JSEE Vol. 23, No. 3, 2021 DOI: 10.48303/jsee.2023.557988.1020

Research Paper

Semi-Supervised Support Vector Machines Algorithms as Classification Methods in Structural Health Monitoring

Hassan Fazeli[1], Mohammad Safi[2*] and Nemat Hassani[3]

1. Ph.D. Candidate in Earthquake Engineering at Civil Engineering College, Abbaspour Technical Campus, Shahid Beheshti University
2. Assistant Professor at Civil Engineering College, Abbaspour Technical Campus, Shahid Beheshti University, *Corresponding Author; email: m_safi@sbu.ac.ir
3. Associate Professor at Civil Engineering College, Abbaspour Technical Campus, Shahid Beheshti University

Received: 17/07/2022
Revised: 25/02/2023
Accepted: 15/03/2023

ABSTRACT

One of the fields in data-based structural health monitoring that has not been widely considered is data classification step. Application of semi-supervised methods in data classification is getting more attention nowadays. In this study, an efficient semi-supervised support vector machine algorithm is used for classifying between healthy and unhealthy stages. For this reason, a combined model-based and data-based approach is taken to determine damage sensitive features. A hybrid approach has been utilized to generate feature vectors. Using vibrational data of structure, dynamic properties are obtained by stochastic subspace state-space system identification methods. Modal strain energy used as damage sensitive features (DSF). Different states of healthy and unhealthy conditions of structure are used to evaluate the effectiveness of proposed algorithm. Also, Support Vector Machines (SVM) algorithm is utilized to compare results. It can be seen that the use of unlabeled data will enhance the effectiveness of classification methods especially in small-number of labeled data. When labeled dataset are large enough, results for both supervised and semi-supervised support vector machines are almost the same.

Keywords:
Statistical pattern recognition; Support vector machines method; Semi-supervised learning algorithms; System Identification

1. Introduction

Civil infrastructures are present in any society, regardless of culture, religion, geographical location and economic development. The safest and most durable structures are those that are well managed. Measurement and monitoring often play a key role in management activities. Structural health monitoring (SHM) is a process that aims to provide accurate and instant information on different conditions of structure, which leads to detecting defined type of damages. This type of data is valuable for client to make decisions about service condition of such structure. Among many monitoring strategies studied so far, vibration-based damage detection shows very efficient results toward civil infrastructure applications.

There has been much recent work in this area; in particular, Doebling et al. [1] and Sohn et al. [2] presented detailed reviews of vibration-based SHM. Because of random and systematic variability in experimentally measured dynamic response data, statistical approaches are necessary to ensure that changes in a structure's measured dynamic response are a result of damage and not caused by operational and environmental variability.

Although much of vibration-based SHM literature focuses on deterministic methods for identifying damage from changes in dynamic system response, we will focus on approaches that follow a statistical pattern recognition paradigm for SHM [3]. This paradigm consists of four steps:

1. Operational evaluation,
2. Data acquisition, (3)
3. Feature extraction, and
4. Statistical classification of features.

The work presented herein focuses on steps (4) of this paradigm.

Traditional damage detection algorithms developed in framework of pattern recognition. Although these algorithms are successful in accurately distinguishing faulty from healthy systems, they can rarely detect the location and severity of damage. In addition, in SHM applications in buildings and bridges, there is no real measured data from different stages of damage, so the pattern recognition approach is to use supervised learning method (the word unsupervised indicates the absence of measurement data on the damage stage in the structure at the time of training, unlike the word supervised that uses both healthy and damaged data for the training stage). Other challenges of data-based methods include selecting appropriate statistical model for damage sensitive features, and appropriate metric for calculating distance between newly extracted features from basic statistical model, hence proper criteria for distinguishing between healthy and unhealthy conditions.

There are important issues with using supervised learning. As an example, one could find that in training phase, there must be a dataset for each type of damage. There are two sources for providing such data, physical testing and mathematical modeling. Both of the mentioned sources have many practical problems, including modeling problems such as complexity of model, as well as test problems such as being expensive for all scenarios and being unusable. In contrast to supervised approach, there is another approach that does not require raw information from different damage classes to classify data, called unsupervised. Of course, this method is only capable of the first stage of damage detection (existence of damage) and in some cases the second stage (location of damage) from the Rytter's hierarchy [4] of damage detection [5-7]. This technique is often called Novelty Detection or Anomaly Detection [8].

Semi-Supervised approach is another method that can be very effective in classification when the number of labeled data is small. In the last few years, Semi-Supervised learning has attracted a lot of attention, and many approaches have been developed in this field of data classification [9-11]. In semi-supervised learning, a classification function is evaluated from a few labeled samples together with a large collection of unlabeled data. Different algorithms suggested to make use of unlabeled data in accordance with labeled ones to improve the classification process [9, 12]. Semi-supervised learning is based on regulating the target function with the marginal distribution. There have been two common assumptions on such distribution, the cluster assumption [13] and the manifold assumption [14].

The cluster assumption says that if there is a connection curve between two points through a high density region, there are probably on the same class label. Thus, the separation margin should be in the lower density region of the space. Implementing this theory results in developing Transductive Support Vector Machines (TSVM) [15] and its several applications, such as TSVM [16] or Semi-Supervised Support Vector Machines (S3VM) [17-18].

The manifold assumption states that the marginal probability distribution underlying the data is supported on or near a low-dimensional manifold, and that the target function should change smoothly along the tangent direction. Many graph based methods have been proposed in this direction [19-21].

In view of the above, it seems that large unlabeled dataset gathered from structures in their service states together with few labeled ones could be a useful source for a machine to perform an efficient classification in machine learning paradigm of structural health monitoring. In this regard, effectiveness of Semi-Supervised Support Vector Machines for classifying datasets related to different structural performance modes (healthy and damaged) has been compared in this research with supervised methods.

2. Materials and Methods

2.1. Semi-Supervised Support Vector Machine S3VM

This method is about the use of transduction technique for learning from small training samples instead of an inductive approach. In the inductive settings, the learner tries to induce a decision function which has a low error rate on the whole distribution of examples for the particular learning task. In many situations, one does not care about the particular decision function, but rather classify a given set of examples (a test set) with as few errors as possible. This is the goal of transductive inference. This method substantially improves the excellent performance of SVMs. Especially for very small training sets, this method reduces the required amount of labeled training data down to a twentieth for some tasks. This is the purpose of the transductive inference.

The setting of Semi-Supervised Support Vector Machine (S3VM) was introduced by Vapnik in [22] and implemented by [23, 16]. For a learning task $P(\vec{x} \mid y) = P(y \mid \vec{x})P(\vec{x})$, the learner L is given a hypothesis space H of functions $h: X \to \{-1.1\}$ and an independent and identically distributed random variable sample S_{train} (i.i.d) of n training examples.

$$(\vec{x}_1.y_1).(\vec{x}_2.y_2).....(\vec{x}_n.y_n) \tag{1}$$

Each training example consists of a document vector $\vec{x} \in X$ and a binary label $y \in \{-1.1\}$. In contrast to the inductive setting, the learner is also given an i.i.d. sample S_{test} of k test examples.

$$\vec{x}_1^*.\vec{x}_2^*.....\vec{x}_k^* \tag{2}$$

The transductive learner L aims to select a function $h_L = L(S_{train}.S_{test})$ from H using S_{train} and S_{test} so that the expected number of erroneous predictions on test examples is minimized. In linearly non-separable cases, the optimization problem of S3VM is shown as follows:

$$\frac{1}{2}\vec{w}^2 + C\sum_{i=0}^{n} L\left(y_i\left[\vec{w}\cdot\vec{x}_i + b\right]\right) + C^*\sum_{i=n+1}^{m} L\left(y_i^*\left[\vec{w}\cdot\vec{x}_j^* + b\right]\right) \tag{3}$$

With $L(t) = \max(0;1-t)$.

Unfortunately, the last term makes this problem non-convex and difficult to solve [16, 23-28]. The implementation of S3VM that is used in this paper is to perform a standard gradient descent on Equation (3). However, since the latter is not differentiable, we replace it by Equation (4):

$$\frac{1}{m}\sum_{i=n+1}^{m}\vec{w}\cdot\vec{x}_i + b = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{4}$$

with $L^* = \exp(-3t^2)$.

To enforce that all unlabeled data are not put in the same class, an additional constraint is added,

$$\frac{1}{m}\sum_{i=n+1}^{m}\vec{w}\cdot\vec{x}_i + b = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{5}$$

This is in analogy to treatment of the min-cut problem in spectral clustering, which is usually replaced by the normalized cut to enforce balanced solutions [29].

Finally, note that unlike traditional SVM learning algorithms, which solve the problem in the dual, this approach directly solves the problem in the primal. If we want to use a non-linear kernel, it is possible to compute coordinates of each point in the Kernel Principal Component Analysis (KPCA) basis [30]. More directly, one can compute Cholesky decomposition of Gram matrix, $K = \tilde{X}\tilde{X}^T$ and minimize (2) with $x_i \equiv (\tilde{X}_{i.1}.....\tilde{X}_{i.n+m})$.

In [25], another way to enforce the cluster assumption in SVM classification are proposed called Low Density Separation (LDS) as a graph-based algorithm. In many graph-based semi-supervised algorithms, cluster assumption is assigned to the algorithm by smoothening the solution with respect to the graph, resulting in small variation of output function between connected nodes.

Let the graph $G = (V, E)$ be determined from data such that nodes are data points $V = \{x_i\}$. If sparsity is desired, edges are placed between nodes that are nearest neighbors (*NN*), either thresholding the degree (*k-NN*) or the distance ($\epsilon - NN$). Many semi-supervised learning methods operate on nearest neighbor graphs, [32-36]. Usually they do not require data points themselves, but only their pairwise distances along edges. In [25], it is assumed that edges $(i, j) \in E$ are weighted by Euclidean distances $d(i, j) = \| x_i - x_j \|$.

In such a case, pairwise similarities of two points are determined by graphs, which leads to squeezing distances in high density regions while leaving them in low density regions. This idea has been proposed before, e.g. in [37-38] and [39]. It has been implemented and used in Isomap [40], cluster kernels [16], and connectivity clustering [41].

One of the main conditions of cluster assumption is that decision boundary should not cut clusters. For similarity-based classifiers, this achieved by applying low similarities to pairs of points in different clusters. Parzen window density that estimates with Gaussian kernel of width V, E will use to determine mentioned constraints, as follows:

$$\hat{p}(x') = \frac{1}{\sqrt{\pi}\sigma} \sum_{i=1}^{n+m} \exp\left(-\frac{\| x' - x_i^{\,2} \|}{\sigma^2} \right) \tag{6}$$

If $p \in V^l$ is defined as a path on a graph $G = (V, E)$ with length of $l = |p|$ and if $(p_k, p_{k+1}) \in E$ for $1 \le k < |p|$, this path connect nodes p_1 and $p_{|p|}$. Let $P_{i,j}$ denotes a set of all paths connecting x_i and x_j. The similarity of two points can be defined by:

$$\begin{aligned} &\max_{p \in P_{i,j}} \min_{k<|p|} \hat{p}\left(\frac{1}{2}\left(x_{p_k} + x_{p_{k+1}} \right) \right) \\ &\approx c.\exp\left(-\frac{1}{2\sigma^2} \left(\max_{p \in P_{i,j}} \min_{k<|p|} d\left(p_k + p_{k+1} \right) \right) \right) \\ &\equiv k\left(x_i, x_j \right) \end{aligned} \tag{7}$$

This k, called "connectivity kernel", is positive definite and was suggested for clustering previously [41].

Kernel values do not depend on the length of paths, which may lead to connection of otherwise separated clusters by single outliers ("bridge" points). To avoid this problem, max Equation (7) is "softened" replacing it with

$$smax^{\rho} = \frac{1}{\rho} \ln\left(1 + \sum_{k=1}^{|p|-1} \left(e^{\rho d(p_k + p_{k+1})} - 1 \right) \right) \tag{8}$$

Equation (7) is recovered by taking $\rho \to \infty$.

The proposed method can be summarized as follows:

1. Build nearest neighbor graph G from all (labeled and unlabeled) data.
2. Compute the $n \times (n+m)$ distance matrix D^{ρ} of minimal ρ-path distances according to: $D_{i,j}^{\rho} = \frac{1}{\rho^2} \ln\left(1 + \min_{p \in P_{i,j}} \sum_{k=1}^{|p|-1} \left(e^{\rho d(p_k + p_{k+1})} - 1 \right) \right)^2$ from all labeled points to all points.
3. Perform a non-linear transformation on D^{ρ} to get kernel K, $K_{i,j} = \exp\left(-\frac{D_{i,j}^{\rho}}{2\sigma^2} \right)$

   The linear case corresponds to $\sigma = \infty$ and $K = -\frac{1}{2} H^n D^{\rho} H^{n+m}$, with H^p being the $p \times p$ centering matrix: $H_{i,j}^{p} = 1_{i=j} - 1_{i \le n} / n$
4. Train an SVM with K and predict.

Final LDS algorithm is summarized in [25]. A MATLAB implementation of LDS can be obtained at http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/lds/.

2.2. Evaluation of Damage Using Numerical Experiment

In this section, a study on application of semi-supervised algorithms presented in the previous section is reviewed on a numerical example. A numerical example description is provided next. A simple model of a lumped mass bridge deck has been used to illustrate various stages in the proposed damage assessment, as well as to test its performance. This model is based on numerical experiment in [31].

This model consists of two interconnected spring chains, each of which consists of flexural springs and concentrated masses placed uniformly. This simple model can be used to display simple pedestrian bridges or a simple span of girder bridges: where flexural springs represent different segments of the girder and the lumped masses represent the mass of deck. Due to the structure of the model, its dynamic behavior includes global bending and global torsional modes of the deck.

The specific model used in this research is shown in Figure (1): which has 12 degrees of freedom (DOF) with 12 lumped masses and 20 flexural springs. 12 vibration modes of this model (in the baseline state) includes 6 flexural modes (predominant) and 6 torsional modes (predominant). The predominant word is used to describe the state of the mode, because only if the structure is

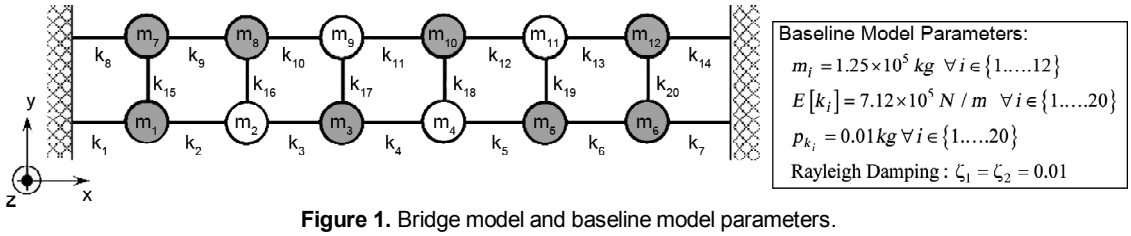

Figure 1. Bridge model and baseline model parameters.

completely different in terms of mass and stiffness distribution, modes become purely flexural and torsional, otherwise modes become flexural-torsional where the flexural or torsional part is predominant. Summary of modal frequency and dominance of modes are presented in Table (1).

Table (2) presents 10 different damage states considered in this study: states 1 to 5 represent the healthy system, states 6 to 9 represent four different damage states with different damage intensities and in different locations. State 10 is a state where a structural reinforcement is done in a specific part of the girder, which leads to an increase in stiffness in one of the springs.

Table 1. Different modes of bridge deck and corresponding frequency.

Mode Number	Type of Mode	Frequency (Hz)
Mode 1	Bending Dominant	1.69
Mode 2	Bending Dominant	3.30
Mode 3	Bending Dominant	4.74
Mode 4	Torsion Dominant	5.63
Mode 5	Bending Dominant	5.94
Mode 6	Torsion Dominant	6.30
Mode 7	Bending Dominant	6.85
Mode 8	Torsion Dominant	7.16
Mode 9	Bending Dominant	7.41
Mode 10	Torsion Dominant	8.01
Mode 11	Torsion Dominant	8.70
Mode 12	Torsion Dominant	9.15

Table 2. Healthy and unhealthy states of structure.

State	Condition	Description	Affected DOFs
1	Undamaged	Baseline Condition	
2	Undamaged	$k_i = 0.99E[k_i]\ \forall i \in \{1.....7\}$	
3	Undamaged	$k_i = 1.01E[k_i]\ \forall i \in \{1.....7\}$	
4	Undamaged	$k_i = 0.99E[k_i]\ \forall i \in \{8.....14\}$	
5	Undamaged	$k_i = 1.01E[k_i]\ \forall i \in \{8.....14\}$	
6	Damaged	$k_1 = 0.70\ E[k_1]$	1
7	Damaged	$k_{16} = 0.80\ E[k_{16}]$	2 and 8
8	Damaged	$k_3 = 0.80\ E[k_3]$	2 and 3
9	Damaged	$k_3 = 0.70\ E[k_3]$	2 and 3
10	Retrofitted	$k_3 = 1.25\ E[k_3]$	2 and 3

To use these data in the classification phase, 30 experiments were performed on each state. As mentioned above, five healthy states are defined, including the state 1 as baseline state, and states 2 and 3 as states in which ambient conditions in -y side of deck change dynamic properties of structure due to a decrease or increase in temperature. Similarly, in states 4 and 5, the structure is affected by ambient conditions in +y side: In each experiment, parameters defining the model are randomly disturbed. For each spring stiffness? k_i, its stiffness value in rth simulation is obtained from Equation (9), as follows:

$$k_i^r = E[k_i] + \mathcal{U}(-p_k.p_k)E[k_i] \tag{9}$$

where $E[k_i]$ is mean value of spring stiffness in that state and is determined in Table (2). Parameter $\mathcal{U}(l_l.l_r)$ is a uniform probability distribution between $(l_l.l_r)$. The value of p_k is presented in Table (2). While change in stiffness parameters presented in Table (2) denotes a systematic change, induced by temperature, damage, etc., and is constant in all experiments in each state, random turbulence indicates statistical model and fluctuations of performance in each state, so they have different values in each state in each of 30 different experiments.

The disturbed model in each experiment is excited by a Gaussian white noise as input force. The resulting acceleration response is corrupted by adding Gaussian white noise sequences with an average of 10% of the quadratic root mean, to simulate measured noise. Instrument arrangement for interpreting acceleration response of structure is defined on the basis of complete instrumentation, i.e. 12 sensors measure acceleration responses of all concentrated masses.

To prepare the required dataset, a code was developed in MATLAB software, which uses SAP2000 software interface called SAP API to analyze the model. SAP2000 software was used

to obtain model acceleration response. Here's how it works: In the written code, first using the SAP API, a mathematical model is created in SAP2000 based on the arrangement defined in Figure (1) and parameters presented in Table (2), by modeling springs, lumped masses and support conditions. Next, stiffness values of each spring are randomly disturbed based on the range specified in Table (2) and initial mean stiffness value presented in each state, and then are assigned to corresponding modeled springs. Then, Gaussian white noise is added to the model as input force in the center of the deck. White noise is defined with a mean value of zero and a Gaussian distribution for 120 seconds with a time-step of 0.001 and its sample frequency rate is 1/120000. The loaded model is analyzed by linear time history analysis and results of acceleration history of each lumped mass are recorded. As mentioned before, to consider measurement noise of sensors, acceleration history of each lumped mass is again perturbed by a Gaussian white noise with an RMS of 10% and stored in a file. This code prepared in MATLAB generates each state of the model in each experiment in SAP2000 software and stores output data.

Sets of acceleration response measured in each experiment are used to identify dynamic properties of the structure using Subspace State-Space System Identification (4SID) method. Since the introduction of stochastic subspace identification (SSI) by Peeters and Roeck [42], subspace methods have become popular system identi?cation tools used by civil and mechanical engineering communities. The SSI algorithm is just one member of the general subspace state-space system identification (4SID) family [43] and is recognized as a significant achievement of theoretical dynamics and control communities [44]. Since 4SID was introduced to the civil engineering community as a set of output-only system identification methods, a rigorous mathematical mapping of state-space model parameters to physical parameters of the system has yet to be undertaken. The lack of a mathematical mapping has limited 4SID as a purely black-box data-driven tool whose results are dif?cult to interpret by engineers.

Methods associated with 4SID are generally categorized into two groups: realization-based and direct 4SID methods [43]. Realization-based 4SID methods and their origins in the seminal work of Ho and Kalman [45] and offer a means of extracting state-space models from the extended observability matrix. At the core of the realization-based 4SID methods is the need for a reliable estimate of system impulse responses, often termed Markov parameters (MPs); the extended observability matrix is estimated directly from MPs. In contrast, direct 4SID methods, also referred to as data-driven subspace identification in the civil engineering community [46], strive to estimate a state-space model directly from an arbitrary set of input and output sequences (i.e., without requiring the estimation of system impulse response functions). Extensive research in the 1970s and 1980s (e.g., stochastic realization [47]) led to the establishment of numerous direct 4SID numerical algorithms such as multivariable output-error state-space (MOESP) [48] and numerical algorithms for subspace state-space system identification (N4SID) methods [49]. Generally, direct 4SID methods are simply referred to as subspace methods or subspace system identifications. A more detailed description of N4SID algorithm can be found from works of Van Overschee and De Moor [49-50].

MATLAB software were used to apply Subspace State-Space System Identification method. A convenience of using N4SID is the availability of a MATLAB function (i.e., n4sid) for its execution [51]. Stored data of the previous section is used to identify dynamic model of system. Modal parameters are then extracted from the identified model. Modal parameter data are then used in combination with spatial information of sensors (concentrated masses) to obtain shape modes of structure.

In this study, measured data in time domain are first converted into frequency response data. This Frequency Response Function (FRF) is then used to identify the state-space model of the existing bridge deck. In MATLAB, it is possible to use time domain data to identify dynamic models, but the use of FRF data compresses very large datasets to smaller samples. This can also be used to set

the model behavior estimate in relevant frequency ranges.

The state-space model is set as follows:

- A 48-order continuous time model is estimated.
- The model includes a feedthrough term. Because once we set our analysis to frequencies less than 11 Hz, but the bandwidth of the deck is more than this value.
- In this study, covariance parameters were not calculated to speed up calculations.

FRF value varies widely along the frequency range. A specific weighting is used to ensure that low values are emphasized equally with high values. Using the Normalized Root - Mean - Square - Error NRMSE, the percentage of consistency between data and the model can be observed.

Using the iterative nonlinear least-squares refinement of the model, consistency between data and the model can be improved. The stability of modal parameters should be controlled by changing the order of the model. The order 48 used in these calculations has shown high stability in frequency and damping. The stability diagram generated for the baseline experiment is shown in Figure (2). In the control theory, the model order is calculated using residual error analysis on the predicted model. In the civil engineering, frequency and damping stabilization diagrams are used to calculate the model order.

After preparing the model, modal parameters, including natural frequencies (fn), damping coefficients corresponding to modes (dr) are extracted. Mode shape after extraction from the model is used to calculate the modal strain energy as a feature vector used in the classification section. By considering the mathematical model of the deck, it is possible to calculate the relative stored energy in each mode and through calculating relative deformations of each spring (using $E = \frac{1}{2}k(\Delta x)^2$).

After calculating modal strain energy for each mode in each experiment, a dataset with dimensions of 10×30 (which contains 30 of 12×1 feature vector) is yielded for all states defined in Table (2). This dataset will be used to compare semi-supervised method presented in previous chapter.

The information about applied dataset is presented in Table (3).

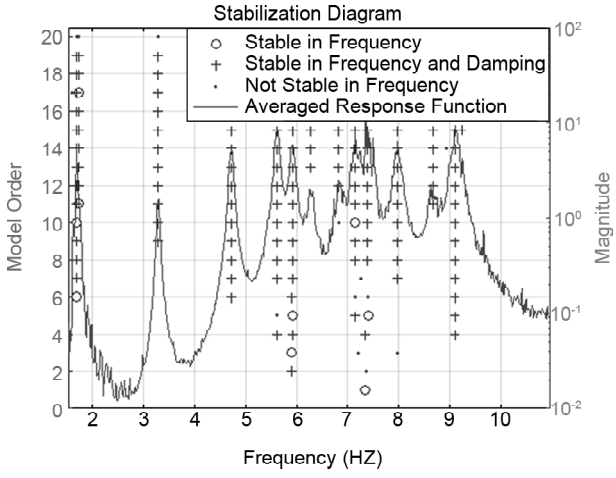

Figure 2. Frequency stabilization diagram for the experiment of state 1.

Table 3. Information on dataset.

Dataset	Classes	Dimensions	Points	Labeled
SHM1	6	12	300	60

2.3. Experiments

As mentioned earlier, in this study, a comparison between a semi-supervised algorithm and support vector machines are presented. For SVM, we use the Spider machine learning package for MATLAB [13].

For each algorithm introduced before, the classification procedure was performed for five conditions regarding the number of labeled data. The classification was carried out for 2-labeled data for each class, then this was repeated for 4-labled, 6-labeled, 10-labeled and 15-labeled data representing 6.7%, 13.3%, 20%, 33.3% and 50% of data being labeled in each class. Therefore, labeled data selected for training sets are chosen by 2 to 15-fold cross-validation process.

For each algorithm, values for a number of parameters have to be fixed. In practical applications, this is usually done by cross-validation (CV). While this is no major problem for two parameters (like SVMs have), it is impractical for five parameters of the LDS algorithm. To reduce this number, we fix three of them in advance, as shown in Table (4).

Table 4. Free and fixed parameters.

Algorithm	Free Parameters	Fixed Parameters
SVM	σ, C	
LDS	σ, ρ	$\sigma = \infty$, k = n + m, $\delta = 0.1$

We are interested in the best possible performance, and simply select parameter values minimizing the test error. As presented in Table (5), we select combinations of values on a finite grid as follows:

Table 5. Parameters.

Parameter	Value
Width, σ	2^{-3}, 2^{-2}, 2^{-1}, 2^{0}, 2^{1}, 2^{2}, 2^{3}
Exponent, ρ	0, 2^{0}, 2^{1}, 2^{2}, 2^{3}, 2^{4}, $+\infty$
Penalty, C	10^{-1}, 10^{0}, 10^{1}, 10^{2}
Degree, k	10, 100, all
Regularize, λ	4^{-2}, 4^{-1}, 4^{0}, 4^{1}, 4^{2}

3. Results

One of the issues to be considered in this section is that SVM-based algorithms have the ability to separate between two data classes, i.e. they are in the form of a dichotomy, but data separation should be performed between six data classes in this study. The usual way to use SVMs in multi-class problems is to compare one class with all other data classes, also called One-vs-rest. Another way is to use SVMs to compare two classes of data, in which all classes must be compared in pairs. In this research, the second method has been used, which leads us to confusion matrixes.

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix [10] is a specific Table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents instances in an actual class while each column represents instances in a predicted class, or vice versa - both variants are found in literature [11]. The name stems from the fact that it makes it easy to see whether the system is confusing two classes (i.e. commonly mislabeling one as another).

As described in previous sections, the dataset used in this section was prepared and used based on the classification algorithms introduced for the various states defined in Table (1). To compare algorithms, three acceptable criteria in the field of classification have been used. These criteria are Accuracy, Precision and Recall, which are suitable for displaying type I and II errors of classification algorithms. Statistical hypothesis testing defines two types of errors: Type I error, also called false positive, is equal to the rejection of positive samples, a simple example of this error could be a doctor diagnosing a disease in a patient while disease does not exist in the patient. In contrast to type I error, type II error is defined, which is also called false negative, indicating that negative samples are not rejected, such as when a doctor does not diagnose a disease that exists in a patient. Therefore, the Precision criterion is equal to the ratio of the number of positive samples to the number of positive samples provided by the algorithm. The high value of this criterion indicates a slight error rate of the first type. Recall criterion is the ratio of the number of correct positive samples provided by the algorithm to the actual number of positive samples in the total data in the dataset.

As stated before, data have classified in 10 conditional states. First five states are representing healthy states. Thus, in this study results are arranged in a way that states 1 to 5 are combined to a unique healthy state. Consequently, results will be divided between six dissimilar states. Confusion matrixes are presented in Tables (6) and (7).

Corresponding values of accuracy, precision and recall for confusion matrices shown above are displayed in Table (8). It should be noted that these tables are just for 6-L test procedure.

Again, the value of accuracy for each algorithm

Table 6. Confusion matrix for testing data - LDS algorithm - 6-Labeled data.

Predicted	New-Class 1 Healthy State	Class_6	Class_7	Class_8	Class_9	Class_10
True New-Class 1	105	3	3	0	0	9
True Class 6	4	17	1	0	0	2
True Class 7	1	4	17	0	0	2
True Class 8	0	0	0	18	5	1
True Class 9	0	1	0	6	17	0
True Class 10	0	0	0	0	0	24

are calculated for each test (regarding the number of labeled data those tests are called 2L, 4L, 6L, 10L and 15L for test with 2, 4, 6, 10 and 15 labeled data in the training set, respectively), and are shown in Figure (3). In Figure (4) and Figure (5), the mean value for Recall and Precision for each state are again displayed for each classification test, respectively.

Table 7. Confusion matrix for testing data - SVM algorithm - 6-Labeled data.

Predicted	New-Class 1 Healthy State	Class_6	Class_7	Class_8	Class_9	Class_10
True New-Class 1	119	0	0	0	1	0
True Class 6	2	20	0	0	2	0
True Class 7	4	1	17	0	2	0
True Class 8	5	0	0	15	4	0
True Class 9	2	0	0	4	18	0
True Class 10	1	0	0	0	0	23

Table 8. Accuracy, Precision and Recall for Confusion Matrices presented before.

				Predicted class	New-Class_1	Class_6	Class_7	Class_8	Class_9	Class_10
LDS	6L	Accuracy	0.83	Precision	0.95	0.68	0.81	0.75	0.77	0.63
				Recall	0.88	0.71	0.71	0.75	0.71	1.00
SVM	6L	Accuracy	0.88	Precision	0.89	0.95	1.00	0.79	0.67	1.00
				Recall	0.99	0.83	0.71	0.63	0.75	0.96

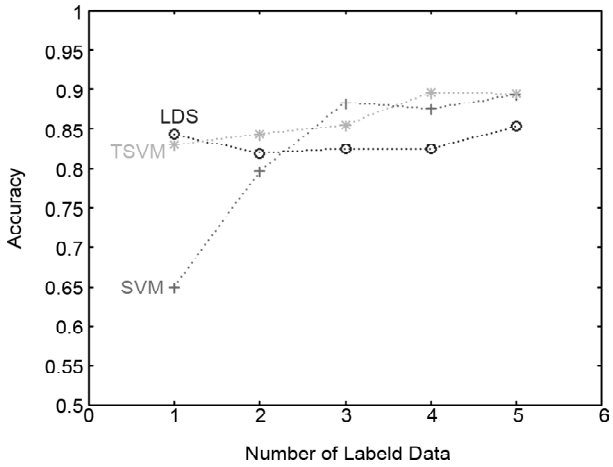

Figure 3. Value of Accuracy for each algorithm in each test.

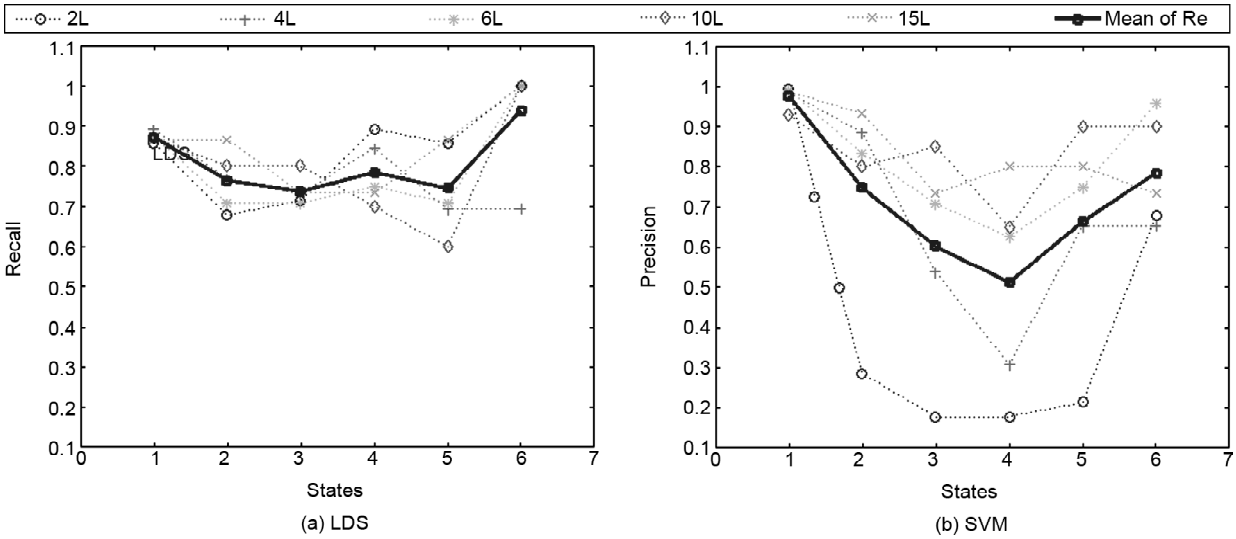

Figure 4. Values of Recall in each test.

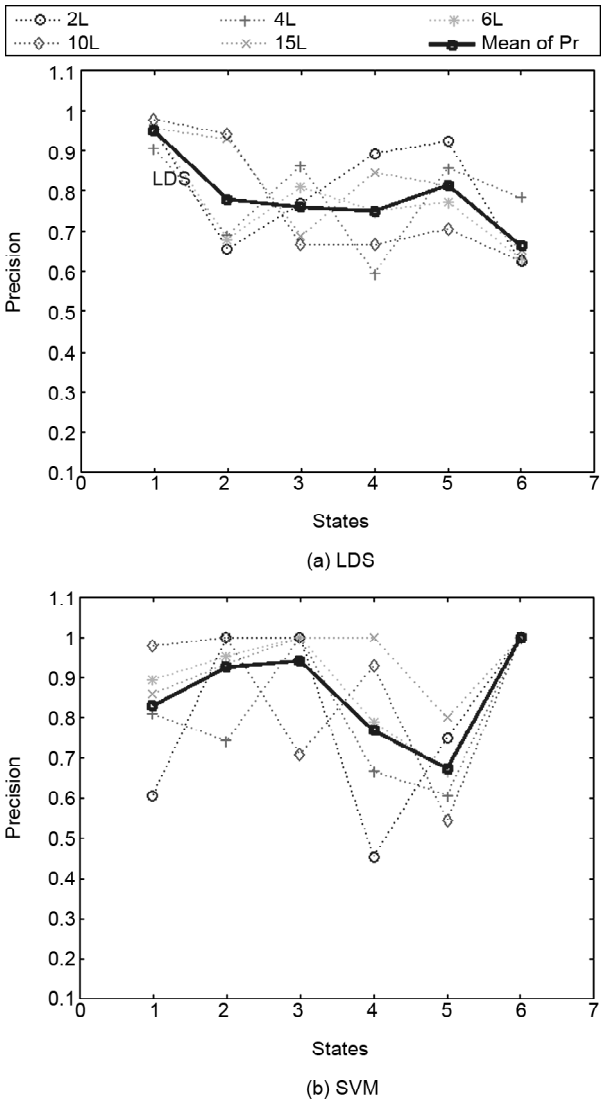

Figure 5. Values of Precision in each test.

4. Discussion

We now investigate details of classification problem by LDS algorithm. Since overall results of the accuracy of these methods are not comparable to verified results of mentioned algorithms, we decided to consider five states of healthy structure being a unique healthy state. For this matter, confusion matrices should be corrected by merging results of classes 1 to 5 in on new class. In order to compare effects of the number of labeled data, classification is performed for different levels of labeled data that is available in the training set. The focus of discussion will be on performance of mentioned algorithms considering different labeled data number.

In Tables (6) and (7), the confusion matrix calculated for the test with 6-labeled test are displayed for LDS and SVM algorithms, respectively. These tables were determined as a basic formats of results of this study, and other results were calculated from these matrices. In Table (8), corresponding values of three criteria were displayed. Those criteria were visualized for whole data. The discussion will be developed in three parts for accuracy, recall and precision criteria. The overall accuracy for SVM is approximately good, but it even touches very small values in 2-L and 4-L tests. As the number of labeled data increases, accuracy also increases, but for 15-L test (50% labeled) accuracy descends a little. LDS algorithm shows higher overall accuracy in 2-L test, but in other tests it lies below SVM (in 6-L and higher). In 2-L, accuracy of LDS is near 85%, that clearly shows the better functionality of semi-supervised algorithms in lack of labeled data. In 4-L, SVM reaches to the near of other methods, and in 6-L a higher, the overall accuracy of SVM get higher than LDS. In 4-L, SVM and LDS are showing approximately similar values, with the accuracy almost 78% and 85%, respectively, which is almost comparable by results of [25] for dataset USPST and COIL20.

In Figure (4), recall metric is displayed for mentioned algorithms in three different plots. Each plot contains five different diagrams of recall values for each state for five specific training conditions, and also mean value of all five mentioned conditions. For all states, results will be discussed in two parts, first the mean value of recall will be studied and then recall for each state will be compared to each other with regard to number of labeled data. The pattern of mean recall values for six new classes are almost similar in all three techniques, noting that mean-value for new class 1 is around 1, declaring a very high accuracy in this healthy state (which is the combination of first five health state). This result implies that algorithms are very delicate to healthy state, maybe because of well-managed feature vector. Considering the mean value of recall, two semi-supervised techniques are showing better results than SVM. Specifically speaking about the pattern of results, it can be seen that results for states 6, 9 and 10 are a bit higher in SVM. Best results for the mean value for LDS method is attained from state 10 (retrofitted model), with value about 90%. In SVM, the highest recall

value for mean value is around 80%, but it is for classes 10 and 9, respectively. For class 1 (healthy state), SVM result in higher mean recall value than LDS. For class 6 (in which the near-support element's stiffness is reduced to 70% of its initial value), LDS results are the highest (approximately 80%), and SVM is the lowest (approximately 70%), but the difference is not considerable. Each algorithm does have higher mean recall value for state 6 than 7, but that pattern for state 8 and 9 is not similar for each technique. For SVM, recall value for state 8 is less than 9, which is logical since state 9 presents a more severe damage than state 8 in a similar position. In LDS, mean recall value for states 8 and 9 are relatively close, in a way that for state 8 is a little higher.

Now, state-by-state comparison of recall values for different number of labeled data is presented. Almost every test shows a great recall value for state 1 (healthy state). In 2-L test, semi-supervised algorithms show much better results than SVM, implying that the efficiency of semi-supervised algorithms in low labeled tests are appropriate. For example, recall values for 2-L test in state 6 for LDS and SVM are respectively 70%, 70% and 30%. However, between LDS and SVM, the former generates better results in 2-L test. This outcome is followed by 4-L test, except that SVM shows a great result in state 6 but in other states its results are similar to 2-L tests. Again, in 4-L test, LDS shows better results than SVM. For 6-L and others, recall value is in a similar range of [70%, 90%], that indicates with more labeled data, the efficiency of semi-supervised and supervised algorithms is similar. Generally speaking, almost in every test, recall value in state 9 is higher than state 8 for SVM, which is rational because the reduction in stiffness is greater in state 9, but in LDS its reverse, which is a negative point. SVM has low recall value for states 6-9 in 2-L (about 20% to 30%), but for state 10 its outcome is desirable (about 70%), and it maintains the pattern of mean-value results. In class 8, SVM does not show good recall value, and LDS's outcome for class 9 does not fit the pattern of mean-value. From these diagrams it can be mentioned that 6-L is a minimum number of labeled data SVM needs to extract acceptable results, but for LDS 2-L is sufficient.

The precision criteria for each algorithm, which are calculated from confusion matrices and are presented in Figure (5), are to be studied next. Then again, we consider mean-value for comparing the pattern of results and review each labeled test condition to rate performance of methods. The pattern of mean value is approximately similar for SVM, but it is different for LDS. The main difference is state 9 in LDS has greater precision than state 8, but in SVM it is reversed. This criterion has a greater mean value in state 1 for LDS than others, but in state 6 and 7 LDS gets lower values than other two methods. State 10 has a very good mean precision value in SVM, but it is not proper in LDS. The pattern for SVM methods is somehow similar, being high for class 6 and 10, class 7 is higher than 8 and 9, but in LDS the pattern is different and has higher value in state 9 and lower in state 10. The precision mean-value for class 9 is smaller than class 8 for SVM, and it is reversed in LDS. Back to confusion matrices, this can be explained by considering the greater number of predicted data for class 9, that it actually belongs to class 8. This brings the idea that LDS method is sensitive enough to small changes in material properties and also for SVM maybe another feature vector can be used to overcome these issues. It also shows the greater sensitivity for LDS in damage locations not close to supports. For class 7 to 9, the results of SVM are better match to actual results and are better than LDS. Now studying the class by class for various labeling conditions, for 2-L and 4-L tests, all techniques are showing great precision in class 6. The value of this metric in class 7 for 2-L for SVM is a better than LDS (being about 1 to 90% and 80%, respectively), but in class 8 and 9 the LDS has the highest value.

5. Conclusion

In this paper, we try to investigate the efficiency of the semi-supervised algorithm to classify between the healthy and damaged states of structures. In the pattern recognition paradigm, the extraction of the feature vectors is executed by determination of the dynamic properties of the

structure in a model-based approach utilizing subspace state-space system identification procedure, while the classification process is accomplished by the LDS Algorithms semi-supervised techniques. To compare the practicality of the classification algorithm, we use a well-known supervised algorithm (SVM). The LDS show suitable results for low-labeled-data tests, but as the number of labeled data gets higher, the efficiency of the SVM reaches the other algorithm.

References

1. Doebling, S., Farrar, C., Prime, M., and Shevitz, D. (1998) A review of damage Identification methods that examine changes in dynamic properties. *Shock Vib. Dig.*, **30**, 91-105.
2. Sohn, H., Farrar, C.R., Hemez, F.M., Shunk, D.S., Stinemates, D.W., Nadler, B.R., and Czarnecki, J.J. (2004) *A Review of Structural Health Monitoring Literature From 1996-2001*. Los Alamos National Laboratory, Report No. LA-13976-MS.
3. Farrar, C.R. and Worden, K. (2007) *An Introduction to Structural Health Monitoring*. Philos. Trans. R. Soc. London, Ser. A, **365**, 303-315.
4. Rytter, A. (1993) *Vibration Based Inspection of Civil Engineering Structures*. Building Technology and Structural Engineering, Aalborg University, Aalborg.
5. Bishop, C.M. (1994) Novelty detection and neural network validation. *IEEE Proceedings - Vision and Image Signal Processing*, **141**, 217-222.
6. Markou, M. and Singh, S. (2003a) Novelty detection - a review. Part I: statistical approaches. *Signal Processing*, **83**, 2481-2497.
7. Markou, M. and Singh, S. (2003b) Novelty detection - a review. Part II: neural network based approaches. *Signal Processing*, **83**, 2499-2521.
8. Worden, K. (1997) Structural fault detection using a novelty measure. *Journal of Sound and Vibration*, **201**, 85-101.
9. Chapelle, O., Scolkopf, B., and Zien, A. (2006) *Semi-Supervised Learning*. MIT Press, Cambridge, MA.
10. Zhou, Z.-H. and Li., M. (2010) Semi-supervised learning by disagreement. *Knowl. Inf. Syst.*, **24**(3), 415-439.
11. Zhu., X. (2006) *Semi-Supervised Learning Literature Survey*. Technical Report 1530, Dept. Comp. Sci., Univ. Wisconsin-Madison.
12. Zhu, X., Goldberg, A.B. (2009) Introduction to semi-supervised learning. *Synth Lect Artif Intell Mach Learn*, **3**(1), 1-130.
13. Chapelle, O., Weston, J., and Scholkopf, B. (2002) Cluster kernels for semi-supervised learning. In: *Proceedings of 16th Annual Neural Information Processing Systems Conference*, 585-592.
14. Belkin, M., Niyogi, P., and Sindhwani, V. (2006) Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, **7**, 2399-2434.
15. Chapelle, O., Vapnik, V., and Weston, J. (1999) *Transductive Inference for Estimating Values of Functions*. NIPS, 421-427.
16. Joachims, T. (1999) Transductive inference for text classification using support vector machines. In: *Proceedings of the Sixteenth International Conference*, **99**, 200-209.
17. Bennett, K. and Demiriz, A. (1999) Semi-supervised support vector machines. In: *Advances in Neural Information Processing Systems*, **11**, 368-374.
18. Chapelle, O., Sindhwani, V., and Keerthi, S.S. (2008) Optimization techniques for semi-supervised support vector machines. *J. Mach. Learn. Res.*, **9**, 203-233.
19. Joachims, T. (2003) Transductive Learning via Spectral Graph Partitioning. In ICML.
20. Belkin, M., Niyogi, P., and Sindhwani, V. (2006) Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, **7**, 2399-2434.

21. Zhu, X. (2005) *Semi-Supervised Learning with Graphs*. Carnegie Mellon University, language technologies institute, school of computer science.

22. Vapnik, V. (1998) *Statistical Learning Theory*. Wiley.

23. Kristin, P., Bennett, Ayhan Demiriz (1999) Semi-supervised support vector machines. In: *Proceedings of the Conference on Advances in Neural Information Processing Systems II*, Cambridge, MA, USA, 68-374.

24. Bie, T.D. and Cristianini, N. (2004) Convex methods for transduction. *Advances in Neural Information Proceeding Systems*, **16**, MIT Press, p. 73.

25. Chapelle, O. and Zien, A. (2005) Semi-supervised classi?cation by low density separation. *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, Barbados, 57-64.

26. Chapelle, O., Sindhwani, V., and Keerthi, S. (2006) Branch and bound for semi-supervised support vector machines. *Advances in Neural Information Processing Systems*, MIT Press, Cambridge, **17**, 217-224.

27. Choi, H., Kim, J., and Kim, Y. (2010) A sparse large margin semi-supervised learning method. *J. Korean Stat. Soc.*, **39**(4), 479p.

28. Collobert, R., Sinz, F., Weston, J., and Bottou, L. (2006) Large scale transductive SVMs. *J. Mach. Learn.*, **7**, 1687-1712.

29. Gieseke, F., Airola, A., Pahikkala, T., Kramer, and Sparse, O. (2012) Quasi-Newton optimization for semi-supervised support vector machines. *Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, 45-54.

30. Hermes, L. and Buhmann, J.M. (2000) Feature selection for support vector machines. *Proceedings of 15th International Conference on Pattern Recognition*, **2**, 712-715.

31. Betti, R. (2014) *Combining Model Based and Data Based Techniques in a Robust Bridge Health Monitoring Algorithm*. Rutgers University. Center for Advanced Infrastructure and Transportation, CAIT-UTC-015.

32. Belkin, M., Matveeva, I., and Niyogi, P. (2004) *Regularization and Semi-Supervised Learning on Large Graphs*. In COLT.

33. Kondor, R.I. and Lafferty, J. (2002) *Diffusion Kernels on Graphs and Other Discrete Structures*. In ICML.

34. Szummer, M. and Jaakkola (2001) *Partially Labeled Classification with Markov Random Walks*. In NIPS, 14.

35. Zhu, X., Ghahramani, Z., and Lafferty, J. (2003) *Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions*. In ICML.

36. Zhou, D., Bousquet, O., Lal, T., Weston, J., and SchAolkopf, B. (2003) *Learning with Local and Global Consistency*. In NIPS, 16.

37. Chapelle, O. (2003) *Support Vector Machines: Induction Principle, Adaptive Tuning and Prior Knowledge*. Ph.D. Thesis, LIP 6.

38. Vincent, P. and Bengio, Y. (2003) *Density-Sensitive Metrics and Kernels*. Presented at the Snowbird Learning Workshop.

39. Bousquet, O., Chapelle, O., and Hein, M. (2004) *Measure Based Regularization*. In NIPS.

40. Tenenbaum, J.B., de Silva V., and Langford, J.C. (2000) A global geometric framework for nonlinear dimensionality reduction. *Science*, **290**(5500), 2319-2323.

41. Fischer, B., Roth, V., and Buhmann, J.M. (2004) *Clustering with the Connectivity Kernel*. In NIPS, **16**.

42. Peeters, B. and Roeck, G.D. (1999) Reference-Based Stochastic Subspace Identification for Output-Only Modal Analysis. *Mechanical Systems and Signal Processing*, **13**(6), 855-878.

43. Viberg, M. (1995) Subspace-based methods for the identification of linear time-invariant systems. *Automatica*, **31**(12), 1835-1851.

44. Gevers, M. (2003) A personal view on the

development of system identification. In: *13th Federation of Automatic Control Symposium on System Identification (IFAC SYSID)*, Rotterdam, Netherlands.

45. Ho, B.L. and Kalman, R.E. (1965) Effective construction of linear state-variable models from input-output functions. *Regelungtechnik*, **12**, 545-548.

46. Peeters, B. and Ventura, C.E. (2003) Comparative study of modal analysis techniques for bridge dynamic characteristics. *Mechanical Systems and Signal Processing*, **17**(5), 965-988.

47. Akaike, H. (1974) Stochastic theory of minimal realization. *IEEE Transactions on Automatic Control*, **19**(6), 667-674.

48. Verhaegen, M. (1994) Identification of the deterministic part of MIMO state space models given in innovations form from input-output data. *Automatica*, **30**(1), 61-74.

49. Van Overschee, P. and De Moor, B. (1994) N4SID: subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, **30**(1), 75-93.

50. Van Overschee, P. and De Moor, B. (1996) *Subspace Identification for Linear Systems*. Kluwer Academic Publishers: Dordrecht, Netherlands.

51. Ljung, L. (2009) *System Identification Toolbox 7*. The Mathworks.