives was influenced by both DI and DL, with higher false positives observed for lower DIs.

It is important to note that, when MU is considered and if measuring train loading configurations is possible, $\Delta\boldsymbol{\varphi}$ should be used to train and test the developed POD-ANN framework to improve damage detection accuracy. When measuring train configurations and loads are challenging, $\widehat{\boldsymbol{\varphi}}_{exp}^{d}$ (estimated damage scenario POMs based on field measurements) should be used to train and test the framework. Lower damage detection accuracy would be expected when compared against the known loading scenario, however.

To reduce the influence of modeling uncertainty on the damage detection accuracy, it is recommended that an analytical model of the monitored bridge is developed and that the model is calibrated against full-scale testing data.

Chapter 8 : Conclusions

An automatic, hybrid experimental-numerical SHM framework was developed and evaluated using a combination of numerical and graphical techniques. Adopted techniques used Proper Orthogonal Decomposition (POD) and Proper Orthogonal Modes (POMs) in conjunction with Artificial Neural Network (ANNs) to locate and identify damage features. The framework was developed analytically and initially validated using measured strains from a SHM system deployed on a steel truss span. Framework robustness was then examined by investigating effectiveness for a different bridge system and using different sensor types and arrangements. Modeling uncertainties were also examined with ΔPOMs being identified as a robust damage feature, one that is largely independent of modeling uncertainty.

It was concluded that, for the bridge configurations that were studied:

- The developed hybrid experimental-numerical approach for damage detection was robust over a range of damage locations and intensities;
- The proposed method successfully detects damage using strain outputs induced by unknown, nonstationary external inputs;
- Strain, rotation or displacement sensors could be used to locate damage;
- Automated data cleansing prior to POM ($\boldsymbol{\varphi}$) extraction was necessary to reduce discrepancies caused by nonstationary inputs;
- Automated peak picking reduced $\boldsymbol{\varphi}$ discrepancy associated with speed variations when selecting data for the snapshot matrix used to calculate $\boldsymbol{\varphi}$;

- The method is robust enough to predict damage supplied from highly noisy signals; and
- $\Delta\boldsymbol{\varphi}$ is largely independent of MU and can be used as a robust damage feature.

Environmental variation influence was not included in the current research work; however, since the framework implements measured responses, environmental effects could be easily incorporated in training procedures when data are available. It is also important to mention that strain signals were recorded during a short time-window during train passage, which was believed to reduce environmental variations effects.

To further enhance study findings, environmental variation effects on the developed POD-ANN framework should be investigated. In addition, the developed hybrid experimental-analytical framework should be further validated by imposing damage to in-service bridges.

The framework was developed with the ultimate goal of detecting riveted steel railway bridge fatigue damage. Conclusions are limited to the truss and plate girder spans with damage was simulated one location at a time. If fatigue damage is expected to happen at multiple locations simultaneously, the training set should include multiple damage location scenarios. It is also important to note that, for unstable fatigue crack growth where the structure does not show significant change in its stiffness prior to fracture, the framework may not be capable of detecting damage with sufficient advance warning.

The proposed framework is believed to be applicable to other bridges by following the steps outlined in Fig. 60 with the following additional recommendations being:

1. Developing a calibrated model for the investigated bridge under routine loading.
2. Proposing an optimal instrumentation plan that uses robust sensors near suspected damage prone locations.
3. Determining a suitable strain threshold for data processing and RMS range for the investigated bridge to provide a reliable ANN training set.

References

1. Phares BM, Rolander DD, Graybeal BA, Washer GA. Reliability of Visual Bridge Inspection. Public Roads. 2001;64(5).

2. Nasrollahi M, Provines J, Connor R, Washer G. New Framework for Risk-Based Inspection of Highway Bridges. Journal of Bridge Engineering. 2016 Apr 1,;21(4):4015077.

3. Vagnoli M, Remenyte-Prescott R, Andrews J. Railway bridge structural health monitoring and fault detection: State-of-the-art methods and future challenges. Structural Health Monitoring. 2018;17(4):971-1007.

4. Brownjohn JM, De Stefano A, Xu Y, Wenzel H, Aktan AE. Vibration-based monitoring of civil infrastructure: challenges and successes. Journal of Civil Structural Health Monitoring. 2011;1(3-4):79-95.

5. Webb GT, Vardanega PJ, Middleton CR. Categories of SHM deployments: technologies and capabilities. J Bridge Eng. 2014;20(11):04014118.

6. Carden EP, Fanning P. Vibration based condition monitoring: a review. Structural health monitoring. 2004;3(4):355-77.

7. Azarbayejani M, El-Osery AI, Choi KK, Taha MR. A probabilistic approach for optimal sensor allocation in structural health monitoring. Smart Mater Struct. 2008;17(5):055019.

8. Meo M, Zumpano G. On the Optimal Sensor Placement Techniques for a Bridge Structure. Engineering Structures. 2005 Aug;27(10):1488-97.

9. Liu W, Gao W, Sun Y, Xu M. Optimal Sensor Placement for Spatial Lattice Structure Based on Genetic Algorithms. J Sound Vibrat. 2008;317(1):175-89.

10. Bellino A, Fasana A, Garibaldi L, Marchesiello S. PCA-based detection of damage in time-varying systems. Mechanical Systems and Signal Processing. 2010;24(7):2250-60.

11. Ni YQ. Structural health monitoring of cable-supported bridges based on vibration measurements. Proceedings of the 9th International Conference on Structural Dynamics, EURODYN; ; 2014.

12. Zhou XT, Ni YQ, Zhang FL. Damage localization of cable-supported bridges using modal frequency data and probabilistic neural network. Mathematical Problems in Engineering. 2014;2014.

13. Kim C, Morita T, Oshima Y, Sugiura K. A Bayesian approach for vibration-based long-term bridge monitoring to consider environmental and operational changes. Smart Structures and Systems. 2015;15(2):395-408.

14. Cunha A, Caetano E, Ribeiro P, Müller G. Vibration-based SHM of a centenary bridge: a comparative study between two different automated OMA techniques. preservation. 2011;1:12.

15. Guo W, Orcesi AD, Cremona CF, Santos JP, Yang SZ, Dieleman L. A vibration-based framework for structural health monitoring of railway bridges. Life-Cycle and Sustainability of Civil Infrastructure Systems: Proceedings of the Third International Symposium on Life-Cycle Civil Engineering (IALCCE'12), Vienna, Austria, October 3-6, 2012; CRC Press; 2012.

16. Moaveni B, He X, Conte JP, Restrepo JI, Panagiotou M. System identification study of a 7-story full-scale building slice tested on the UCSD-NEES shake table. J Struct Eng. 2010;137(6):705-17.

17. Yan A, Kerschen G, De Boe P, Golinval J. Structural damage diagnosis under varying environmental conditions—part I: a linear analysis. Mechanical Systems and Signal Processing. 2005;19(4):847-64.

18. Vanlanduit S, Parloo E, Cauberghe B, Guillaume P, Verboven P. A robust singular value decomposition for damage detection under changing operating conditions and structural uncertainties. J Sound Vibrat. 2005;284(3-5):1033-50.

19. Shane C, Jha R. Proper orthogonal decomposition based algorithm for detecting damage location and severity in composite beams. Mechanical systems and signal processing. 2011;25(3):1062-72.

20. Eftekhar Azam S, Mariani S, Attari N. Online damage detection via a synergy of proper orthogonal decomposition and recursive Bayesian filters. Nonlinear Dyn. 2017;89(2):1489-511.

21. Eftekhar Azam S. Online damage detection in structural systems: applications of proper orthogonal decomposition, and kalman and particle filters. . 2014.

22. Lanata F, Del Grosso A. Damage detection and localization for continuous static monitoring of structures using a proper orthogonal decomposition of signals. Smart Mater Struct. 2006;15(6):1811.

23. Jin C, Jang S, Sun X, Li J, Christenson R. Damage detection of a highway bridge under severe temperature changes using extended Kalman filter trained neural network. Journal of Civil Structural Health Monitoring. 2016;6(3):545-60.

24. Zang C, Friswell MI, Imregun M. Structural damage detection using independent component analysis. Structural Health Monitoring. 2004;3(1):69-83.

25. Dworakowski Z, Dragan K, Stepinski T. Artificial neural network ensembles for fatigue damage detection in aircraft. J Intell Mater Syst Struct. 2017;28(7):851-61.

26. Gu J, Gul M, Wu X. Damage detection under varying temperature using artificial neural networks. Structural Control and Health Monitoring. 2017;24(11):e1998.

27. Sbarufatti C. Optimization of an artificial neural network for fatigue damage identification using analysis of variance. Structural Control and Health Monitoring. 2017;24(9):e1964.

28. Moaveni B, Conte JP, Hemez FM. Uncertainty and sensitivity analysis of damage identification results obtained using finite element model updating. Computer-Aided Civil and Infrastructure Engineering. 2009;24(5):320-34.

29. Moaveni B, Barbosa AR, Conte JP, Hemez FM. Uncertainty analysis of system identification results obtained for a seven-story building slice tested on the UCSD-NEES shake table. Structural Control and Health Monitoring. 2014;21(4):466-83.

30. Papadimitriou C, Papadioti D. Component mode synthesis techniques for finite element model updating. Comput Struct. 2013;126:15-28.

31. Song W. Generalized minimum variance unbiased joint input-state estimation and its unscented scheme for dynamic systems with direct feedthrough. Mechanical Systems and Signal Processing. 2018;99:886-920.

32. Moaveni B, Behmanesh I. Effects of changing ambient temperature on finite element model updating of the Dowling Hall Footbridge. Eng Struct. 2012;43:58-68.

33. Rakoczy AM. Development of system reliability models for railway bridges. The University of Nebraska-Lincoln; 2012.

34. Haghani R, Al-Emrani M, Heshmati M. Fatigue-Prone Details in Steel Bridges. Buildings. 2012 Nov 12,;2(4):456-76.

35. Rageh A, Linzell D. Optimized Health Monitoring Plans for a Steel, Double-Track Railway Bridge [dissertation]. University of Nebraska-Lincoln; 2018.

36. Imam B, Righiniotis TD, Chryssanthopoulos MK. Fatigue Assessment of Riveted Railway Bridges. Steel Structures. 2005;5(5):485-94.

37. Al-Emrani M. Fatigue Performance of Stringer-to-Floor-Beam Connections in Riveted Railway Bridges. Journal of Bridge Engineering. 2005 Mar;10(2):179-85.

38. Al-Emrani M, Akesson B, Kliger R. Overlooked Secondary Effects in Open-Deck Truss Bridges. Struct Eng Int. 2004;14(4):307-12.

39. Chotickai P, Kanchanalai T. Field Testing and Performance Evaluation of a Through-Plate Girder Railway Bridge. Transportation Research Record: Journal of the Transportation Research Board. 2010(2172):132-41.

40. Imam B, Righiniotis TD, Chryssanthopoulos MK. Connection Fixity Effects on Stress Histories in Riveted Rail Bridges. Bridge Maintenance, Safety, Management and Cost; ; 2004.

41. Spyrakos CC, Raftoyiannis IG, Ermopoulos JC. Condition Assessment and Retrofit of a Historic Steel-Truss Railway Bridge. Journal of Constructional Steel Research. 2004;60(8):1213-25.

42. Tobias DH, Foutch DH, Choros J. Investigation of an Open Deck Through-Truss Railway Bridge: Work Train Tests. 1993.

43. Farrar CR, Worden K. Structural health monitoring: a machine learning perspective. John Wiley & Sons; 2012.

44. Shi J, Spencer Jr BF, Chen S. Damage detection in shear buildings using different estimated curvature. Structural Control and Health Monitoring. 2018;25(1):e2050.

45. Roy K. Structural damage identification using mode shape slope and curvature. J Eng Mech. 2017;143(9):04017110.

46. Moaveni B, He X, Conte JP, Restrepo JI. Damage identification study of a seven-story full-scale building slice tested on the UCSD-NEES shake table. Struct Saf. 2010;32(5):347-56.

47. Taciroglu E, Ghahari SF, Abazarsa F. Efficient model updating of a multi-story frame and its foundation stiffness from earthquake records using a timoshenko beam model. Soil Dyn Earthquake Eng. 2017;92:25-35.

48. Tan ZX, Thambiratnam DP, Chan T, Razak HA. Detecting damage in steel beams using modal strain energy based damage index and Artificial Neural Network. Eng Failure Anal. 2017;79:253-62.

49. Ashory M, Ghasemi-Ghalebahman A, Kokabi M. An efficient modal strain energy-based damage detection for laminated composite plates. Advanced Composite Materials. 2017:1-16.

50. Worden K, Manson G, Fieller NR. Damage detection using outlier analysis. J Sound Vibrat. 2000;229(3):647-67.

51. Achenbach JD. Structural health monitoring–What is the prescription? Mech Res Commun. 2009;36(2):137-42.

52. O'Connor SM, Zhang Y, Lynch JP, Ettouney MM, Jansson PO. Long-term performance assessment of the Telegraph Road Bridge using a permanent wireless monitoring system and automated statistical process control analytics. Structure and Infrastructure Engineering. 2017;13(5):604-24.

53. Thiene M, Galvanetto U. Impact location in composite plates using proper orthogonal decomposition. Mech Res Commun. 2015;64:1-7.

54. Kim Y, Eun H. Comparison of Damage Detection Methods Depending on FRFs within Specified Frequency Ranges. Advances in Materials Science and Engineering. 2017;2017.

55. Ruotolo R, Surace C. Using SVD to detect damage in structures with different operational conditions. J Sound Vibrat. 1999;226(3):425-39.

56. Galvanetto U, Violaris G. Numerical investigation of a new damage detection method based on proper orthogonal decomposition. Mechanical Systems and Signal Processing. 2007;21(3):1346-61.

57. Xia Q, Tian YD, Zhu XW, Xu DW, Zhang J. Structural damage detection by principle component analysis of long-gauge dynamic strains. Struct Eng Mech. 2015;54(2):379-92.

58. Azam SE, Mariani S. Investigation of computational and accuracy issues in POD-based reduced order modeling of dynamic structural systems. Eng Struct. 2013;54:150-67.

59. Dervilis N, Choi M, Taylor SG, Barthorpe RJ, Park G, Farrar CR, et al. On damage diagnosis for a wind turbine blade using pattern recognition. J Sound Vibrat. 2014;333(6):1833-50.

60. Ou Y, Chatzi EN, Dertimanis VK, Spiridonakos MD. Vibration-based experimental damage detection of a small-scale wind turbine blade. Structural Health Monitoring. 2017;16(1):79-96.

61. Rageh A, Linzell DG, Azam SE. Automated, strain-based, output-only bridge damage detection. Journal of Civil Structural Health Monitoring. 2018;8(5):833-46.

62. Eftekhar Azam S, Rageh A, Linzell D. Damage detection in structural systems utilizing artificial neural networks and proper orthogonal decomposition. Structural Control and Health Monitoring:e2288.

63. Behmanesh I, Moaveni B, Papadimitriou C. Probabilistic damage identification of a designed 9-story building using modal data in the presence of modeling errors. Eng Struct. 2017;131:542-52.

64. Law SS, Li XY, Lu ZR. Structural damage detection from wavelet coefficient sensitivity with model errors. J Eng Mech. 2006;132(10):1077-87.

65. Astroza R, Alessandri A. Effects of model uncertainty in nonlinear structural finite element model updating by numerical simulation of building structures. Structural Control and Health Monitoring. 2019;26(3):e2297.

66. Chatzi EN, Smyth AW, Masri SF. Experimental application of on-line parametric identification for nonlinear hysteretic systems with model uncertainty. Struct Saf. 2010;32(5):326-37.

67. Lee JJ, Lee JW, Yi JH, Yun CB, Jung HY. Neural networks-based damage detection for bridges considering errors in baseline finite element models. J Sound Vibrat. 2005;280(3-5):555-78.

68. Astroza R, Alessandri A, Conte JP. A dual adaptive filtering approach for nonlinear finite element model updating accounting for modeling uncertainty. Mechanical Systems and Signal Processing. 2019;115:782-800.

69. Hu W, Cunha Á, Caetano E, Rohrmann R, Said S, Teng J. Comparison of different statistical approaches for removing environmental/operational effects for massive data continuously collected from footbridges. Structural Control and Health Monitoring. 2017;24(8).

70. Silva M, Santos A, Santos R, Figueiredo E, Sales C, Costa JC. Deep principal component analysis: An enhanced approach for structural damage identification. Structural Health Monitoring. 2018:1475921718799070.

71. Ghahari SF, Abazarsa F, Taciroglu E. Blind modal identification of non-classically damped structures under non-stationary excitations. Structural Control and Health Monitoring. 2017;24(6).

72. Abazarsa F, Nateghi F, Ghahari SF, Taciroglu E. Extended blind modal identification technique for nonstationary excitations and its verification and validation. J Eng Mech. 2015;142(2):04015078.

73. Pearson K. LIII. On lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science. 1901;2(11):559-72.

74. Karhunen K. Über lineare Methoden in der Wahrscheinlichkeitsrechnung. Sana; 1947.

75. Klema V, Laub A. The singular value decomposition: Its computation and some applications. IEEE Transactions on automatic control. 1980;25(2):164-76.

76. Liang YC, Lee HP, Lim SP, Lin WZ, Lee KH, Wu CG. Proper orthogonal decomposition and its applications—Part I: Theory. J Sound Vibrat. 2002;252(3):527-44.

77. Feeny BF, Kappagantu R. On the physical interpretation of proper orthogonal modes in vibrations. J Sound Vibrat. 1998;211(4):607-16.

78. Kerschen G, Golinval J. Physical interpretation of the proper orthogonal modes using the singular value decomposition. J Sound Vibrat. 2002;249(5):849-65.

79. Feeny BF. On proper orthogonal co-ordinates as indicators of modal activity. J Sound Vibrat. 2002;255(5):805-18.

80. Georgiou I. Advanced proper orthogonal decomposition tools: using reduced order models to identify normal modes of vibration and slow invariant manifolds in the dynamics of planar nonlinear rods. Nonlinear Dyn. 2005;41(1-3):69-110.

81. Kerschen G, Golinval J, Vakakis AF, Bergman LA. The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: an overview. Nonlinear Dyn. 2005;41(1-3):147-69.

82. North GR. Empirical orthogonal functions and normal modes. J Atmos Sci. 1984;41(5):879-87.

83. Amezquita-Sanchez JP, Adeli H. Feature extraction and classification techniques for health monitoring of structures. Scientia Iranica.Transaction A, Civil Engineering. 2015;22(6):1931.

84. Bishop CM. Pattern recognition and machine learning, 2006. 대한토목학회지. 2012;60(1):78.

85. MacKay DJ. Bayesian interpolation. Neural Comput. 1992;4(3):415-47.

86. Nguyen D, Widrow B. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. Neural Networks, 1990., 1990 IJCNN International Joint Conference on; IEEE; 1990.

87. Waszczyszyn Z. Fundamentals of artificial neural networks. In: Neural networks in the analysis and design of structures. Springer; 1999. p. 1-51.

88. Foresee FD, Hagan MT. Gauss-Newton approximation to Bayesian learning. Proceedings of the 1997 international joint conference on neural networks; Piscataway: IEEE; 1997.

89. Moler C, Little J, Bangert S. Matlab User's Guide-The Language of Technical Computing. The MathWorks, Sherborn, Mass. 2001.

90. Howard D, Mark B, Martin H. Neural network toolbox for use with MATLAB. User's Guide Version. 1998;3.

91. Buljak V, Maier G. Identification of residual stresses by instrumented elliptical indentation and inverse analysis. Mech Res Commun. 2012;41:21-9.

92. Thiene M, Galvanetto U. Impact location in composite plates using proper orthogonal decomposition. Mech Res Commun. 2015;64:1-7.

93. Azam SE, Mariani S. Online damage detection in structural systems via dynamic inverse analysis: A recursive Bayesian approach. Eng Struct. 2018;159:28-45.

Appendix 1: POD-ANN Framework Application to Varying Spans, Responses and Instrumentations

In this appendix, more damage detection results of the research completed in Chapter 6 are shown. The appendix contains results for:

(i) The truss and plate girder spans.

(ii) The proposed instrumentation plans in Chapter 6.

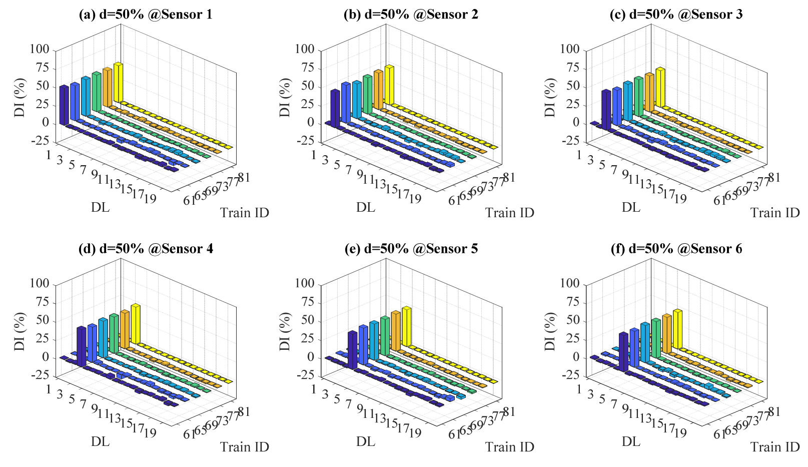
(iii) Various structural responses including strains, rotation, acceleration and displacement.

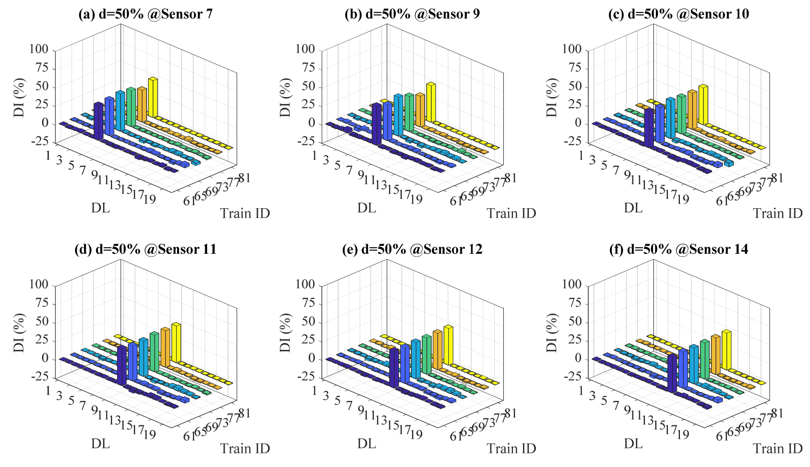
The appendix also includes other analyses completed to show the effect on changing the span-length on the developed damage detection framework.

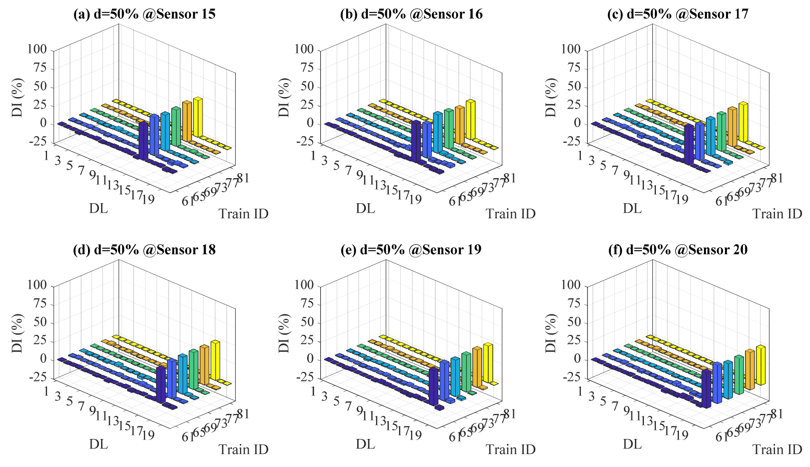
1. Truss span:

1.1 Instrumentation plan 1

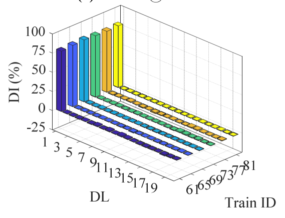
Damage detection with *Strain* responses and DI 50%

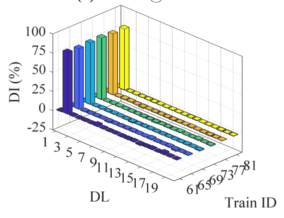


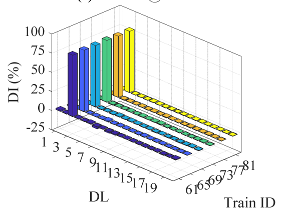


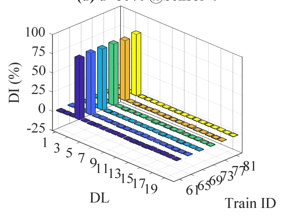


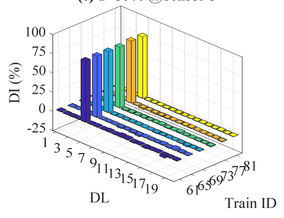
Damage detection with ***Strain*** responses and DI 80%

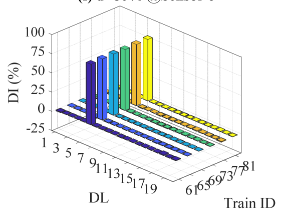

(a) d=80% @Sensor 1

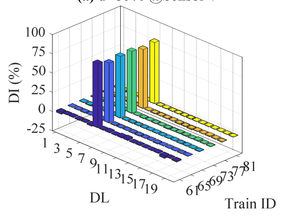

(b) d=80% @Sensor 2

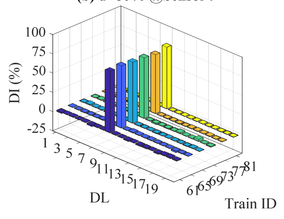

(c) d=80% @Sensor 3

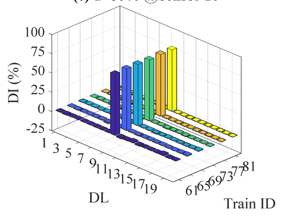

(d) d=80% @Sensor 4

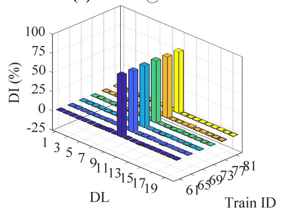

(e) d=80% @Sensor 5

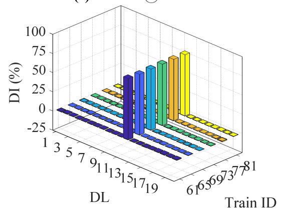

(f) d=80% @Sensor 6

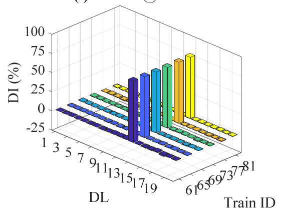

(a) d=80% @Sensor 7

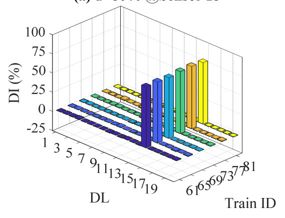

(b) d=80% @Sensor 9

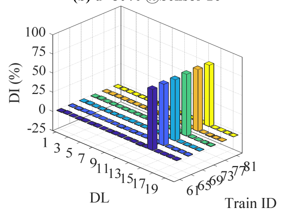

(c) d=80% @Sensor 10

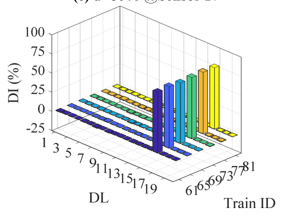

(d) d=80% @Sensor 11

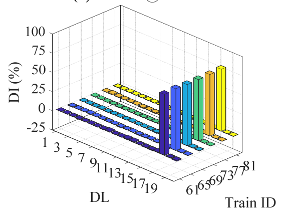

(e) d=80% @Sensor 12

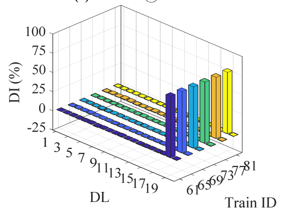

(f) d=80% @Sensor 14

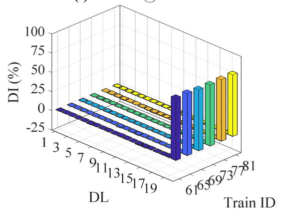

(a) d=80% @Sensor 15


(b) d=80% @Sensor 16


(c) d=80% @Sensor 17

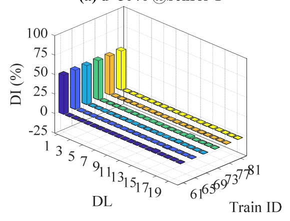

(d) d=80% @Sensor 18


(e) d=80% @Sensor 19

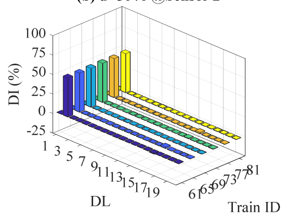

(f) d=80% @Sensor 20

Damage detection with ***Rotation*** responses and DI 50%

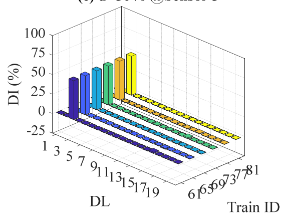
(a) d=50% @Sensor 1



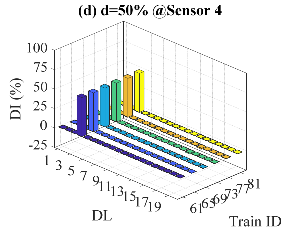
(b) d=50% @Sensor 2



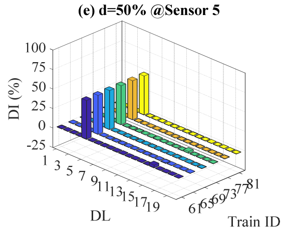
(c) d=50% @Sensor 3



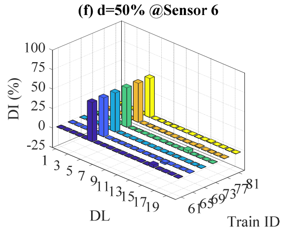
(d) d=50% @Sensor 4



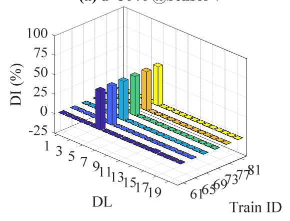
(e) d=50% @Sensor 5



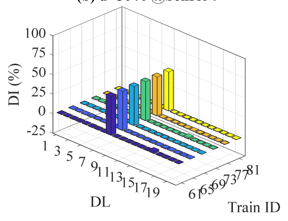
(f) d=50% @Sensor 6



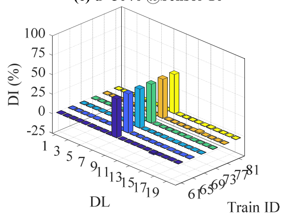
(a) d=50% @Sensor 7



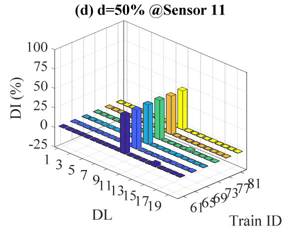
(b) d=50% @Sensor 9



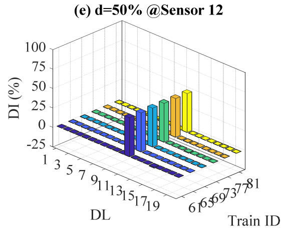
(c) d=50% @Sensor 10



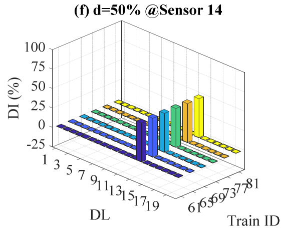
(d) d=50% @Sensor 11



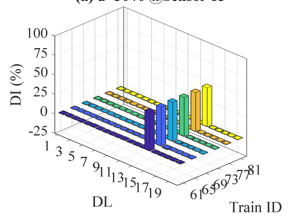
(e) d=50% @Sensor 12



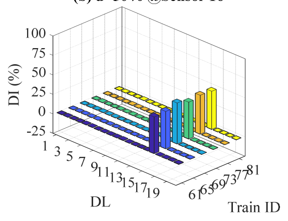
(f) d=50% @Sensor 14



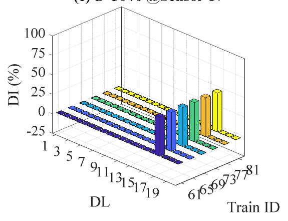
(a) d=50% @Sensor 15



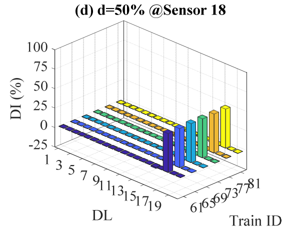
(b) d=50% @Sensor 16



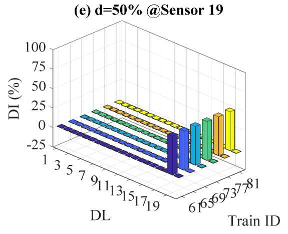
(c) d=50% @Sensor 17



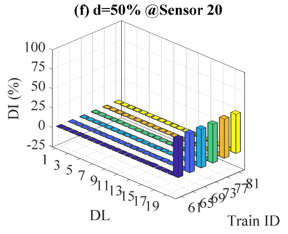
(d) d=50% @Sensor 18



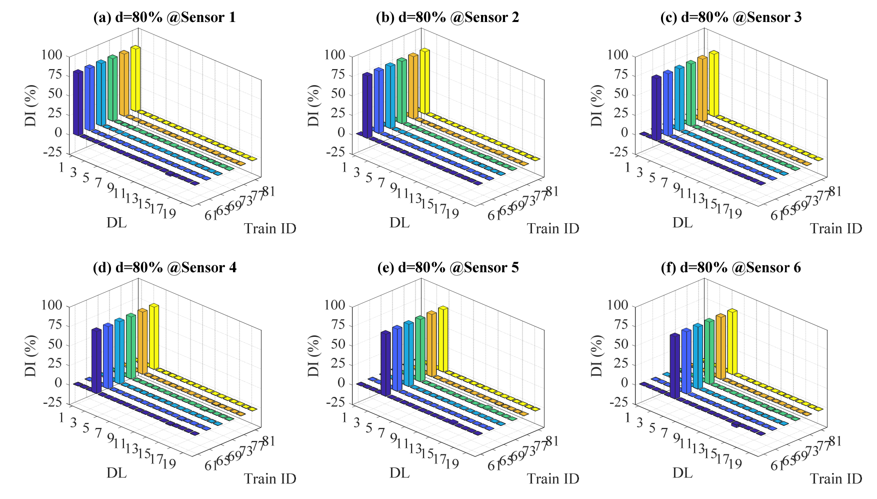
(e) d=50% @Sensor 19

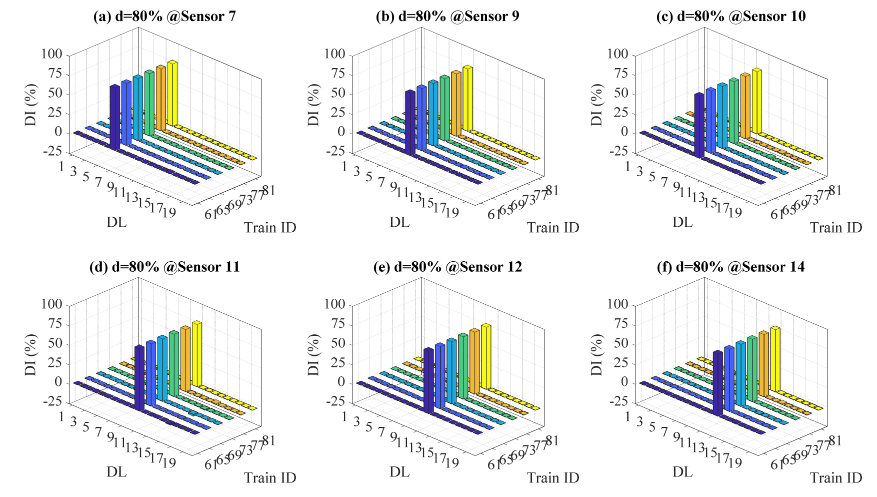


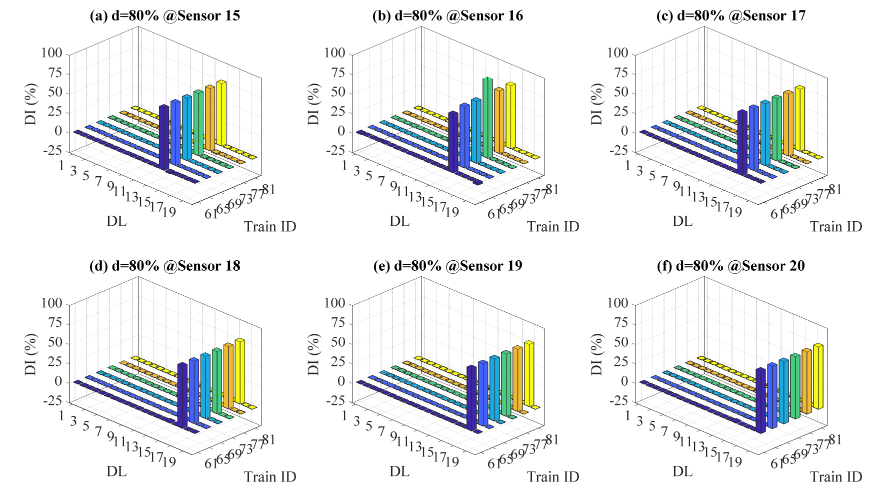
(f) d=50% @Sensor 20



Damage detection with ***Rotation*** responses and DI 80%

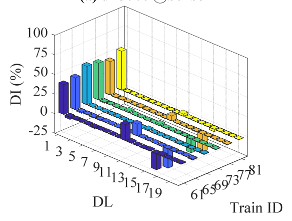




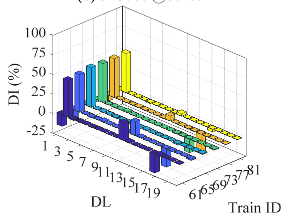


Damage detection with ***Displacement*** responses and DI 50%

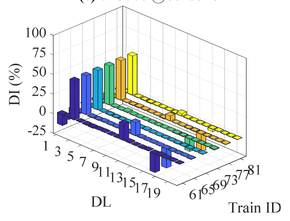
(a) d=50% @Sensor 1



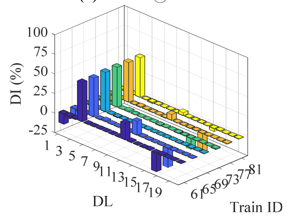
(b) d=50% @Sensor 2



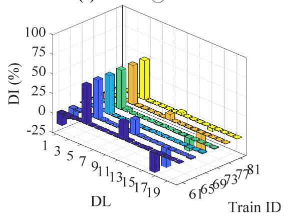
(c) d=50% @Sensor 3



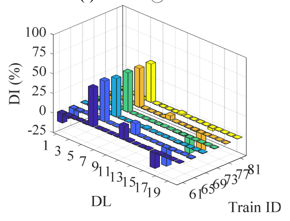
(d) d=50% @Sensor 4



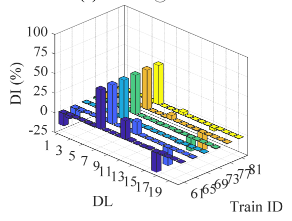
(e) d=50% @Sensor 5



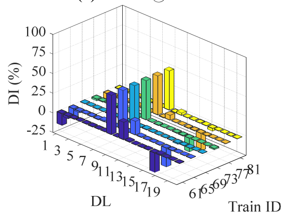
(f) d=50% @Sensor 6



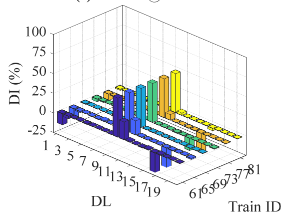
(a) d=50% @Sensor 7



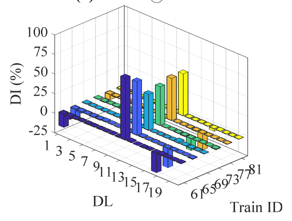
(b) d=50% @Sensor 9



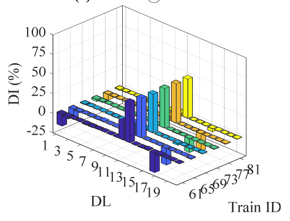
(c) d=50% @Sensor 10



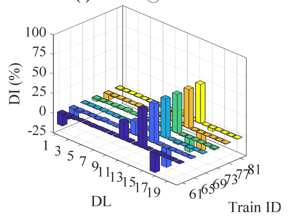
(d) d=50% @Sensor 11



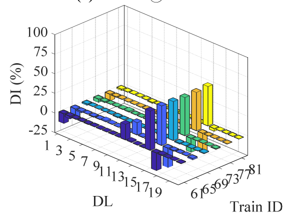
(e) d=50% @Sensor 12



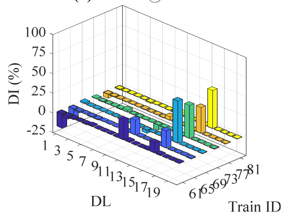
(f) d=50% @Sensor 14



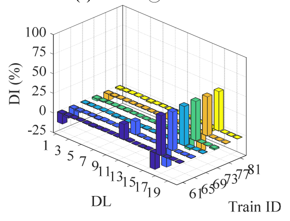
(a) d=50% @Sensor 15



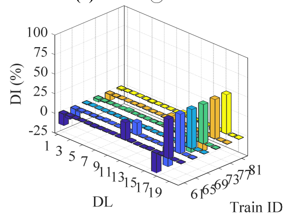
(b) d=50% @Sensor 16



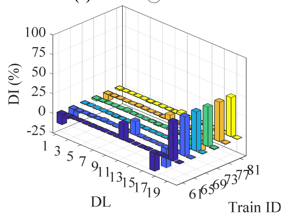
(c) d=50% @Sensor 17



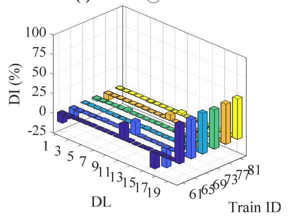
(d) d=50% @Sensor 18



(e) d=50% @Sensor 19

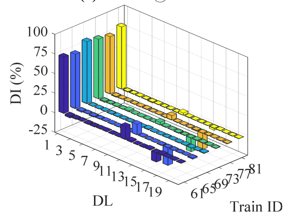


(f) d=50% @Sensor 20

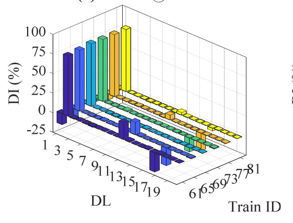


Damage detection with ***Displacement*** responses and DI 80%

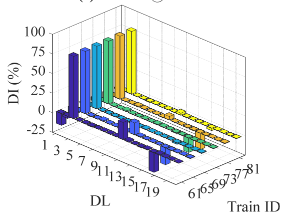
(a) d=80% @Sensor 1



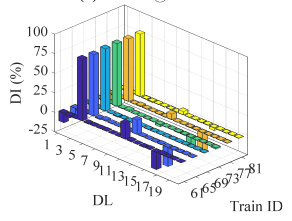
(b) d=80% @Sensor 2



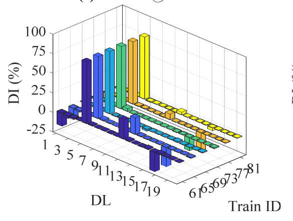
(c) d=80% @Sensor 3



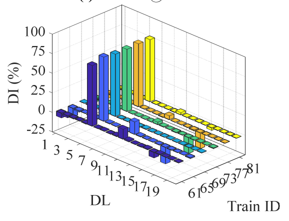
(d) d=80% @Sensor 4



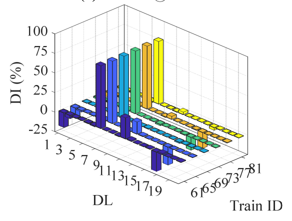
(e) d=80% @Sensor 5



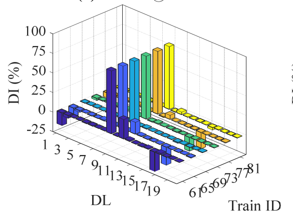
(f) d=80% @Sensor 6



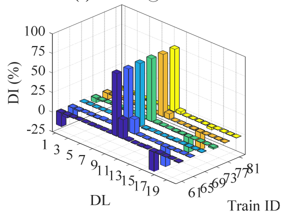
(a) d=80% @Sensor 7



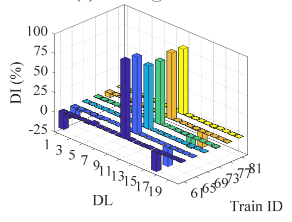
(b) d=80% @Sensor 9



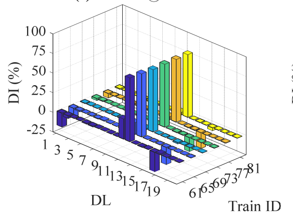
(c) d=80% @Sensor 10



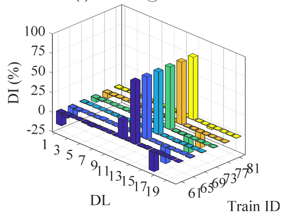
(d) d=80% @Sensor 11



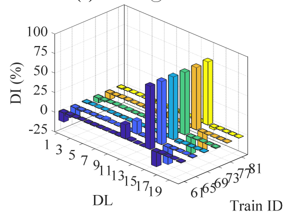
(e) d=80% @Sensor 12



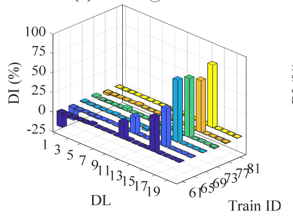
(f) d=80% @Sensor 14



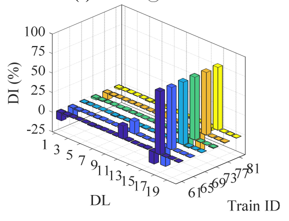
(a) d=80% @Sensor 15



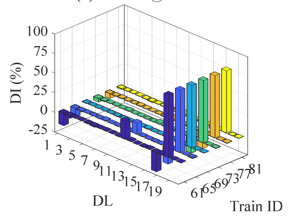
(b) d=80% @Sensor 16



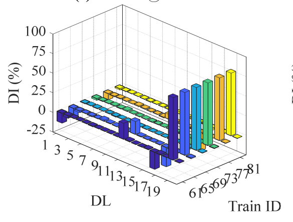
(c) d=80% @Sensor 17



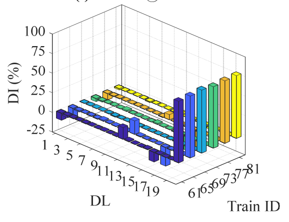
(d) d=80% @Sensor 18



(e) d=80% @Sensor 19

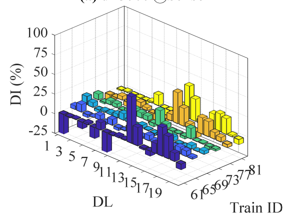


(f) d=80% @Sensor 20

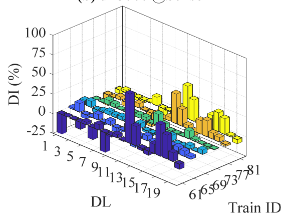


Damage detection with ***Acceleration*** responses and DI 50%

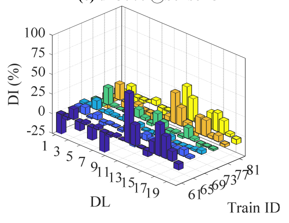
(a) d=50% @Sensor 1



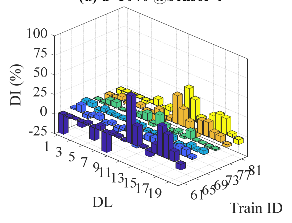
(b) d=50% @Sensor 2



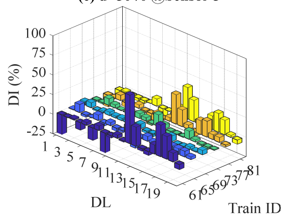
(c) d=50% @Sensor 3



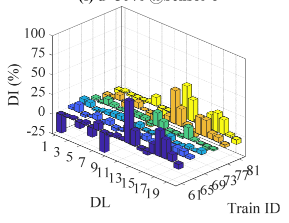
(d) d=50% @Sensor 4



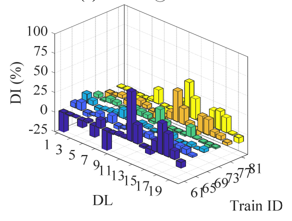
(e) d=50% @Sensor 5



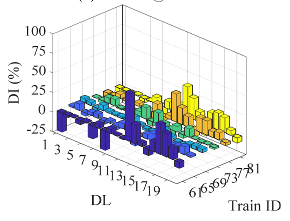
(f) d=50% @Sensor 6



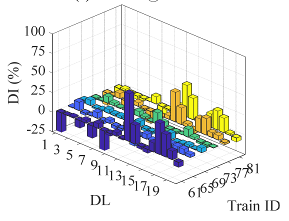
(a) d=50% @Sensor 7



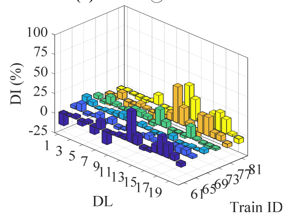
(b) d=50% @Sensor 9



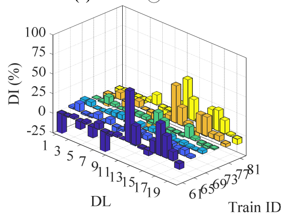
(c) d=50% @Sensor 10



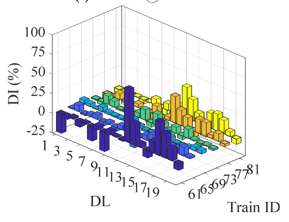
(d) d=50% @Sensor 11



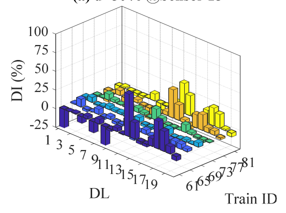
(e) d=50% @Sensor 12



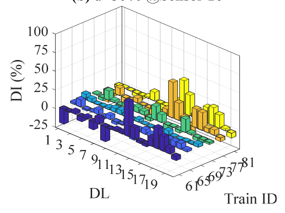
(f) d=50% @Sensor 14



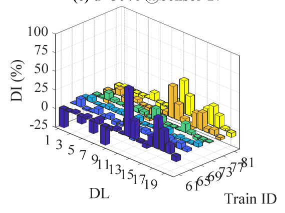
(a) d=50% @Sensor 15



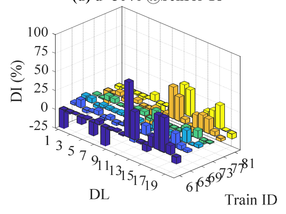
(b) d=50% @Sensor 16



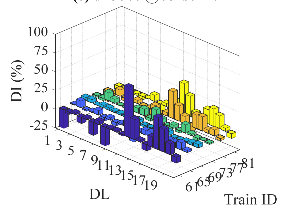
(c) d=50% @Sensor 17



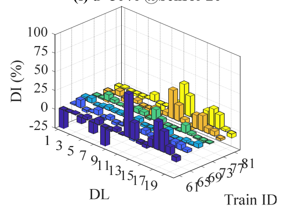
(d) d=50% @Sensor 18



(e) d=50% @Sensor 19

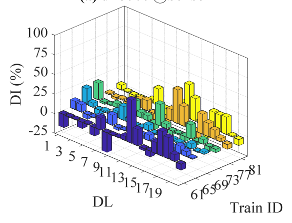


(f) d=50% @Sensor 20

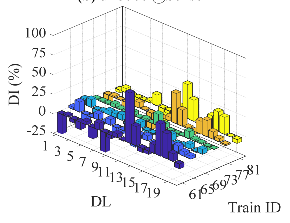


Damage detection with ***Acceleration*** responses and DI 80%

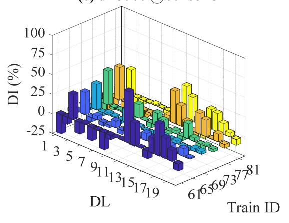
(a) d=80% @Sensor 1



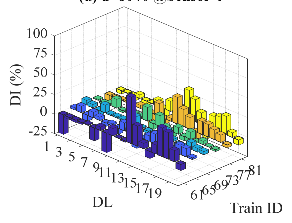
(b) d=80% @Sensor 2



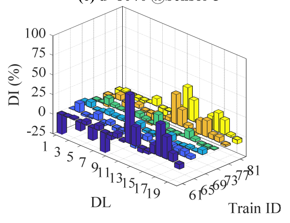
(c) d=80% @Sensor 3



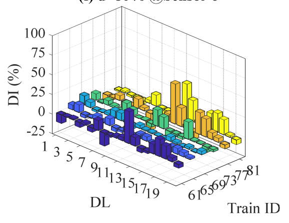
(d) d=80% @Sensor 4



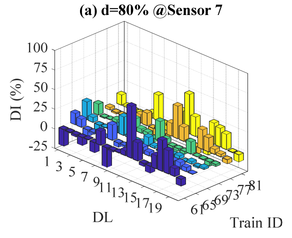
(e) d=80% @Sensor 5



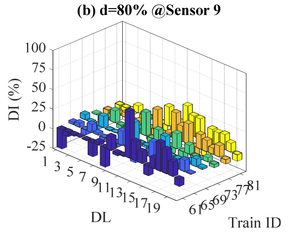
(f) d=80% @Sensor 6



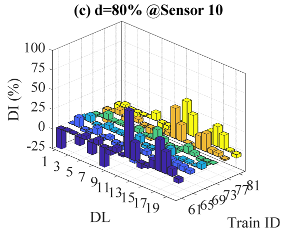
(a) d=80% @Sensor 7



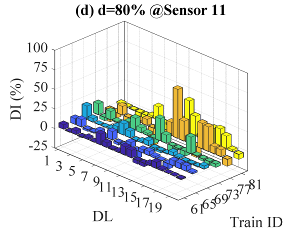
(b) d=80% @Sensor 9



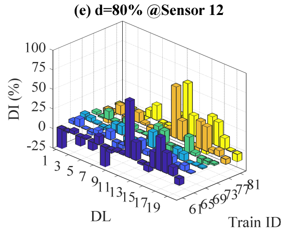
(c) d=80% @Sensor 10



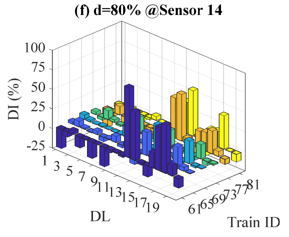
(d) d=80% @Sensor 11



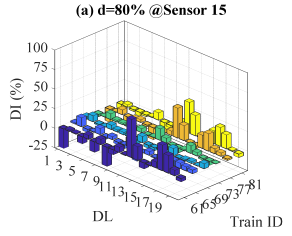
(e) d=80% @Sensor 12



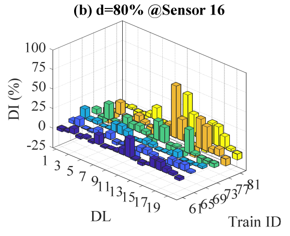
(f) d=80% @Sensor 14



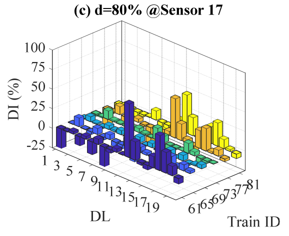
(a) d=80% @Sensor 15



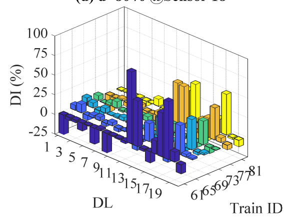
(b) d=80% @Sensor 16



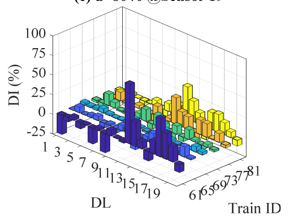
(c) d=80% @Sensor 17



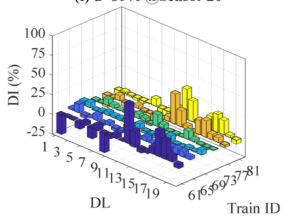
(d) d=80% @Sensor 18



(e) d=80% @Sensor 19

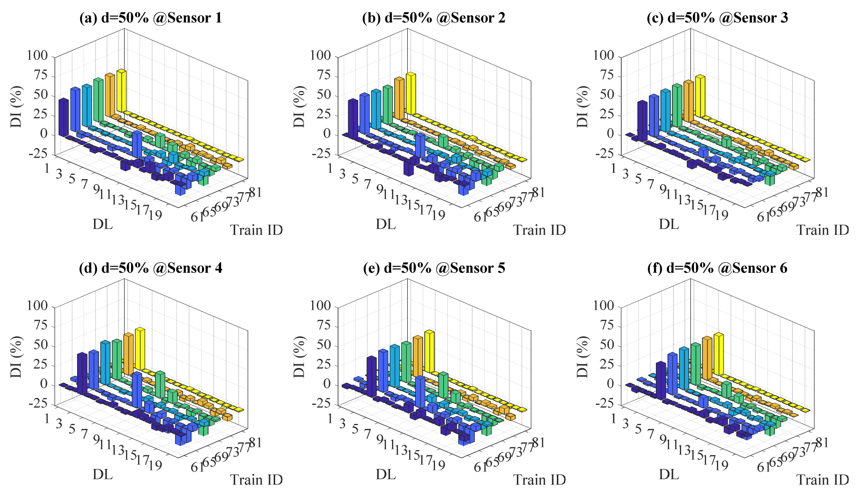


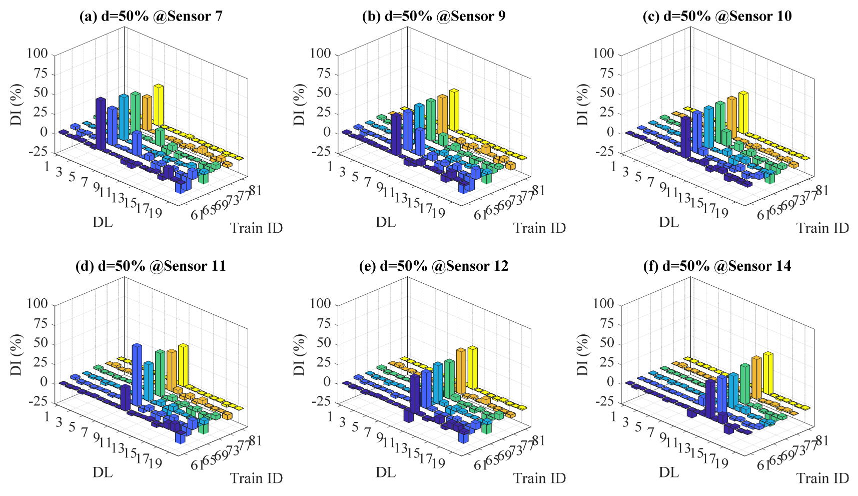
(f) d=80% @Sensor 20

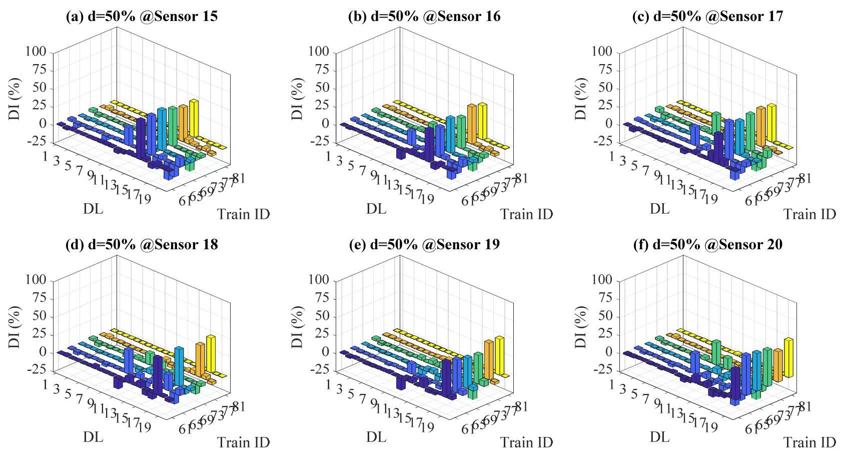


1.2 Instrumentation plan 2

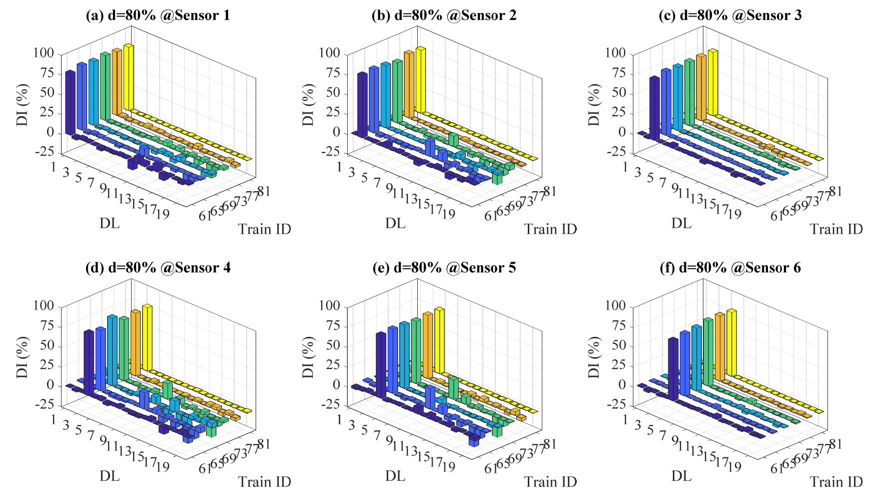
Damage detection with *<u>Strain</u>* responses and DI 50%

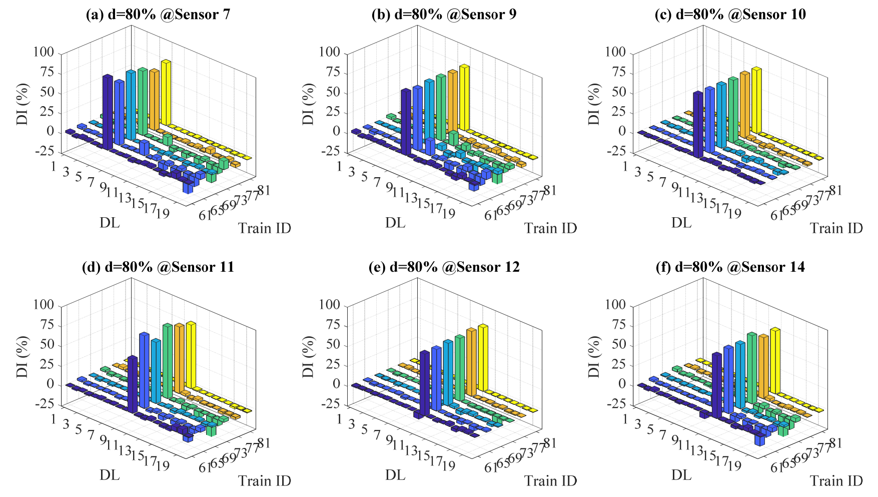


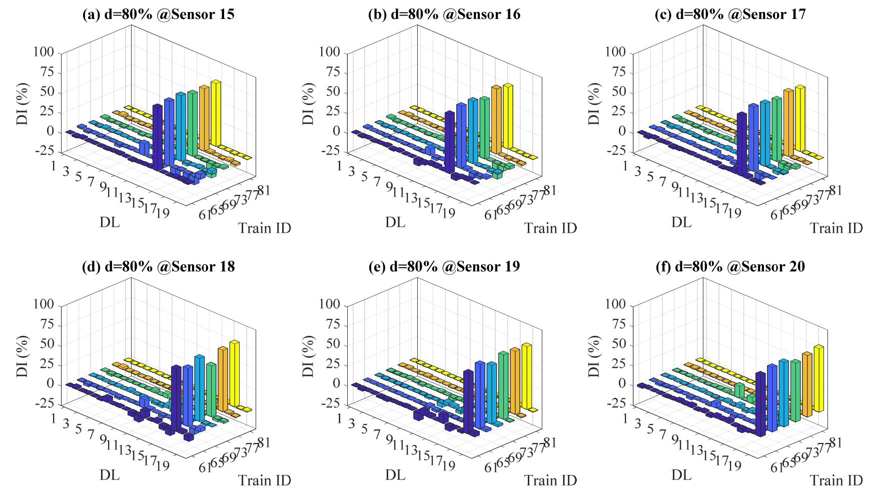




Damage detection with ***Strain*** responses and DI 80%

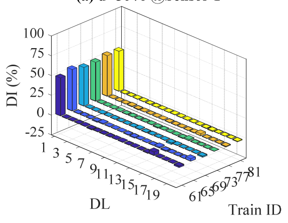




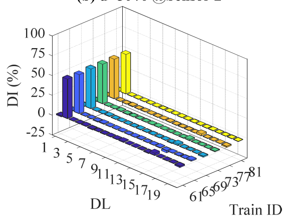


Damage detection with ***Rotation*** responses and DI 50%

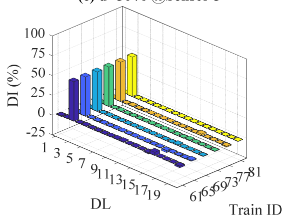
(a) d=50% @Sensor 1



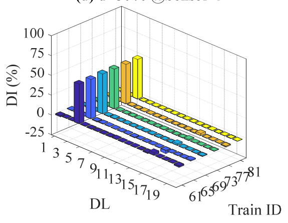
(b) d=50% @Sensor 2



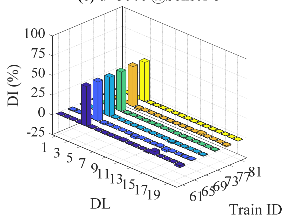
(c) d=50% @Sensor 3



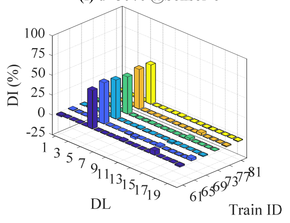
(d) d=50% @Sensor 4



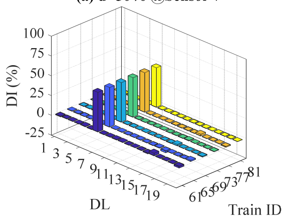
(e) d=50% @Sensor 5



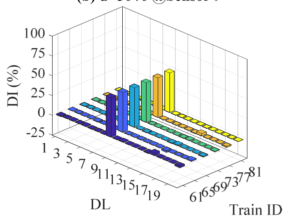
(f) d=50% @Sensor 6



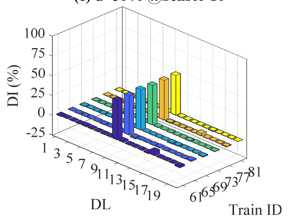
(a) d=50% @Sensor 7



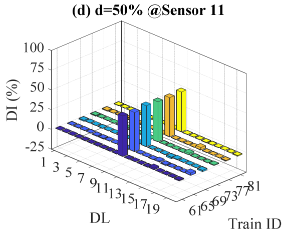
(b) d=50% @Sensor 9



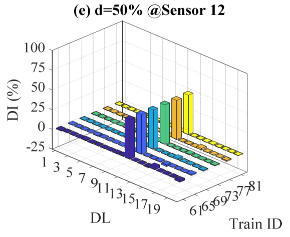
(c) d=50% @Sensor 10



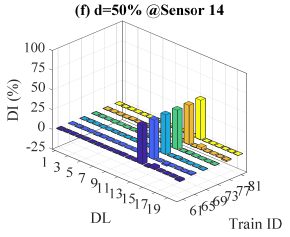
(d) d=50% @Sensor 11



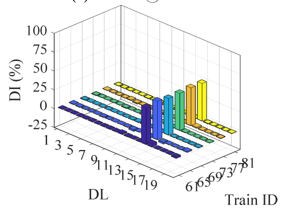
(e) d=50% @Sensor 12



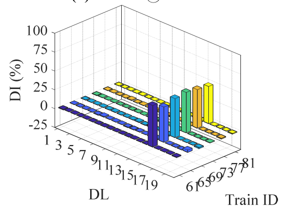
(f) d=50% @Sensor 14



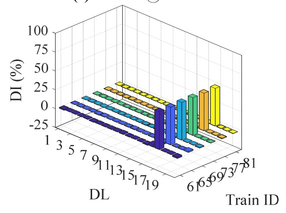
(a) d=50% @Sensor 15



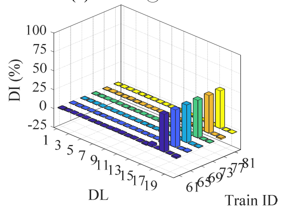
(b) d=50% @Sensor 16



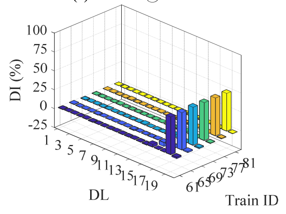
(c) d=50% @Sensor 17



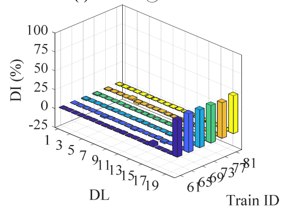
(d) d=50% @Sensor 18



(e) d=50% @Sensor 19

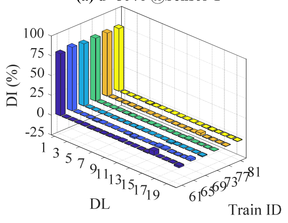


(f) d=50% @Sensor 20

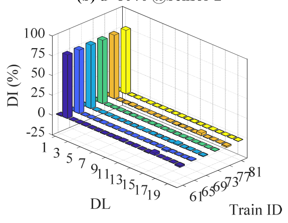


Damage detection with ***Rotation*** responses and DI 80%

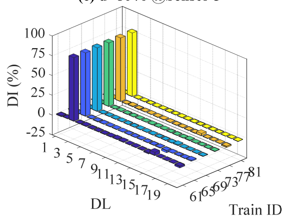
(a) d=80% @Sensor 1



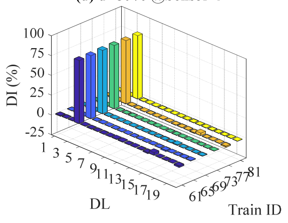
(b) d=80% @Sensor 2



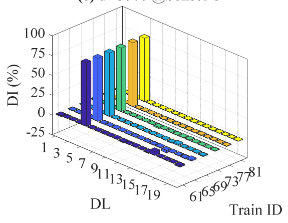
(c) d=80% @Sensor 3



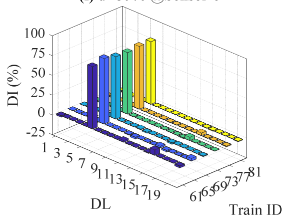
(d) d=80% @Sensor 4



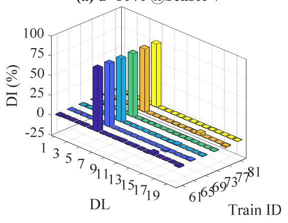
(e) d=80% @Sensor 5



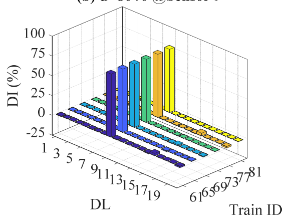
(f) d=80% @Sensor 6



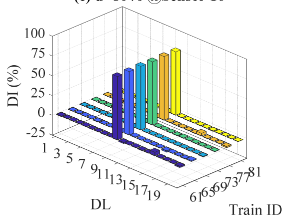
(a) d=80% @Sensor 7



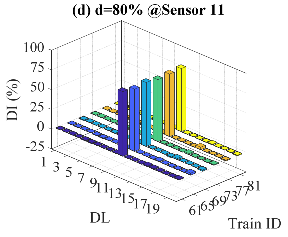
(b) d=80% @Sensor 9



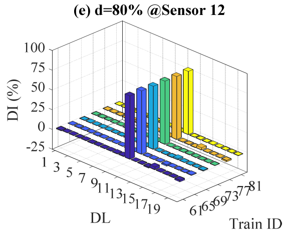
(c) d=80% @Sensor 10



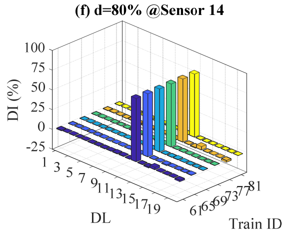
(d) d=80% @Sensor 11



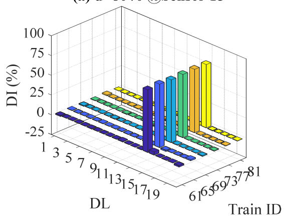
(e) d=80% @Sensor 12



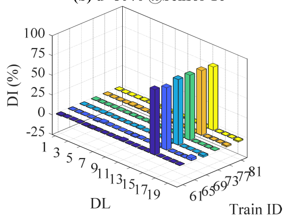
(f) d=80% @Sensor 14



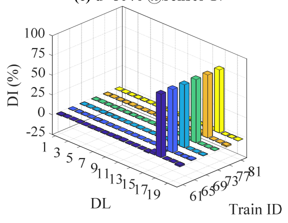
(a) d=80% @Sensor 15



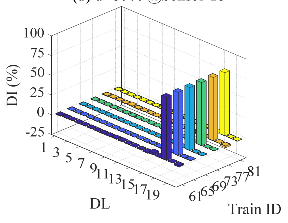
(b) d=80% @Sensor 16



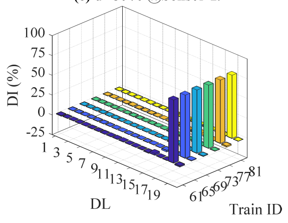
(c) d=80% @Sensor 17



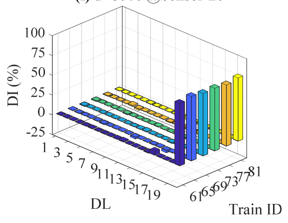
(d) d=80% @Sensor 18



(e) d=80% @Sensor 19

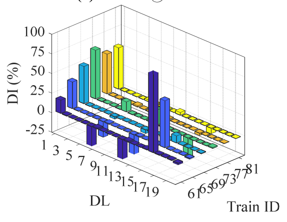


(f) d=80% @Sensor 20

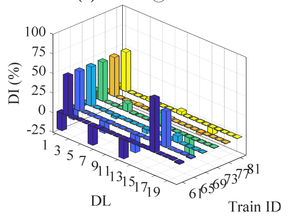


Damage detection with ***Displacement*** responses and DI 50%

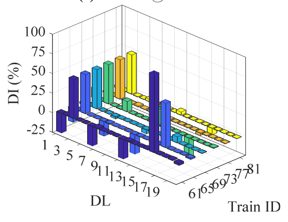
(a) d=50% @Sensor 1



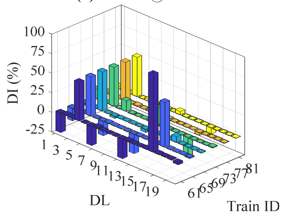
(b) d=50% @Sensor 2



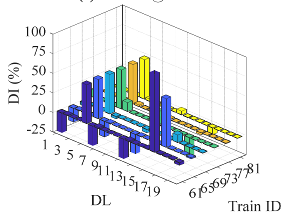
(c) d=50% @Sensor 3



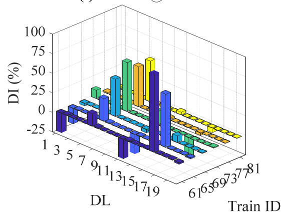
(d) d=50% @Sensor 4



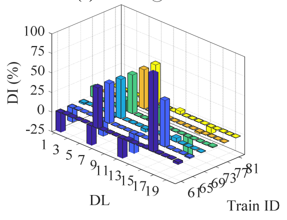
(e) d=50% @Sensor 5



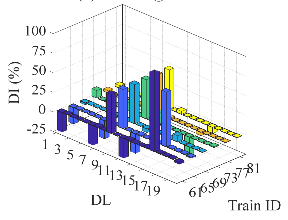
(f) d=50% @Sensor 6



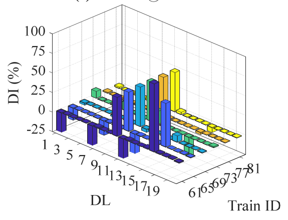
(a) d=50% @Sensor 7



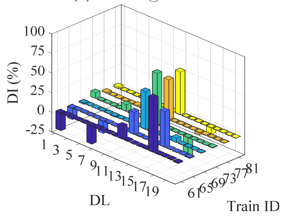
(b) d=50% @Sensor 9



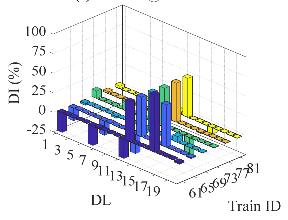
(c) d=50% @Sensor 10



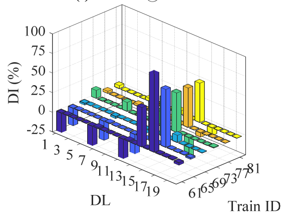
(d) d=50% @Sensor 11



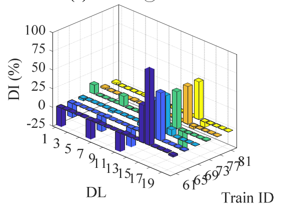
(e) d=50% @Sensor 12



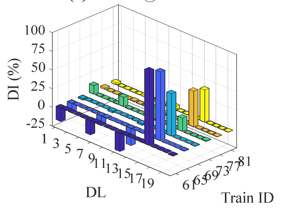
(f) d=50% @Sensor 14



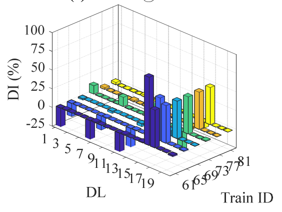
(a) d=50% @Sensor 15



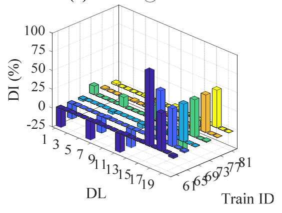
(b) d=50% @Sensor 16



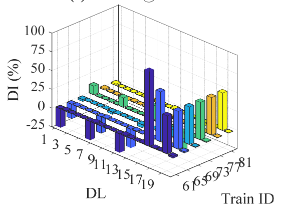
(c) d=50% @Sensor 17



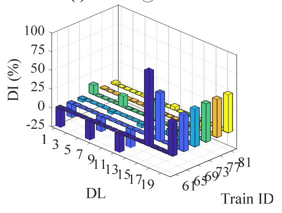
(d) d=50% @Sensor 18



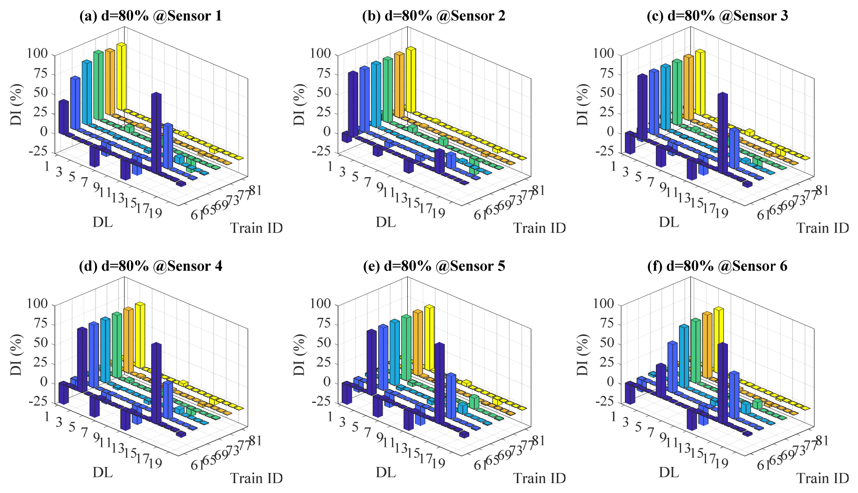
(e) d=50% @Sensor 19

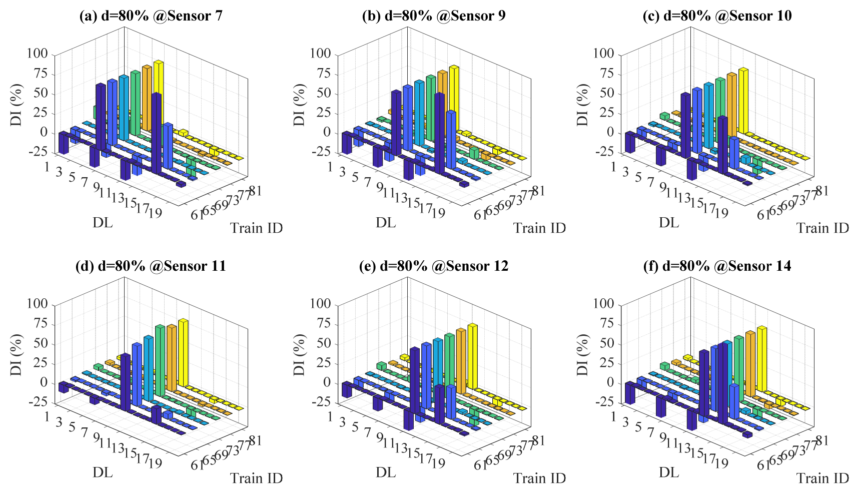


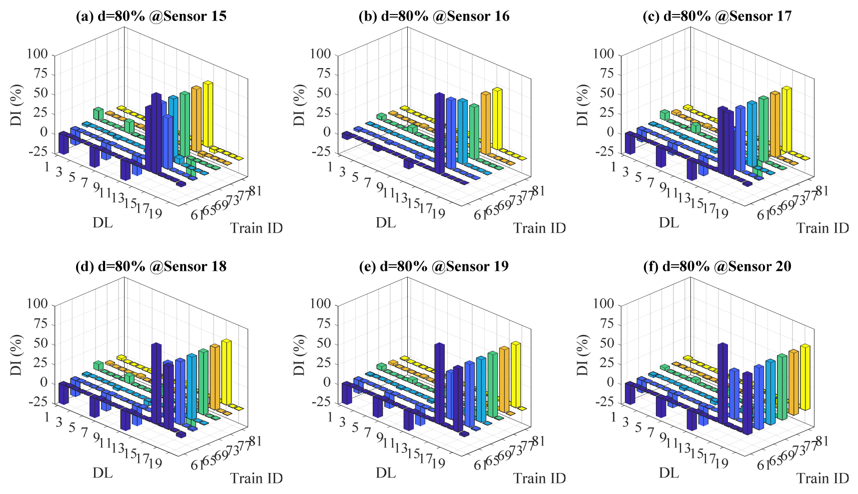
(f) d=50% @Sensor 20



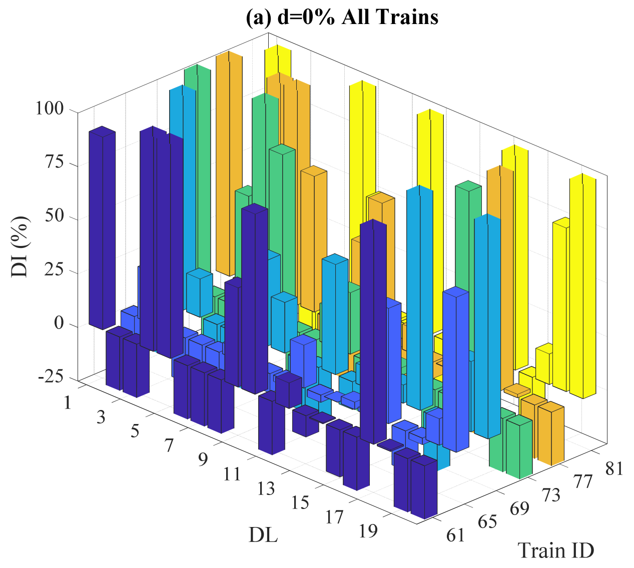
Damage detection with *Displacement* responses and DI 80%

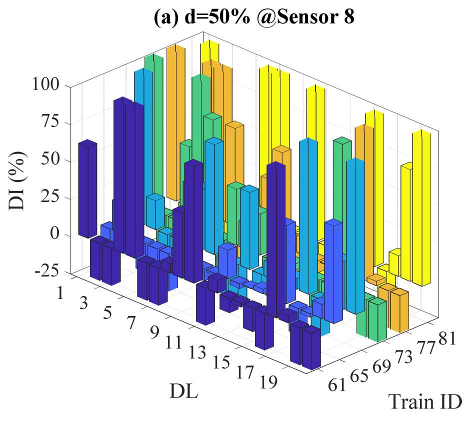


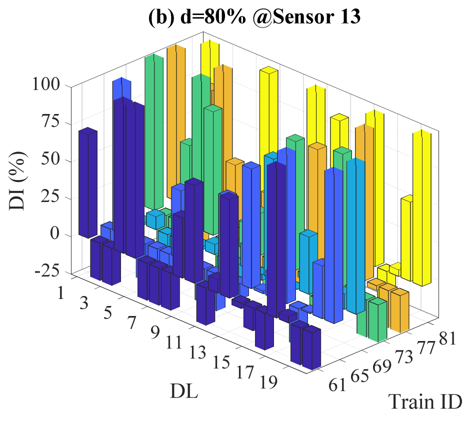




Damage detection with *Acceleration* responses

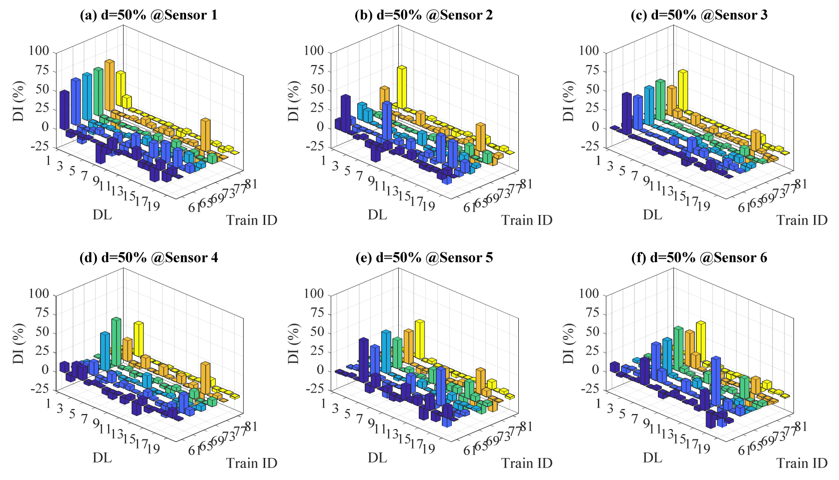


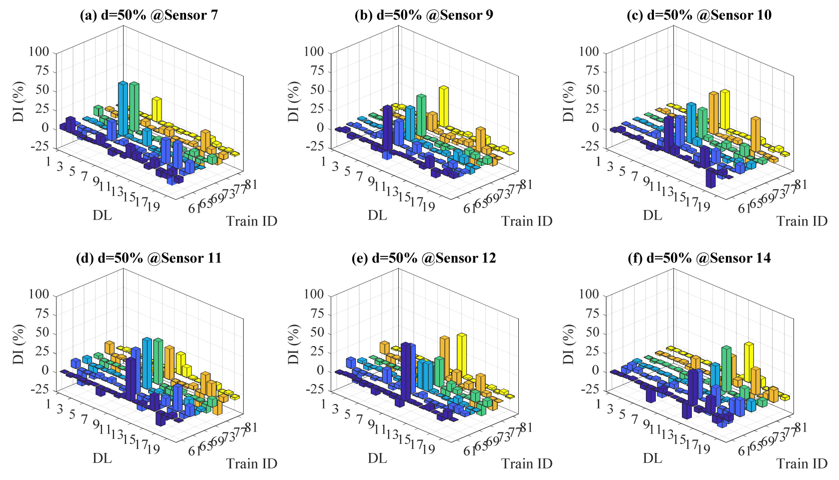


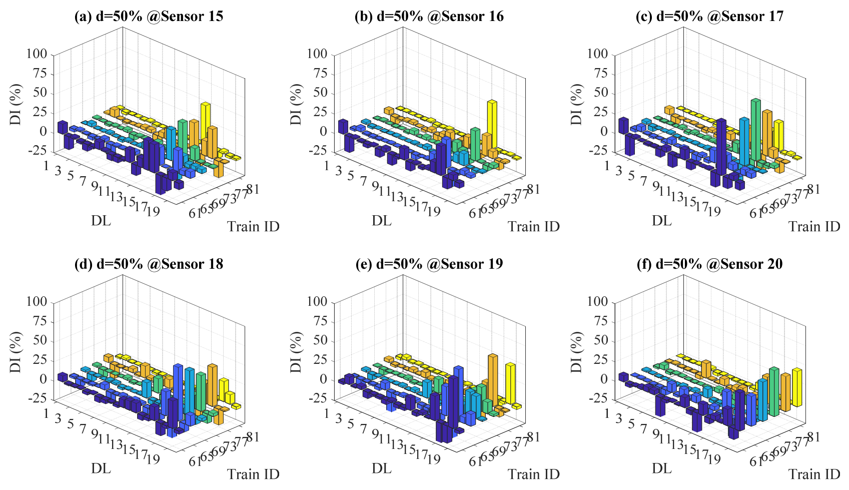


1.3 Instrumentation plan 3

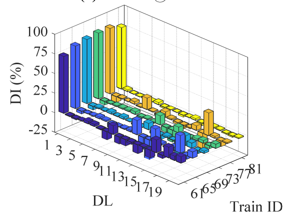
Damage detection with ***Strain*** responses and DI 50%

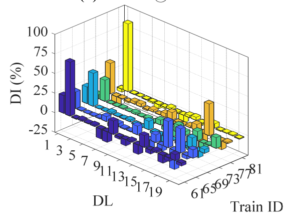


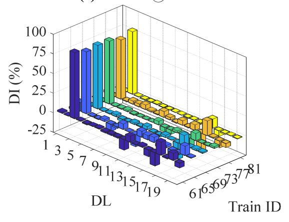


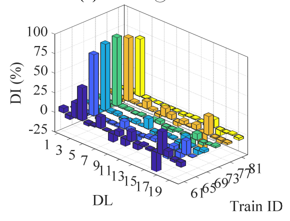


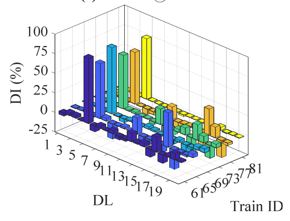
Damage detection with ***Strain*** responses and DI 80%

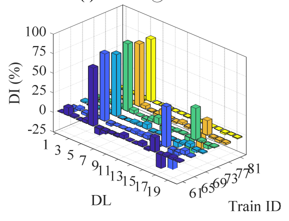

(a) d=80% @Sensor 1

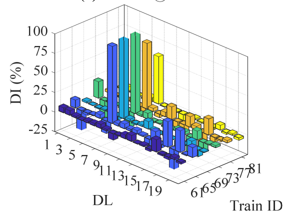

(b) d=80% @Sensor 2

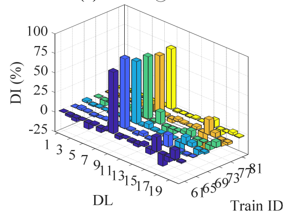

(c) d=80% @Sensor 3

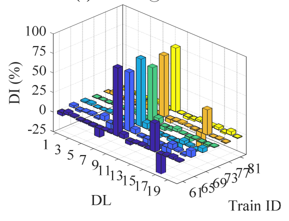

(d) d=80% @Sensor 4

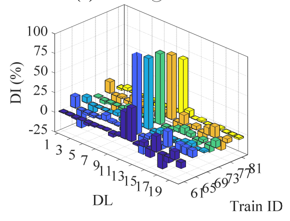

(e) d=80% @Sensor 5

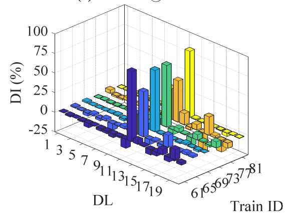

(f) d=80% @Sensor 6

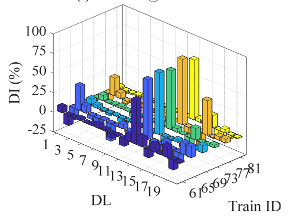

(a) d=80% @Sensor 7

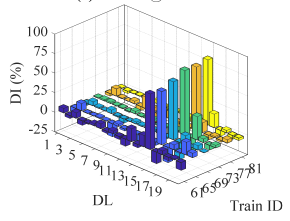

(b) d=80% @Sensor 9

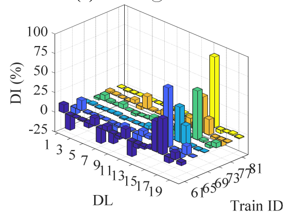

(c) d=80% @Sensor 10

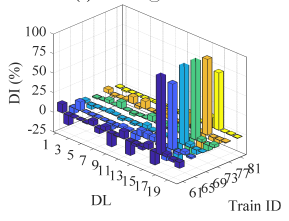

(d) d=80% @Sensor 11

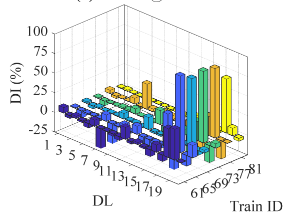

(e) d=80% @Sensor 12

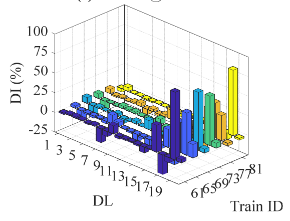

(f) d=80% @Sensor 14

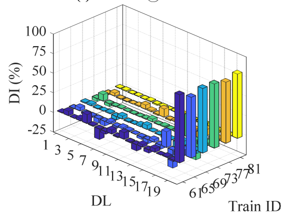

(a) d=80% @Sensor 15


(b) d=80% @Sensor 16


(c) d=80% @Sensor 17

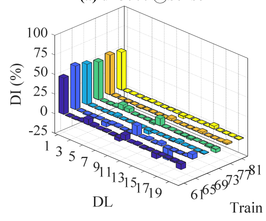

(d) d=80% @Sensor 18


(e) d=80% @Sensor 19

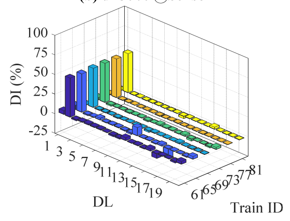

(f) d=80% @Sensor 20

Damage detection with ***Rotation*** responses and DI 50%

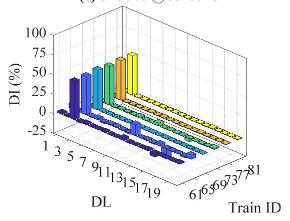
(a) d=50% @Sensor 1



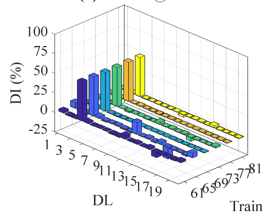
(b) d=50% @Sensor 2



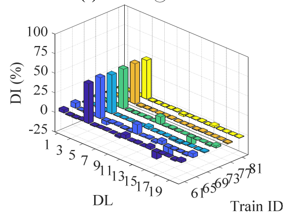
(c) d=50% @Sensor 3



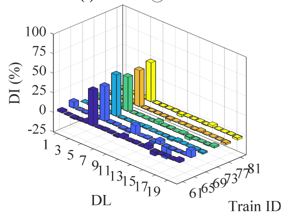
(d) d=50% @Sensor 4



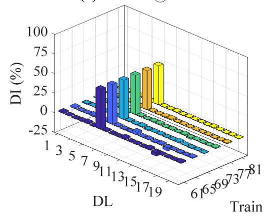
(e) d=50% @Sensor 5



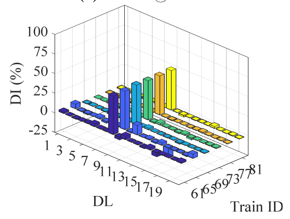
(f) d=50% @Sensor 6



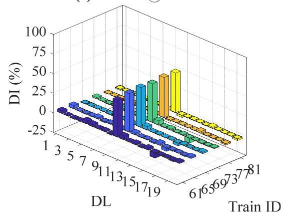
(a) d=50% @Sensor 7



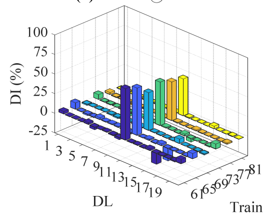
(b) d=50% @Sensor 9



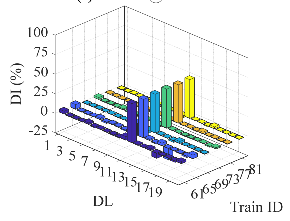
(c) d=50% @Sensor 10



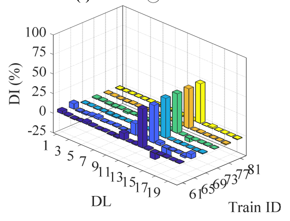
(d) d=50% @Sensor 11



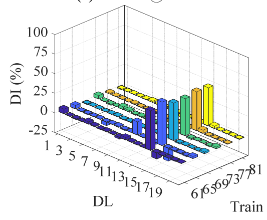
(e) d=50% @Sensor 12



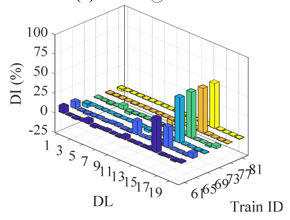
(f) d=50% @Sensor 14



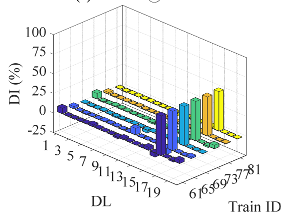
(a) d=50% @Sensor 15



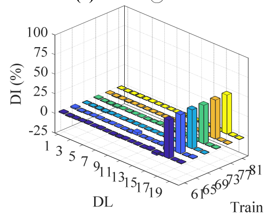
(b) d=50% @Sensor 16



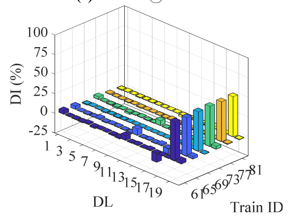
(c) d=50% @Sensor 17



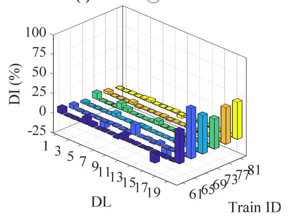
(d) d=50% @Sensor 18



(e) d=50% @Sensor 19

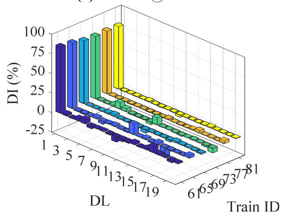


(f) d=50% @Sensor 20

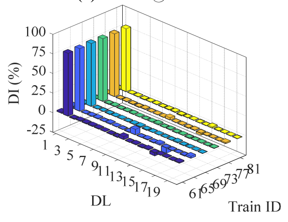


Damage detection with ***Rotation*** responses and DI 80%

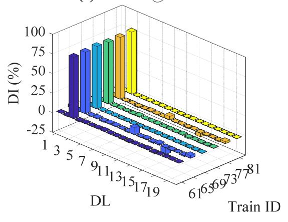
(a) d=80% @Sensor 1



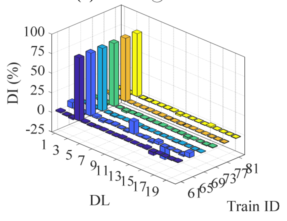
(b) d=80% @Sensor 2



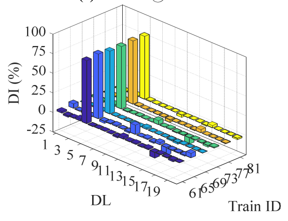
(c) d=80% @Sensor 3



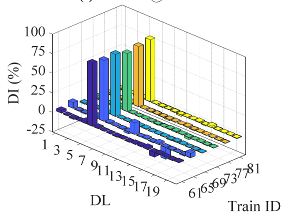
(d) d=80% @Sensor 4



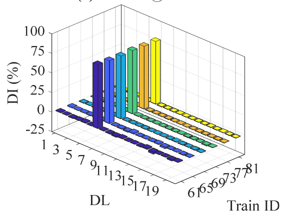
(e) d=80% @Sensor 5



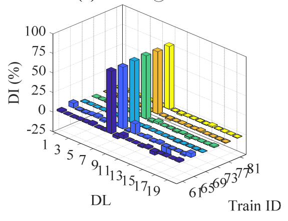
(f) d=80% @Sensor 6



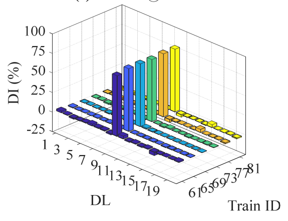
(a) d=80% @Sensor 7



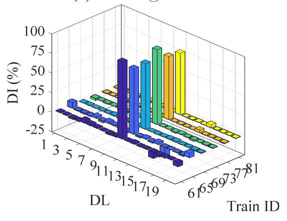
(b) d=80% @Sensor 9



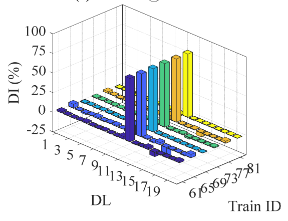
(c) d=80% @Sensor 10



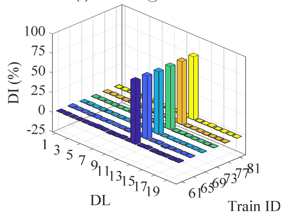
(d) d=80% @Sensor 11



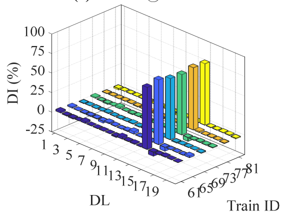
(e) d=80% @Sensor 12



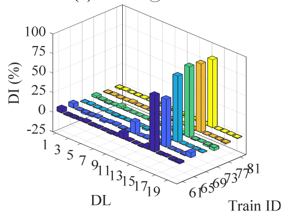
(f) d=80% @Sensor 14



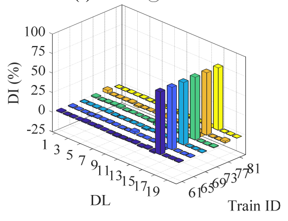
(a) d=80% @Sensor 15



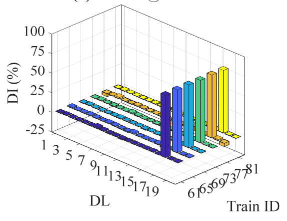
(b) d=80% @Sensor 16



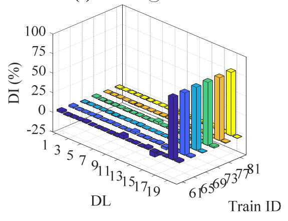
(c) d=80% @Sensor 17



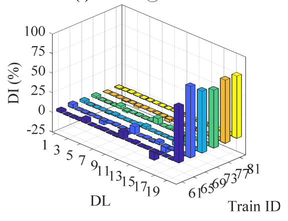
(d) d=80% @Sensor 18



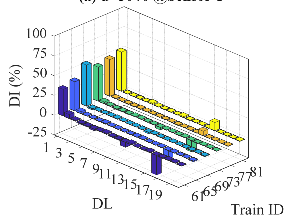
(e) d=80% @Sensor 19



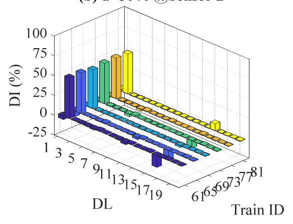
(f) d=80% @Sensor 20



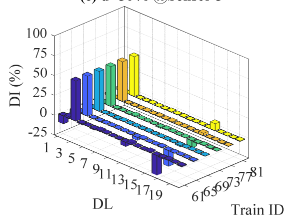
Damage detection with ***Displacement*** responses and DI 50%



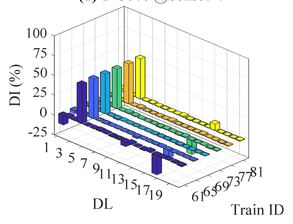
(a) d=50% @Sensor 1



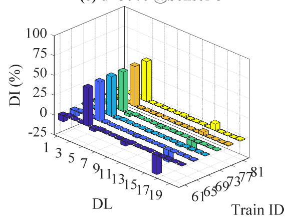
(b) d=50% @Sensor 2



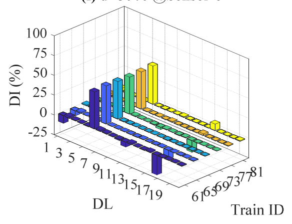
(c) d=50% @Sensor 3



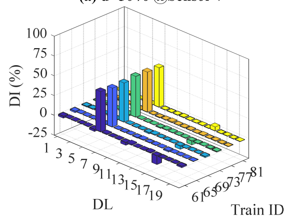
(d) d=50% @Sensor 4



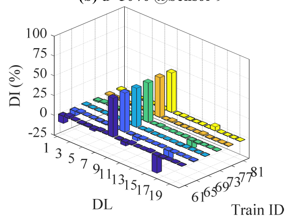
(e) d=50% @Sensor 5



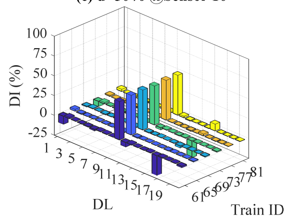
(f) d=50% @Sensor 6



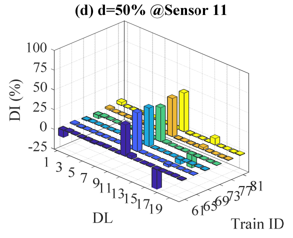
(a) d=50% @Sensor 7



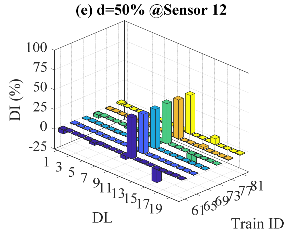
(b) d=50% @Sensor 9



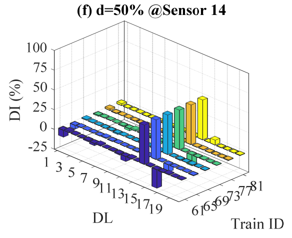
(c) d=50% @Sensor 10



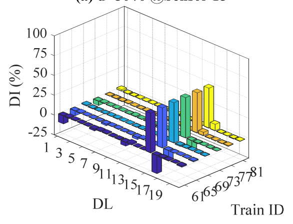
(d) d=50% @Sensor 11



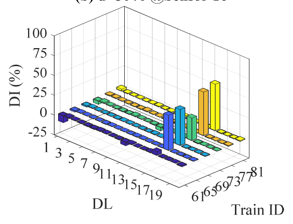
(e) d=50% @Sensor 12



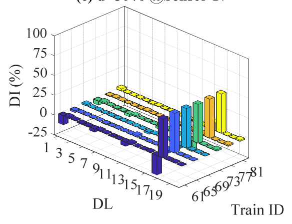
(f) d=50% @Sensor 14



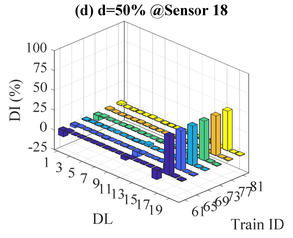
(a) d=50% @Sensor 15



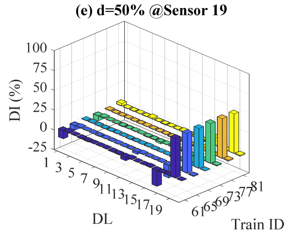
(b) d=50% @Sensor 16



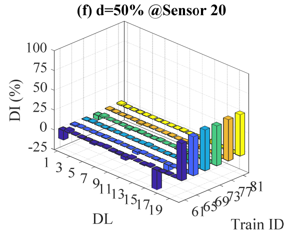
(c) d=50% @Sensor 17



(d) d=50% @Sensor 18



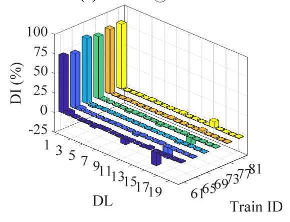
(e) d=50% @Sensor 19



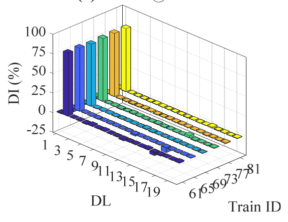
(f) d=50% @Sensor 20

Damage detection with ***Displacement*** responses and DI 80%

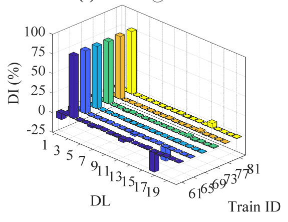
(a) d=80% @Sensor 1



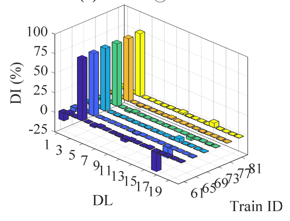
(b) d=80% @Sensor 2



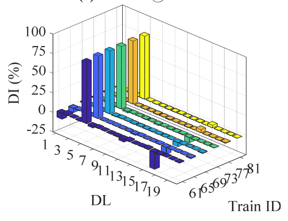
(c) d=80% @Sensor 3



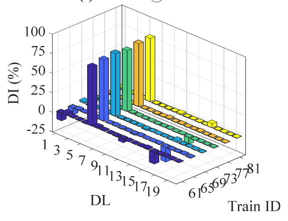
(d) d=80% @Sensor 4



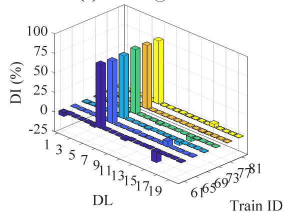
(e) d=80% @Sensor 5



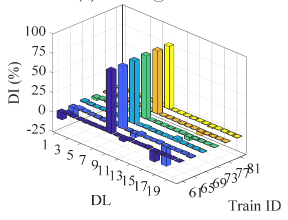
(f) d=80% @Sensor 6



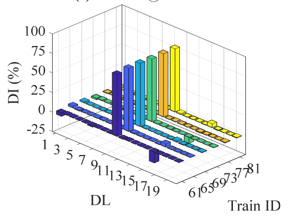
(a) d=80% @Sensor 7



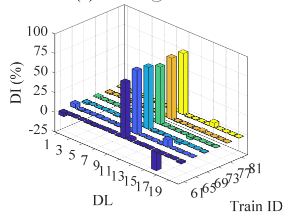
(b) d=80% @Sensor 9



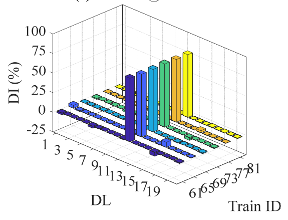
(c) d=80% @Sensor 10



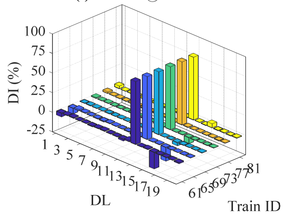
(d) d=80% @Sensor 11



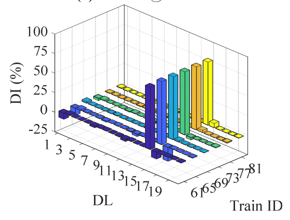
(e) d=80% @Sensor 12



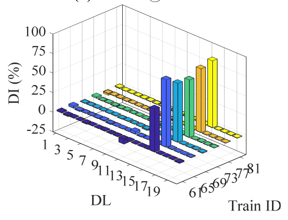
(f) d=80% @Sensor 14



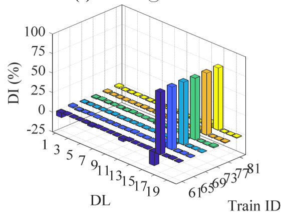
(a) d=80% @Sensor 15



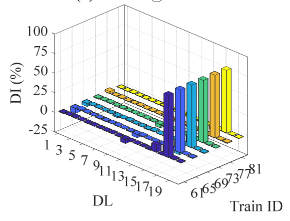
(b) d=80% @Sensor 16



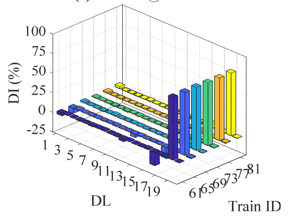
(c) d=80% @Sensor 17



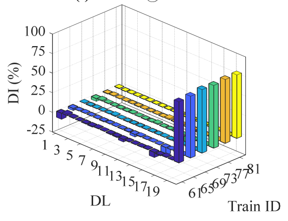
(d) d=80% @Sensor 18



(e) d=80% @Sensor 19

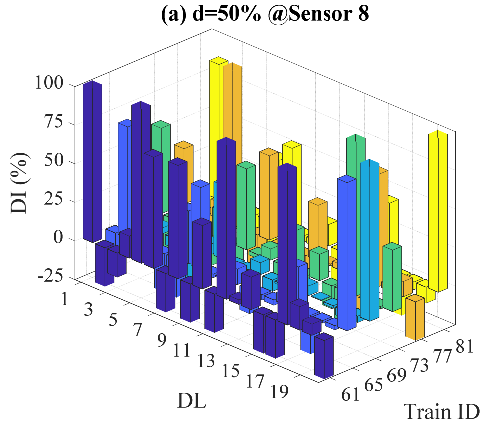


(f) d=80% @Sensor 20

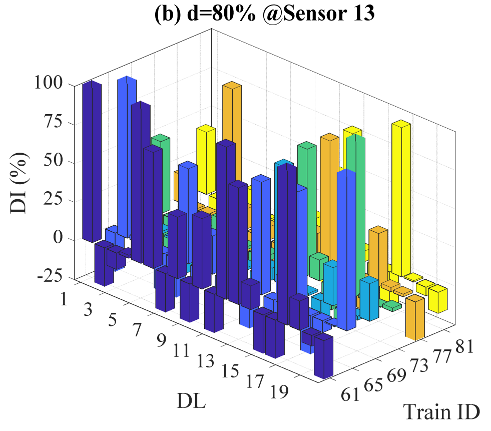


Damage detection with ***Acceleration*** ***res*ponses**

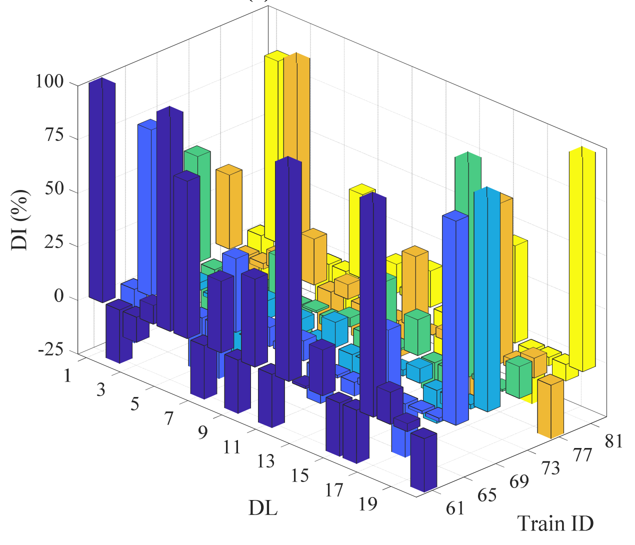
(a) d=50% @Sensor 8



(b) d=80% @Sensor 13



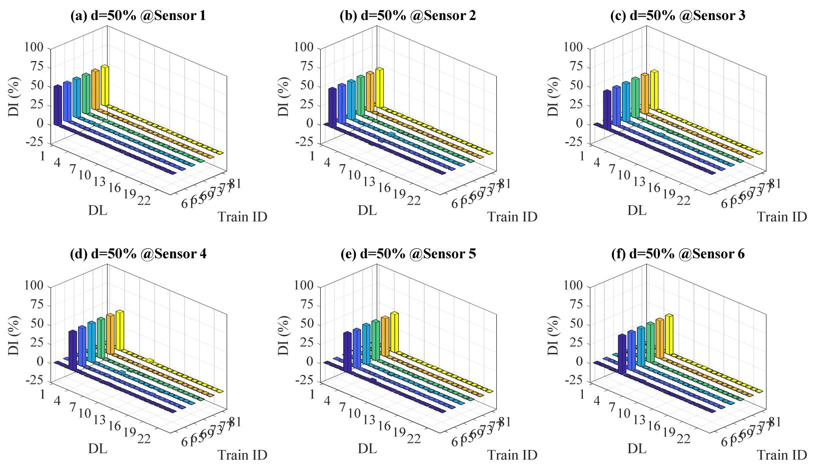
(a) d=0% All Trains

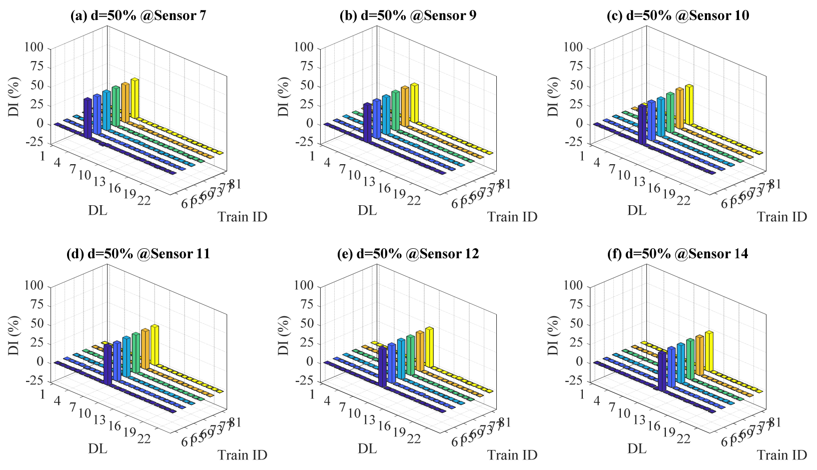


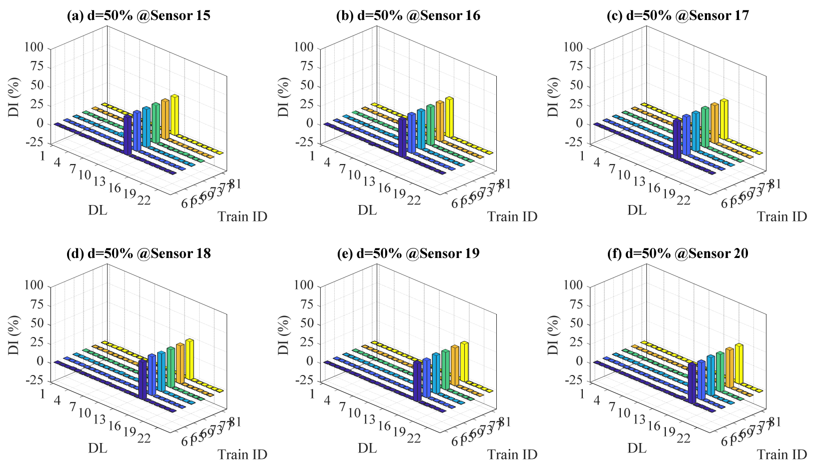
2. Plate girder span:

2.1 Instrumentation plan 1

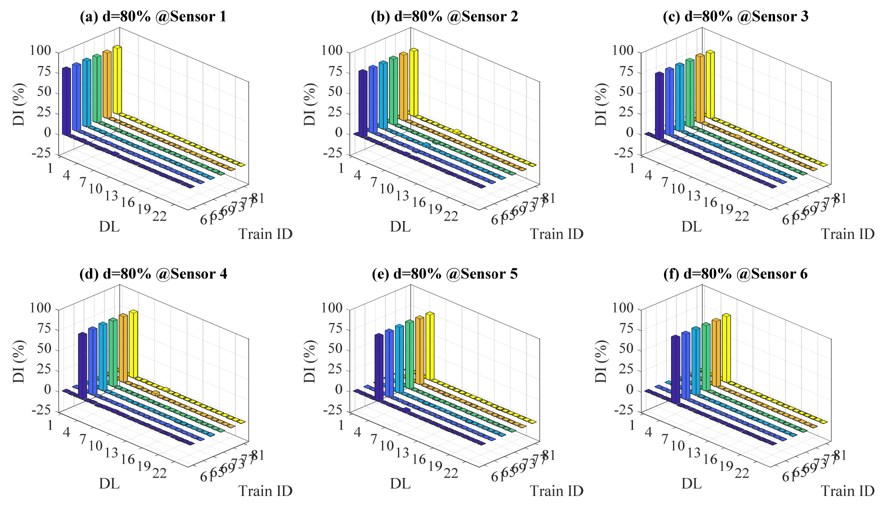
Damage detection with *Strain* responses and DI 50%

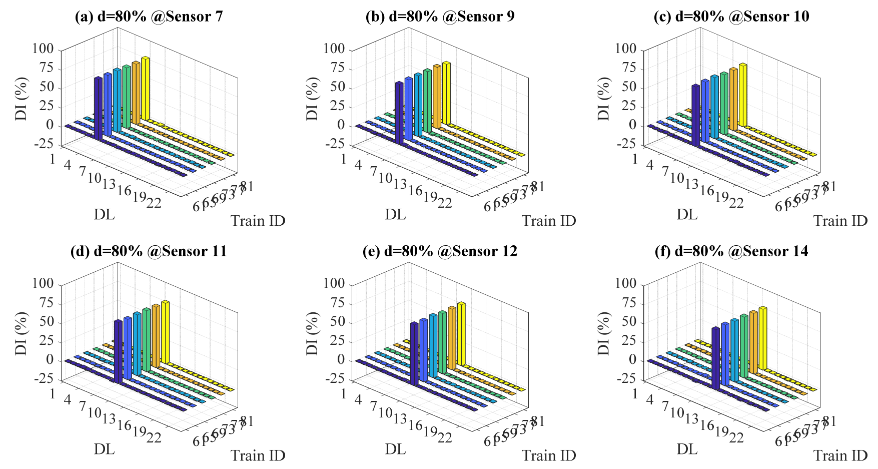


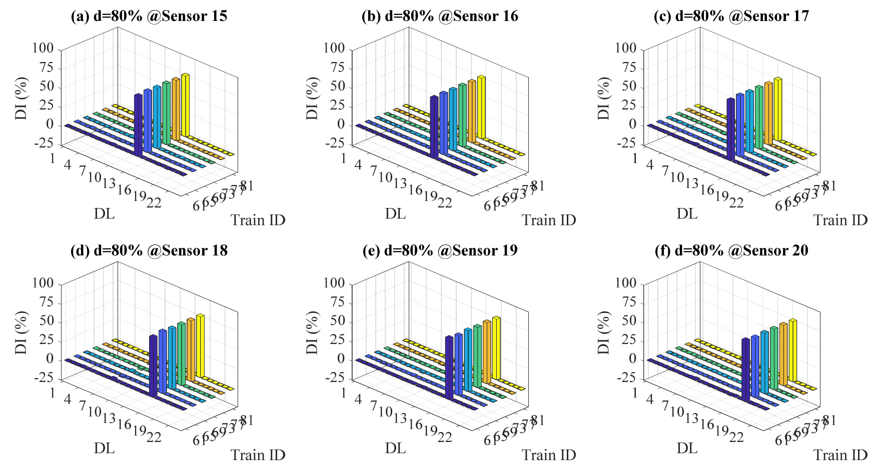




Damage detection with ***<u>Strain</u>*** responses and DI 80%

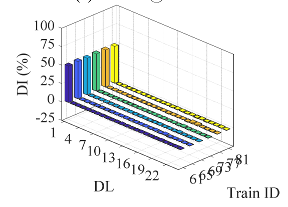




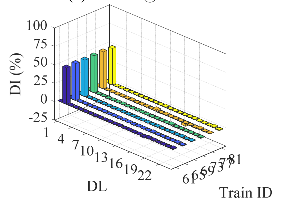


Damage detection with ***Rotation*** responses and DI 50%

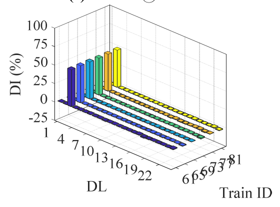
(a) d=50% @Sensor 1



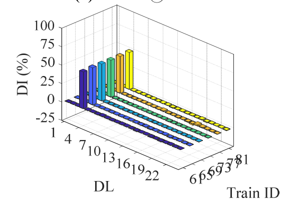
(b) d=50% @Sensor 2



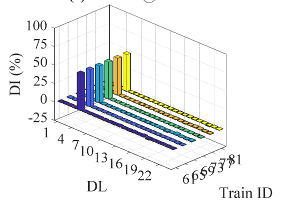
(c) d=50% @Sensor 3



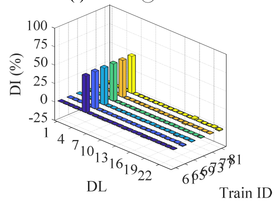
(d) d=50% @Sensor 4



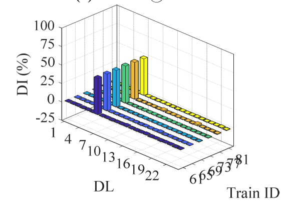
(e) d=50% @Sensor 5



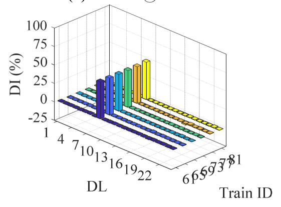
(f) d=50% @Sensor 6



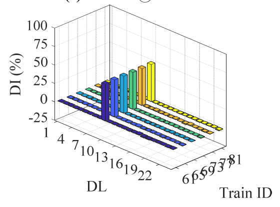
(a) d=50% @Sensor 7



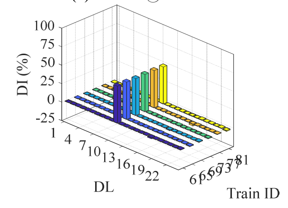
(b) d=50% @Sensor 9



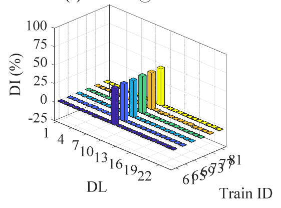
(c) d=50% @Sensor 10



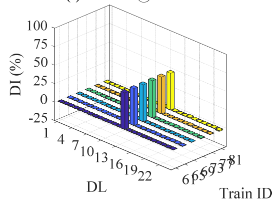
(d) d=50% @Sensor 11



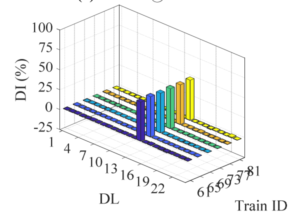
(e) d=50% @Sensor 12



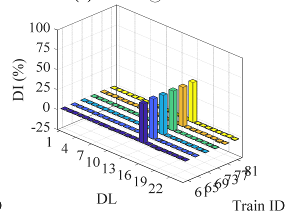
(f) d=50% @Sensor 14



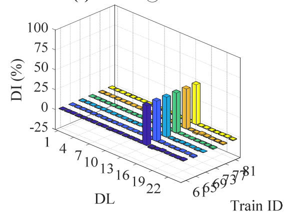
(a) d=50% @Sensor 15



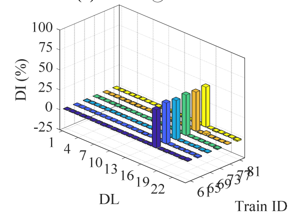
(b) d=50% @Sensor 16



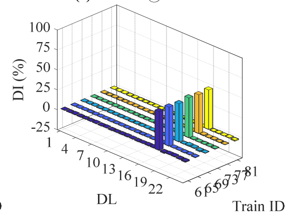
(c) d=50% @Sensor 17



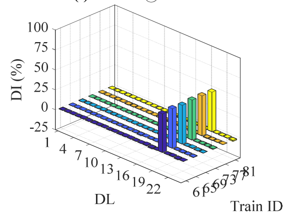
(d) d=50% @Sensor 18



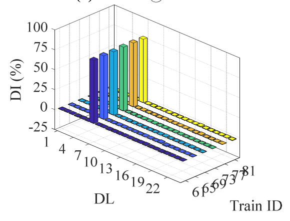
(e) d=50% @Sensor 19

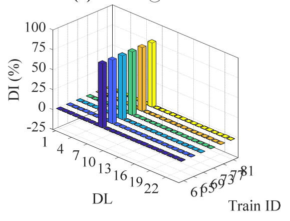


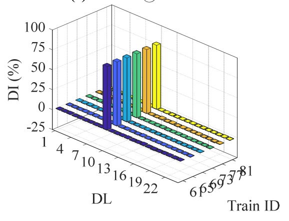
(f) d=50% @Sensor 20

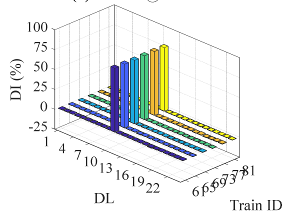


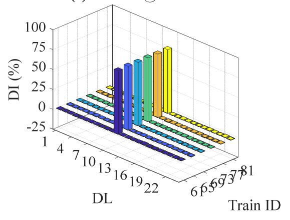
Damage detection with ***Rotation*** responses and DI 80%

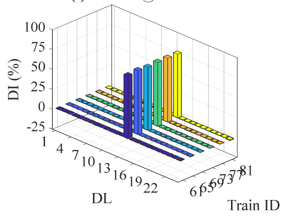

(a) d=80% @Sensor 7

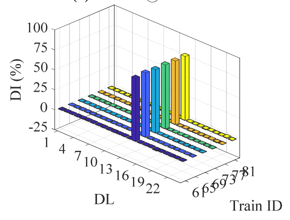

(b) d=80% @Sensor 9

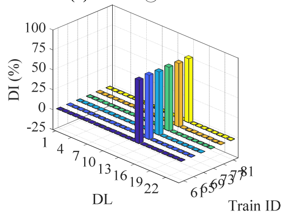

(c) d=80% @Sensor 10

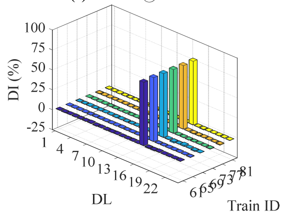

(d) d=80% @Sensor 11

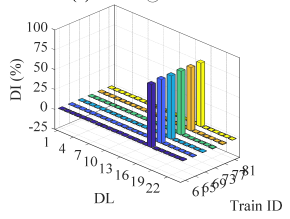

(e) d=80% @Sensor 12

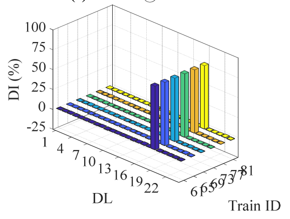

(f) d=80% @Sensor 14

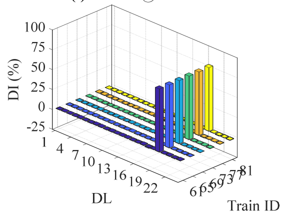

(a) d=80% @Sensor 15

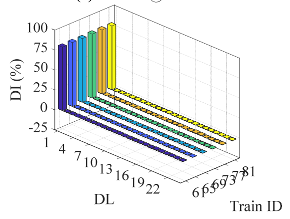

(b) d=80% @Sensor 16

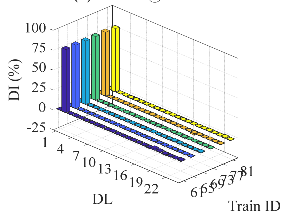

(c) d=80% @Sensor 17

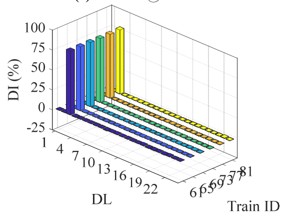

(d) d=80% @Sensor 18

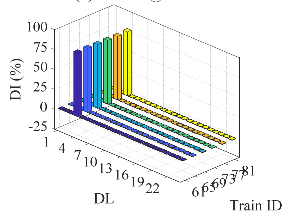

(e) d=80% @Sensor 19

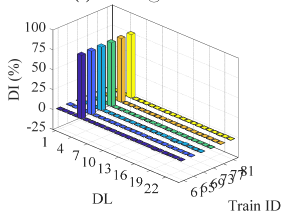

(f) d=80% @Sensor 20

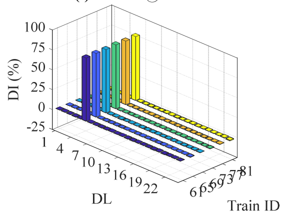

(a) d=80% @Sensor 1


(b) d=80% @Sensor 2


(c) d=80% @Sensor 3

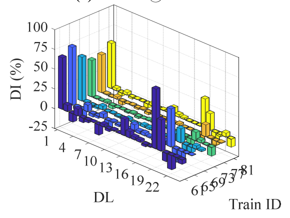

(d) d=80% @Sensor 4


(e) d=80% @Sensor 5

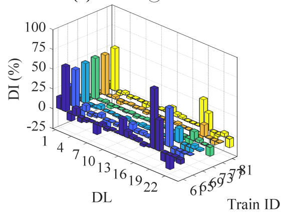

(f) d=80% @Sensor 6

Damage detection with ***Displacement*** responses and DI 50%

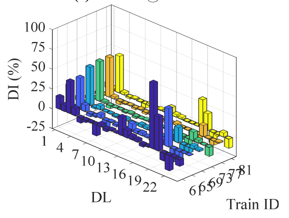
(a) d=50% @Sensor 1



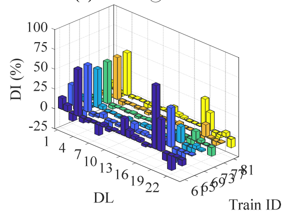
(b) d=50% @Sensor 2



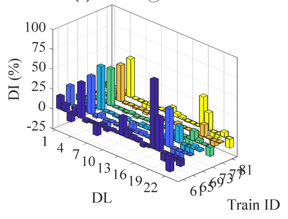
(c) d=50% @Sensor 3



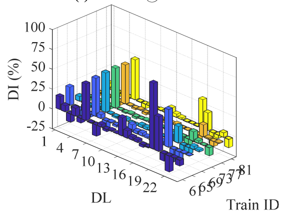
(d) d=50% @Sensor 4



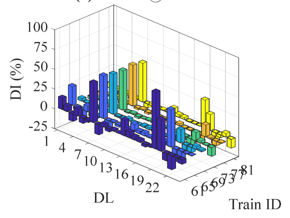
(e) d=50% @Sensor 5



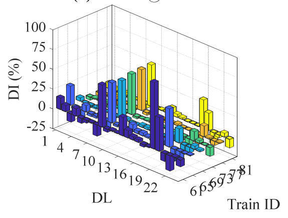
(f) d=50% @Sensor 6



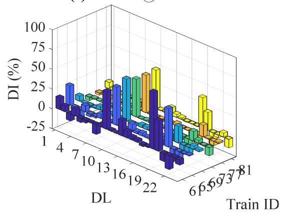
(a) d=50% @Sensor 7



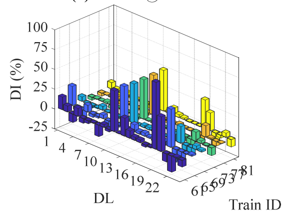
(b) d=50% @Sensor 9



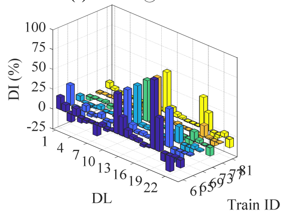
(c) d=50% @Sensor 10



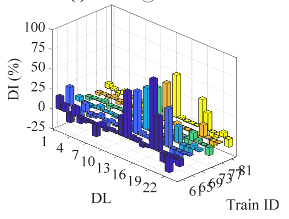
(d) d=50% @Sensor 11



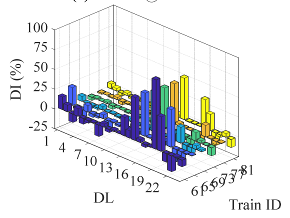
(e) d=50% @Sensor 12



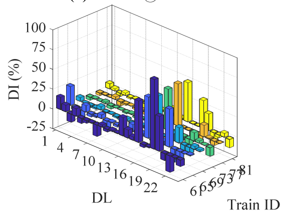
(f) d=50% @Sensor 14



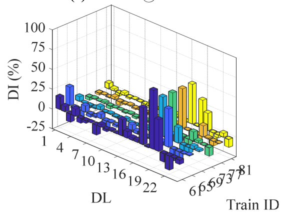
(a) d=50% @Sensor 15



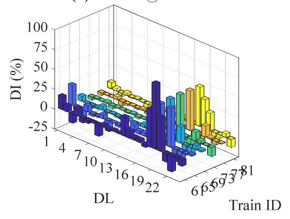
(b) d=50% @Sensor 16



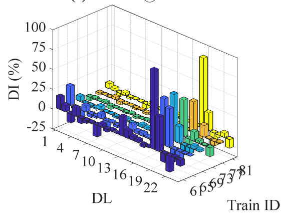
(c) d=50% @Sensor 17



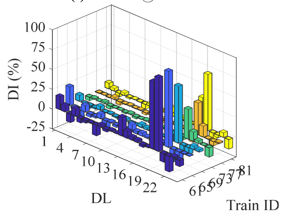
(d) d=50% @Sensor 18



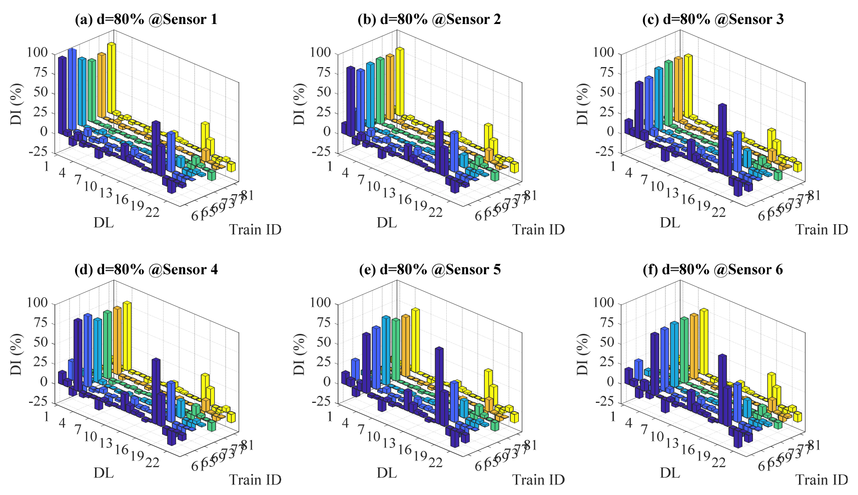
(e) d=50% @Sensor 19

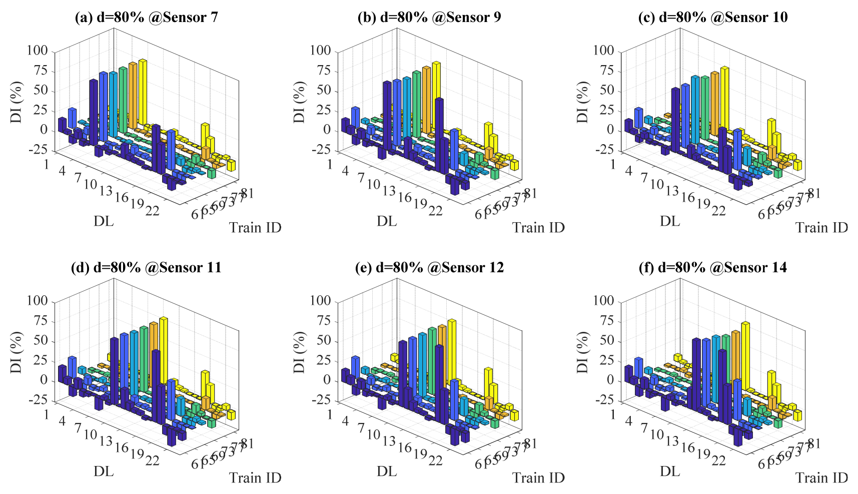


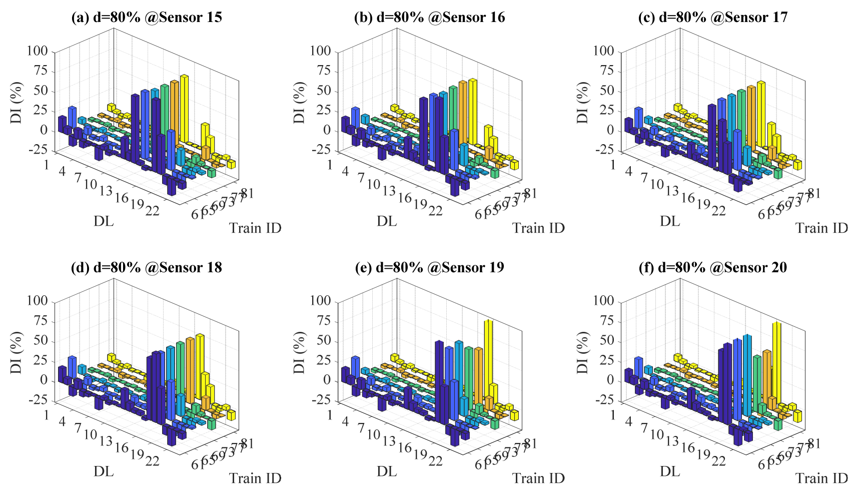
(f) d=50% @Sensor 20



Damage detection with ***<u>Displacement</u>*** responses and DI 80%

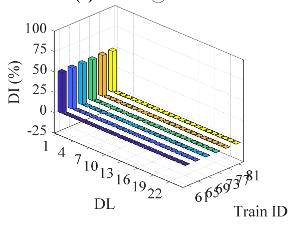




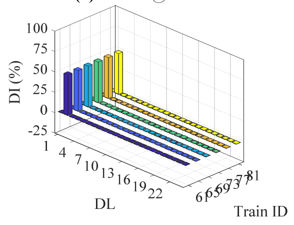


Damage detection with ***Acceleration*** responses and DI 50%

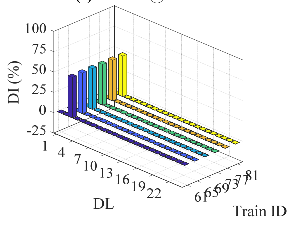
(a) d=50% @Sensor 1



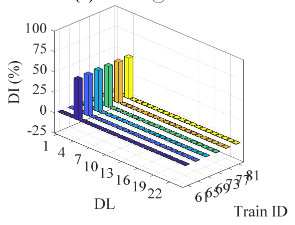
(b) d=50% @Sensor 2



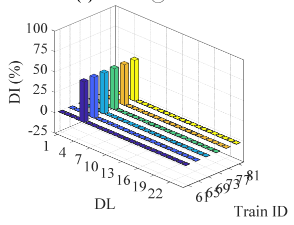
(c) d=50% @Sensor 3



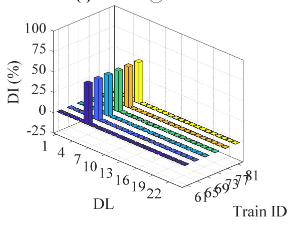
(d) d=50% @Sensor 4



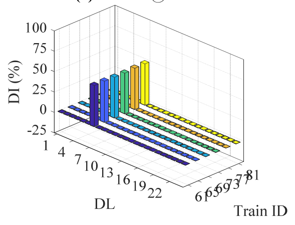
(e) d=50% @Sensor 5



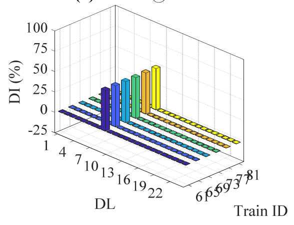
(f) d=50% @Sensor 6



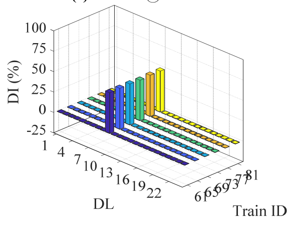
(a) d=50% @Sensor 7



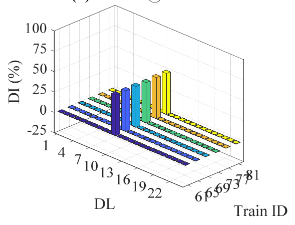
(b) d=50% @Sensor 9



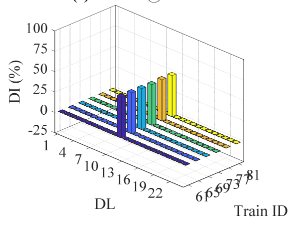
(c) d=50% @Sensor 10



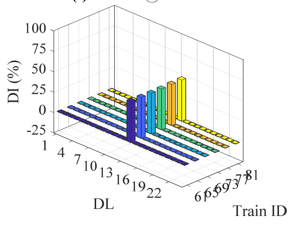
(d) d=50% @Sensor 11



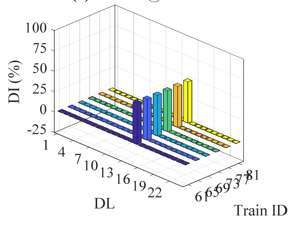
(e) d=50% @Sensor 12



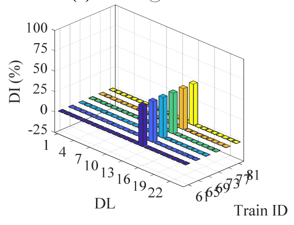
(f) d=50% @Sensor 14



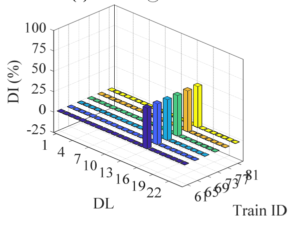
(a) d=50% @Sensor 15



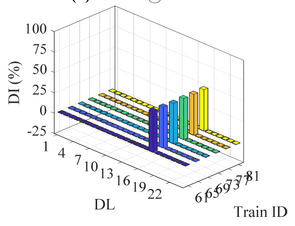
(b) d=50% @Sensor 16



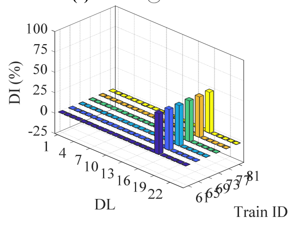
(c) d=50% @Sensor 17



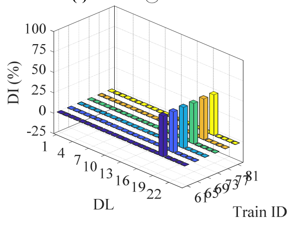
(d) d=50% @Sensor 18



(e) d=50% @Sensor 19

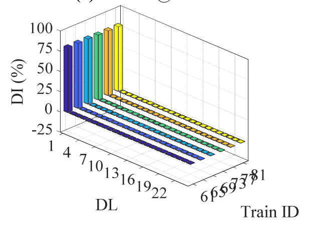


(f) d=50% @Sensor 20

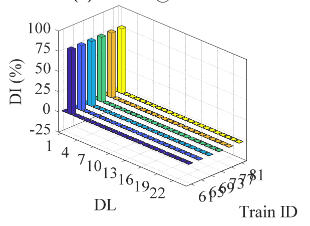


Damage detection with ***Acceleration*** responses and DI 80%

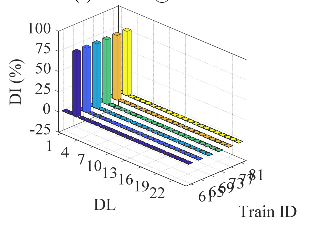
(a) d=80% @Sensor 1



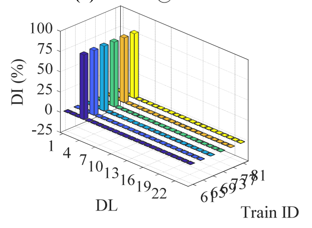
(b) d=80% @Sensor 2



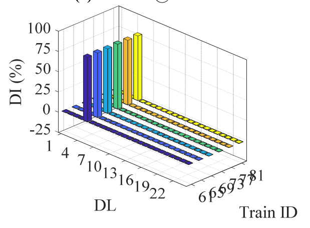
(c) d=80% @Sensor 3



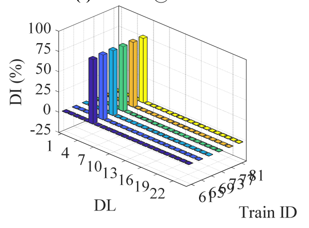
(d) d=80% @Sensor 4



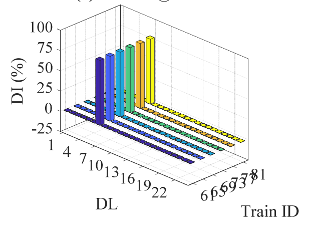
(e) d=80% @Sensor 5



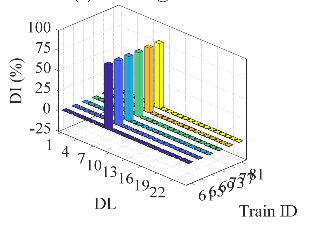
(f) d=80% @Sensor 6



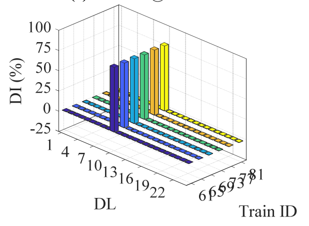
(a) d=80% @Sensor 7



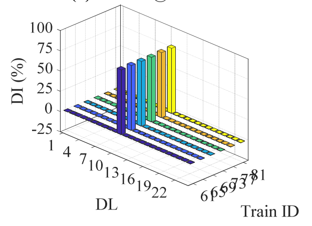
(b) d=80% @Sensor 9



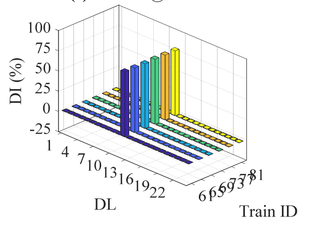
(c) d=80% @Sensor 10



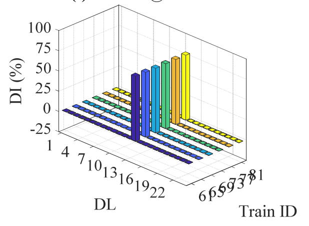
(d) d=80% @Sensor 11



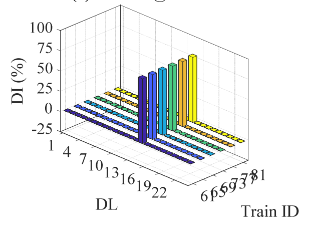
(e) d=80% @Sensor 12



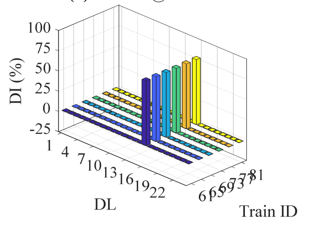
(f) d=80% @Sensor 14



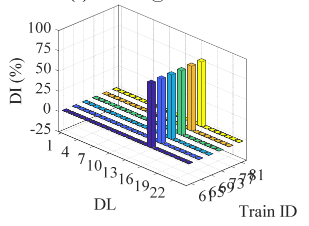
(a) d=80% @Sensor 15



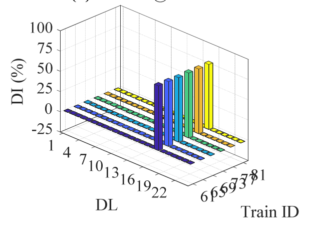
(b) d=80% @Sensor 16



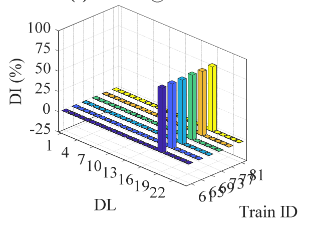
(c) d=80% @Sensor 17



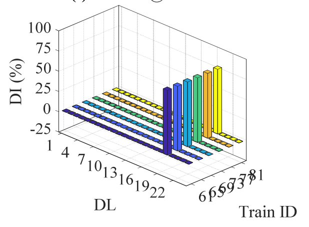
(d) d=80% @Sensor 18



(e) d=80% @Sensor 19



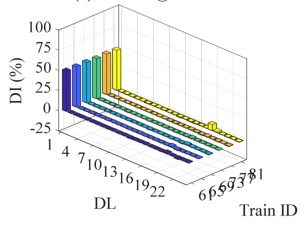
(f) d=80% @Sensor 20



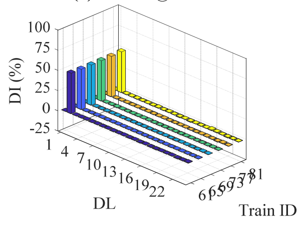
2.2 Instrumentation plan 2

Damage detection with *Strain* responses and DI 50%

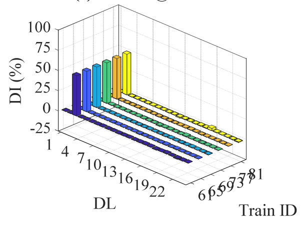
(a) d=50% @Sensor 1



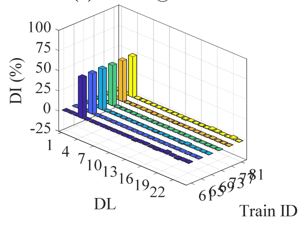
(b) d=50% @Sensor 2



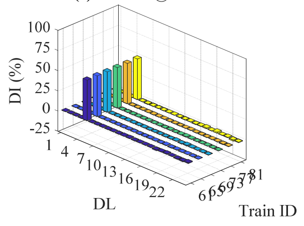
(c) d=50% @Sensor 3



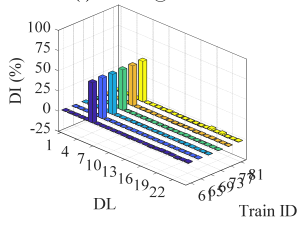
(d) d=50% @Sensor 4



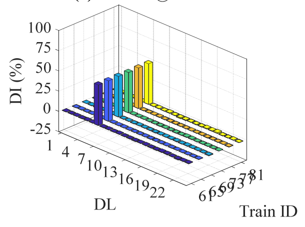
(e) d=50% @Sensor 5



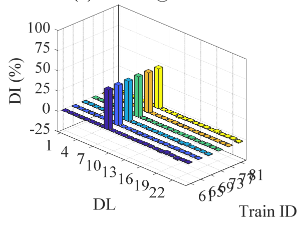
(f) d=50% @Sensor 6



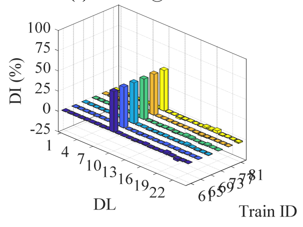
(a) d=50% @Sensor 7



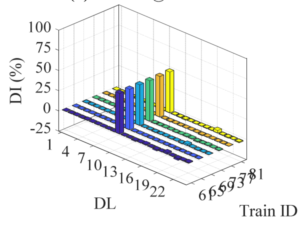
(b) d=50% @Sensor 9



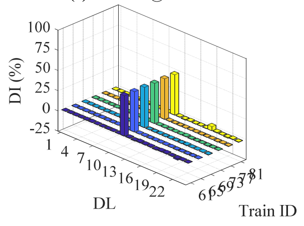
(c) d=50% @Sensor 10



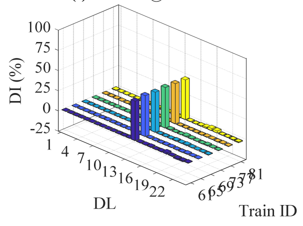
(d) d=50% @Sensor 11



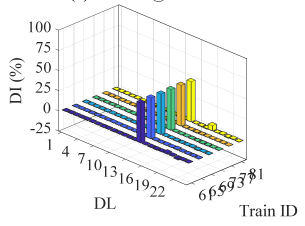
(e) d=50% @Sensor 12



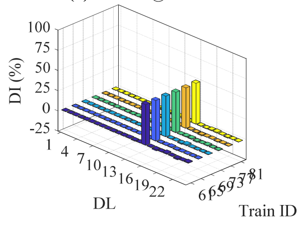
(f) d=50% @Sensor 14



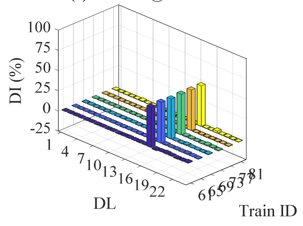
(a) d=50% @Sensor 15



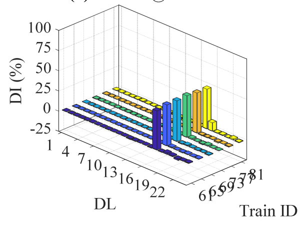
(b) d=50% @Sensor 16



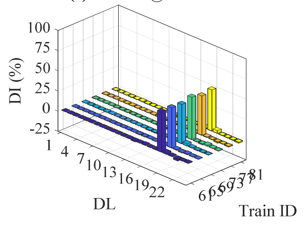
(c) d=50% @Sensor 17



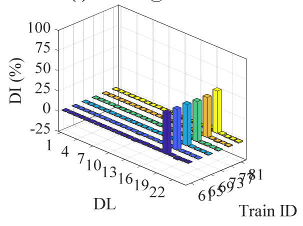
(d) d=50% @Sensor 18



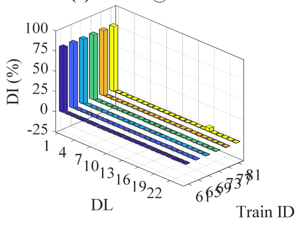
(e) d=50% @Sensor 19



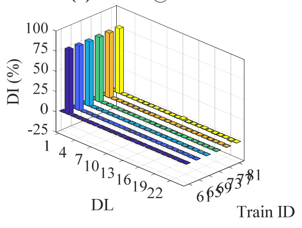
(f) d=50% @Sensor 20



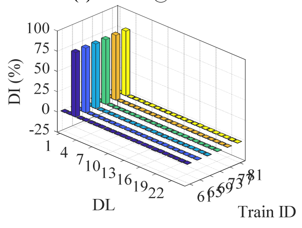
Damage detection with ***Strain*** responses and DI 80%



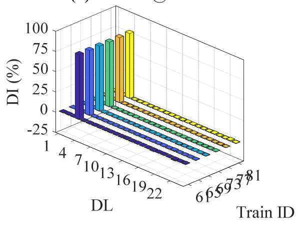
(a) d=80% @Sensor 1



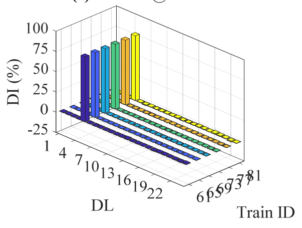
(b) d=80% @Sensor 2



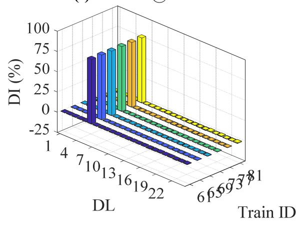
(c) d=80% @Sensor 3



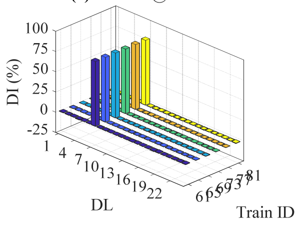
(d) d=80% @Sensor 4



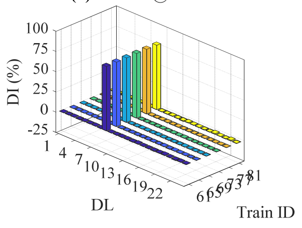
(e) d=80% @Sensor 5



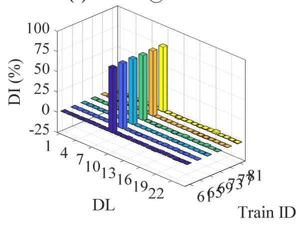
(f) d=80% @Sensor 6



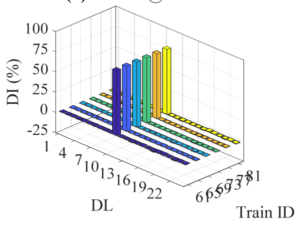
(a) d=80% @Sensor 7



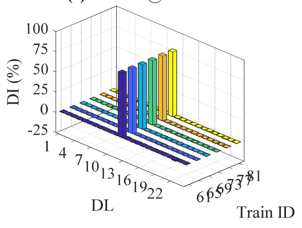
(b) d=80% @Sensor 9



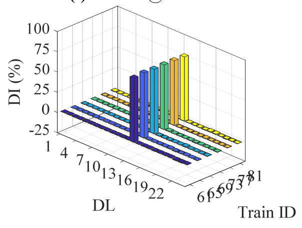
(c) d=80% @Sensor 10



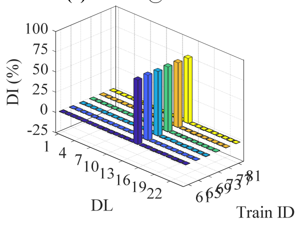
(d) d=80% @Sensor 11



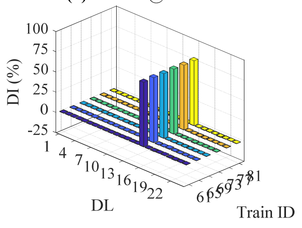
(e) d=80% @Sensor 12



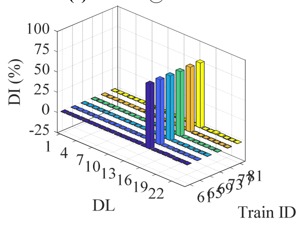
(f) d=80% @Sensor 14



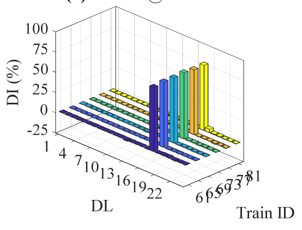
(a) d=80% @Sensor 15



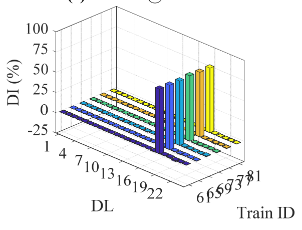
(b) d=80% @Sensor 16



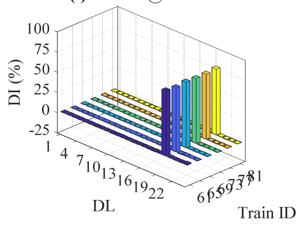
(c) d=80% @Sensor 17



(d) d=80% @Sensor 18



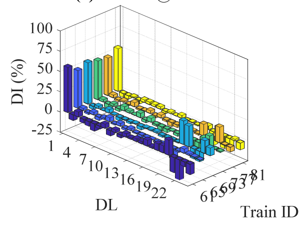
(e) d=80% @Sensor 19



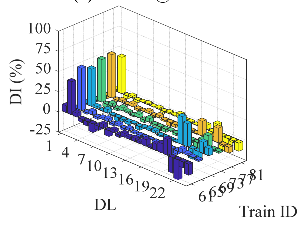
(f) d=80% @Sensor 20

Damage detection with ***<u>Rotation</u>*** responses and DI 50%

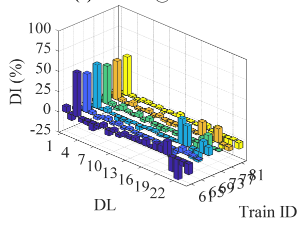
(a) d=50% @Sensor 1



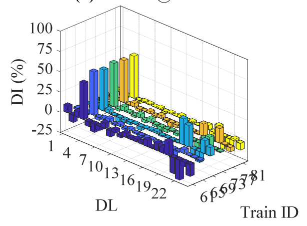
(b) d=50% @Sensor 2



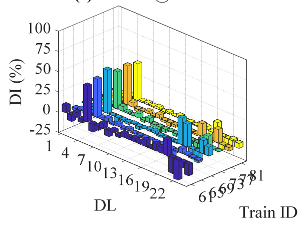
(c) d=50% @Sensor 3



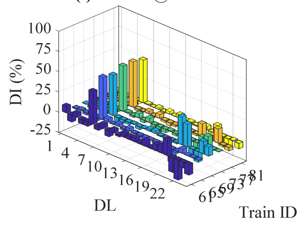
(d) d=50% @Sensor 4



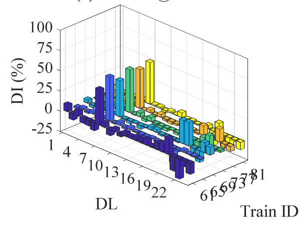
(e) d=50% @Sensor 5



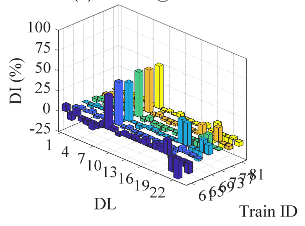
(f) d=50% @Sensor 6



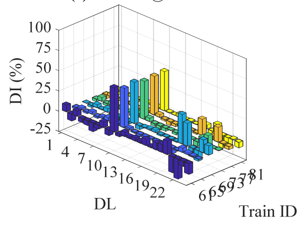
(a) d=50% @Sensor 7



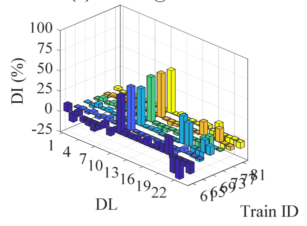
(b) d=50% @Sensor 9



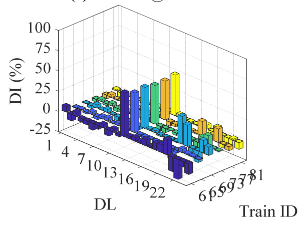
(c) d=50% @Sensor 10



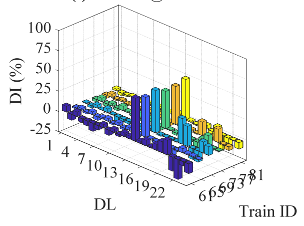
(d) d=50% @Sensor 11



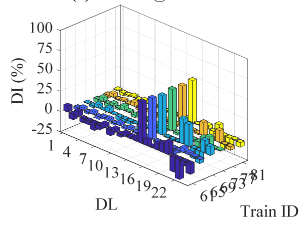
(e) d=50% @Sensor 12



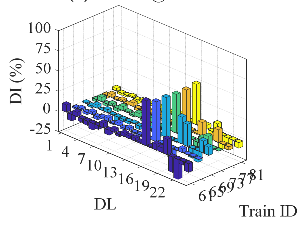
(f) d=50% @Sensor 14



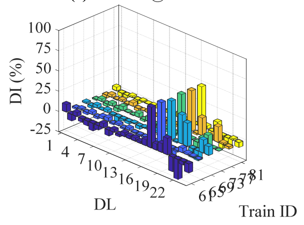
(a) d=50% @Sensor 15



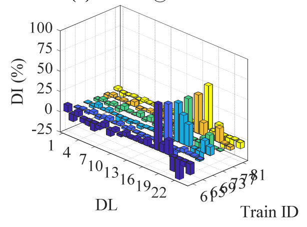
(b) d=50% @Sensor 16



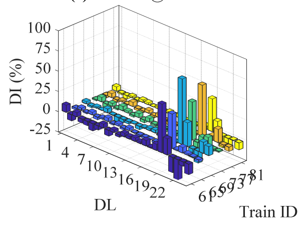
(c) d=50% @Sensor 17



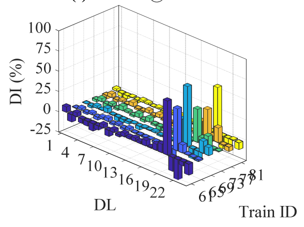
(d) d=50% @Sensor 18



(e) d=50% @Sensor 19

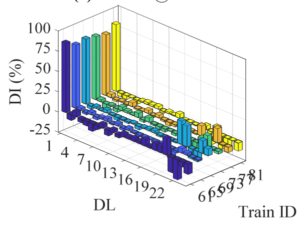


(f) d=50% @Sensor 20

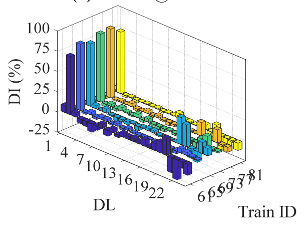


Damage detection with ***Rotation*** responses and DI 80%

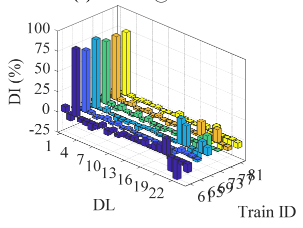
(a) d=80% @Sensor 1



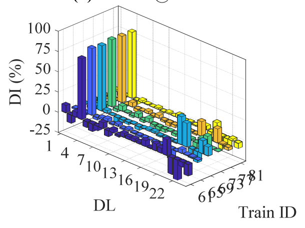
(b) d=80% @Sensor 2



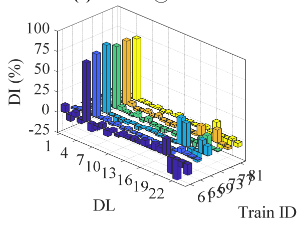
(c) d=80% @Sensor 3



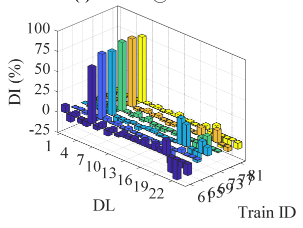
(d) d=80% @Sensor 4



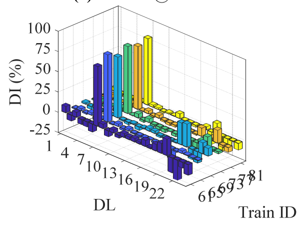
(e) d=80% @Sensor 5



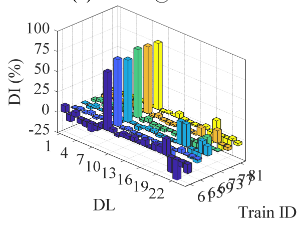
(f) d=80% @Sensor 6



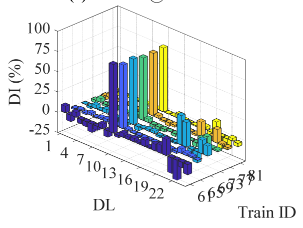
(a) d=80% @Sensor 7



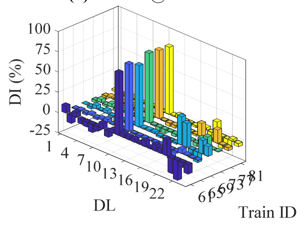
(b) d=80% @Sensor 9



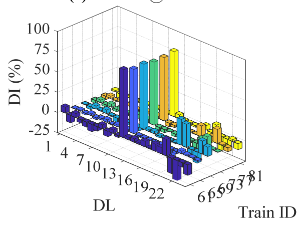
(c) d=80% @Sensor 10



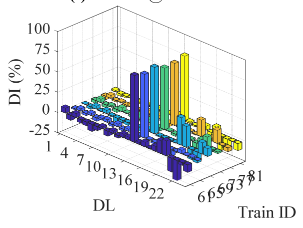
(d) d=80% @Sensor 11



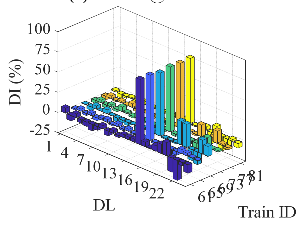
(e) d=80% @Sensor 12



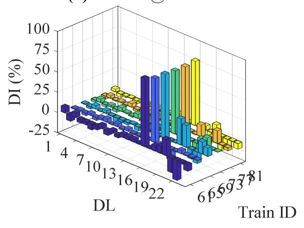
(f) d=80% @Sensor 14



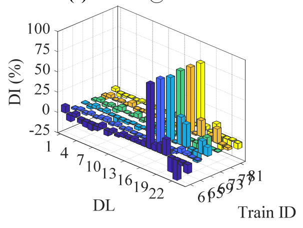
(a) d=80% @Sensor 15



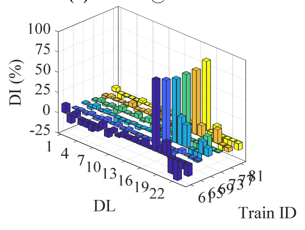
(b) d=80% @Sensor 16



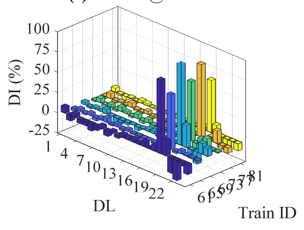
(c) d=80% @Sensor 17



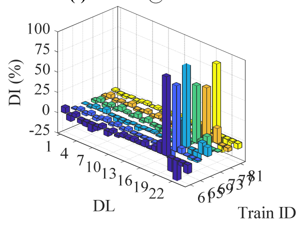
(d) d=80% @Sensor 18



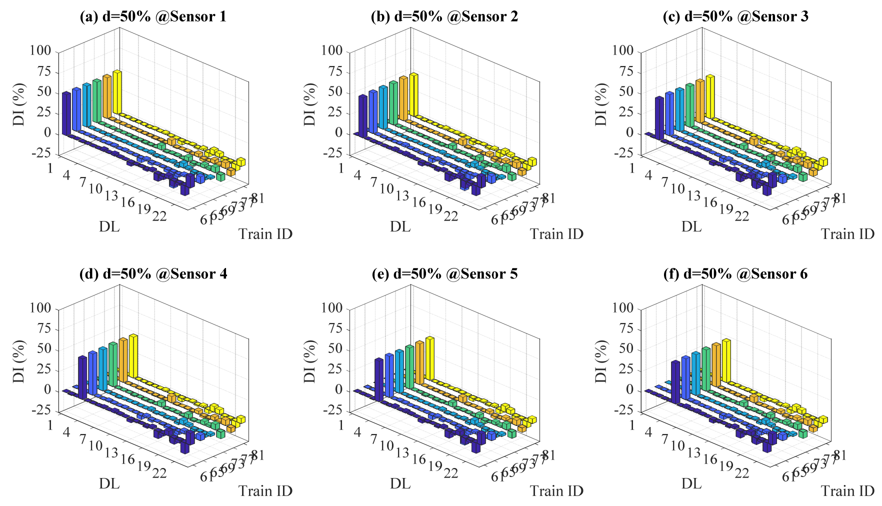
(e) d=80% @Sensor 19

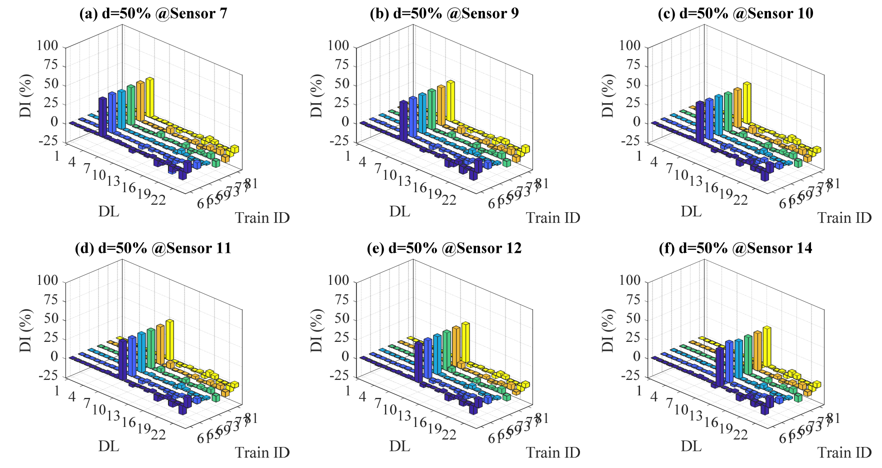


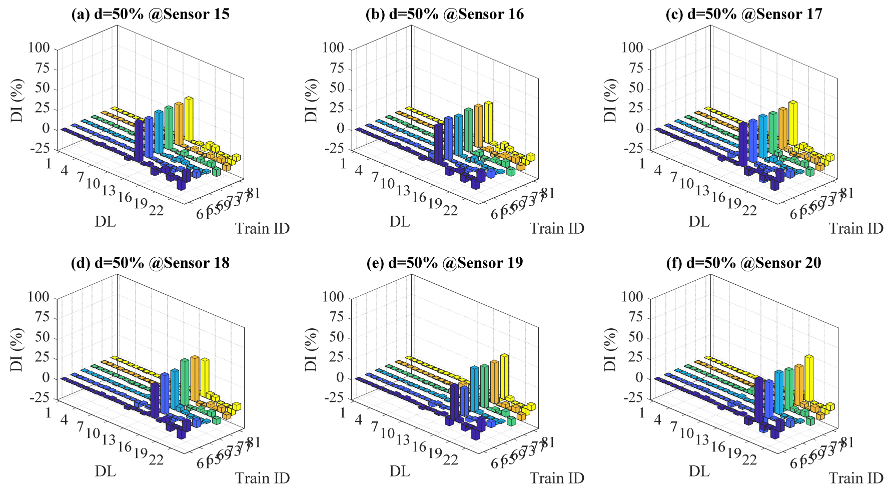
(f) d=80% @Sensor 20



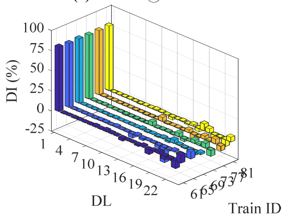
Damage detection with ***Displacement*** responses and DI 50%

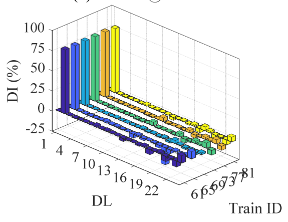


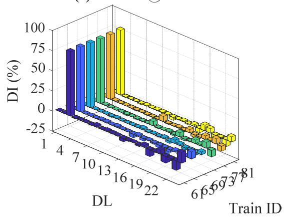


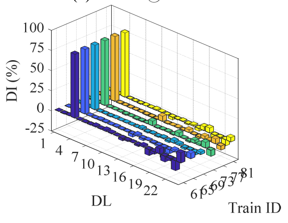


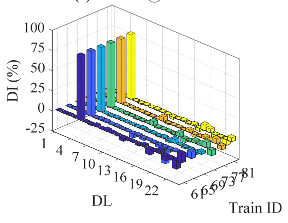
Damage detection with *Displacement* responses and DI 80%

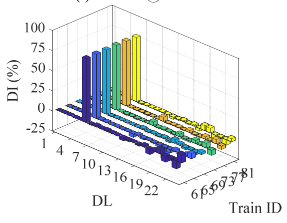

(a) d=80% @Sensor 1

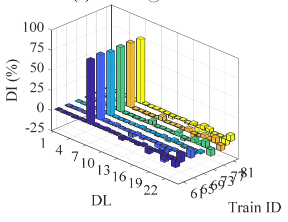

(b) d=80% @Sensor 2

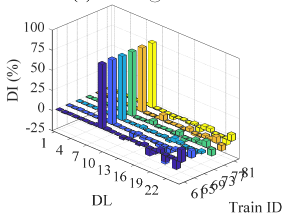

(c) d=80% @Sensor 3

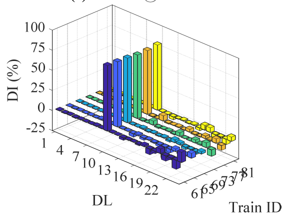

(d) d=80% @Sensor 4

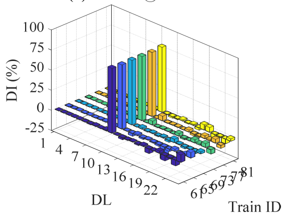

(e) d=80% @Sensor 5

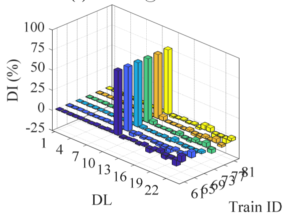

(f) d=80% @Sensor 6

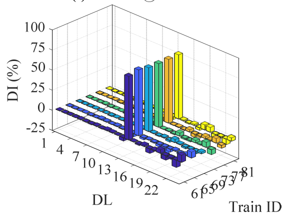

(a) d=80% @Sensor 7

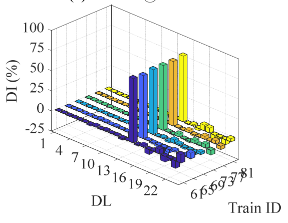

(b) d=80% @Sensor 9

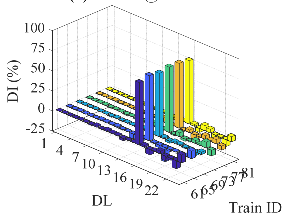

(c) d=80% @Sensor 10

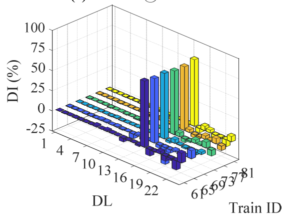

(d) d=80% @Sensor 11

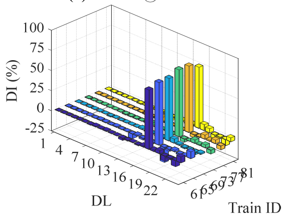

(e) d=80% @Sensor 12

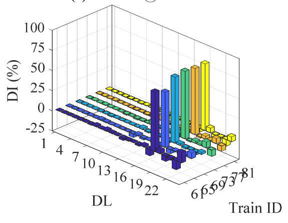

(f) d=80% @Sensor 14

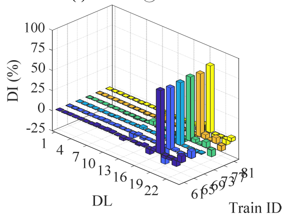

(a) d=80% @Sensor 15


(b) d=80% @Sensor 16


(c) d=80% @Sensor 17

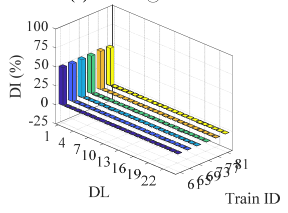

(d) d=80% @Sensor 18


(e) d=80% @Sensor 19

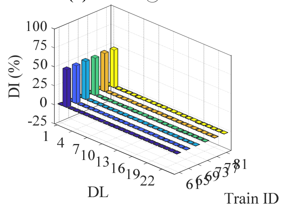

(f) d=80% @Sensor 20

Damage detection with ***Acceleration*** responses

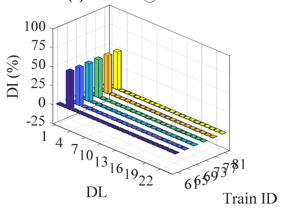
(a) d=50% @Sensor 1



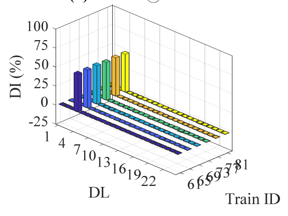
(b) d=50% @Sensor 2



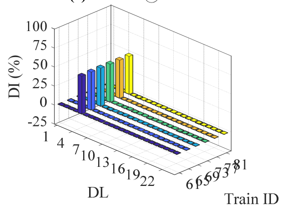
(c) d=50% @Sensor 3



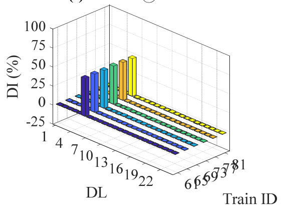
(d) d=50% @Sensor 4



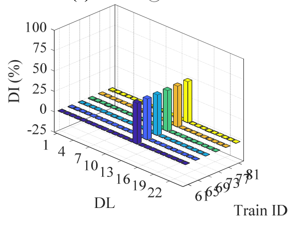
(e) d=50% @Sensor 5



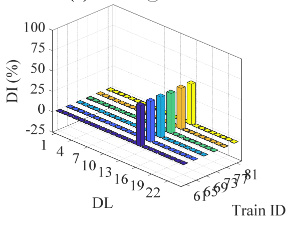
(f) d=50% @Sensor 6



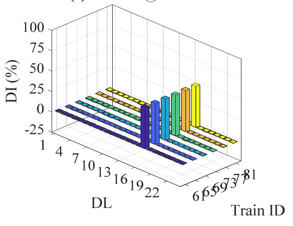
(a) d=50% @Sensor 15



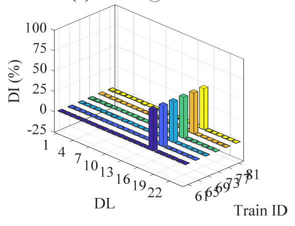
(b) d=50% @Sensor 16



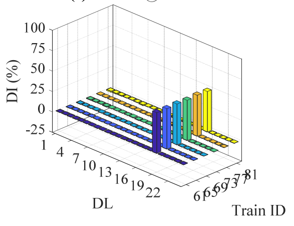
(c) d=50% @Sensor 17



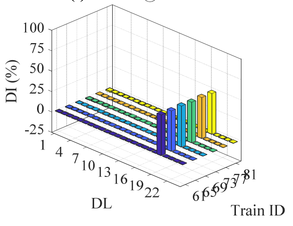
(d) d=50% @Sensor 18



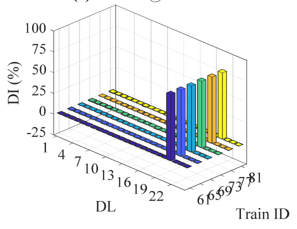
(e) d=50% @Sensor 19



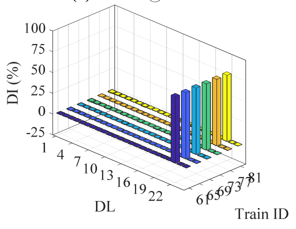
(f) d=50% @Sensor 20



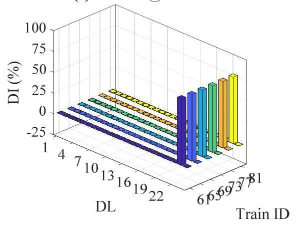
(a) d=80% @Sensor 21



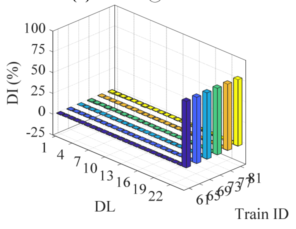
(b) d=80% @Sensor 22



(c) d=80% @Sensor 23

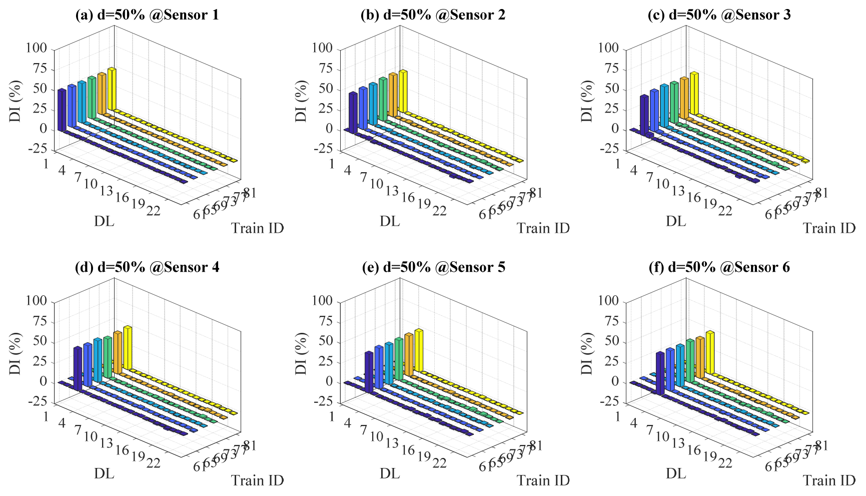


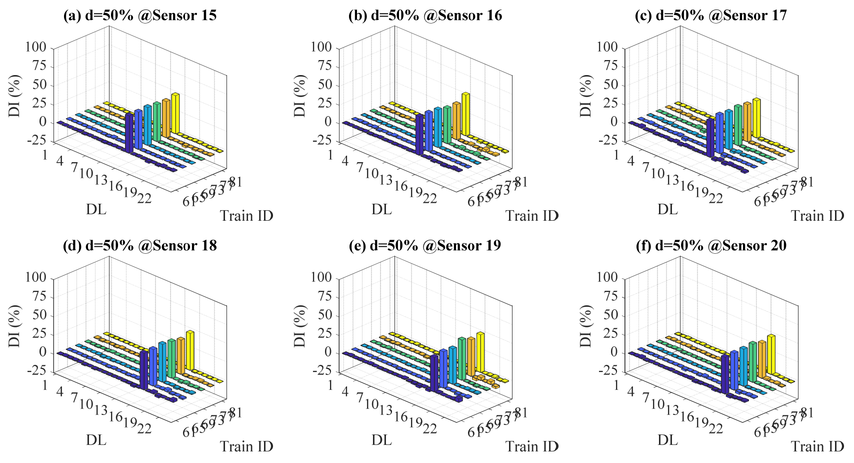
(d) d=80% @Sensor 24

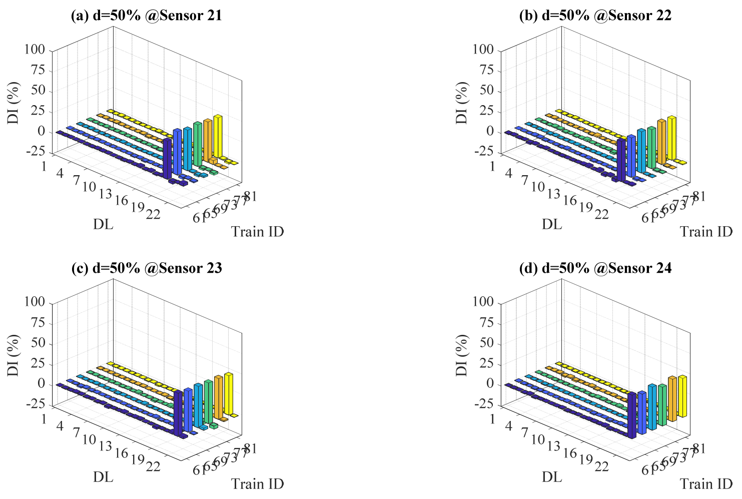


2.3 Instrumentation plan 3

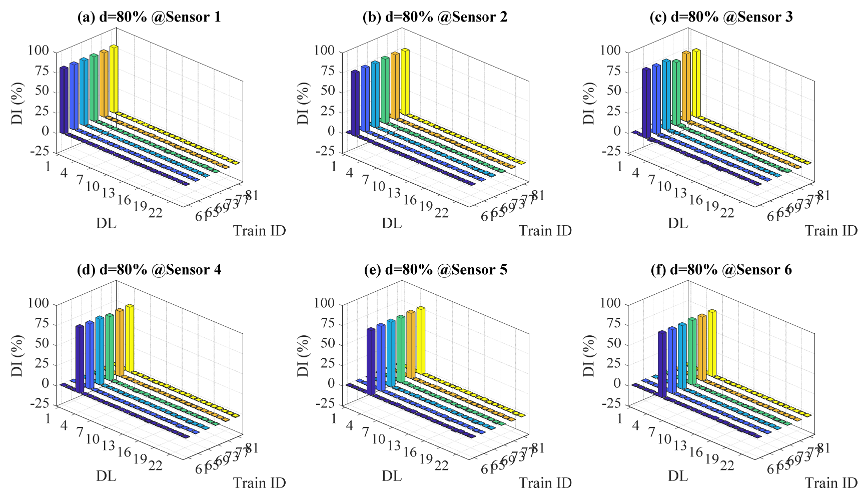
Damage detection with *Strain* responses and DI 50%

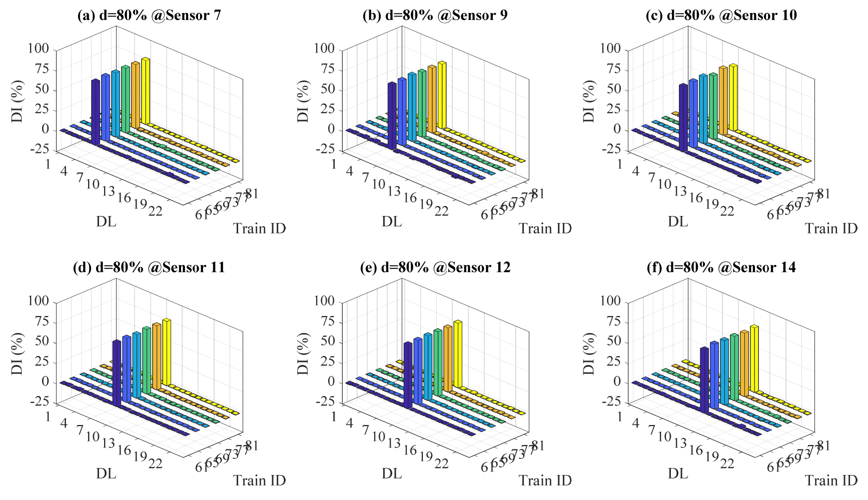


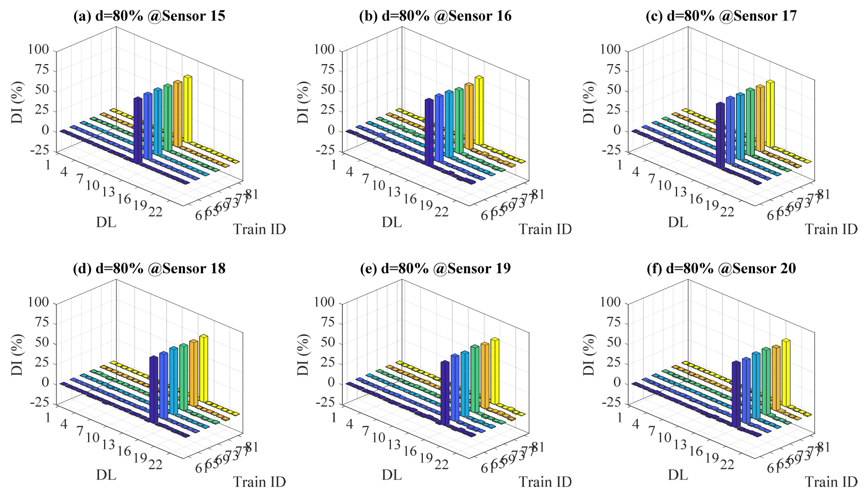




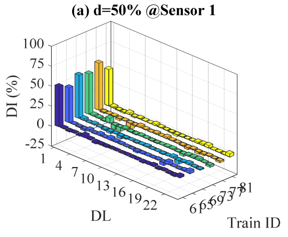
Damage detection with ***Strain*** responses and DI 80%

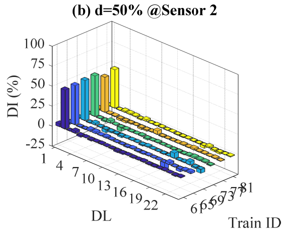


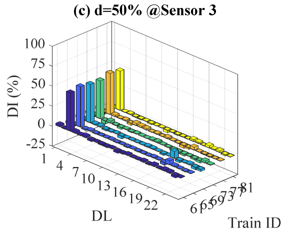


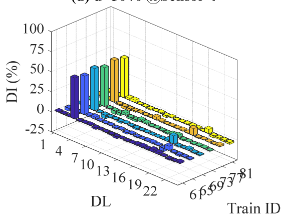


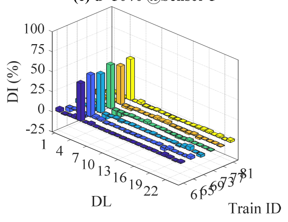
Damage detection with ***Rotation*** responses and DI 50%

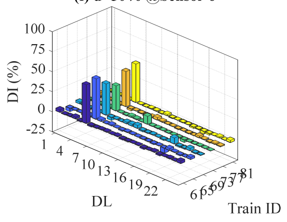
(a) d=50% @Sensor 1


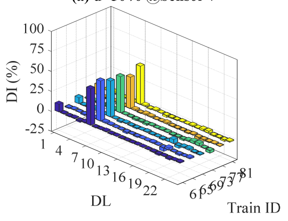
(b) d=50% @Sensor 2


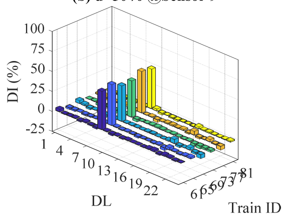
(c) d=50% @Sensor 3


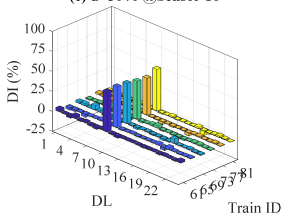
(d) d=50% @Sensor 4


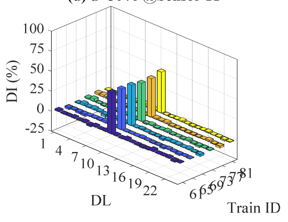
(e) d=50% @Sensor 5


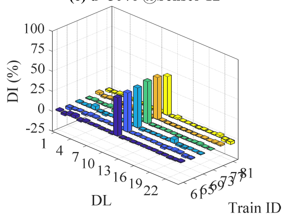
(f) d=50% @Sensor 6


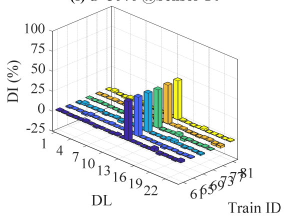
(a) d=50% @Sensor 7


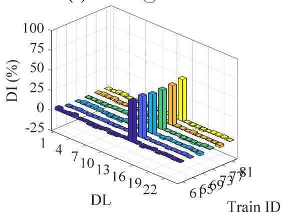
(b) d=50% @Sensor 9


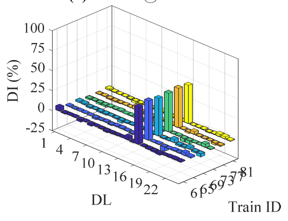
(c) d=50% @Sensor 10


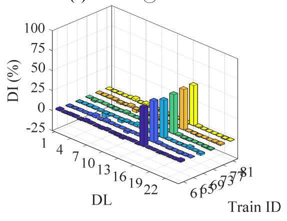
(d) d=50% @Sensor 11


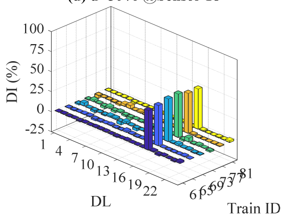
(e) d=50% @Sensor 12


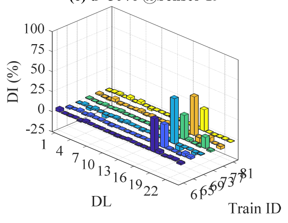
(f) d=50% @Sensor 14


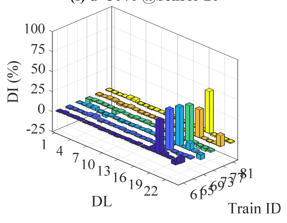
(a) d=50% @Sensor 15


(b) d=50% @Sensor 16


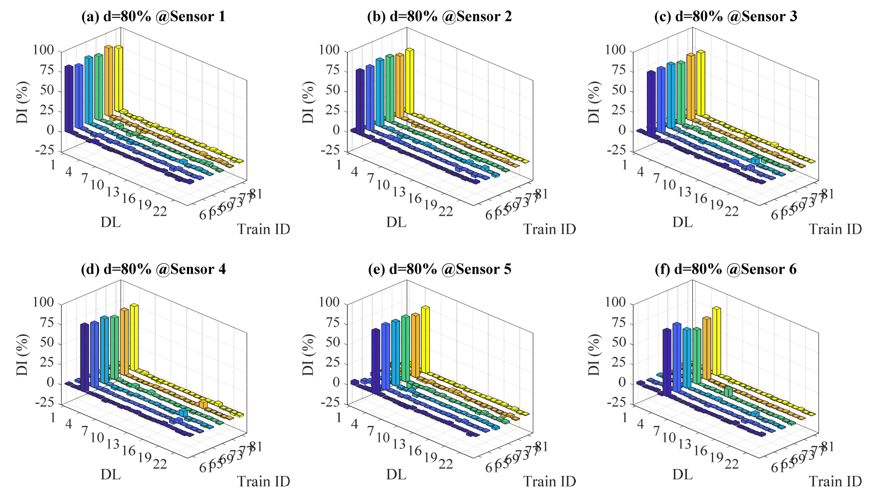
(c) d=50% @Sensor 17


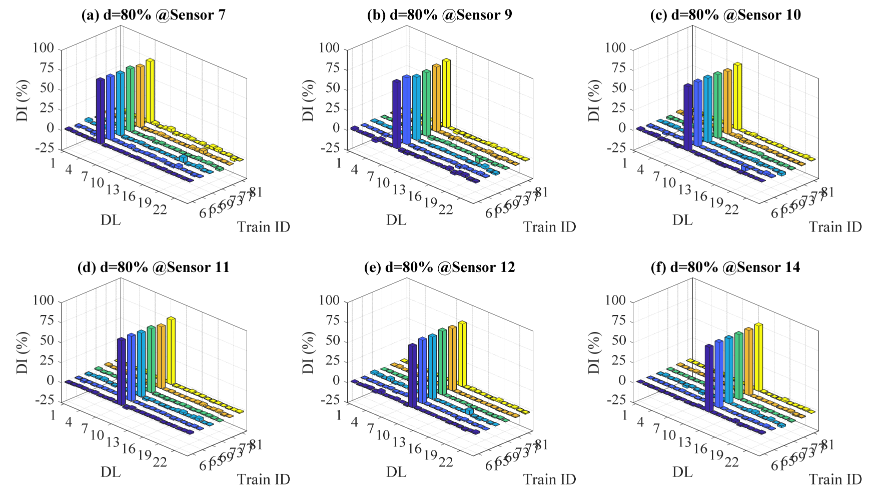
(d) d=50% @Sensor 18


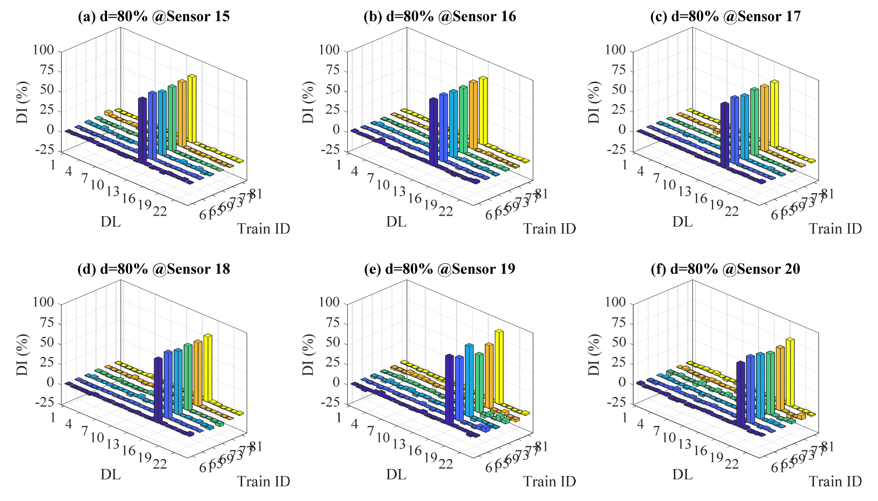
(e) d=50% @Sensor 19


(f) d=50% @Sensor 20


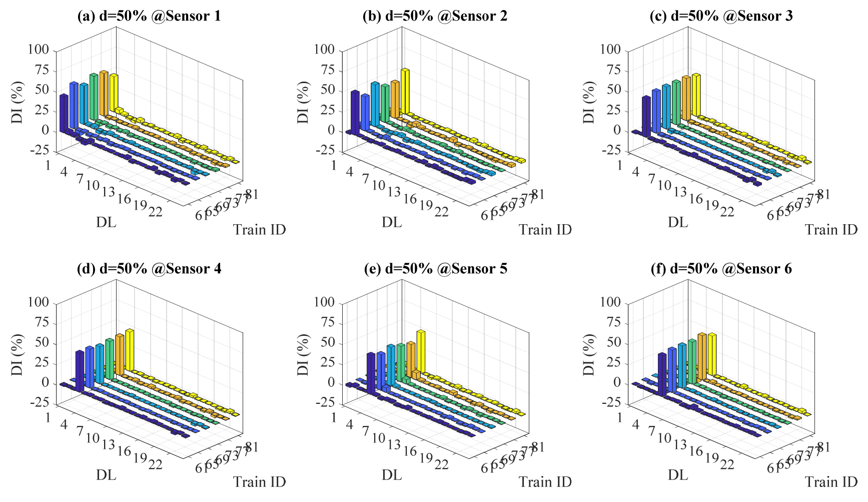
Damage detection with ***Rotation*** responses and DI 80%

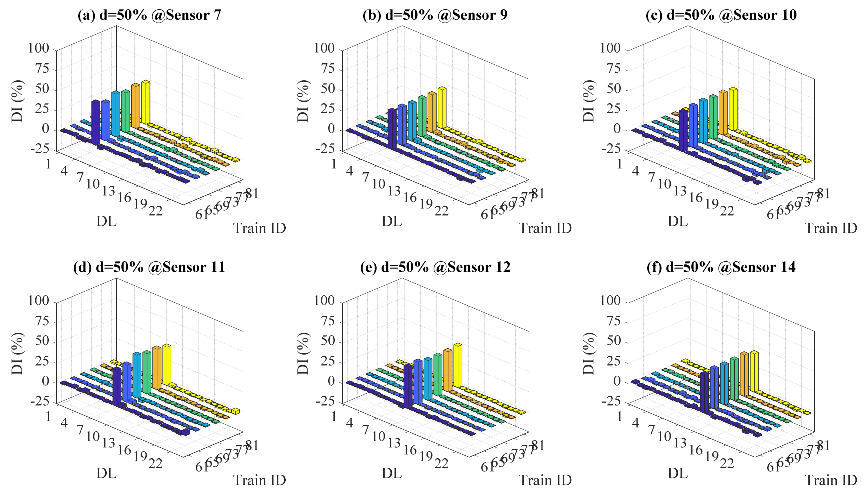


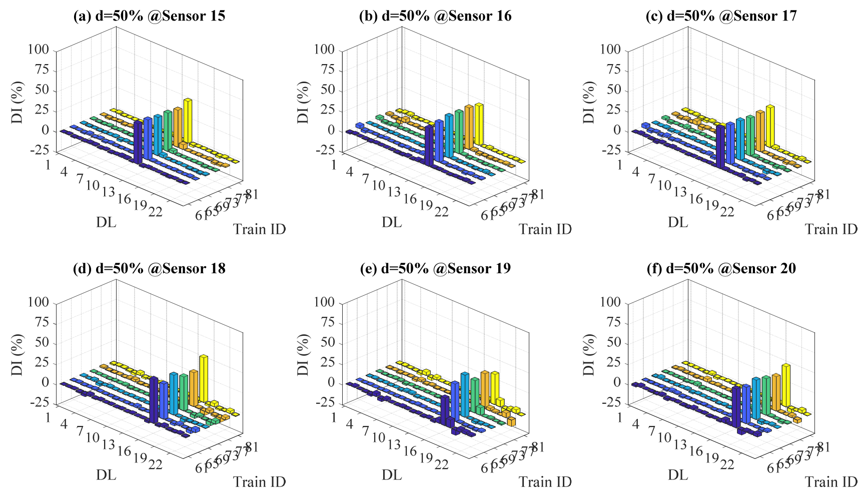




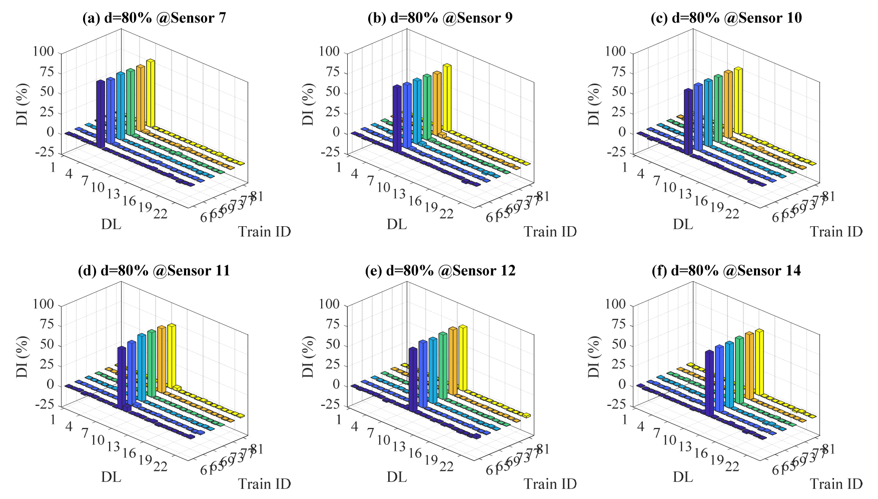
Damage detection with ***Displacement*** responses and DI 50%

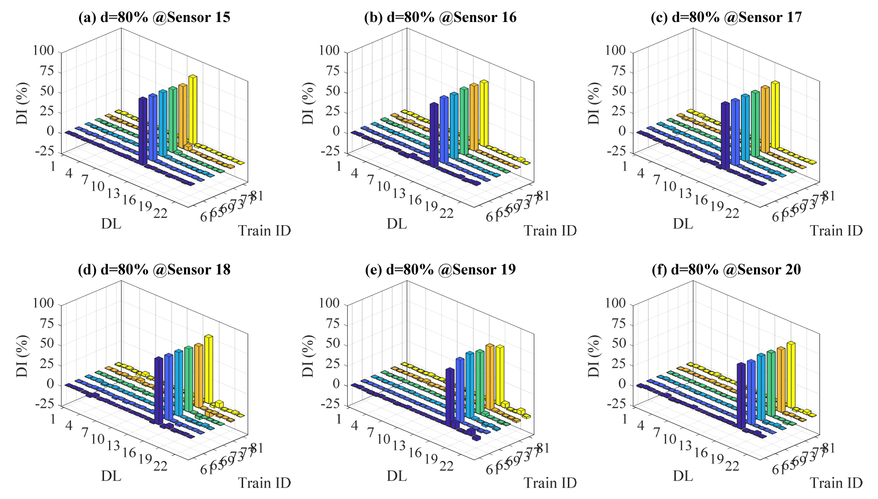


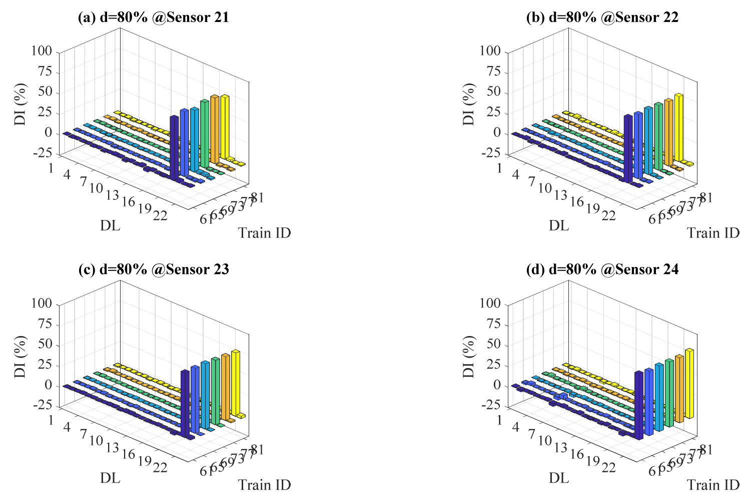




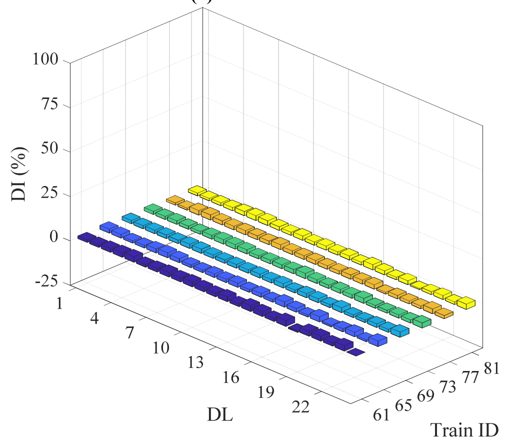
Damage detection with *Displacement* responses and DI 80%



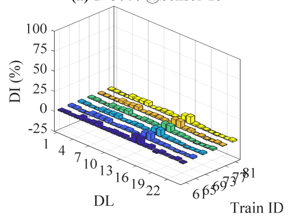




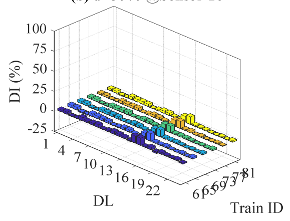
Damage detection with ***Acceleration*** *res*ponses



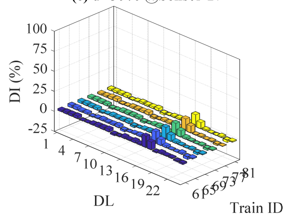
(a) d=0% All Trains



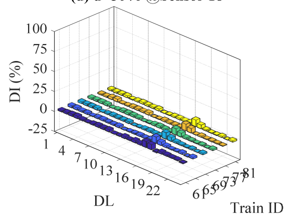
(a) d=50% @Sensor 15



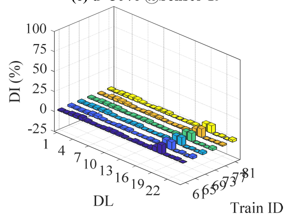
(b) d=50% @Sensor 16



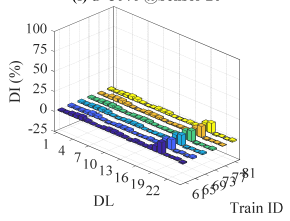
(c) d=50% @Sensor 17



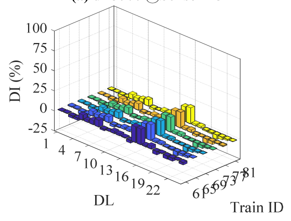
(d) d=50% @Sensor 18



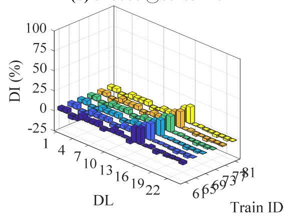
(e) d=50% @Sensor 19



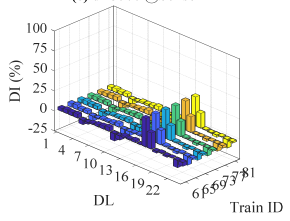
(f) d=50% @Sensor 20



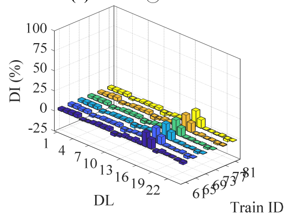
(a) d=80% @Sensor 15



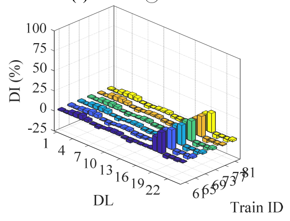
(b) d=80% @Sensor 16



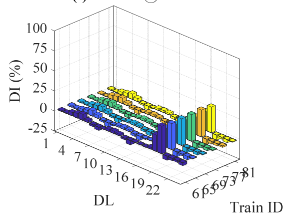
(c) d=80% @Sensor 17



(d) d=80% @Sensor 18



(e) d=80% @Sensor 19



(f) d=80% @Sensor 20

Appendix 2: MATLAB Coding Used to Simulate Trains, Damage and Extracting Internal Forces

In this appendix, MATLAB and SAP2000 OAPI connection MATLAB code is shown. This MATLAB code used in simulating trains, damage and extracting internal forces at the selected locations.

Clean-up the workspace & command window

```
clear all;
close all;
```

Definging Files Paths and Names

Excel file for Reading and summarizing loads [Type the Excel file name and extension]

```
ExcelFileName = ('HBD_WILD data for Br. 86.49 Columbus (8-8-16).xlsx');

% SAP2000, $2k file path and file name [% Insert the SAp model location and
% Name]

FilePath = 'E:\PHD\Reading And Modelling Trains Automatically\Modelling WILD Trains, Sorted RMS,
Getting Results and POMS';
ModelName = 'TRUSS (146 C-C)_Model (3)_WILD 81 Trains';
```

Starting time

```
tic
```

Required Input Fields

```
% Fixed end ratio for healthy state condition
EFR_H = 7111000; %kip-in @ end fixity ratio of 67% of continuous

% Desired damage intensities
DI = 10:10:100;         % as percentage of Healthy State - Change each iteration
EFR_D = ((100-DI)/100)*EFR_H; %kip-in

Nsensors = 20;               %Number of sensors or monitored locations
N_Axle_Model = 50;        %Type the number of axles to be used in the Model
N_TRN_sorted_RMS = 10; %N trains to used in modeling when TRNS sorted by RMS
LoadDurationSAP = 12; % defined loading duration in SAP seconds
LoadDisSAP = 0.005;   % Defined time step in SAP seconds
Nrows = LoadDurationSAP/LoadDisSAP+1;% Number of rows in each load case output
```

Section Prop for stringers

```
Area=46.438;    %in^2   Stringer cross sectional area
Sbot=735.67;    %in^3   Stringer section bottom modulus
```

Loading the Excel File

```
WheelLoads=readtable(ExcelFileName);
```

Extracting Unique trains ID, and Count

```
[TRN_ID_Unique,ia,idx] = unique(WheelLoads.PSNG_SMRY_ID,'stable');

%Number of trains by [(counting the AXLE distance of 0.000)/2]
TRN_Count =(nnz(WheelLoads.AXLE_DIST==0))/2;
```

SnapShotMatrix, POMs_1 and Damage Intensities empty matrices

POMs zeros matrix having dimensions [Nsensors*(Ntrains*Nsensors)]

```
POMs_1_RMS=zeros(Nsensors,N_TRN_sorted_RMS*((length(DI)*Nsensors)+1));%matrix
[Ndamage*Nsensors+1]

% SnapShotMatrix empty matrix
SnapShotMatrix_Initial=zeros(int16(Nrows),Nsensors,N_TRN_sorted_RMS*((length(DI)*Nsensors)+1));

% Damage intensity matrix
D_Vectors = zeros(Nsensors,N_TRN_sorted_RMS*((length(DI)*Nsensors)+1));%matrix
[Ndamage*Nsensors+1]
```

Creating a Matrix of Zeros - Filled Later - to Summarize Trains Data

```
Trains_Summary=zeros(length(TRN_ID_Unique),9);
```

Dividing the Whole Excel table into Different Unique Trains

This provides a matrix containing train data for each unique train Trains sequence in the list are sorted based on time

```
for TR=1:length(TRN_ID_Unique)
    TRN_DATA_ByTime{TR}=WheelLoads(WheelLoads.PSNG_SMRY_ID==TRN_ID_Unique(TR),:);
end
```

Extracting Important Statistics

```
for TR=1:length(TRN_ID_Unique)
    %Train Axle vertical load = Wheel 1 load + Wheel 2 load
    TRN_AXLE_VLoad{TR}=(TRN_DATA_ByTime{TR}.WHL_VERT_AVG_WGT(TRN_DATA_ByTime{TR}.WHL_ID==1))+...
        (TRN_DATA_ByTime{TR}.WHL_VERT_AVG_WGT(TRN_DATA_ByTime{TR}.WHL_ID==2));
    %Train Maximum axle vertical load
    TRN_AXLE_max_VLoad{TR}=max(TRN_AXLE_VLoad{TR});
    %Train minimum axle vertical load
    TRN_AXLE_min_VLoad{TR}=min(TRN_AXLE_VLoad{TR});
    %Train mean / average axle vertical load
    TRN_AXLE_mean_VLoad{TR}=mean(TRN_AXLE_VLoad{TR});
    %Train axle equivalent uniform load (Sum axle loads / Train length)
    TRN_AXLE_EQ_ULoad{TR}=(sum(TRN_AXLE_VLoad{TR}))/(sum(TRN_DATA_ByTime{TR}.AXLE_DIST)/12)*2;
    %Train average axle distance
    TRN_AXLE_mean_DIST{TR}= (sum(TRN_DATA_ByTime{TR}.AXLE_DIST)/2/12)/...
        (max(TRN_DATA_ByTime{TR}.AXLE_ON_TRN_SEQ));
    %Car Average length
    Car_mean_Length{TR}= (sum(TRN_DATA_ByTime{TR}.AXLE_DIST)/2/12)/...
        (max(TRN_DATA_ByTime{TR}.EQMT_ON_TRN_SEQ));
    %Train Symbol
    TRN_SYMBB{TR}= TRN_DATA_ByTime{TR}.TRN_SYMB(1,1);
end
```

Filling in the train summary as a matrix (numbers)

```
for TR=1:length(TRN_ID_Unique)
    %Filling in trains PSNG_SMRY_ID
    Trains_Summary(:,1)=TRN_ID_Unique;
    %Filling in trains ID
    Trains_Summary(TR,2)=TRN_DATA_ByTime{TR}.TRN_ID(1,1);
    %Filling in number of cars for each train
    Trains_Summary(TR,3)=max(TRN_DATA_ByTime{TR}.EQMT_ON_TRN_SEQ);
    %Filling in Train Maximum axle vertical load
    Trains_Summary(TR,4)=TRN_AXLE_max_VLoad{TR};
    %Filling in Train Minimum axle vertical load
    Trains_Summary(TR,5)=TRN_AXLE_min_VLoad{TR};
    %Filling in Train MEAN axle vertical load
    Trains_Summary(TR,6)=TRN_AXLE_mean_VLoad{TR};
    %Filling in Train axle equivalent uniform load vertical load
    Trains_Summary(TR,7)=TRN_AXLE_EQ_ULoad{TR};
    %Filling in Train average axle distance
    Trains_Summary(TR,8)=TRN_AXLE_mean_DIST{TR};
    %Filling in Train average car length
    Trains_Summary(TR,9)=Car_mean_Length{TR};
    %Extracting trains symbols as string, not numbers
    TRN_SYMB(TR,1) = TRN_SYMBB{TR}(1,1);
end
```

Extracting Trains Sorted by Equivalent Uniform Load

Sorting Trains_Summary by the equivalent uniform load to get index

```
[~,Original_index] = sortrows(Trains_Summary,7);

% Sorting Data using the above-extracted index vector
for TR=1:length(TRN_ID_Unique)
    % Sorting Trains from Lowest EqUniform to Highest EqUniform
    TRN_DATA_ByUniform{TR}=TRN_DATA_ByTime{Original_index(TR,1)};
end
```

Extracting modeling trains based on the defined number of axles

Then modeling EQ Uniform Load sorts trains for the defined No of axels (i.e., Sum of axle loads/length of the modeled train)

```
for TR=1:length(TRN_ID_Unique)
    % Extracting trains with a reduced number of axles for modeling
    TRN_MOD1{TR}=TRN_DATA_ByUniform{TR}(TRN_DATA_ByUniform{TR}.AXLE_ON_TRN_SEQ <=
N_Axle_Model,:);

    % Extracting modeling trains axle loads (Wheel 1 + Wheel 2)
    TRN_MOD_AXL_LOD{TR}=(TRN_MOD1{TR}.WHL_VERT_AVG_WGT(TRN_MOD1{TR}.WHL_ID==1))+...
        (TRN_MOD1{TR}.WHL_VERT_AVG_WGT(TRN_MOD1{TR}.WHL_ID==2));
    % Modelling trains axle equivalent uniform load (Sum axle loads / Train length)
    TRN_MOD_EqULoad{TR}=(sum(TRN_MOD_AXL_LOD{TR}))/(sum(TRN_MOD1{TR}.AXLE_DIST)/12)*2;

    % Converting the TRN_MOD_EqULoad{i} to a vertical column to be sorted
    % easier
    TRN_MOD_EqULoad_Vector(TR,1)=(TRN_MOD_EqULoad{TR});
end

% Getting modeled trains index vector when sorting by EQ Uniform load
[~,MOD_index]=sortrows(TRN_MOD_EqULoad_Vector);

for TR=1:length(TRN_ID_Unique)
    % Final modeling trains tables sorted by EQ Uniform load (low to high)
    TRN_MOD{TR}=TRN_MOD1{MOD_index(TR,1)};
    % Extracting Wheel 1 loading, trains sorted by EQ Uniform load
    TRN_MOD_WHL1{TR}=TRN_MOD{TR}(TRN_MOD{TR}.WHL_ID == 1,:);
    % Extracting Wheel 2 loading, trains sorted by EQ Uniform load
    TRN_MOD_WHL2{TR}=TRN_MOD{TR}(TRN_MOD{TR}.WHL_ID == 2,:);
    %Summarized Tables for Model Loading Configurations only (Distance,
    %load) sorted by EQ Uniform load (low to high)
    LODG_CONF_MOD{TR}=[(TRN_MOD_WHL1{TR}.AXLE_DIST),(TRN_MOD_WHL1{TR}.WHL_VERT_AVG_WGT),...
       (TRN_MOD_WHL2{TR}.WHL_VERT_AVG_WGT)];
end
```

Sorting modeling trains by RMS of stress matrix generated from former code

Then modeling trains sorted by EQ Uniform Load will be sorted again based on the RMS of stress matrix

```
% Loading Trains ID sorted based on RMS of stresses
load('ID_RMS_Sorted');
ID_SortedBy_RMS = ID_rms;

for TR=1:length(TRN_ID_Unique)
    % Final modeling trains tables sorted by EQ Uniform load (low to high)
    TRN_MOD_RMS{TR}=TRN_MOD{ID_SortedBy_RMS(1,TR)};
    % Extracting Wheel 1 loading, trains sorted by EQ Uniform load
    TRN_MOD_RMS_WHL1{TR}=TRN_MOD_RMS{TR}(TRN_MOD_RMS{TR}.WHL_ID == 1,:);
    % Extracting Wheel 2 loading, trains sorted by EQ Uniform load
    TRN_MOD_RMS_WHL2{TR}=TRN_MOD_RMS{TR}(TRN_MOD_RMS{TR}.WHL_ID == 2,:);
    %Summarized Tables for Model Loading Configurations only (Distance,
    %load) sorted by EQ Uniform load (low to high)

LODG_CONF_MOD{TR}=[(TRN_MOD_RMS_WHL1{TR}.AXLE_DIST),(TRN_MOD_RMS_WHL1{TR}.WHL_VERT_AVG_WGT),...
       (TRN_MOD_RMS_WHL2{TR}.WHL_VERT_AVG_WGT)];
end
```

Selecting the highest stresses RMS Trains to be modelled

Number of trains is predefined above (N_TRN_sorted_RMS)

```
LODG_CONF_RMS_MOD = LODG_CONF_MOD((length(LODG_CONF_MOD)-
N_TRN_sorted_RMS)+1:length(LODG_CONF_MOD));
```

Creating a Summary TABLE as strings to include the TRN SYMB

```
PSNG_SMRY_ID = Trains_Summary(:,1); %PSNG SMRY ID
TRN_ID = Trains_Summary(:,2); %TRN ID
TRN_SYMB = TRN_SYMB(:,1); %TRN SYMB
```

```
CARS_COUNT = Trains_Summary(:,3); %CARS COUNT
TRN_AXLE_maxVL = Trains_Summary(:,4); %AXLE MAXIMUM VERICAL LOAD
TRN_AXLE_minVL = Trains_Summary(:,5); %AXLE MINIMUM VERICAL LOAD
TRN_AXLE_meanVL = Trains_Summary(:,6); %AXLE MEAN VERICAL LOAD
TRN_AXLE_EqUniVL = Trains_Summary(:,7); %AXLE EQ UNIFORM LOAD
TRN_AXLE_meanDIST = Trains_Summary(:,8); %AXLE EQ UNIFORM LOAD
TRN_CAR_meanLENGTH = Trains_Summary(:,9); %AXLE EQ UNIFORM LOAD

%%%%Writing the above data into a TABLE
Trains_Summary_Table = table(PSNG_SMRY_ID,TRN_ID,TRN_SYMB,CARS_COUNT,...
    TRN_AXLE_maxVL, TRN_AXLE_minVL, TRN_AXLE_meanVL, TRN_AXLE_EqUniVL,...
   TRN_AXLE_meanDIST,TRN_CAR_meanLENGTH);

%%%%Sorting the table by the axle equivalent uniform load
Trains_Summary_Table = sortrows(Trains_Summary_Table,8);
```

Trains set Short Summary

Developing a short Summary for the whole trains set

```
Cars_Count_max = max(Trains_Summary(:,3));%max car counts
Cars_Count_min = min(Trains_Summary(:,3));%min car counts
TRN_AXLE_VL_max_kip = max(Trains_Summary(:,4));%maximum axle load (of maximum)
TRN_AXLE_VL_min_kip = min(Trains_Summary(:,5));%minimum axle load (of minimum)
TRN_AXLE_VL_Mean_max_kip = max(Trains_Summary(:,6));%maximum axle mean
TRN_AXLE_VL_Mean_min_kip = min(Trains_Summary(:,6));%minimum axle mean
TRN_AXLE_Uniform_max_kip_ft = max(Trains_Summary(:,7));%maximum uniform load
TRN_AXLE_Uniform_min_kip_ft = min(Trains_Summary(:,7));%minimum uniform load

%%%% Writing summary table
Train_Data_ShortSummary = table(Cars_Count_max, Cars_Count_min,...
    TRN_AXLE_VL_max_kip, TRN_AXLE_VL_min_kip, TRN_AXLE_VL_Mean_max_kip,...
    TRN_AXLE_VL_Mean_min_kip, TRN_AXLE_Uniform_max_kip_ft,...
    TRN_AXLE_Uniform_min_kip_ft);
```

Saving Results

```
save('Trains_Data_Summary','Trains_Summary_Table', 'Train_Data_ShortSummary',...
    'TRN_DATA_ByTime','TRN_DATA_ByUniform','TRN_MOD','TRN_MOD_WHL1',...
    'TRN_MOD_WHL2','LODG_CONF_MOD');
```

Opening SAP2000

Set the following flag to true to manually specify the path to SAP2000.exe

```
%%this allows for a connection to a version of SAP2000 other than the latest installation
%%otherwise the latest installed version of SAP2000 will be launched
SpecifyPath = false;
```

If the above flag is set to true, specify the path to ETABS below

```
ProgramPath = 'C:\Program Files\Computers and Structures\SAP2000 19\SAP2000.exe';
```

Full path to API dll

```
%%set it to the installation folder
APIDLLPath = 'C:\Program Files\Computers and Structures\SAP2000 19\SAP2000v19.dll';
```

Create API helper object

```
a = NET.addAssembly(APIDLLPath);

helper = SAP2000v19.Helper;

helper = NET.explicitCast(helper,'SAP2000v19.cHelper');

if SpecifyPath

    %%create an instance of the SapObject from the specified path

    SapObject = helper.CreateObject(ProgramPath);

else

    %%create an instance of the SapObject from the latest installed ETABS

    SapObject = helper.CreateObjectProgID('CSI.SAP2000.API.SapObject');

end
SapObject = NET.explicitCast(SapObject,'SAP2000v19.cOAPI');

helper = 0;
```

Start Sap2000 application

```
SapObject.ApplicationStart;
```

Create Sap Model object

```
SapModel = NET.explicitCast(SapObject.SapModel,'SAP2000v19.cSapModel');
```

%%

```
for TR=1:length(LODG_CONF_RMS_MOD)
```

Writing Results as Text.$2k file to be imported by SAP

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Writing TABLE:  "VEHICLES 3 - GENERAL VEHICLES 2 - LOADS"
fileID=fopen('text1.$2k','w');
% The heading of the table
Txt_Veh_Loads = 'TABLE:  "VEHICLES 3 - GENERAL VEHICLES 2 - LOADS"\r\n';
fprintf(fileID,Txt_Veh_Loads);
% The content of the table

    % Counting the number of wheels
    No_of_Wheels = length(LODG_CONF_RMS_MOD{TR});
    % Creating a zeros matrix of size (No of Wheels, one column)
    I = zeros(No_of_Wheels,1);
    % Filling in the all of the I matrix cells by the value of (i)
    I(:,1)=TR;

    % Writing the loading configurations for a vehicle presenting (Wheel 1)
    Txt0_WHL1 ='    VehName="Train#%1.0f-WHL1"   LoadType="Fixed Length"   UnifLoad=0
AxleLoad=%4.2f\r\n';
    Txt1_WHL1 = '    VehName="Train#%1.0f-WHL1"   LoadType="Fixed Length"   UnifLoad=0
AxleLoad=%4.2f   MinDist=%4.2f\r\n';
    A0_WHL1=[I(1,1),LODG_CONF_RMS_MOD{TR}(1,2)];

A1_WHL1=[I(2:No_of_Wheels,1),LODG_CONF_RMS_MOD{TR}(2:No_of_Wheels,2),LODG_CONF_RMS_MOD{TR}(2:No_o
f_Wheels,1)];

    % Writing into the text file
    fprintf(fileID,Txt0_WHL1, A0_WHL1');
    fprintf(fileID,Txt1_WHL1, A1_WHL1');

    % Writing the loading configurations for a vehicle presenting (Wheel 2)
    Txt0_WHL2 ='    VehName="Train#%1.0f-WHL2"   LoadType="Fixed Length"   UnifLoad=0
AxleLoad=%4.2f\r\n';
    Txt1_WHL2 = '    VehName="Train#%1.0f-WHL2"   LoadType="Fixed Length"   UnifLoad=0
AxleLoad=%4.2f   MinDist=%4.2f\r\n';
    A0_WHL2=[I(1,1),LODG_CONF_RMS_MOD{TR}(1,3)];

A1_WHL2=[I(2:No_of_Wheels,1),LODG_CONF_RMS_MOD{TR}(2:No_of_Wheels,3),LODG_CONF_RMS_MOD{TR}(2:No_o
f_Wheels,1)];

    % Writing into the text file
    fprintf(fileID,Txt0_WHL2, A0_WHL2');
    fprintf(fileID,Txt1_WHL2, A1_WHL2');
fclose(fileID);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fileID=fopen('text2.$2k','w');
%%%% Writing TABLE:  "VEHICLES 4 - VEHICLE CLASSES"
% The heading of the table
Txt_Veh_Class = 'TABLE:  "VEHICLES 4 - VEHICLE CLASSES"\r\n';
fprintf(fileID,Txt_Veh_Class);

% The content of the table
    % Writing VehClass for vehicles presenting (Wheel 1)
    VehClass_WHL1 ='   VehClass="Train#%1.0f-WHL1"   VehName="Train#%1.0f-WHL1"
ScaleFactor=1\r\n';
    % Writing VehClass for vehicles presenting (Wheel 2)
    VehClass_WHL2 ='   VehClass="Train#%1.0f-WHL2"   VehName="Train#%1.0f-WHL2"
ScaleFactor=1\r\n';

    % Writing into the text file
    fprintf(fileID,VehClass_WHL1, [TR,TR]);
    fprintf(fileID,VehClass_WHL2, [TR,TR]);
fclose(fileID);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fileID=fopen('text3.$2k','w');
%%%% Writing TABLE:  "LOAD PATTERN DEFINITIONS"
% The heading of the table
Txt_LoadPtrn = 'TABLE:  "LOAD PATTERN DEFINITIONS"\r\n';
fprintf(fileID,Txt_LoadPtrn);

% The content of the table
    LoadPtrnDead ='   LoadPat=DEAD   DesignType=Dead   SelfWtMult=1   GUID=33df69bc-8efe-4379-
ae17-2740fe6a5463\r\n';
    LoadPtrn ='   LoadPat="Train#%1.0f-Tr#1"   DesignType="Vehicle Live"   SelfWtMult=0
GUID=%1.0fe1581a-522c-44f5-b02e-a18bee0731af\r\n';

    % Writing into the text file
    fprintf(fileID,LoadPtrnDead);
    fprintf(fileID,LoadPtrn, [TR,TR]);
fclose(fileID);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fileID=fopen('text4.$2k','w');
%%%% Writing TABLE:  "CASE - MULTISTEP STATIC 1 - LOAD ASSIGNMENTS"
% The heading of the table
Txt_MultiStatic1 = 'TABLE:  "CASE - MULTISTEP STATIC 1 - LOAD ASSIGNMENTS"\r\n';
fprintf(fileID,Txt_MultiStatic1);

% The content of the table
    MultiStep1 ='   Case="Train#%1.0f-Tr#1"   LoadType="Load pattern"   LoadName="Train#%1.0f-
Tr#1"   LoadSF=1\r\n';

    % Writing into the text file
    fprintf(fileID,MultiStep1, [TR,TR]);
fclose(fileID);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
fileID=fopen('text5.$2k','w');
%%%% Writing TABLE:  "LOAD CASE DEFINITIONS"
% The heading of the table
Txt_Loadcase = 'TABLE:  "LOAD CASE DEFINITIONS"\r\n';
fprintf(fileID,Txt_Loadcase);

% The content of the table
    DeadLoadcase ='   Case=DEAD   Type=LinStatic   InitialCond=Zero   DesTypeOpt="Prog Det"
DesignType=Dead   DesActOpt="Prog Det"   DesignAct=Non-Composite   AutoType=None
RunCase=Yes\r\n';
    ModalLoadcase ='   Case=MODAL   Type=LinModal   InitialCond=Zero   DesTypeOpt="Prog Det"
DesignType=Other   DesActOpt="Prog Det"   DesignAct=Other   AutoType=None   RunCase=No\r\n';
    VehLoadcase ='   Case="Train#%1.0f-Tr#1"   Type=LinMSStat   InitialCond=Zero
DesTypeOpt="Prog Det"   DesignType="Vehicle Live"   DesActOpt="Prog Det"   DesignAct="Short-Term
Composite"   AutoType=None   RunCase=Yes\r\n';
    % Writing into the text file
    fprintf(fileID,DeadLoadcase);
    fprintf(fileID,ModalLoadcase);
    fprintf(fileID,VehLoadcase, TR);
fclose(fileID);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fileID=fopen('text6.$2k','w');
%%%% Writing TABLE:  "MULTI-STEP MOVING LOAD 1 - GENERAL"
% The heading of the table
Txt_MultiStepMoving = 'TABLE:  "MULTI-STEP MOVING LOAD 1 - GENERAL"\r\n';
fprintf(fileID,Txt_MultiStepMoving);

% The content of the table
    MultiStepMoving ='   LoadPat="Train#%1.0f-Tr#1"   LoadDur=12   LoadDisc=0.005\r\n';

    % Writing into the text file
    fprintf(fileID,MultiStepMoving, TR);
fclose(fileID);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fileID=fopen('text7.$2k','w');
%%%% Writing TABLE:  "MULTI-STEP MOVING LOAD 2 - VEHICLE DATA"
%%%% Defining the Multi-Static load pattern factors {Vehicle, lane,speed..}
% The heading of the table
Txt_Veh_LoadPat = 'TABLE:  "MULTI-STEP MOVING LOAD 2 - VEHICLE DATA"\r\n';
fprintf(fileID,Txt_Veh_LoadPat);

% The content of the table
    % Load pattern factors for vehicle presenting (Wheel 1)
    VehLoadPat_WHL1 ='   LoadPat="Train#%1.0f-Tr#1"   Vehicle="Train#%1.0f-WHL1"   Lane=LANE(2)-
STR(3)   Station=0   StartTime=0   Direction=Forward   Speed=739.2\r\n';
    % Load pattern factors for vehicle presenting (Wheel 2)
    VehLoadPat_WHL2 ='   LoadPat="Train#%1.0f-Tr#1"   Vehicle="Train#%1.0f-WHL2"   Lane=LANE(2)-
STR(4)   Station=0   StartTime=0   Direction=Forward   Speed=739.2\r\n';

    % Writing into the text file
    fprintf(fileID,VehLoadPat_WHL1, [TR,TR]);
    fprintf(fileID,VehLoadPat_WHL2, [TR,TR]);
fclose(fileID);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
fileID=fopen('text8.$2k','w');
%%%% Writing TABLE:  "VEHICLES 2 - GENERAL VEHICLES 1 - GENERAL"
Txt_Veh_General = 'TABLE:  "VEHICLES 2 - GENERAL VEHICLES 1 - GENERAL"\r\n';
fprintf(fileID,Txt_Veh_General);

% The content of the table
    % Vehicle presenting (Wheel 1)
    VehGeneral_WHL1 ='   VehName="Train#%1.0f-WHL1"   StayInLane=No\r\n';
    % Vehicle presenting (Wheel 2)
    VehGeneral_WHL2 ='   VehName="Train#%1.0f-WHL2"   StayInLane=No\r\n';

    % Writing into the text file
    fprintf(fileID,VehGeneral_WHL1, TR);
    fprintf(fileID,VehGeneral_WHL2, TR);
% Closing the text file
fclose(fileID);
```

Replacing the above Tables in The Original SAP2000 $2k file with the

Above written tables From WILD DATA - Extract table name from replacement file.

```
 Content1 = fileread( 'text1.$2k' ) ;
 Content2 = fileread( 'text2.$2k' ) ;
 Content3 = fileread( 'text3.$2k' ) ;
 Content4 = fileread( 'text4.$2k' ) ;
 Content5 = fileread( 'text5.$2k' ) ;
 Content6 = fileread( 'text6.$2k' ) ;
 Content7 = fileread( 'text7.$2k' ) ;
 Content8 = fileread( 'text8.$2k' ) ;

 VehLoads = regexp( Content1, '(?<=TABLE:\s+")[^"]+', 'match', 'once' ) ;
 VehClass = regexp( Content2, '(?<=TABLE:  "VEHICLES 4 - VEHICLE CLASSES")[^"]+', 'match', 'once'
) ;
 LoadPattern = regexp( Content3, '(?<=TABLE:  "LOAD PATTERN DEFINITIONS")[^"]+', 'match', 'once'
) ;
 CaseMultistep = regexp( Content4, '(?<=TABLE:  "CASE - MULTISTEP STATIC 1 - LOAD
ASSIGNMENTS")[^"]+', 'match', 'once' ) ;
 LoadCase = regexp( Content5, '(?<=TABLE:  "LOAD CASE DEFINITIONS")[^"]+', 'match', 'once' ) ;
 MultiStepMoving1 = regexp( Content6, '(?<=TABLE:  "MULTI-STEP MOVING LOAD 1 - GENERAL")[^"]+',
'match', 'once' ) ;
 MultiStepMoving2 = regexp( Content7, '(?<=TABLE:  "MULTI-STEP MOVING LOAD 2 - VEHICLE
DATA")[^"]+', 'match', 'once' ) ;
 GeneralVehicle = regexp( Content8, '(?<=TABLE:  "VEHICLES 2 - GENERAL VEHICLES 1 -
GENERAL")[^"]+', 'match', 'once' ) ;

 % - Build a pattern for matching what to replace, based on the table name.
 pattern1 = sprintf( 'TABLE:\\s+"%s.*?(?=[\\r\\n]TABLE)', VehLoads) ;
 pattern2 = sprintf( 'TABLE:  "VEHICLES 4 - VEHICLE CLASSES"%s.*?(?=[\\r\\n]END TABLE DATA)',
VehClass) ;
 pattern3 = sprintf( 'TABLE:  "LOAD PATTERN DEFINITIONS"%s.*?(?=[\\r\\n]TABLE)', LoadPattern) ;
 pattern4 = sprintf( 'TABLE:  "CASE - MULTISTEP STATIC 1 - LOAD
ASSIGNMENTS"%s.*?(?=[\\r\\n]TABLE)', CaseMultistep ) ;
 pattern5 = sprintf( 'TABLE:  "LOAD CASE DEFINITIONS"%s.*?(?=[\\r\\n]TABLE)', LoadCase ) ;
 pattern6 = sprintf( 'TABLE:  "MULTI-STEP MOVING LOAD 1 - GENERAL"%s.*?(?=[\\r\\n]TABLE)',
MultiStepMoving1 ) ;
 pattern7 = sprintf( 'TABLE:  "MULTI-STEP MOVING LOAD 2 - VEHICLE DATA"%s.*?(?=[\\r\\n]TABLE)',
MultiStepMoving2 ) ;
 pattern8 = sprintf( 'TABLE:  "VEHICLES 2 - GENERAL VEHICLES 1 - GENERAL"%s.*?(?=[\\r\\n]TABLE)',
GeneralVehicle ) ;

 % - Replace in original content.
 originalSAP_s2k = fileread( [ModelName,'.$2k']) ;
 originalSAP_s2k = regexprep( originalSAP_s2k, pattern1, Content1) ;
 originalSAP_s2k = regexprep( originalSAP_s2k, pattern2, Content2) ;
 originalSAP_s2k = regexprep( originalSAP_s2k, pattern3, Content3) ;
 originalSAP_s2k = regexprep( originalSAP_s2k, pattern4, Content4) ;
 originalSAP_s2k = regexprep( originalSAP_s2k, pattern5, Content5) ;
 originalSAP_s2k = regexprep( originalSAP_s2k, pattern6, Content6) ;
 originalSAP_s2k = regexprep( originalSAP_s2k, pattern7, Content7) ;
 originalSAP_s2k = regexprep( originalSAP_s2k, pattern8, Content8) ;

 % - Export updated content.
 fId = fopen( [ModelName,'.$2k'], 'w');
 fwrite( fId, originalSAP_s2k ) ;
 fclose( fId ) ;
```

%%

Starting SAP model and Analysis

Initialize model

```
ret = SapModel.InitializeNewModel;
```

Open an Existing model

```
File = NET.explicitCast(SapModel.File,'SAP2000v19.cFile');

ret = File.OpenFile([FilePath, '\',ModelName,'.$2k']);
```

Hide Application, Hide = Unhide

SapObject.Hide;

%%

Getting POMs and SnapShot Matrices for Healthy Model, first

Unlock the model

```
ret = SapModel.SetModelIsLocked(false);
```

Switch to k-in units

```
ret = SapModel.SetPresentUnits(SAP2000v19.eUnits.kip_in_F);
```

Assigning Frame Releases Assuming 76% fixity ratio Defined above

```
FrameObj = NET.explicitCast(SapModel.FrameObj,'SAP2000v19.cFrameObj');
Start = NET.createArray('System.Boolean',12);%% From VBA Function
End = NET.createArray('System.Boolean',12);%% From VBA Function

% Release for stringers left side

Start_Releases_1 = NET.createArray('System.Double',6);%% From VBA Function
End_Releases_1 = NET.createArray('System.Double',6);%% From VBA Function

for ii = 5 : 6

    Start(ii) = true(); %% Activate Releases in Y and X Directions at start

end

Start_Releases_1(5) = 0; %%Frame Partial Fixity in Y-direction at start
Start_Releases_1(6) = EFR_H; %%Frame Partial Fixity in X-direction at End @ 67% of continuous

for jj = 5 : 6

    End(jj) = false();%% Activate Releases in Y and X Directions at start

end

End_Releases_1 (5) = 0; %%Frame Partial Fixity in Y-direction at End
End_Releases_1 (6) = EFR_H; %%Frame Partial Fixity in X-direction at End @ 67% of continuous

for iii=1:20
    SetReleases = FrameObj.SetReleases(['STR-L-
',num2str(iii)],Start,End,Start_Releases_1,End_Releases_1);
end
```

```
for iii=1:4
    SetReleases = FrameObj.SetReleases(['STR-LL-
',num2str(iii)],Start,End,Start_Releases_1,End_Releases_1);
end

% Release for stringers right side

Start_Releases_2 = NET.createArray('System.Double',6);%% From VBA Function
End_Releases_2 = NET.createArray('System.Double',6);%% From VBA Function

for ii = 5 : 6

    Start(ii) = false(); %% Activate Releases in Y and X Directions at start

end

Start_Releases_2(5) = 0; %%Frame Partial Fixity in Y-direction at start
Start_Releases_2(6) = EFR_H; %%Frame Partial Fixity in X-direction at End @ 67% of continuous

for jj = 5 : 6

    End(jj) = true();%% Activate Releases in Y and X Directions at start

end

End_Releases_2 (5) = 0; %%Frame Partial Fixity in Y-direction at End
End_Releases_2 (6) = EFR_H; %%Frame Partial Fixity in X-direction at End @ 67% of continuous


for iiii=1:24
    SetReleases = FrameObj.SetReleases(['STR-R-
',num2str(iiii)],Start,End,Start_Releases_2,End_Releases_2);
end
```

Refresh view, update (initialize) zoom

```
View = NET.explicitCast(SapModel.View,'SAP2000v19.cView');
ret = View.RefreshView(0, false());
```

Save model, write the model path

```
ret = File.Save([FilePath, '\',ModelName,'.sdb']);
```

Run model (this will create the analysis model)

```
Analyze = NET.explicitCast(SapModel.Analyze,'SAP2000v19.cAnalyze');
% Select model case to run
ret = Analyze.SetRunCaseFlag('DEAD',true);%%might be removed
ret = Analyze.SetRunCaseFlag('MODAL',false);%%might be removed
ret = Analyze.SetRunCaseFlag(['Train#',num2str(TR),'-Tr#1'],true);%%The code main purpose
ret = Analyze.RunAnalysis();
```

Getting Modal Analysis Parameters

Switch to k-ft units

```
ret = SapModel.SetPresentUnits(SAP2000v19.eUnits.kip_ft_F);

% Creating results file
AnalysisResults = NET.explicitCast(SapModel.Results,'SAP2000v19.cAnalysisResults');
AnalysisResultsSetup =
NET.explicitCast(AnalysisResults.Setup,'SAP2000v19.cAnalysisResultsSetup');

% Select results cases
AnalysisResultsSetup.DeselectAllCasesAndCombosForOutput;
AnalysisResultsSetup.SetCaseSelectedForOutput(['Train#',num2str(TR),'-Tr#1']);

% Extracting results step by step (2)
AnalysisResultsSetup.SetOptionMultiStepStatic(2)


%Getting Internal Forces
NumberResults = 0;

Obj = NET.createArray('System.String',2);
ObjSta = NET.createArray('System.Double',2);
Elm = NET.createArray('System.String',2);
ElmSta = NET.createArray('System.Double',2);
ACase = NET.createArray('System.String',2);
StepType = NET.createArray('System.String',2);
StepNum = NET.createArray('System.Double',2);
P= NET.createArray('System.Double',2);
V2 = NET.createArray('System.Double',2);
V3 = NET.createArray('System.Double',2);
T = NET.createArray('System.Double',2);
M2 = NET.createArray('System.Double',2);
M3 = NET.createArray('System.Double',2);

[~, NumberResults, Obj,ObjSta, Elm,ElmSta, ACase, StepType, StepNum,...
    P, V2, V3, T, M2, M3] = AnalysisResults.FrameForce('STR_ENDS',...
    SAP2000v19.eItemTypeElm.GroupElm, NumberResults, Obj,ObjSta,...
    Elm,ElmSta, ACase, StepType, StepNum, P, V2, V3, T, M2, M3);

%Converting results into a readable MATLAB format
```

```
PP_H1(:,1)=(P.double);
VV2_H1(:,1)=(V2.double);
VV3_H1(:,1)=(V3.double);
TT_H1(:,1)=(T.double);
MM2_H1(:,1)=(M2.double);
MM3_H1(:,1)=(M3.double);
OObjSta_H1(:,1)= (ObjSta.double);
EElmSta_H1(:,1)= (ElmSta.double);
SStepNum_H1(:,1)= (StepNum.double);

OObj_H1(:,1)= cell(Obj.Length,1);
for j = 1:Obj.Length
    OObj_H1{j,1} = char(Obj(j));
end

EElm_H1= cell(Elm.Length,1);
for j = 1:Elm.Length
    EElm_H1{j,1} = char(Elm(j));
end

AACase_H1= cell(ACase.Length,1);
for j = 1:ACase.Length
    AACase_H1{j,1} = char(ACase(j));
end

SStepType_H1=cell(StepType.Length,1);
for j = 1:StepType.Length
    SStepType_H1{j,1} = char(StepType(j));
end

% Extracting numerical results into a Matrix

 sapResults_H=[EElmSta_H1, SStepNum_H1, PP_H1,VV2_H1,VV3_H1,MM2_H1,MM3_H1];

 % Extracting all results into a Matrix
 sapResults_Table_H=[array2table(OObj_H1),array2table(OObjSta_H1), array2table(EElm_H1),...
 array2table(EElmSta_H1),array2table(AACase_H1),array2table(SStepType_H1),...
 array2table(SStepNum_H1),array2table(PP_H1),array2table(VV2_H1),...
 array2table(VV3_H1),array2table(TT_H1),array2table(MM2_H1),array2table(MM3_H1)];

% Filtering tables by a station
SapResults_Filtered_H=sapResults_Table_H(sapResults_Table_H.OObjSta_H1==0,:);

% Sort and Extract final tables
D_Str_Ends_H=sortrows(SapResults_Filtered_H,1);
```

Damage Internal Forces and Bottom Stresses

```
M3_H=table2array(D_Str_Ends_H(:,13)); % Bending moment
P_H = table2array(D_Str_Ends_H(:,8)); % Axial force
BotStr_H=((P_H)/Area+(M3_H)*12/Sbot); % Bottom stresses
```

```
TimeStep_H =table2array(D_Str_Ends_H(:,7)); %Time step Number
Nrows_H =(length(BotStr_H))/Nsensors;   %number of steps ot dt/train

% Snapshot matrix using stresses
SnapShot_Model_Str_End_H = [BotStr_H(1:(Nrows_H),:),...
    BotStr_H((Nrows_H+1):(2*Nrows_H),:),...
    BotStr_H((2*Nrows_H+1):(3*Nrows_H),:),...
    BotStr_H((3*Nrows_H+1):(4*Nrows_H),:),...
    BotStr_H((4*Nrows_H+1):(5*Nrows_H),:),...
    BotStr_H((5*Nrows_H+1):(6*Nrows_H),:),...
    BotStr_H((6*Nrows_H+1):(7*Nrows_H),:),...
    BotStr_H((7*Nrows_H+1):(8*Nrows_H),:),...
    BotStr_H((8*Nrows_H+1):(9*Nrows_H),:),...
    BotStr_H((9*Nrows_H+1):(10*Nrows_H),:),...
    BotStr_H((10*Nrows_H+1):(11*Nrows_H),:),...
    BotStr_H((11*Nrows_H+1):(12*Nrows_H),:),...
    BotStr_H((12*Nrows_H+1):(13*Nrows_H),:),...
    BotStr_H((13*Nrows_H+1):(14*Nrows_H),:),...
    BotStr_H((14*Nrows_H+1):(15*Nrows_H),:),...
    BotStr_H((15*Nrows_H+1):(16*Nrows_H),:),...
    BotStr_H((16*Nrows_H+1):(17*Nrows_H),:),...
    BotStr_H((17*Nrows_H+1):(18*Nrows_H),:),...
    BotStr_H((18*Nrows_H+1):(19*Nrows_H),:),...
    BotStr_H((19*Nrows_H+1):(20*Nrows_H),:)];
```

Extracting POMs, Healthy Model

```
[POM_Str_End_H, SVD_Str_End_H,~] = svd((SnapShot_Model_Str_End_H(100:2100,:))');

% Getting POMs#1 alone
POM1_H=POM_Str_End_H(:,1);
```

Writing SnapShot matrix, POMs and for Health Cases

```
%Healthy models
Loc_H =(TR-1)*Nsensors*length(DI)+TR;
SnapShotMatrix_Initial(:,:,Loc_H)=SnapShot_Model_Str_End_H;%snapshot matrix
POMs_1_RMS(:,Loc_H)=POM1_H; % Writing Healthy Models POMS into POMs_1 matrix
```

%%

Getting POMs and SnapShot Matrices for Damaged Cases, Iterative process

Iteration Loops for each damage intensity

```
for kk=1:length(DI)
for k=1:Nsensors     % Number of stringer ends, Assumed to be instrumented
```

Unlock the model

```
ret = SapModel.SetModelIsLocked(false);
```

switch to k-in units

```
ret = SapModel.SetPresentUnits(SAP2000v19.eUnits.kip_in_F);
```

Asigning Frame Releases for Healthy Model As 76% of Rigid

```
FrameObj = NET.explicitCast(SapModel.FrameObj,'SAP2000v19.cFrameObj');
Start = NET.createArray('System.Boolean',12);%% From VBA Function
End = NET.createArray('System.Boolean',12);%% From VBA Function

% Release for stringers left side

Start_Releases_1 = NET.createArray('System.Double',6);%% From VBA Function
End_Releases_1 = NET.createArray('System.Double',6);%% From VBA Function

for ii = 5 : 6

    Start(ii) = true(); %% Activate Releases in Y and X Directions at start

end

Start_Releases_1(5) = 0; %%Frame Partial Fixity in Y-direction at start
Start_Releases_1(6) = EFR_H; %%Frame Partial Fixity in X-direction at End @ 67% of continuous

for jj = 5 : 6

    End(jj) = false();%% Activate Releases in Y and X Directions at start

end

End_Releases_1 (5) = 0; %%Frame Partial Fixity in Y-direction at End
End_Releases_1 (6) = EFR_H; %%Frame Partial Fixity in X-direction at End @ 67% of continuous

for iii=1:20
    SetReleases = FrameObj.SetReleases(['STR-L-
',num2str(iii)],Start,End,Start_Releases_1,End_Releases_1);
end
for iii=1:4
    SetReleases = FrameObj.SetReleases(['STR-LL-
',num2str(iii)],Start,End,Start_Releases_1,End_Releases_1);
end

% Release for stringers right side

Start_Releases_2 = NET.createArray('System.Double',6);%% From VBA Function
End_Releases_2 = NET.createArray('System.Double',6);%% From VBA Function

for ii = 5 : 6

    Start(ii) = false(); %% Activate Releases in Y and X Directions at start

end

Start_Releases_2(5) = 0; %%Frame Partial Fixity in Y-direction at start
Start_Releases_2(6) = EFR_H; %%Frame Partial Fixity in X-direction at End @ 67% of continuous

for jj = 5 : 6

    End(jj) = true();%% Activate Releases in Y and X Directions at start

end

End_Releases_2 (5) = 0; %%Frame Partial Fixity in Y-direction at End
End_Releases_2 (6) = EFR_H; %%Frame Partial Fixity in X-direction at End @ 67% of continuous


for iiii=1:24
    SetReleases = FrameObj.SetReleases(['STR-R-
',num2str(iiii)],Start,End,Start_Releases_2,End_Releases_2);
end
```

Assigning DAMAGE EFFECTS as RELEASES

Release for stringers left side

```
Start_Releases_D = NET.createArray('System.Double',6);%% From VBA Function
End_Releases_D = NET.createArray('System.Double',6);%% From VBA Function

for ii = 5 : 6

    Start(ii) = true(); %% Activate Releases in Y and X Directions at start

end

Start_Releases_D(5) = 0; %%Frame Partial Fixity in Y-direction at start
Start_Releases_D(6) = EFR_D(kk); %%Frame Partial Fixity in X-direction at End @ 67% of continuous

for jj = 5 : 6
```

```
    End(jj) = false();%% Activate Releases in Y and X Directions at start

end

End_Releases_D (5) = 0; %%Frame Partial Fixity in Y-direction at End
End_Releases_D (6) = EFR_D(kk); %%Frame Partial Fixity in X-direction at End @ 67% of continuous

SetReleases(k) = FrameObj.SetReleases(['STR-L-
',num2str(k)],Start,End,Start_Releases_D,End_Releases_D);
```

Refresh view, update (initialize) zoom

```
View = NET.explicitCast(SapModel.View,'SAP2000v19.cView');
ret = View.RefreshView(0, false());
```

Save model, write the model path

```
ret = File.Save([FilePath, '\',ModelName,'.sdb']);
```

Run model (this will create the analysis model)

```
Analyze = NET.explicitCast(SapModel.Analyze,'SAP2000v19.cAnalyze');
% Select model case to run
ret = Analyze.SetRunCaseFlag('DEAD',true);%%might be removed
ret = Analyze.SetRunCaseFlag('MODAL',false);%%might be removed
ret = Analyze.SetRunCaseFlag(['Train#',num2str(TR),'-Tr#1'],true);%%The code main purpose
ret = Analyze.RunAnalysis();
```

Getting Internal Forces

switch to k-ft units

```
ret = SapModel.SetPresentUnits(SAP2000v19.eUnits.kip_ft_F);

% Creating results file
AnalysisResults = NET.explicitCast(SapModel.Results,'SAP2000v19.cAnalysisResults');
AnalysisResultsSetup =
NET.explicitCast(AnalysisResults.Setup,'SAP2000v19.cAnalysisResultsSetup');

% Select results cases
AnalysisResultsSetup.DeselectAllCasesAndCombosForOutput;
AnalysisResultsSetup.SetCaseSelectedForOutput(['Train#',num2str(TR),'-Tr#1']);
```

```
% Extracting results step by step (2)
AnalysisResultsSetup.SetOptionMultiStepStatic(2)


%Getting Internal Forces
NumberResults = 0;

Obj = NET.createArray('System.String',2);
ObjSta = NET.createArray('System.Double',2);
Elm = NET.createArray('System.String',2);
ElmSta = NET.createArray('System.Double',2);
ACase = NET.createArray('System.String',2);
StepType = NET.createArray('System.String',2);
StepNum = NET.createArray('System.Double',2);
P= NET.createArray('System.Double',2);
V2 = NET.createArray('System.Double',2);
V3 = NET.createArray('System.Double',2);
T = NET.createArray('System.Double',2);
M2 = NET.createArray('System.Double',2);
M3 = NET.createArray('System.Double',2);

[~, NumberResults, Obj,ObjSta, Elm,ElmSta, ACase, StepType, StepNum,...
    P, V2, V3, T, M2, M3] = AnalysisResults.FrameForce('STR_ENDS',...
    SAP2000v19.eItemTypeElm.GroupElm, NumberResults, Obj,ObjSta,...
    Elm,ElmSta, ACase, StepType, StepNum, P, V2, V3, T, M2, M3);

%Converting results into a readable MATLAB format
PP_D1(:,1)=(P.double);
VV2_D1(:,1)=(V2.double);
VV3_D1(:,1)=(V3.double);
TT_D1(:,1)=(T.double);
MM2_D1(:,1)=(M2.double);
MM3_D1(:,1)=(M3.double);
OObjSta_D1(:,1)= (ObjSta.double);
EElmSta_D1(:,1)= (ElmSta.double);
SStepNum_D1(:,1)= (StepNum.double);

OObj_D1(:,1)= cell(Obj.Length,1);
for j = 1:Obj.Length
    OObj_D1{j,1} = char(Obj(j));
end

EElm_D1= cell(Elm.Length,1);
for j = 1:Elm.Length
    EElm_D1{j,1} = char(Elm(j));
end

AACase_D1= cell(ACase.Length,1);
for j = 1:ACase.Length
    AACase_D1{j,1} = char(ACase(j));
end

SStepType_D1=cell(StepType.Length,1);
for j = 1:StepType.Length
```

```
    SStepType_D1{j,1} = char(StepType(j));
end

% Extracting numerical results into a Matrix

 sapResults=[EElmSta_D1, SStepNum_D1, PP_D1,VV2_D1,VV3_D1,MM2_D1,MM3_D1];

 % Extracting all results into a Matrix
 sapResults_Table=[array2table(OObj_D1),array2table(OObjSta_D1), array2table(EElm_D1),...
 array2table(EElmSta_D1),array2table(AACase_D1),array2table(SStepType_D1),...
 array2table(SStepNum_D1),array2table(PP_D1),array2table(VV2_D1),...
 array2table(VV3_D1),array2table(TT_D1),array2table(MM2_D1),array2table(MM3_D1)];

% Filtering tables by a station
SapResults_Filtered_D=sapResults_Table(sapResults_Table.OObjSta_D1==0,:);

% Sort and Extract final tables
D_Str_Ends_D=sortrows(SapResults_Filtered_D,1);
```

Damage Internal Forces and Bottom Stresses

```
M3_D=table2array(D_Str_Ends_D(:,13)); % Bending moment
P_D = table2array(D_Str_Ends_D(:,8)); % Axial force
BotStr_D=((P_D)/Area+(M3_D)*12/Sbot); % Bottom stresses
TimeStep_D =table2array(D_Str_Ends_D(:,7)) ;%Time step Number
Nrows_D =(length(BotStr_D))/Nsensors  ; %number of steps ot dt/train

% Snapshot matrix using stresses
SnapShot_Model_Str_End_D = [BotStr_D(1:(Nrows_D),:),...
    BotStr_D((Nrows_D+1):(2*Nrows_D),:),...
    BotStr_D((2*Nrows_D+1):(3*Nrows_D),:),...
    BotStr_D((3*Nrows_D+1):(4*Nrows_D),:),...
    BotStr_D((4*Nrows_D+1):(5*Nrows_D),:),...
    BotStr_D((5*Nrows_D+1):(6*Nrows_D),:),...
    BotStr_D((6*Nrows_D+1):(7*Nrows_D),:),...
    BotStr_D((7*Nrows_D+1):(8*Nrows_D),:),...
    BotStr_D((8*Nrows_D+1):(9*Nrows_D),:),...
    BotStr_D((9*Nrows_D+1):(10*Nrows_D),:),...
    BotStr_D((10*Nrows_D+1):(11*Nrows_D),:),...
    BotStr_D((11*Nrows_D+1):(12*Nrows_D),:),...
    BotStr_D((12*Nrows_D+1):(13*Nrows_D),:),...
    BotStr_D((13*Nrows_D+1):(14*Nrows_D),:),...
    BotStr_D((14*Nrows_D+1):(15*Nrows_D),:),...
    BotStr_D((15*Nrows_D+1):(16*Nrows_D),:),...
    BotStr_D((16*Nrows_D+1):(17*Nrows_D),:),...
    BotStr_D((17*Nrows_D+1):(18*Nrows_D),:),...
    BotStr_D((18*Nrows_D+1):(19*Nrows_D),:),...
    BotStr_D((19*Nrows_D+1):(20*Nrows_D),:)];
```

Extracting POMs, Damaged Cases

```
[POM_Str_End_D, SVD_Str_End_D,~] = svd((SnapShot_Model_Str_End_D(100:2100,:))');

% Getting POMs#1 alone
POM1_D=POM_Str_End_D(:,1);
```

Writing SnapShot matrix, POMs and Damage Intensities values in the matrix

```
%Defected or Damaged models
Loc_D=(TR-1)*Nsensors*length(DI)+TR+(((kk-1)*Nsensors)+k);
POMs_1_RMS(:,Loc_D)=POM1_D;
SnapShotMatrix_Initial(:,:,Loc_D)=SnapShot_Model_Str_End_D;
%Damage Intensities Matrix
D_Vectors(k,Loc_D)=DI(kk);
```

Damage iterative process (POMs and SnapShotMatrices)

```
end
end
```

End OF the Iterative Process for Defined number of Trains

```
end
```

Delete Unwanted Text Files

```
for TR=1:8
    delete (['text',num2str(TR),'.$2k'])
end
```

Switching SnapShotMatrix to Simple Array

```
for TR=1:length(POMs_1_RMS)
    SnapShotMatrix_RMS{TR}=SnapShotMatrix_Initial(:,:,TR);
end
```

End Time

```
EndTime=toc/(60*60)     %  154.71 hours
```

Writing Output in Part 2

```
POMs_1_RMS_1to5=POMs_1_RMS;
SnapShotMatrix_RMS_1to5=SnapShotMatrix_RMS;
```

Writing results for each Intensity case, for the 20 stringer

```
save('Trains_Results_WILD_Higher_RMS_10Trains','POMs_1_RMS','SnapShotMatrix_RMS');

save('Trains_Results_WILD_Higher_RMS_10Trains_1to5','POMs_1_RMS_1to5','SnapShotMatrix_RMS_1to5');
```

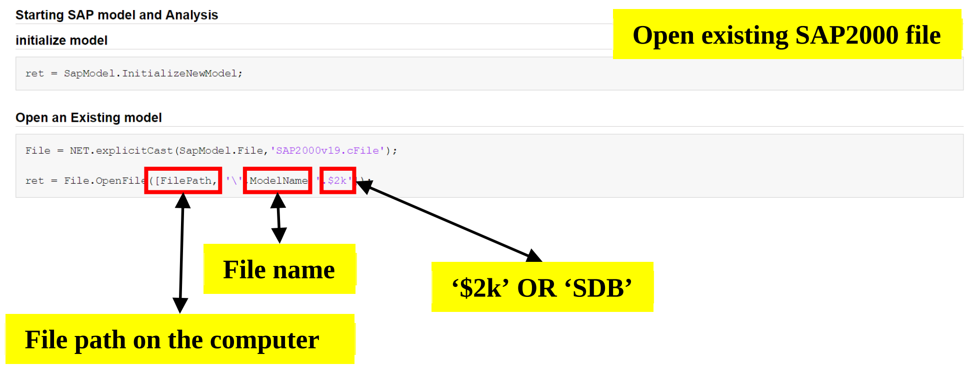
Published with MATLAB® R2017a

Appendix 3: SAP2000 OAPI and MATLAB Coding Explanation

In this appendix, MATLAB and SAP2000 OAPI connection MATLAB code is shown. This MATLAB code used in simulating trains, damage and extracting internal forces at the selected locations.

SAP2000 OAPI and Matlab coding explanation

1. Opening an existing SAP2000



Starting SAP model and Analysis

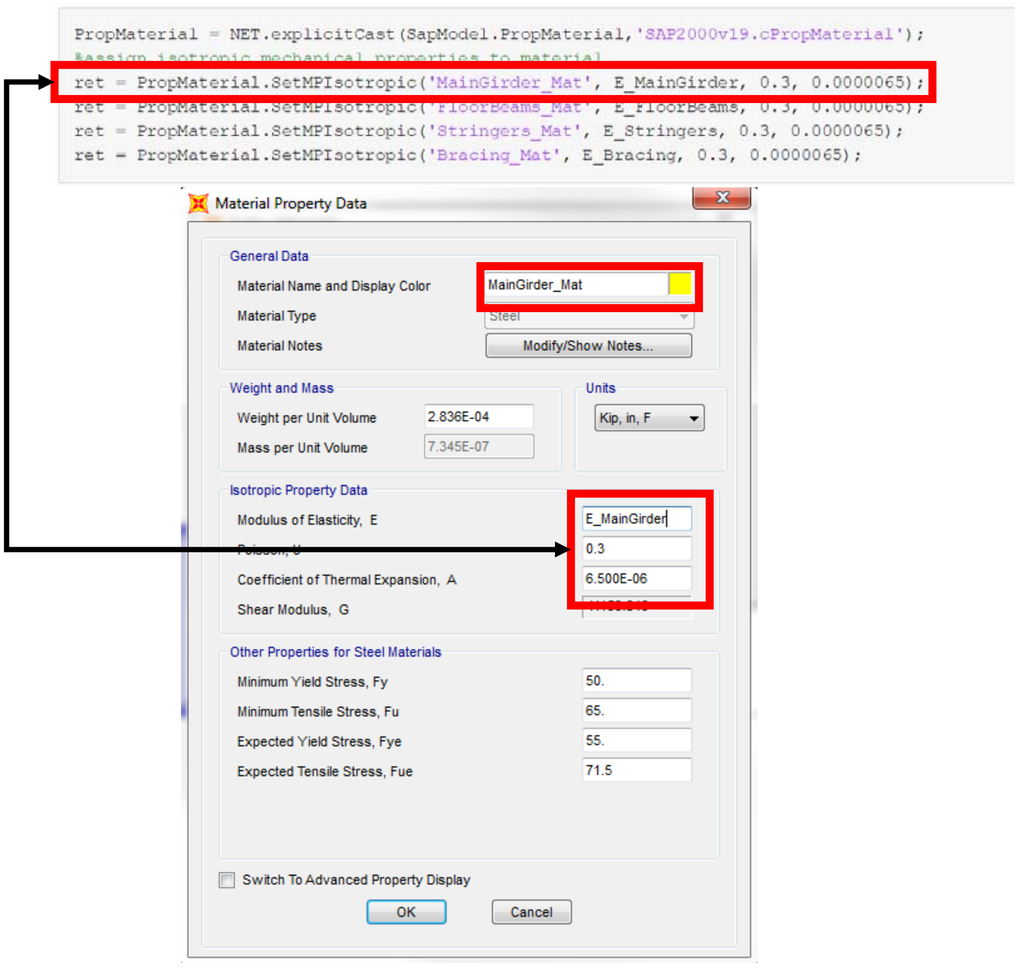
initialize model

```
ret = SapModel.InitializeNewModel;
```

Open an Existing model

```
File = NET.explicitCast(SapModel.File,'SAP2000v19.cFile');

ret = File.OpenFile([FilePath, '\',ModelName,'.$2k']);
```

2. Assigning material properties



```
PropMaterial = NET.explicitCast(SapModel.PropMaterial,'SAP2000v19.cPropMaterial');
%assign isotropic mechanical properties to material
ret = PropMaterial.SetMPIsotropic('MainGirder_Mat', E_MainGirder, 0.3, 0.0000065);
ret = PropMaterial.SetMPIsotropic('FloorBeams_Mat', E_FloorBeams, 0.3, 0.0000065);
ret = PropMaterial.SetMPIsotropic('Stringers_Mat', E_Stringers, 0.3, 0.0000065);
ret = PropMaterial.SetMPIsotropic('Bracing_Mat', E_Bracing, 0.3, 0.0000065);
```

Material Property Data

General Data

Material Name and Display Color	MainGirder_Mat
Material Type	Steel
Material Notes	Modify/Show Notes...

Weight and Mass

Weight per Unit Volume	2.836E-04
Mass per Unit Volume	7.345E-07

Units: Kip, in, F

Isotropic Property Data

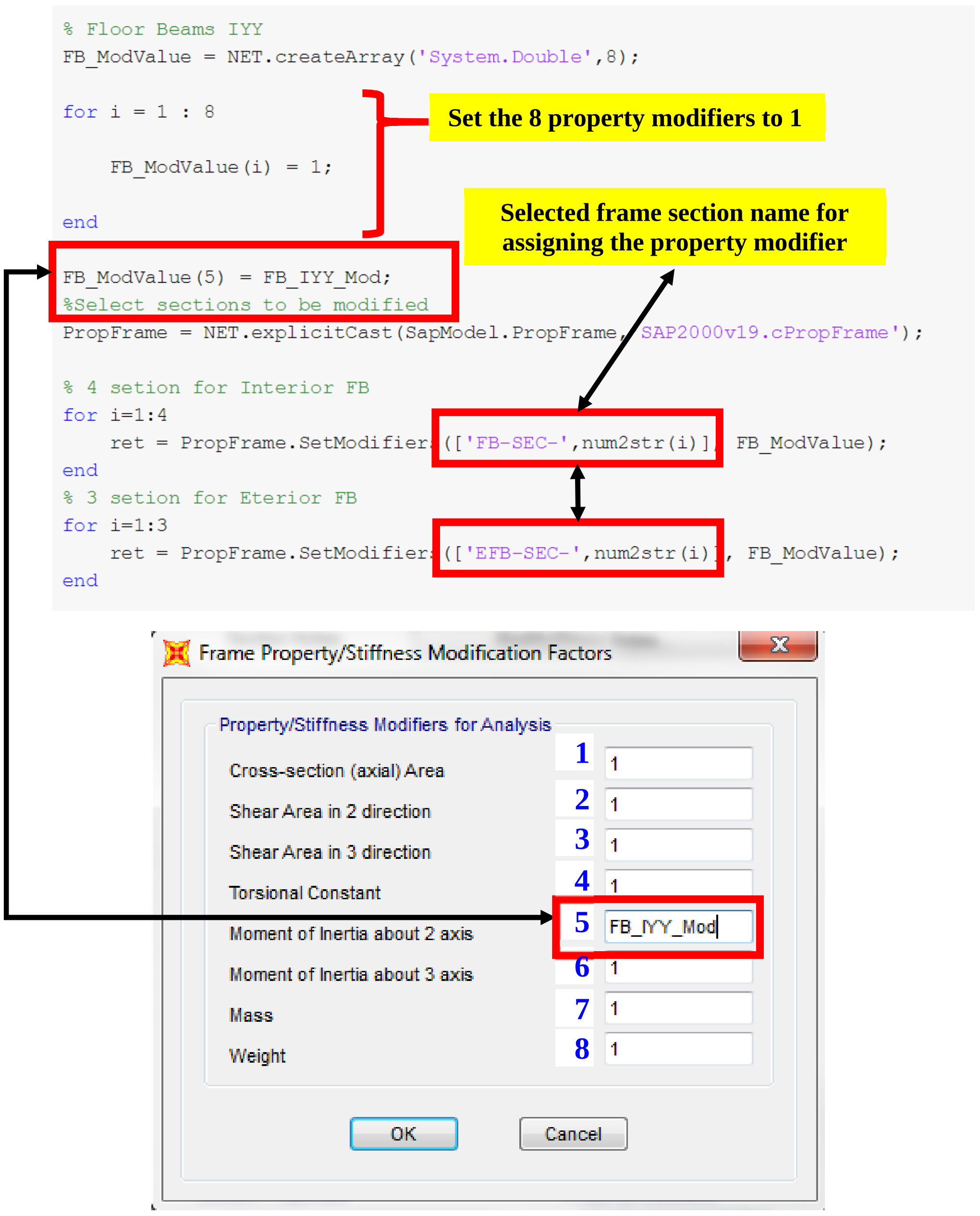
Modulus of Elasticity, E	E_MainGirder
Poisson, U	0.3
Coefficient of Thermal Expansion, A	6.500E-06
Shear Modulus, G	[illegible]

Other Properties for Steel Materials

Minimum Yield Stress, Fy	50.
Minimum Tensile Stress, Fu	65.
Expected Yield Stress, Fye	55.
Expected Tensile Stress, Fue	71.5

☐ Switch To Advanced Property Display

3. Assigning frame element property modifier about Y-Direction



```
% Floor Beams IYY
FB_ModValue = NET.createArray('System.Double',8);

for i = 1 : 8

    FB_ModValue(i) = 1;

end
FB_ModValue(5) = FB_IYY_Mod;
%Select sections to be modified
PropFrame = NET.explicitCast(SapModel.PropFrame,'SAP2000v19.cPropFrame');

% 4 setion for Interior FB
for i=1:4
    ret = PropFrame.SetModifiers(['FB-SEC-',num2str(i)], FB_ModValue);
end
% 3 setion for Eterior FB
for i=1:3
    ret = PropFrame.SetModifiers(['EFB-SEC-',num2str(i)], FB_ModValue);
end
```

Set the 8 property modifiers to 1

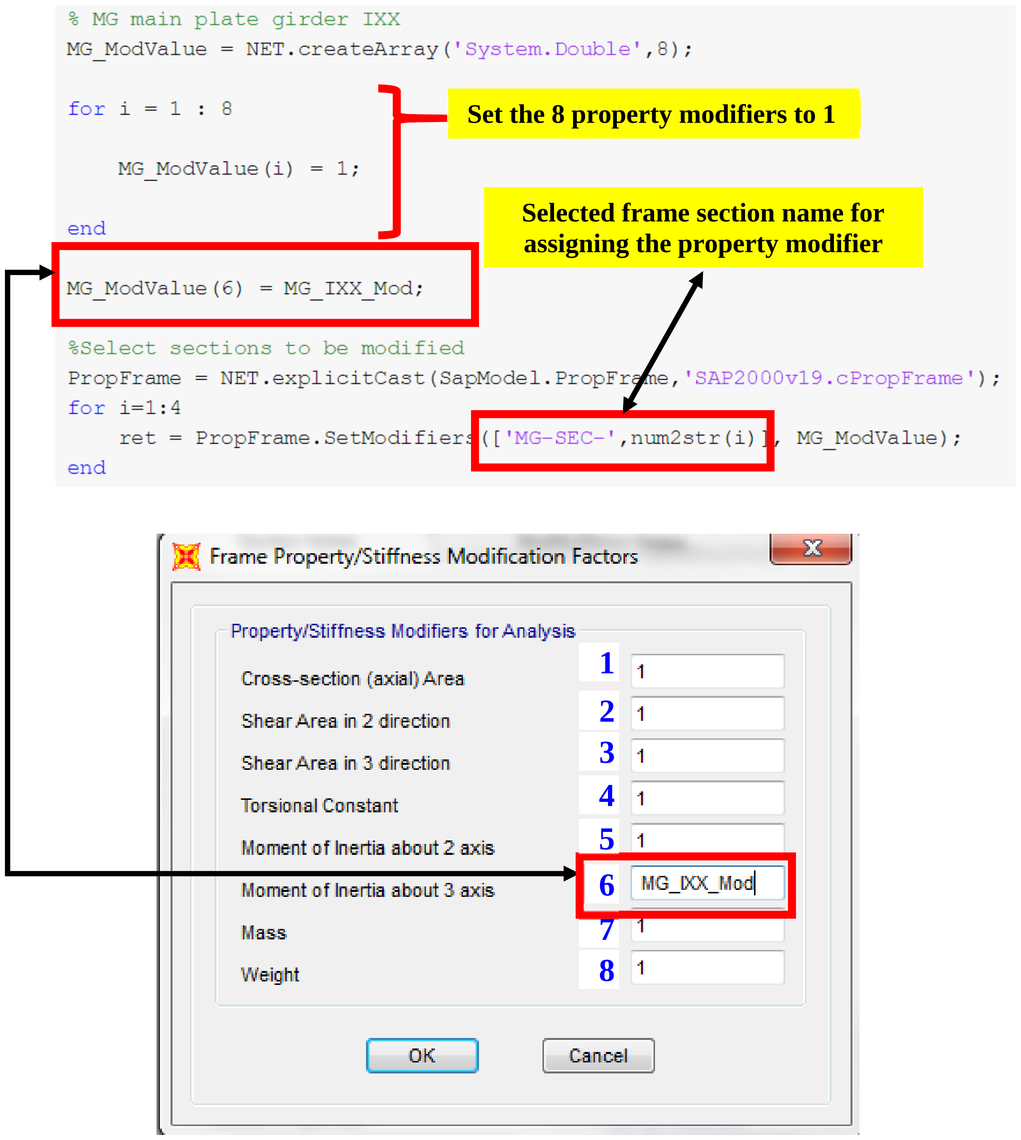
Selected frame section name for assigning the property modifier

Frame Property/Stiffness Modification Factors

Property/Stiffness Modifiers for Analysis

Cross-section (axial) Area	1	1
Shear Area in 2 direction	2	1
Shear Area in 3 direction	3	1
Torsional Constant	4	1
Moment of Inertia about 2 axis	5	FB_IYY_Mod
Moment of Inertia about 3 axis	6	1
Mass	7	1
Weight	8	1

4. Assigning frame element property modifier about X-Direction



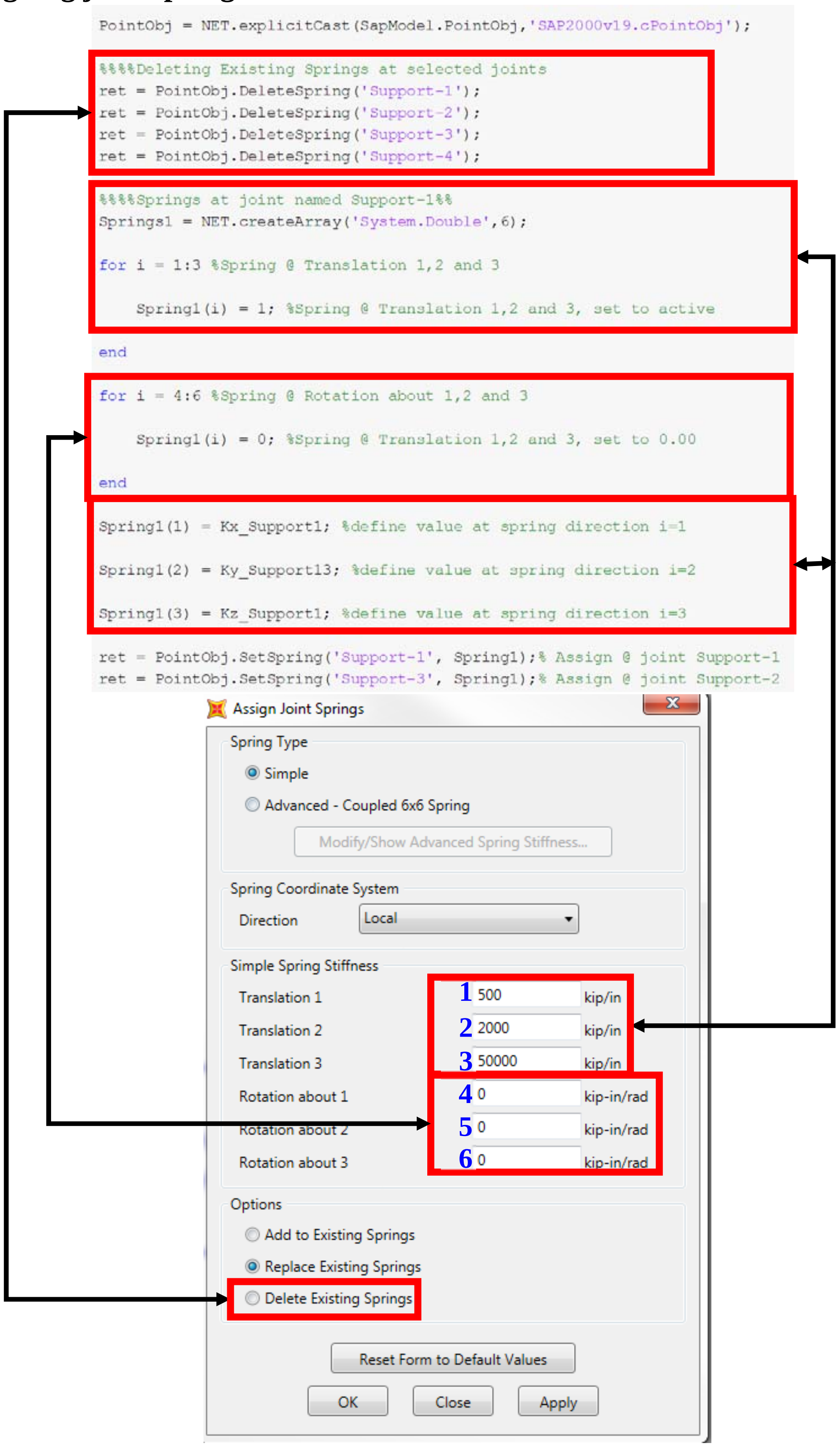
```
% MG main plate girder IXX
MG_ModValue = NET.createArray('System.Double',8);

for i = 1 : 8

    MG_ModValue(i) = 1;

end

MG_ModValue(6) = MG_IXX_Mod;

%Select sections to be modified
PropFrame = NET.explicitCast(SapModel.PropFrame,'SAP2000v19.cPropFrame');
for i=1:4
    ret = PropFrame.SetModifiers(['MG-SEC-',num2str(i)], MG_ModValue);
end
```

Set the 8 property modifiers to 1
Selected frame section name for assigning the property modifier
Frame Property/Stiffness Modification Factors
OK
Cancel

Property/Stiffness Modifiers for Analysis

Cross-section (axial) Area	1	1
Shear Area in 2 direction	2	1
Shear Area in 3 direction	3	1
Torsional Constant	4	1
Moment of Inertia about 2 axis	5	1
Moment of Inertia about 3 axis	6	MG_IXX_Mod
Mass	7	1
Weight	8	1

5. Assigning joint springs



```
PointObj = NET.explicitCast(SapModel.PointObj,'SAP2000v19.cPointObj');

%%%%Deleting Existing Springs at selected joints
ret = PointObj.DeleteSpring('Support-1');
ret = PointObj.DeleteSpring('Support-2');
ret = PointObj.DeleteSpring('Support-3');
ret = PointObj.DeleteSpring('Support-4');

%%%%Springs at joint named Support-1%%
Springs1 = NET.createArray('System.Double',6);

for i = 1:3 %Spring @ Translation 1,2 and 3

    Spring1(i) = 1; %Spring @ Translation 1,2 and 3, set to active

end

for i = 4:6 %Spring @ Rotation about 1,2 and 3

    Spring1(i) = 0; %Spring @ Translation 1,2 and 3, set to 0.00

end

Spring1(1) = Kx_Support1; %define value at spring direction i=1

Spring1(2) = Ky_Support13; %define value at spring direction i=2

Spring1(3) = Kz_Support1; %define value at spring direction i=3

ret = PointObj.SetSpring('Support-1', Spring1);% Assign @ joint Support-1
ret = PointObj.SetSpring('Support-3', Spring1);% Assign @ joint Support-2
```

Assign Joint Springs

Spring Type: [✓ Simple] Advanced - Coupled 6x6 Spring

Spring Coordinate System: Direction: Local

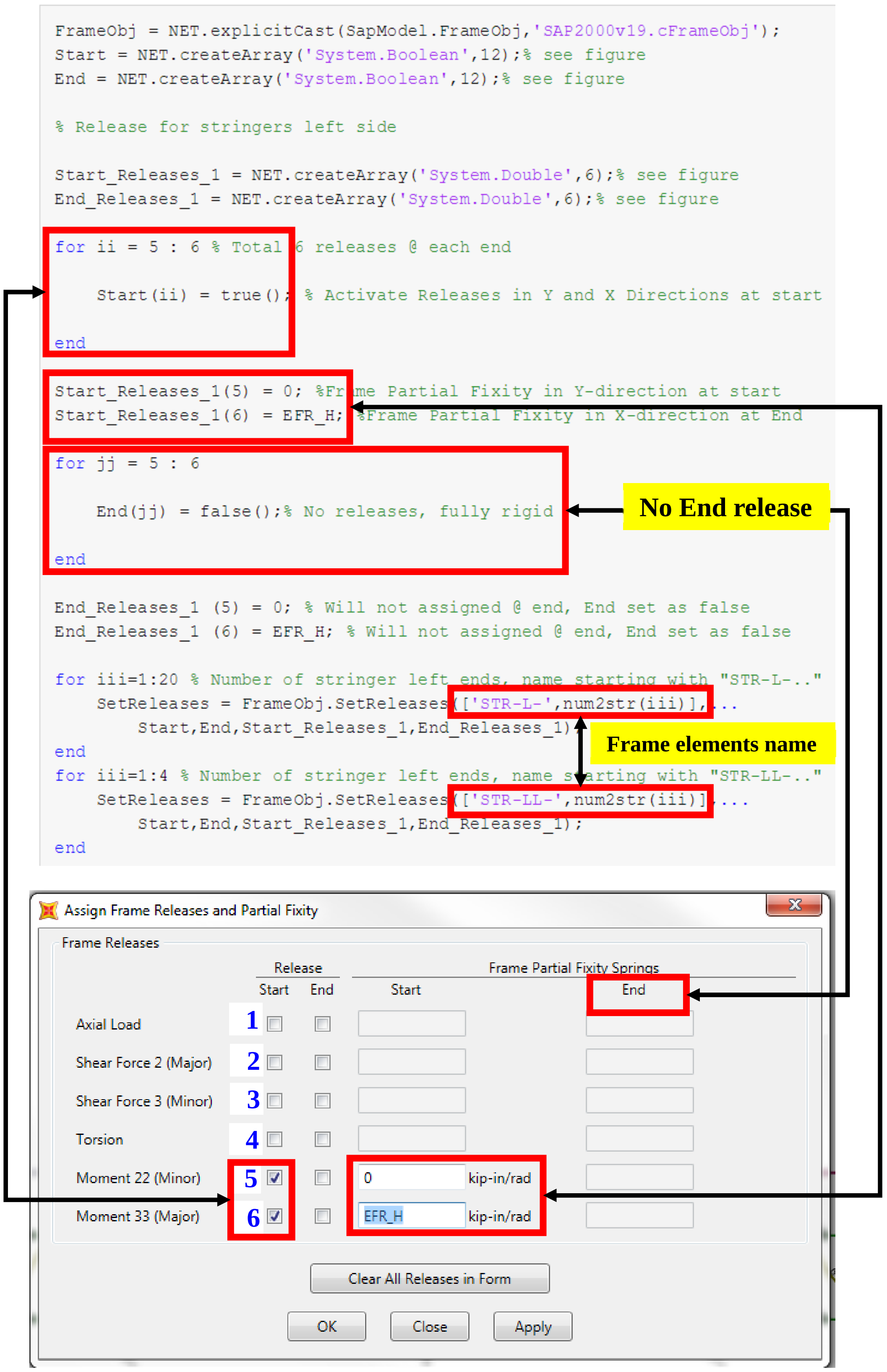
Simple Spring Stiffness

Translation 1	1	500	kip/in
Translation 2	2	2000	kip/in
Translation 3	3	50000	kip/in
Rotation about 1	4	0	kip-in/rad
Rotation about 2	5	0	kip-in/rad
Rotation about 3	6	0	kip-in/rad

Options: Add to Existing Springs [✓ Replace Existing Springs] Delete Existing Springs

Modify/Show Advanced Spring Stiffness...
Reset Form to Default Values
OK
Close
Apply

6. Assigning frame element end releases



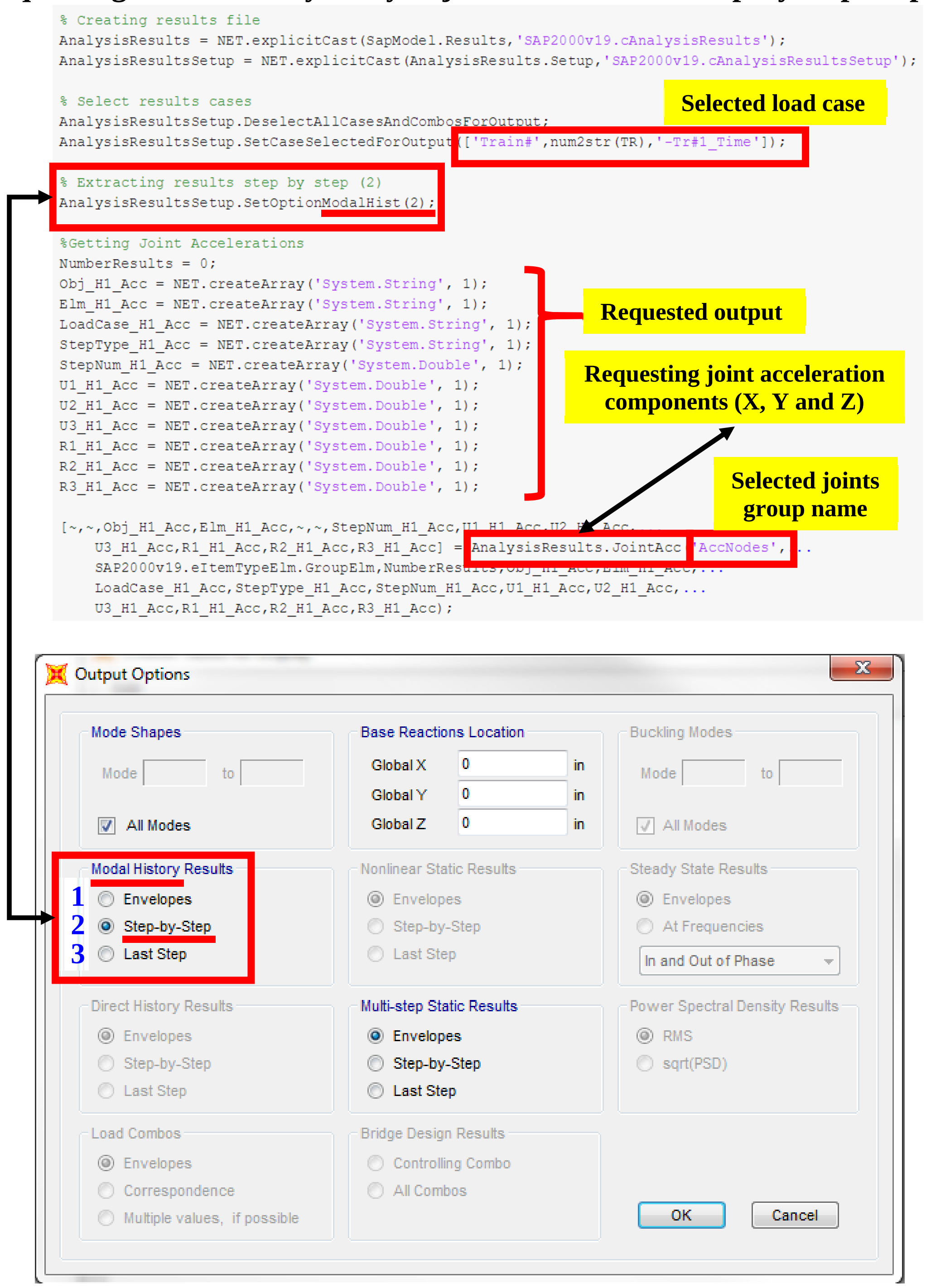
```
FrameObj = NET.explicitCast(SapModel.FrameObj,'SAP2000v19.cFrameObj');
Start = NET.createArray('System.Boolean',12);% see figure
End = NET.createArray('System.Boolean',12);% see figure

% Release for stringers left side

Start_Releases_1 = NET.createArray('System.Double',6);% see figure
End_Releases_1 = NET.createArray('System.Double',6);% see figure

for ii = 5 : 6 % Total 6 releases @ each end

    Start(ii) = true(); % Activate Releases in Y and X Directions at start

end

Start_Releases_1(5) = 0; %Frame Partial Fixity in Y-direction at start
Start_Releases_1(6) = EFR_H; %Frame Partial Fixity in X-direction at End

for jj = 5 : 6

    End(jj) = false();% No releases, fully rigid

end

End_Releases_1 (5) = 0; % Will not assigned @ end, End set as false
End_Releases_1 (6) = EFR_H; % Will not assigned @ end, End set as false

for iii=1:20 % Number of stringer left ends, name starting with "STR-L-.."
    SetReleases = FrameObj.SetReleases(['STR-L-',num2str(iii)],...
        Start,End,Start_Releases_1,End_Releases_1);
end
for iii=1:4 % Number of stringer left ends, name starting with "STR-LL-.."
    SetReleases = FrameObj.SetReleases(['STR-LL-',num2str(iii)],...
        Start,End,Start_Releases_1,End_Releases_1);
end
```

Assign Frame Releases and Partial Fixity

Frame Releases

		Release Start	Release End	Frame Partial Fixity Springs Start		Frame Partial Fixity Springs End
Axial Load	1	☐	☐			
Shear Force 2 (Major)	2	☐	☐			
Shear Force 3 (Minor)	3	☐	☐			
Torsion	4	☐	☐			
Moment 22 (Minor)	5	☑	☐	0	kip-in/rad	
Moment 33 (Major)	6	☑	☐	EFR_H	kip-in/rad	

7. Requesting modal history analysis joint accelerations step-by-step output


```
% Creating results file
AnalysisResults = NET.explicitCast(SapModel.Results,'SAP2000v19.cAnalysisResults');
AnalysisResultsSetup = NET.explicitCast(AnalysisResults.Setup,'SAP2000v19.cAnalysisResultsSetup');

% Select results cases
AnalysisResultsSetup.DeselectAllCasesAndCombosForOutput;
AnalysisResultsSetup.SetCaseSelectedForOutput(['Train#',num2str(TR),'-Tr#1_Time']);

% Extracting results step by step (2)
AnalysisResultsSetup.SetOptionModalHist(2);

%Getting Joint Accelerations
NumberResults = 0;
Obj_H1_Acc = NET.createArray('System.String', 1);
Elm_H1_Acc = NET.createArray('System.String', 1);
LoadCase_H1_Acc = NET.createArray('System.String', 1);
StepType_H1_Acc = NET.createArray('System.String', 1);
StepNum_H1_Acc = NET.createArray('System.Double', 1);
U1_H1_Acc = NET.createArray('System.Double', 1);
U2_H1_Acc = NET.createArray('System.Double', 1);
U3_H1_Acc = NET.createArray('System.Double', 1);
R1_H1_Acc = NET.createArray('System.Double', 1);
R2_H1_Acc = NET.createArray('System.Double', 1);
R3_H1_Acc = NET.createArray('System.Double', 1);

[~,~,Obj_H1_Acc,Elm_H1_Acc,~,~,StepNum_H1_Acc,U1_H1_Acc,U2_H1_Acc,...
    U3_H1_Acc,R1_H1_Acc,R2_H1_Acc,R3_H1_Acc] = AnalysisResults.JointAcc('AccNodes',...
    SAP2000v19.eItemTypeElm.GroupElm,NumberResults,Obj_H1_Acc,Elm_H1_Acc,...
    LoadCase_H1_Acc,StepType_H1_Acc,StepNum_H1_Acc,U1_H1_Acc,U2_H1_Acc,...
    U3_H1_Acc,R1_H1_Acc,R2_H1_Acc,R3_H1_Acc);
```
Selected load case
Requested output
Requesting joint acceleration components (X, Y and Z)
Selected joints group name
Output Options
1
2
3
OK
Cancel

Mode Shapes: Mode ___ to ___ ; [✓] All Modes

Base Reactions Location: Global X 0 in; Global Y 0 in; Global Z 0 in

Buckling Modes: Mode ___ to ___ ; [✓] All Modes

Modal History Results: Envelopes [✓ Step-by-Step] Last Step

Nonlinear Static Results: [✓ Envelopes] Step-by-Step Last Step

Steady State Results: [✓ Envelopes] At Frequencies; In and Out of Phase

Direct History Results: [✓ Envelopes] Step-by-Step Last Step

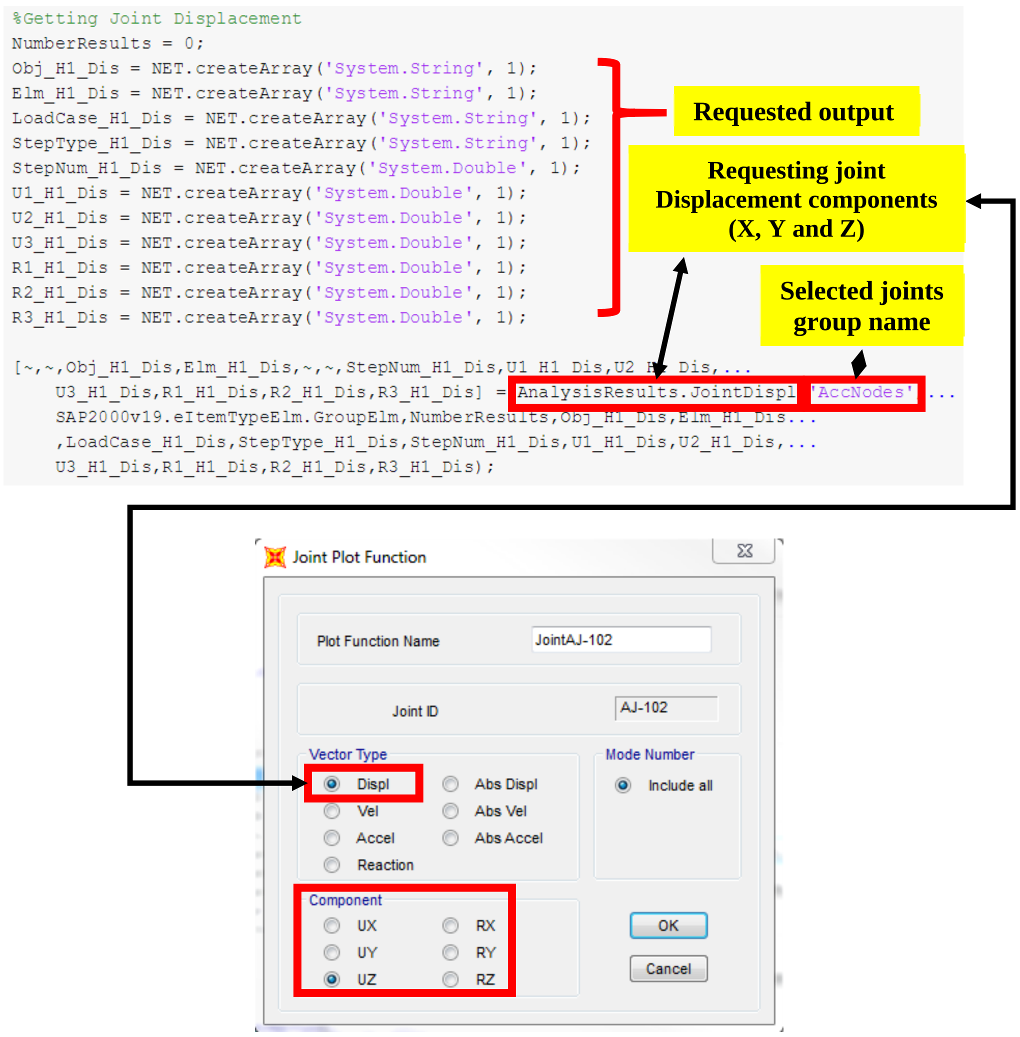
Multi-step Static Results: [✓ Envelopes] Step-by-Step Last Step

Power Spectral Density Results: [✓ RMS] sqrt(PSD)

Load Combos: [✓ Envelopes] Correspondence Multiple values, if possible

Bridge Design Results: Controlling Combo All Combos

8. Requesting joint displacements output



```
%Getting Joint Displacement
NumberResults = 0;
Obj_H1_Dis = NET.createArray('System.String', 1);
Elm_H1_Dis = NET.createArray('System.String', 1);
LoadCase_H1_Dis = NET.createArray('System.String', 1);
StepType_H1_Dis = NET.createArray('System.String', 1);
StepNum_H1_Dis = NET.createArray('System.Double', 1);
U1_H1_Dis = NET.createArray('System.Double', 1);
U2_H1_Dis = NET.createArray('System.Double', 1);
U3_H1_Dis = NET.createArray('System.Double', 1);
R1_H1_Dis = NET.createArray('System.Double', 1);
R2_H1_Dis = NET.createArray('System.Double', 1);
R3_H1_Dis = NET.createArray('System.Double', 1);

[~,~,Obj_H1_Dis,Elm_H1_Dis,~,~,StepNum_H1_Dis,U1_H1_Dis,U2_H1_Dis,...
    U3_H1_Dis,R1_H1_Dis,R2_H1_Dis,R3_H1_Dis] = AnalysisResults.JointDispl('AccNodes',...
    SAP2000v19.eItemTypeElm.GroupElm,NumberResults,Obj_H1_Dis,Elm_H1_Dis...
    ,LoadCase_H1_Dis,StepType_H1_Dis,StepNum_H1_Dis,U1_H1_Dis,U2_H1_Dis,...
    U3_H1_Dis,R1_H1_Dis,R2_H1_Dis,R3_H1_Dis);
```

Requested output
Requesting joint Displacement components (X, Y and Z)
Selected joints group name

Joint Plot Function

Plot Function Name: JointAJ-102

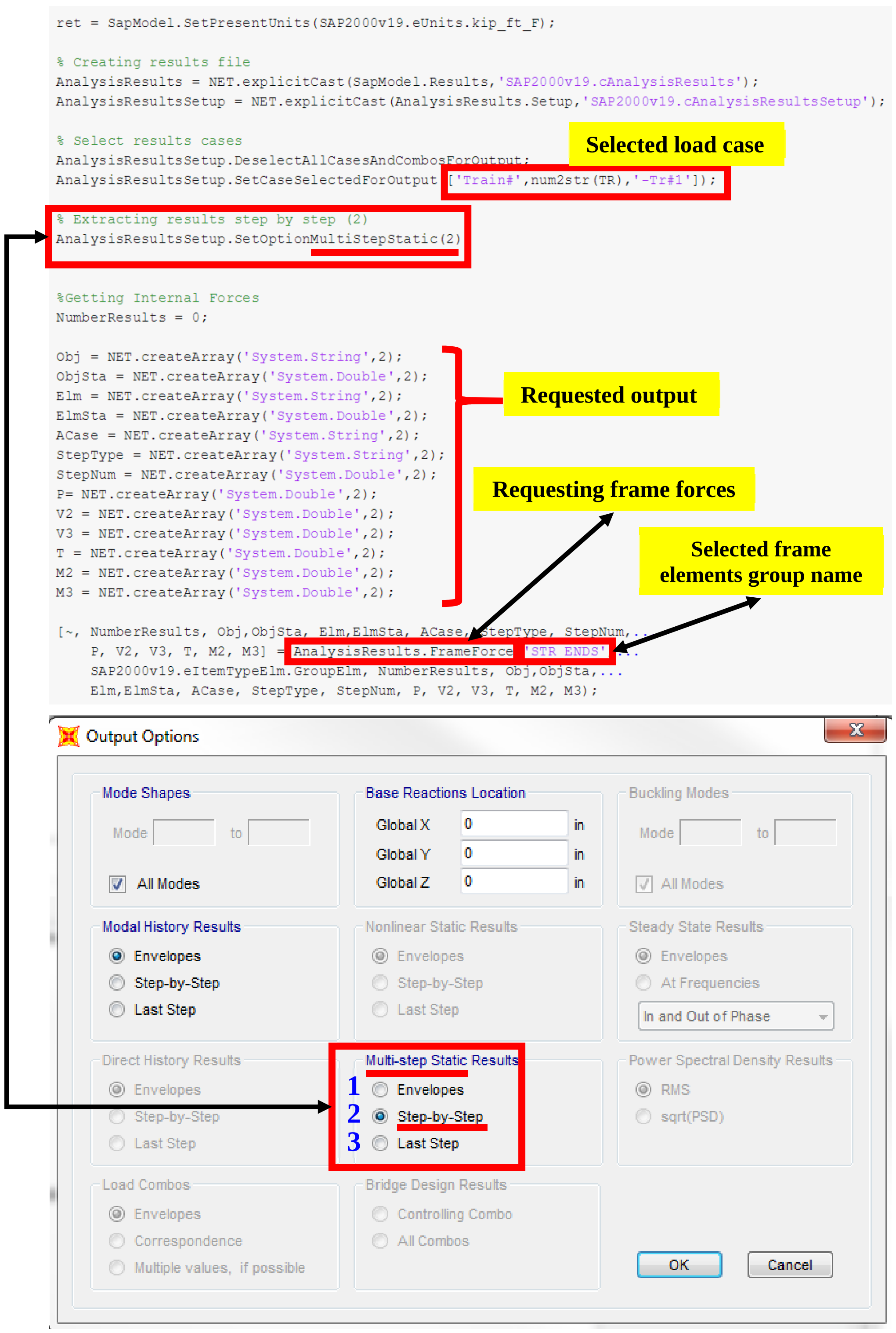
Joint ID: AJ-102

Vector Type: [✓ Displ] Vel Accel Reaction Abs Displ Abs Vel Abs Accel

Mode Number: [✓ Include all]

Component: UX UY [✓ UZ] RX RY RZ

9. Requesting multi-step-static analysis frame element forces step-by-step output



```
ret = SapModel.SetPresentUnits(SAP2000v19.eUnits.kip_ft_F);

% Creating results file
AnalysisResults = NET.explicitCast(SapModel.Results,'SAP2000v19.cAnalysisResults');
AnalysisResultsSetup = NET.explicitCast(AnalysisResults.Setup,'SAP2000v19.cAnalysisResultsSetup');

% Select results cases
AnalysisResultsSetup.DeselectAllCasesAndCombosForOutput;
AnalysisResultsSetup.SetCaseSelectedForOutput ['Train#',num2str(TR),'-Tr#1']);

% Extracting results step by step (2)
AnalysisResultsSetup.SetOptionMultiStepStatic(2)

%Getting Internal Forces
NumberResults = 0;

Obj = NET.createArray('System.String',2);
ObjSta = NET.createArray('System.Double',2);
Elm = NET.createArray('System.String',2);
ElmSta = NET.createArray('System.Double',2);
ACase = NET.createArray('System.String',2);
StepType = NET.createArray('System.String',2);
StepNum = NET.createArray('System.Double',2);
P= NET.createArray('System.Double',2);
V2 = NET.createArray('System.Double',2);
V3 = NET.createArray('System.Double',2);
T = NET.createArray('System.Double',2);
M2 = NET.createArray('System.Double',2);
M3 = NET.createArray('System.Double',2);

[~, NumberResults, Obj,ObjSta, Elm,ElmSta, ACase, StepType, StepNum,...
    P, V2, V3, T, M2, M3] = AnalysisResults.FrameForce 'STR ENDS' ...
    SAP2000v19.eItemTypeElm.GroupElm, NumberResults, Obj,ObjSta,...
    Elm,ElmSta, ACase, StepType, StepNum, P, V2, V3, T, M2, M3);
```

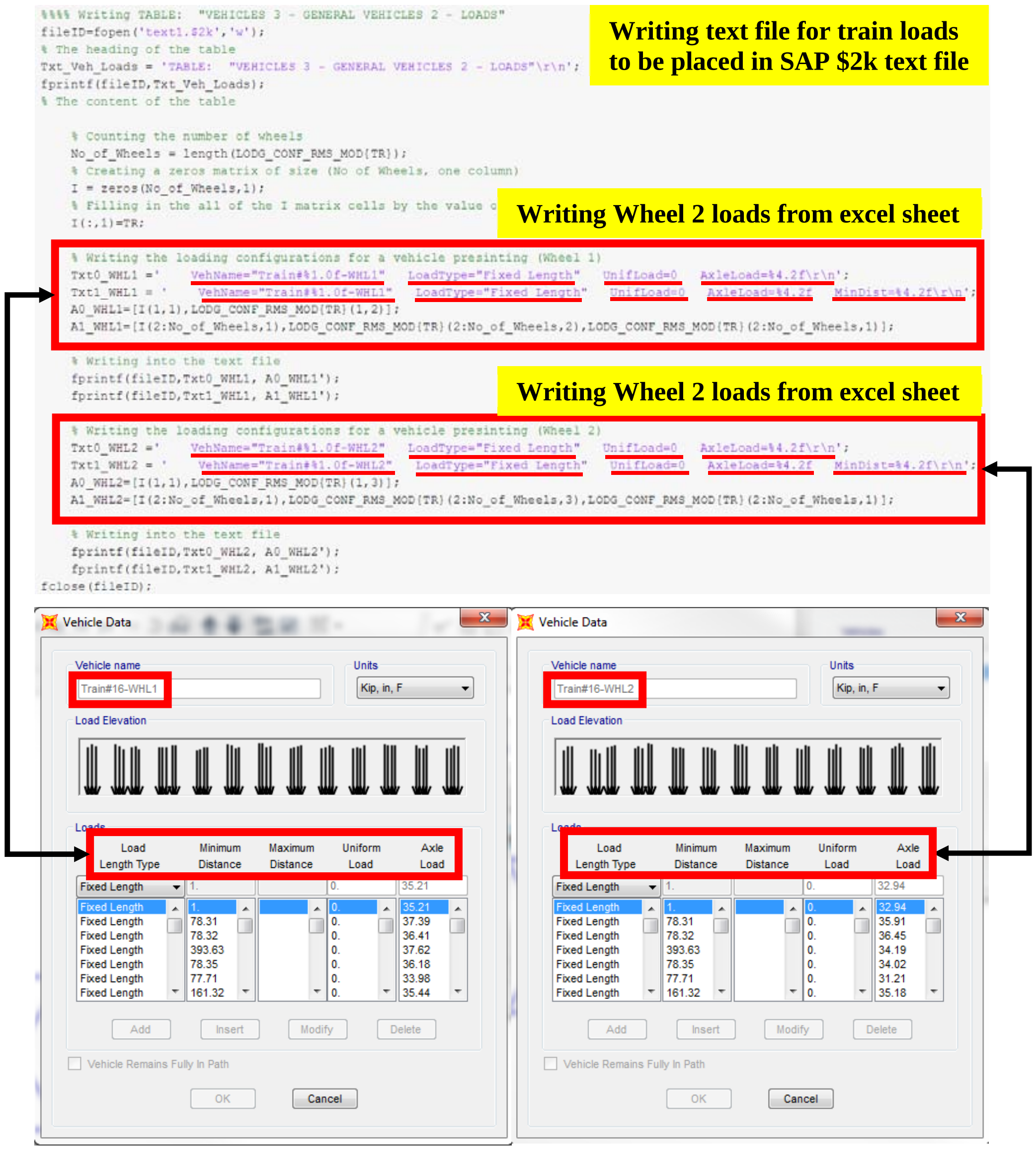
Output Options

Mode Shapes — Mode ___ to ___ ; [✓] All Modes
Base Reactions Location — Global X 0 in; Global Y 0 in; Global Z 0 in
Buckling Modes — Mode ___ to ___ ; [✓] All Modes
Modal History Results — [✓ Envelopes] Step-by-Step Last Step
Nonlinear Static Results — [✓ Envelopes] Step-by-Step Last Step
Steady State Results — [✓ Envelopes] At Frequencies; In and Out of Phase
Direct History Results — [✓ Envelopes] Step-by-Step Last Step
Multi-step Static Results — Envelopes [✓ Step-by-Step] Last Step
Power Spectral Density Results — [✓ RMS] sqrt(PSD)
Load Combos — [✓ Envelopes] Correspondence Multiple values, if possible
Bridge Design Results — Controlling Combo All Combos

OK Cancel

10. Writing train loads related text files and placing them in the SAP2000 original model $2k file. Train loads and names found to affect 8 different locations in the $2k file and as a result 8 different text files will be created and replaced with their corresponding locations in the $2k file automatically

a) Writing train loads in a text file where Matlab read loads, axle spacing and number of axles from an excel sheet provided from the bridge owner automatically.


Writing text file for train loads to be placed in SAP $2k text file
Writing Wheel 2 loads from excel sheet
Writing Wheel 2 loads from excel sheet

```
%%%% Writing TABLE:  "VEHICLES 3 - GENERAL VEHICLES 2 - LOADS"
fileID=fopen('text1.$2k','w');
% The heading of the table
Txt_Veh_Loads = 'TABLE:  "VEHICLES 3 - GENERAL VEHICLES 2 - LOADS"\r\n';
fprintf(fileID,Txt_Veh_Loads);
% The content of the table

    % Counting the number of wheels
    No_of_Wheels = length(LODG_CONF_RMS_MOD{TR});
    % Creating a zeros matrix of size (No of Wheels, one column)
    I = zeros(No_of_Wheels,1);
    % Filling in the all of the I matrix cells by the value o
    I(:,1)=TR;

    % Writing the loading configurations for a vehicle presinting (Wheel 1)
    Txt0_WHL1 ='    VehName="Train#%1.0f-WHL1"   LoadType="Fixed Length"   UnifLoad=0   AxleLoad=%4.2f\r\n';
    Txt1_WHL1 = '    VehName="Train#%1.0f-WHL1"   LoadType="Fixed Length"   UnifLoad=0   AxleLoad=%4.2f   MinDist=%4.2f\r\n';
    A0_WHL1=[I(1,1),LODG_CONF_RMS_MOD{TR}(1,2)];
    A1_WHL1=[I(2:No_of_Wheels,1),LODG_CONF_RMS_MOD{TR}(2:No_of_Wheels,2),LODG_CONF_RMS_MOD{TR}(2:No_of_Wheels,1)];

    % Writing into the text file
    fprintf(fileID,Txt0_WHL1, A0_WHL1');
    fprintf(fileID,Txt1_WHL1, A1_WHL1');

    % Writing the loading configurations for a vehicle presinting (Wheel 2)
    Txt0_WHL2 ='    VehName="Train#%1.0f-WHL2"   LoadType="Fixed Length"   UnifLoad=0   AxleLoad=%4.2f\r\n';
    Txt1_WHL2 = '    VehName="Train#%1.0f-WHL2"   LoadType="Fixed Length"   UnifLoad=0   AxleLoad=%4.2f   MinDist=%4.2f\r\n';
    A0_WHL2=[I(1,1),LODG_CONF_RMS_MOD{TR}(1,3)];
    A1_WHL2=[I(2:No_of_Wheels,1),LODG_CONF_RMS_MOD{TR}(2:No_of_Wheels,3),LODG_CONF_RMS_MOD{TR}(2:No_of_Wheels,1)];

    % Writing into the text file
    fprintf(fileID,Txt0_WHL2, A0_WHL2');
    fprintf(fileID,Txt1_WHL2, A1_WHL2');
fclose(fileID);
```

Vehicle Data

Vehicle name: Train#16-WHL1
Units: Kip, in, F

Load Length Type	Minimum Distance	Maximum Distance	Uniform Load	Axle Load
Fixed Length	1.		0.	35.21
Fixed Length	1.		0.	35.21
Fixed Length	78.31		0.	37.39
Fixed Length	78.32		0.	36.41
Fixed Length	393.63		0.	37.62
Fixed Length	78.35		0.	36.18
Fixed Length	77.71		0.	33.98
Fixed Length	161.32		0.	35.44

Vehicle Remains Fully In Path

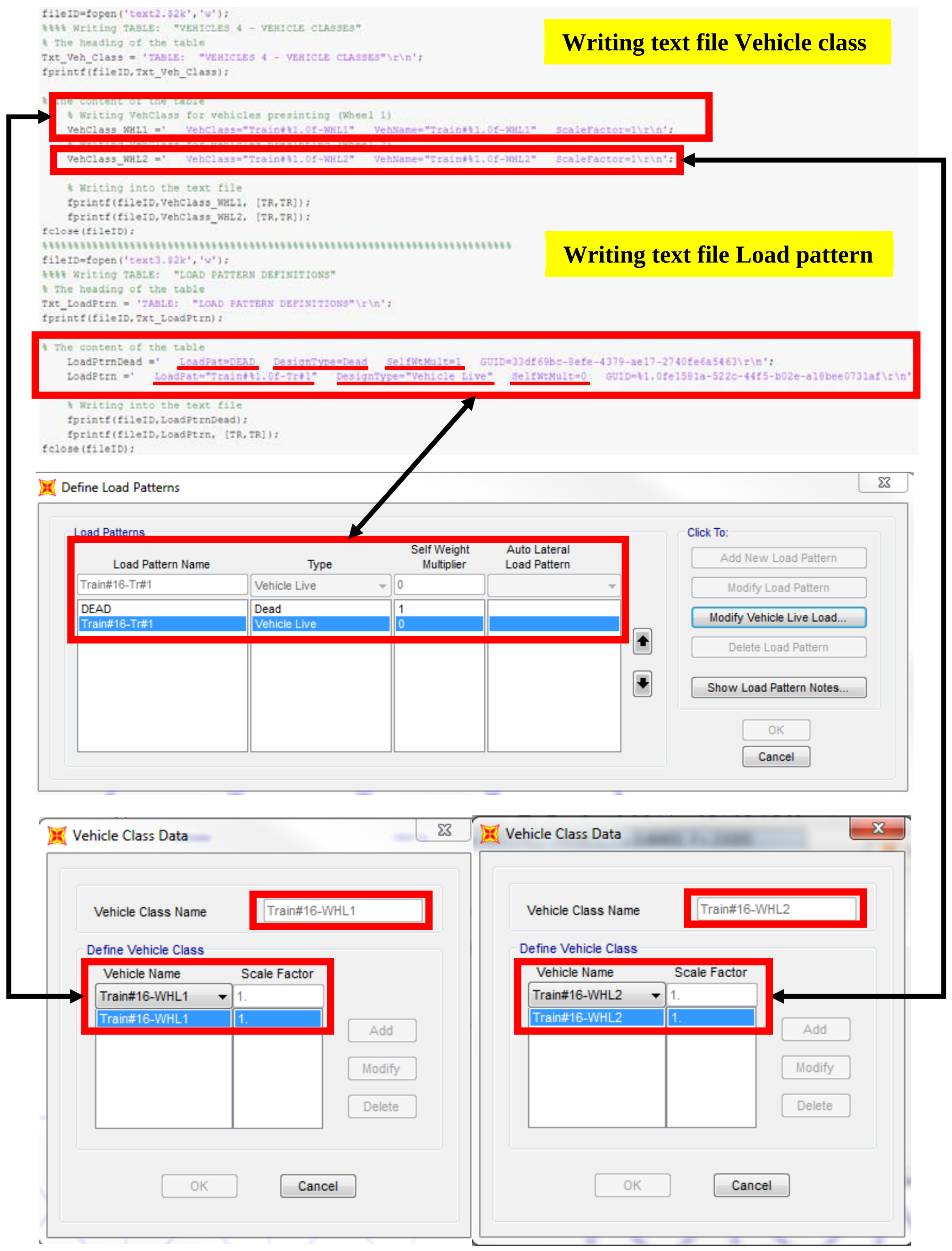
Vehicle Data

Vehicle name: Train#16-WHL2
Units: Kip, in, F

Load Length Type	Minimum Distance	Maximum Distance	Uniform Load	Axle Load
Fixed Length	1.		0.	32.94
Fixed Length	1.		0.	32.94
Fixed Length	78.31		0.	35.91
Fixed Length	78.32		0.	36.45
Fixed Length	393.63		0.	34.19
Fixed Length	78.35		0.	34.02
Fixed Length	77.71		0.	31.21
Fixed Length	161.32		0.	35.18

Vehicle Remains Fully In Path

b) Writing load pattern and vehicle class portions, 2 text files.



Writing text file Vehicle class
Writing text file Load pattern

```
fileID=fopen('text2.$2k','w');
%%%% Writing TABLE:  "VEHICLES 4 - VEHICLE CLASSES"
% The heading of the table
Txt_Veh_Class = 'TABLE:  "VEHICLES 4 - VEHICLE CLASSES"\r\n';
fprintf(fileID,Txt_Veh_Class);

% The content of the table
    % Writing VehClass for vehicles presinting (Wheel 1)
    VehClass_WHL1 ='   VehClass="Train#%1.0f-WHL1"   VehName="Train#%1.0f-WHL1"   ScaleFactor=1\r\n';
    % Writing VehClass for vehicles presinting (Wheel 2)
    VehClass_WHL2 ='   VehClass="Train#%1.0f-WHL2"   VehName="Train#%1.0f-WHL2"   ScaleFactor=1\r\n';

    % Writing into the text file
    fprintf(fileID,VehClass_WHL1, [TR,TR]);
    fprintf(fileID,VehClass_WHL2, [TR,TR]);
fclose(fileID);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fileID=fopen('text3.$2k','w');
%%%% Writing TABLE:  "LOAD PATTERN DEFINITIONS"
% The heading of the table
Txt_LoadPtrn = 'TABLE:  "LOAD PATTERN DEFINITIONS"\r\n';
fprintf(fileID,Txt_LoadPtrn);

% The content of the table
    LoadPtrnDead ='   LoadPat=DEAD   DesignType=Dead   SelfWtMult=1   GUID=33df69bc-8efe-4379-ae17-2740fe6a5463\r\n';
    LoadPtrn ='   LoadPat="Train#%1.0f-Tr#1"   DesignType="Vehicle Live"   SelfWtMult=0   GUID=%1.0fe1581a-522c-44f5-b02e-a18bee0731af\r\n'
    % Writing into the text file
    fprintf(fileID,LoadPtrnDead);
    fprintf(fileID,LoadPtrn, [TR,TR]);
fclose(fileID);
```

Define Load Patterns

Load Patterns

Load Pattern Name	Type	Self Weight Multiplier	Auto Lateral Load Pattern
Train#16-Tr#1	Vehicle Live	0	
DEAD	Dead	1	
Train#16-Tr#1	Vehicle Live	0	

Vehicle Class Data

Vehicle Class Name: Train#16-WHL1

Define Vehicle Class

Vehicle Name	Scale Factor
Train#16-WHL1	1.
Train#16-WHL1	1.

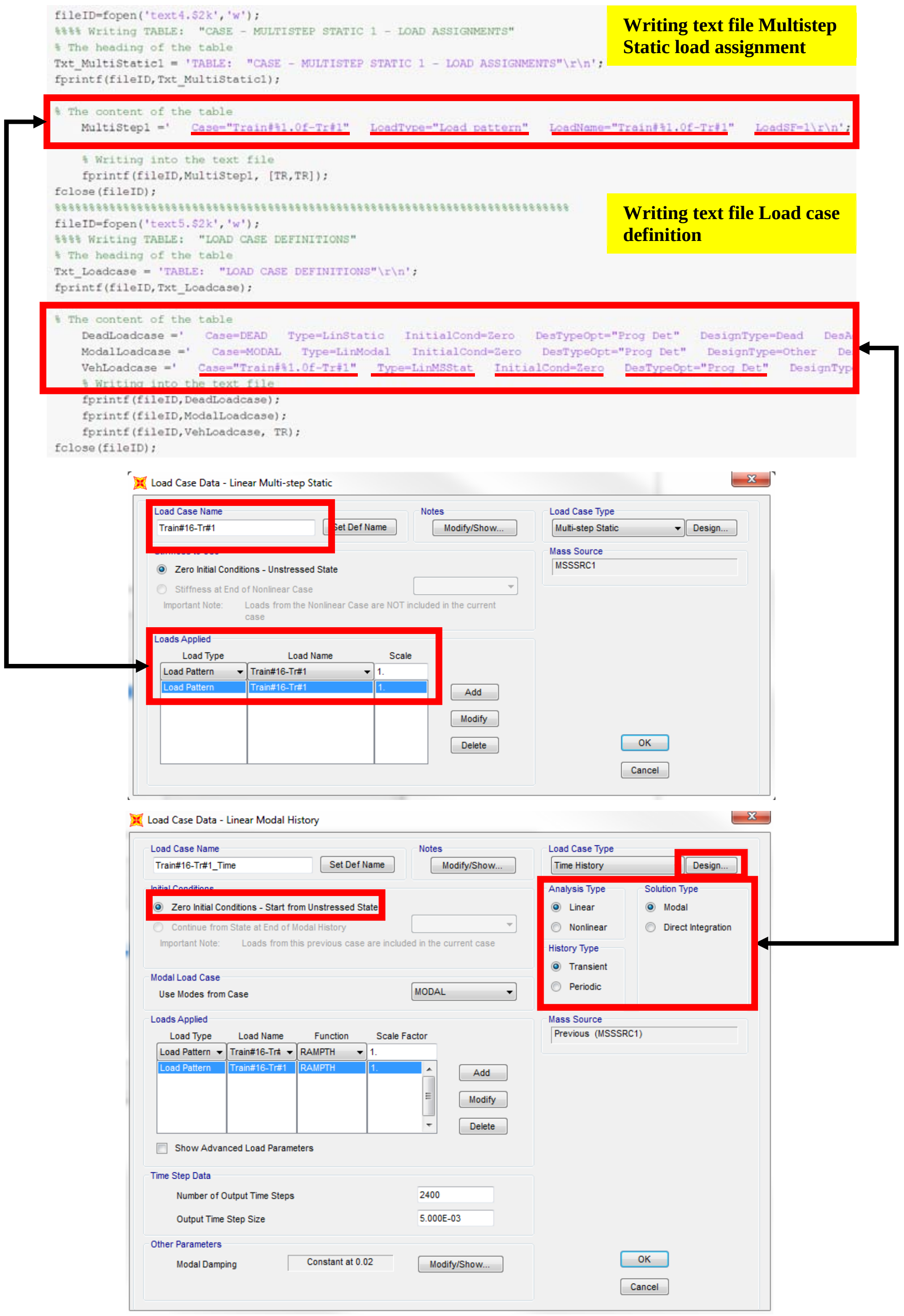
Vehicle Class Data

Vehicle Class Name: Train#16-WHL2

Define Vehicle Class

Vehicle Name	Scale Factor
Train#16-WHL2	1.
Train#16-WHL2	1.

c) Writing multistep-step-static load assignment and load case definition portions, 2 text files.



```
fileID=fopen('text4.$2k','w');
%%%% Writing TABLE:  "CASE - MULTISTEP STATIC 1 - LOAD ASSIGNMENTS"
% The heading of the table
Txt_MultiStatic1 = 'TABLE:  "CASE - MULTISTEP STATIC 1 - LOAD ASSIGNMENTS"\r\n';
fprintf(fileID,Txt_MultiStatic1);

% The content of the table
    MultiStep1 ='   Case="Train#%1.0f-Tr#1"   LoadType="Load pattern"   LoadName="Train#%1.0f-Tr#1"   LoadSF=1\r\n';

    % Writing into the text file
    fprintf(fileID,MultiStep1, [TR,TR]);
fclose(fileID);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fileID=fopen('text5.$2k','w');
%%%% Writing TABLE:  "LOAD CASE DEFINITIONS"
% The heading of the table
Txt_Loadcase = 'TABLE:  "LOAD CASE DEFINITIONS"\r\n';
fprintf(fileID,Txt_Loadcase);

% The content of the table
    DeadLoadcase ='   Case=DEAD   Type=LinStatic   InitialCond=Zero   DesTypeOpt="Prog Det"   DesignType=Dead   DesA
    ModalLoadcase ='   Case=MODAL   Type=LinModal   InitialCond=Zero   DesTypeOpt="Prog Det"   DesignType=Other   De
    VehLoadcase ='   Case="Train#%1.0f-Tr#1"   Type=LinMSStat   InitialCond=Zero   DesTypeOpt="Prog Det"   DesignTyp
    % Writing into the text file
    fprintf(fileID,DeadLoadcase);
    fprintf(fileID,ModalLoadcase);
    fprintf(fileID,VehLoadcase, TR);
fclose(fileID);
```

Writing text file Multistep Static load assignment

Writing text file Load case definition

Load Case Data - Linear Multi-step Static

Load Case Name: Train#16-Tr#1
Load Case Type: Multi-step Static
Mass Source: MSSSRC1
[✓ Zero Initial Conditions - Unstressed State] Stiffness at End of Nonlinear Case
Important Note: Loads from the Nonlinear Case are NOT included in the current case

Loads Applied

Load Type	Load Name	Scale
Load Pattern	Train#16-Tr#1	1.
Load Pattern	Train#16-Tr#1	1.

Load Case Data - Linear Modal History

Load Case Name: Train#16-Tr#1_Time
Load Case Type: Time History
Initial Conditions: [✓ Zero Initial Conditions - Start from Unstressed State] Continue from State at End of Modal History
Important Note: Loads from this previous case are included in the current case
Analysis Type: [✓ Linear] Nonlinear
Solution Type: [✓ Modal] Direct Integration
History Type: [✓ Transient] Periodic
Modal Load Case: Use Modes from Case MODAL
Mass Source: Previous (MSSSRC1)

Loads Applied

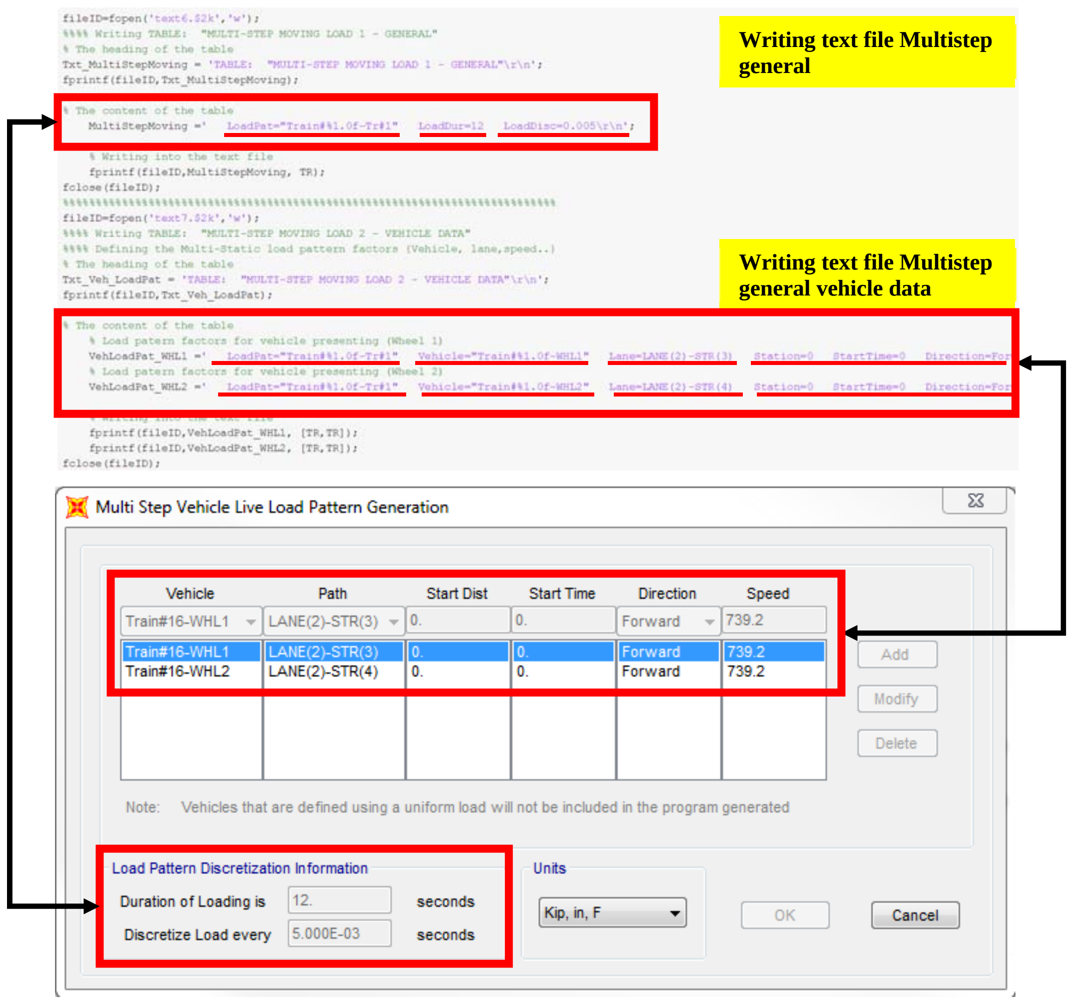
Load Type	Load Name	Function	Scale Factor
Load Pattern	Train#16-Tr#	RAMPTH	1.
Load Pattern	Train#16-Tr#1	RAMPTH	1.

Show Advanced Load Parameters

Time Step Data
Number of Output Time Steps: 2400
Output Time Step Size: 5.000E-03

Other Parameters
Modal Damping: Constant at 0.02

d) Writing multi-step general and multi-step general vehicle data, 2 text files.


Writing text file Multistep general
Writing text file Multistep general vehicle data

```
fileID=fopen('text6.$2k','w');
%%%% Writing TABLE:  "MULTI-STEP MOVING LOAD 1 - GENERAL"
% The heading of the table
Txt_MultiStepMoving = 'TABLE:  "MULTI-STEP MOVING LOAD 1 - GENERAL"\r\n';
fprintf(fileID,Txt_MultiStepMoving);

% The content of the table
    MultiStepMoving ='   LoadPat="Train#%1.0f-Tr#1"   LoadDur=12   LoadDisc=0.005\r\n';
    % Writing into the text file
    fprintf(fileID,MultiStepMoving, TR);
fclose(fileID);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fileID=fopen('text7.$2k','w');
%%%% Writing TABLE:  "MULTI-STEP MOVING LOAD 2 - VEHICLE DATA"
%%%% Defining the Multi-Static load pattern factors (Vehicle, lane,speed..)
% The heading of the table
Txt_Veh_LoadPat = 'TABLE:  "MULTI-STEP MOVING LOAD 2 - VEHICLE DATA"\r\n';
fprintf(fileID,Txt_Veh_LoadPat);

% The content of the table
    % Load patern factors for vehicle presenting (Wheel 1)
    VehLoadPat_WHL1 ='   LoadPat="Train#%1.0f-Tr#1"   Vehicle="Train#%1.0f-WHL1"   Lane=LANE(2)-STR(3)   Station=0   StartTime=0   Direction=For
    % Load patern factors for vehicle presenting (Wheel 2)
    VehLoadPat_WHL2 ='   LoadPat="Train#%1.0f-Tr#1"   Vehicle="Train#%1.0f-WHL2"   Lane=LANE(2)-STR(4)   Station=0   StartTime=0   Direction=For

    % Writing into the text file
    fprintf(fileID,VehLoadPat_WHL1, [TR,TR]);
    fprintf(fileID,VehLoadPat_WHL2, [TR,TR]);
fclose(fileID);
```

Multi Step Vehicle Live Load Pattern Generation

Vehicle	Path	Start Dist	Start Time	Direction	Speed
Train#16-WHL1	LANE(2)-STR(3)	0.	0.	Forward	739.2
Train#16-WHL1	LANE(2)-STR(3)	0.	0.	Forward	739.2
Train#16-WHL2	LANE(2)-STR(4)	0.	0.	Forward	739.2

Note: Vehicles that are defined using a uniform load will not be included in the program generated

Load Pattern Discretization Information

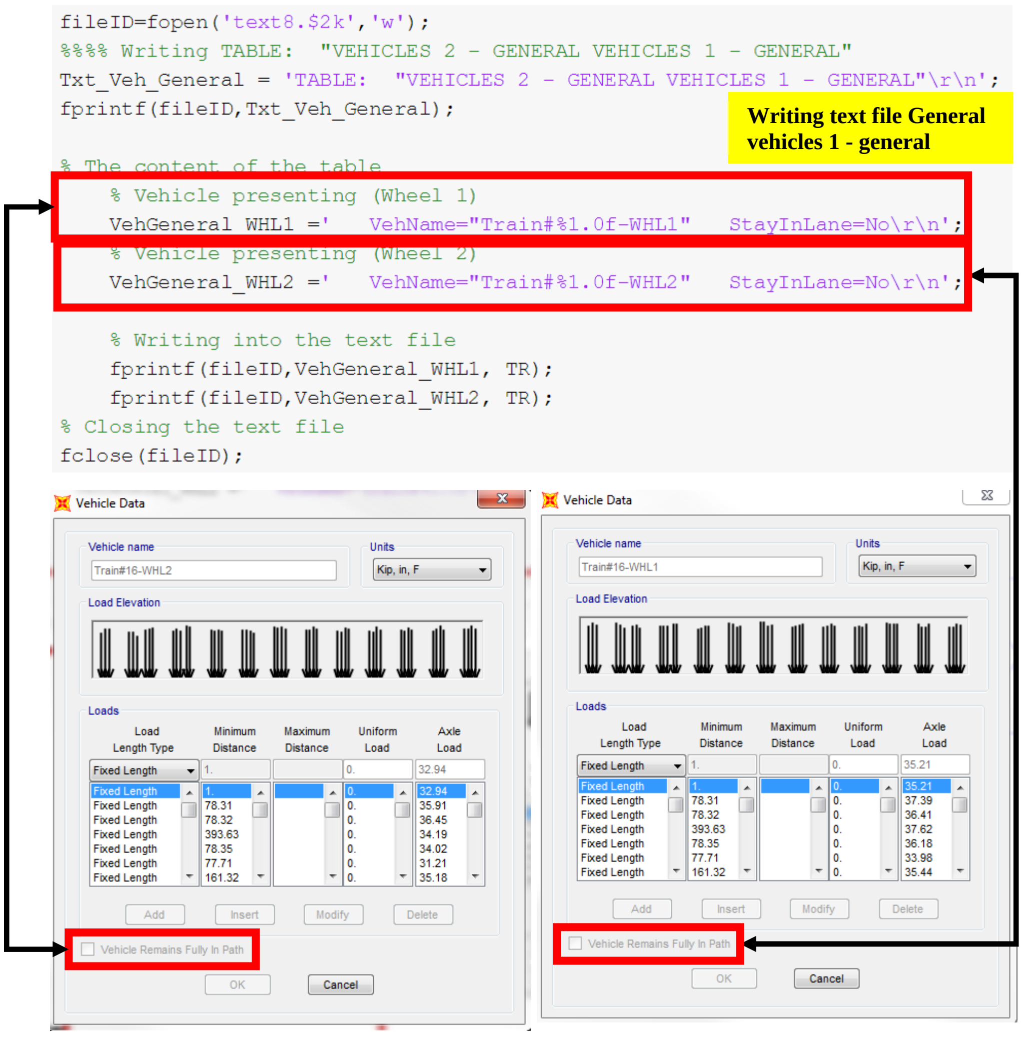
Duration of Loading is 12. seconds

Discretize Load every 5.000E-03 seconds

Units

Kip, in, F

e) Writing general vehicles 1, text files.



```
fileID=fopen('text8.$2k','w');
%%%% Writing TABLE:  "VEHICLES 2 - GENERAL VEHICLES 1 - GENERAL"
Txt_Veh_General = 'TABLE:  "VEHICLES 2 - GENERAL VEHICLES 1 - GENERAL"\r\n';
fprintf(fileID,Txt_Veh_General);

% The content of the table
    % Vehicle presenting (Wheel 1)
    VehGeneral_WHL1 ='   VehName="Train#%1.0f-WHL1"   StayInLane=No\r\n';
    % Vehicle presenting (Wheel 2)
    VehGeneral_WHL2 ='   VehName="Train#%1.0f-WHL2"   StayInLane=No\r\n';

    % Writing into the text file
    fprintf(fileID,VehGeneral_WHL1, TR);
    fprintf(fileID,VehGeneral_WHL2, TR);
% Closing the text file
fclose(fileID);
```

Vehicle name: Train#16-WHL2
Units: Kip, in, F

Loads

Load Length Type	Minimum Distance	Maximum Distance	Uniform Load	Axle Load
Fixed Length	1.		0.	32.94
Fixed Length	1.		0.	32.94
Fixed Length	78.31		0.	35.91
Fixed Length	78.32		0.	36.45
Fixed Length	393.63		0.	34.19
Fixed Length	78.35		0.	34.02
Fixed Length	77.71		0.	31.21
Fixed Length	161.32		0.	35.18

☐ Vehicle Remains Fully In Path

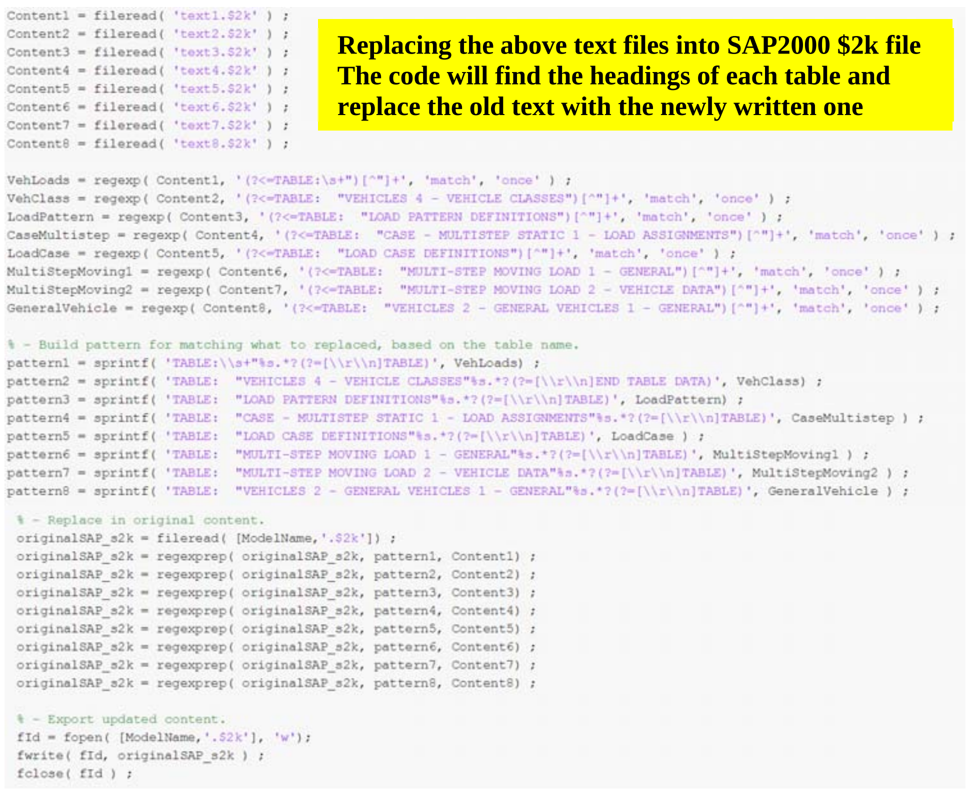
Vehicle name: Train#16-WHL1
Units: Kip, in, F

Loads

Load Length Type	Minimum Distance	Maximum Distance	Uniform Load	Axle Load
Fixed Length	1.		0.	35.21
Fixed Length	1.		0.	35.21
Fixed Length	78.31		0.	37.39
Fixed Length	78.32		0.	36.41
Fixed Length	393.63		0.	37.62
Fixed Length	78.35		0.	36.18
Fixed Length	77.71		0.	33.98
Fixed Length	161.32		0.	35.44

☐ Vehicle Remains Fully In Path

f) Replacing text files generated automatically in (a to f) with the existing old corresponding portions in the SAP2000 $2k file.



Replacing the above text files into SAP2000 $2k file
The code will find the headings of each table and replace the old text with the newly written one

```
Content1 = fileread( 'text1.$2k' ) ;
Content2 = fileread( 'text2.$2k' ) ;
Content3 = fileread( 'text3.$2k' ) ;
Content4 = fileread( 'text4.$2k' ) ;
Content5 = fileread( 'text5.$2k' ) ;
Content6 = fileread( 'text6.$2k' ) ;
Content7 = fileread( 'text7.$2k' ) ;
Content8 = fileread( 'text8.$2k' ) ;

VehLoads = regexp( Content1, '(?<=TABLE:\s+")[^"]+', 'match', 'once' ) ;
VehClass = regexp( Content2, '(?<=TABLE:  "VEHICLES 4 - VEHICLE CLASSES")[^"]+', 'match', 'once' ) ;
LoadPattern = regexp( Content3, '(?<=TABLE:  "LOAD PATTERN DEFINITIONS")[^"]+', 'match', 'once' ) ;
CaseMultistep = regexp( Content4, '(?<=TABLE:  "CASE - MULTISTEP STATIC 1 - LOAD ASSIGNMENTS")[^"]+', 'match', 'once' ) ;
LoadCase = regexp( Content5, '(?<=TABLE:  "LOAD CASE DEFINITIONS")[^"]+', 'match', 'once' ) ;
MultiStepMoving1 = regexp( Content6, '(?<=TABLE:  "MULTI-STEP MOVING LOAD 1 - GENERAL")[^"]+', 'match', 'once' ) ;
MultiStepMoving2 = regexp( Content7, '(?<=TABLE:  "MULTI-STEP MOVING LOAD 2 - VEHICLE DATA")[^"]+', 'match', 'once' ) ;
GeneralVehicle = regexp( Content8, '(?<=TABLE:  "VEHICLES 2 - GENERAL VEHICLES 1 - GENERAL")[^"]+', 'match', 'once' ) ;

% - Build pattern for matching what to replaced, based on the table name.
pattern1 = sprintf( 'TABLE:\\s+"%s.*?(?=[\\r\\n]TABLE)', VehLoads) ;
pattern2 = sprintf( 'TABLE:  "VEHICLES 4 - VEHICLE CLASSES"%s.*?(?=[\\r\\n]END TABLE DATA)', VehClass) ;
pattern3 = sprintf( 'TABLE:  "LOAD PATTERN DEFINITIONS"%s.*?(?=[\\r\\n]TABLE)', LoadPattern) ;
pattern4 = sprintf( 'TABLE:  "CASE - MULTISTEP STATIC 1 - LOAD ASSIGNMENTS"%s.*?(?=[\\r\\n]TABLE)', CaseMultistep ) ;
pattern5 = sprintf( 'TABLE:  "LOAD CASE DEFINITIONS"%s.*?(?=[\\r\\n]TABLE)', LoadCase ) ;
pattern6 = sprintf( 'TABLE:  "MULTI-STEP MOVING LOAD 1 - GENERAL"%s.*?(?=[\\r\\n]TABLE)', MultiStepMoving1 ) ;
pattern7 = sprintf( 'TABLE:  "MULTI-STEP MOVING LOAD 2 - VEHICLE DATA"%s.*?(?=[\\r\\n]TABLE)', MultiStepMoving2 ) ;
pattern8 = sprintf( 'TABLE:  "VEHICLES 2 - GENERAL VEHICLES 1 - GENERAL"%s.*?(?=[\\r\\n]TABLE)', GeneralVehicle ) ;

 % - Replace in original content.
 originalSAP_s2k = fileread( [ModelName,'.$2k']) ;
 originalSAP_s2k = regexprep( originalSAP_s2k, pattern1, Content1) ;
 originalSAP_s2k = regexprep( originalSAP_s2k, pattern2, Content2) ;
 originalSAP_s2k = regexprep( originalSAP_s2k, pattern3, Content3) ;
 originalSAP_s2k = regexprep( originalSAP_s2k, pattern4, Content4) ;
 originalSAP_s2k = regexprep( originalSAP_s2k, pattern5, Content5) ;
 originalSAP_s2k = regexprep( originalSAP_s2k, pattern6, Content6) ;
 originalSAP_s2k = regexprep( originalSAP_s2k, pattern7, Content7) ;
 originalSAP_s2k = regexprep( originalSAP_s2k, pattern8, Content8) ;

 % - Export updated content.
 fId = fopen( [ModelName,'.$2k'], 'w');
 fwrite( fId, originalSAP_s2k ) ;
 fclose( fId ) ;
```