Entropy-Based Sensor Selection Algorithms for Damage Detection in Structural Health Monitoring Systems



Tjen Jimmy

Department of Information Engineering, Computer Science and Mathematics

University of L'Aquila

Advisor: Prof. Alessandro D'Innocenzo

Doctor of Philosophy

L'Aquila 2021



UNIVERSITÀ DEGLI STUDI DELL'AQUILA

DIPARTIMENTO DI INGENGERIA E SCIENZE DELL'INFORMAZIONE E MATEMATICA

Dottorato di Ricerca in Information and Communication Technologies- ICT

Curriculum Systems Engineering, telecommunications and HW/SW platforms.

33 ciclo

Titolo della tesi

Entropy-Based Sensor Selection Algorithms for Damage Detection in SHM Systems

SSD ING-INF/04

Dottorando

Jimmy Tjen

Coordinatore del corso

Prof. Vittorio Cortellessa

Tutor

Prof. Alessandro D'Innocenzo

A.A. 2019/2020

Acknowledgements

This thesis project could not have been finished without the contribution from so many people. I would like to express my gratitude, especially to the following people:

First, to my supervisor, Professor Alessandro D'Innocenzo, who has been a great supervisor for me. I am thankful for his guidance on the work for this thesis, and his advices have been invaluable assets for me. I would like to express my gratitude also to Francesco Smarra, especially for the helps and hints in the coding and for the general idea of the project. As without his help, I would not be able to finish this thesis.

I would like also to thank my family, especially my parents, grandmother and aunt for the great moral and financial support while I was studying abroad. Also, I would like to say thanks to all my brothers, especially Irkham, who has been a great brother figure for me. I thank him for motivating me and for giving me hints, especially for figuring out how to improve the code and algorithm.

And finally, I would like to express my gratitude also to my friends and Mentor especially to: Enrico Reticcioli, Amal Alrish, Marco Mirabilio, Alessandro D'Ortenzio and Vittorio de Iuliis for the great time that we spent together as friends. Also, a special credit goes to Giovanni D. di Girolamo, aka Gianni: I thank him for his great assistances, especially for helping me with taking care of administration letters and for sharing many things about Italian culture.

Finally, I would like to thank every individual that helped me during the process of my study in which I cannot mention them one by one.

Contents

List of Figures **ix**

List of Tables **xi**

List of Abbreviations **xiii**

1 Introduction **1**
1.1 Motivation 1
1.2 State of the Art and Progress Beyond 5

2 Theoretical Background **9**
2.1 Correlation Matrix 10
2.2 Principal Component Analysis (PCA) 11
2.3 Entropy and Information Gain 13
2.3.1 Binary Entropy, Joint Entropy and Conditional Entropy . . 14
2.3.2 Mutual Information 15
2.4 Classification and Regression Tree (CART) Algorithm 16
2.4.1 The Regression Tree 17
2.5 Introduction of Levenberg-Marquardt Algorithm 18
2.5.1 Steepest Descend Algorithm 19
2.5.2 Gauss-Newton Algorithm 20
2.5.3 Levenberg-Marquardt Algorithm 21
2.6 Kalman Filter 22
2.6.1 Extended Kalman Filter 23

3 An Entropy Based Sensor Selection Algorithm for Fault Detection **25**
3.1 PCA-based Damage Detection 26
3.2 Entropy-based Sensor Selection 28
3.3 Results and Numerical Simulations 33
3.3.1 Damage Detection Thresholds 33
3.3.2 Bookshelf Resembling Structure 34
3.3.3 UCI Bridge Column 39
3.3.4 Sheraton Hotel 43
3.4 Discussion 45

4 Regression Trees-based identification for structural damage detection 47
4.1 RT-based Damage Detection 48
4.2 Results and Numerical Simulations 49
4.2.1 Bookshelf Resembling Structure 49
4.2.2 UCI Bridge Column 51
4.2.3 Sheraton Hotel 53
4.3 Discussion 57

5 Poly-Exponential Based Fault Detection Identification Technique 59
5.1 Poly-Exponential Correction Term 60
5.2 Results and Numerical Simulations 61
5.2.1 Bookshelf Resembling Structure 62
5.2.2 UCI Bridge Column 65
5.2.3 Sheraton Hotel 68
5.3 Discussion 68

6 Conclusions and Future Work 71

References 73

List of Figures

1.1 Illustration on how to maintain the linearity for PCA algorithm. . . 3

2.1 Example of the linear correlation between 2 variables: (a) positively correlated; (b) no correlation; (c) negatively correlated. 11
2.2 Plot of the binary entropy function $H(Y)$. 15
2.3 Relationship between entropy and mutual information for Y and Z. 16
2.4 Example of regression tree structure for 2 variables and 3 partitions. Number 1, 2 and 3 are the leaf nodes. 18

3.1 Assembled LANL test structure similar to a 3-tier bookshelf. 35
3.2 Trajectories of $\gamma(k)$: 1^{st} column: nominal case, 2^{nd}-4^{th} column: faulty cases $1-3$; 1^{st} row: all PCA, 2^{nd} row: c-ss PCA, 3^{rd} Row: e-ss PCA. 38
3.3 Accelerometers location on UCI bridge column. 39
3.4 Trajectories of $\gamma(k)$: 1^{st} column: nominal case, 2^{nd}-4^{th} column: faulty cases $1-3$; 1^{st} row: all PCA, 2^{nd} row: c-ss PCA, 3^{rd} Row: e-ss PCA. 42
3.5 Accelerometers Placement Inside Sheraton Hotel. 43

4.1 Trajectories of $\gamma(k)$: 1^{st} column: nominal case, 2^{nd}-4^{th} column: faulty cases $1-3$; 1^{st} row: all RT, 2^{nd} row: c-ss RT, 3^{rd} Row: e-ss RT. . . 52
4.2 Trajectories of $\gamma(k)$: 1^{st} column: nominal case, 2^{nd}-4^{th} column: faulty cases $1-3$. 1^{st} row: all RT, 2^{nd} row: c-ss RT, 3^{rd} Row: e-ss RT. . . 55

List of Tables

3.1 Numerical test spanning r_{ij}^2 for $\theta = 1, 1.05, 1.1$. 31
3.2 (1 - NRMSE) $\times 100\%$ and sensor selection sets, PCA-based model . 36
3.3 Escape time for different damages and techniques. 37
3.4 100 - NRMSE% and sensor selection sets, PCA-based model 41
3.5 Escape time for different damages and techniques. 43
3.6 100 - NRMSE% and sensor selection sets, PCA-based model 44

4.1 (1 - NRMSE)$\times 100\%$ for the RT-based model (sensor selection sets are same as in Table 3.2) 50
4.2 Escape time for different damages and techniques. 51
4.3 (1 - NRMSE)$\times 100\%$ for the RT-based model (sensor selection sets are same as in Table 3.4) 54
4.4 Escape time for different damages and techniques. 56
4.5 (1 - NRMSE)$\times 100\%$ for the RT-based model (sensor selection sets are same as in Table 3.6) 56

5.1 Model predictive accuracy (1 - NRMSE) $\times 100\%$ for e-ss PCA and e-ss PCA + PE correction term 63
5.2 Escape time for PE correction term model and e-ss PCA for 3 different damage cases. 64
5.3 Model predictive accuracy (1 - NRMSE) $\times 100\%$ for e-ss PCA and e-ss PCA + PE correction term 66
5.4 Escape time for PE correction term model and e-ss PCA for 3 different damage cases. 67
5.5 Model predictive accuracy (1 - NRMSE) $\times 100\%$ for e-ss PCA and e-ss PCA + PE correction term 69

List of Abbreviations

AR . . . . . Auto Regressive.

CART . . . . . Classification And Regression Tree.

c-ss . . . . . Correlation-based subset selection.

EVD . . . . . Eigen Value Decomposition.

e-ss . . . . . Entropy-based subset selection.

FA . . . . . Factor Analysis.

GNA . . . . . Gauss Newton Algorithm.

LM . . . . . Levenberg Marquardt.

opt-ss . . . . . Optimal subset selection.

PC . . . . . Principal Component.

PE . . . . . Poly-Exponential

PCA . . . . . Principal Component Analysis.

POC . . . . . Principal Orthogonal Component.

RT . . . . . Regression Tree.

SHM . . . . . Structural Health Monitoring.

SVD . . . . . Singular Value Decomposition.

TVAR . . . . . Time Varying Auto Regressive.

Thesis Summary

In this thesis, we consider the problem of structural damage detection. In particular there are 3 main contributions presented in this thesis:

First, we provided a novel algorithm based on information theory so called the entropy based sensor selection (e-ss) algorithm to select a subset of sensors in order to improve the predictive model accuracy and fault detection sensitivity. We demonstrated on an experimental dataset that the combination of our algorithms outperforms other approaches based on Principal Component Analysis (PCA) and correlation based subset selection (c-ss), obtaining a model prediction accuracy improvement from 3% up to 7%, while improving the damage detection sensitivity up to 59%, and reducing the number of required accelerometers up to 80%.

For the second contribution, We proposed a novel damage detection methodology based on the combination of Kalman filtering with a model identification algorithm based on Auto Regressive(AR) system identification and Regression Trees (RT). All experimental setups show the capability of our novel RT-based approaches in predicting faults, which also outperform the previously described PCA-based models. The damage detection validation shows that e-ss RT performs better than c-ss and all-RT, in the prediction accuracy and the damage detection sensitivity while significantly reduce the number of sensors.

And finally, for the third contribution, we presented a new fault detection identification technique based on the Poly-Exponential (PE) correction term and nonlinear Kalman filtering on residuals. The numerical simulation results show that e-ss PCA + PE correction term is able to improve the model predictive accuracy w.r.t the ess-PCA up to 9% and 2% on averages. Furthermore, the proposed algorithm is capable of detecting faults faster than the e-ss PCA, while also potentially reduce the number of false positives.

In general, The presented techniques, which are validated on three different experimental datasets, provide evidences that the proposed algorithms outperform some previous approaches, improving the prediction accuracy and the damage detection sensitivity while potentially reducing the number of required sensors.

Abstract

It is often the case that small faults in a structure lead to irreparable damages that deliver a huge financial loss or even pose safety risks. Thus, an early fault detection is necessary, such that these unfortunate events can be avoided. In this thesis, the problem of structural damage detection is considered. In particular there are 3 main contributions: First, a novel sensors selection algorithm based on the concepts of entropy and information gain from information theory is developed, to reduce the number of sensors without affecting, or even improving, model accuracy; Second, a novel technique based on Kalman filtering and on a combination of Regression Trees theory from Machine Learning and Auto Regressive (AR) system identification from control theory is derived, to build models that can be used to detect structural damages. Finally, a new fault detection algorithm based on Poly-Exponential (PE) models and nonlinear Kalman filtering on the residual is introduced, which is able to enhance the sensitivity of the proposed fault detection algorithm and improve the data prediction quality for some accelerometers in a notably margin. The presented techniques are validated on three different experimental datasets, providing evidence that the proposed algorithms outperform some previous approaches, improving the prediction accuracy and the damage detection sensitivity while reducing the number of sensors.

1 Introduction

Contents

1.1 Motivation 1
1.2 State of the Art and Progress Beyond 5

It is often the case that small faults in a structure lead to massive failures. Hence, an early fault detection is compulsory in order to avoid these kinds of catastrophic events. However, a structure might be too complex, such that it is no longer a trivial task to do the maintenance process. Thus, one might ask: is there a way to perform a fault detection which is cost efficient yet precise?

1.1 Motivation

In the modern era, it is expected that civil engineering structures are equipped with large networks of sensors (e.g. accelerometers), collecting nominal and non-nominal(i.e., when a damage on the structure occur), data related to the structural health. Such collected data can be used to derive, using system identification techniques, a mathematical model of the nominal (and possibly faulty) structure behavior, so that existing techniques for early damage detection can be applied, see e.g. [1–3]. This strategy is particularly relevant, especially when the behavior of

the structure is too complex to be derived using physics-based modeling or finite element methods. In this case, the cost/effort to derive a predictive model is in most cases impractical or even impossible. Hence, data-driven approaches are necessary [4–6]. There are several ways to identify a mathematical model based on historical data, one among these is the Principal Component Analysis (PCA) [7].

PCA consists of an orthogonal projection of a dataset to a lower dimensional space such that the variance is maximized. Through the dimension reduction, PCA can unveil a much simpler pattern that can be extracted using Eigen Value Decomposition (EVD) via the sample covariance matrix [8]. This pattern is in general very sensitive to structural changes (e.g. damages) and can be used for structural damage detection. There are several research papers that utilized this paradigm, see e.g. [9], [10], [11], [12] and [13]. In general, all these techniques demonstrate the capability of PCA-based models to detect damages of a structure.

While PCA is a versatile and robust method, it also has some limitations. First, PCA is performed by calculating the EVD of the data: hence, the complexity of the algorithm increases as the number of sensors increases. In particular, the algorithm complexity of eigenvalue computation for an $n \times n$ matrix is $\mathrm{O}(n^3)$ with the standard Singular Value Decomposition (SVD) algorithm [14]. As a consequence, even PCA can be computationally intractable or inefficient when the number of sensors is very large. Another limitation is that the principal component of PCA is given as a linear combination of the original data: hence, if the linearity assumption is not satisfied PCA does not provide the desired prediction accuracy [15]. Furthermore, on high multidimensional data, upward bias in sample eigenvalue and inconsistencies of sample eigenvectors can appear, and PCA provides unexpected results [16].

To address the above issues, we provide in this thesis the following 3 contributions:

1. We propose an idea to solve the linearity issue of PCA. In particular, we derive a novel sensors selection methodology, based on the concepts of entropy and information gain from information theory, to drastically reduce the number

of sensors without affecting (and, in our experimental case, the proposed algorithm improved the accuracy by 3 to 7%) the predictive model accuracy;

2. We develop a novel damage detection methodology based on Kalman filtering exploiting a recently developed model identification algorithm [17–19], which are based on the Auto Regressive (AR) system identification and the Regression Trees (RT) theory to also capture the nonlinear behaviors from historical data;

3. We improve the fault detection algorithm above by further exploring the nonlinear behavior of residuals using the Poly-Exponential model.

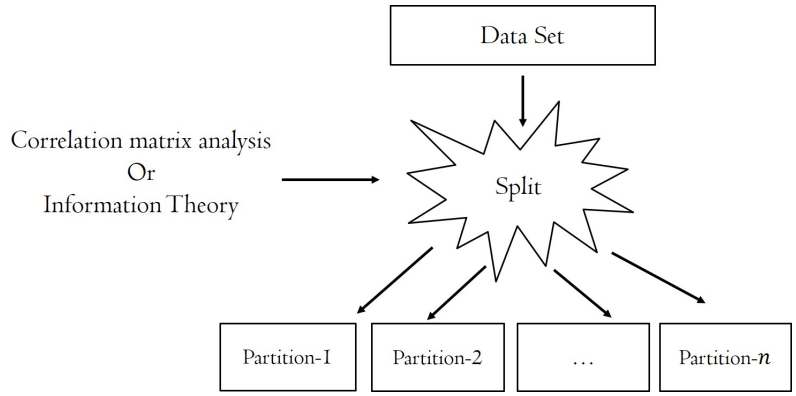


Figure 1.1: Illustration on how to maintain the linearity for PCA algorithm.

In Chapter 3, we develop a concept to solve the linearity issue presented in PCA. In order to solve this issue, we proposed an idea of partitioning the whole time series dataset into smaller subset based on the similarities (i.e. the linear correlation) between variables. We offered 2 ways to create these subsets: via the correlation matrix analysis and the entropy analysis. The correlation matrix analysis is an heuristic approach which exploits the correlation matrix to generate subset of data. Each entry of the correlation matrix is called the correlation coefficient. This coefficient tells us how strong is the linear correlation between 2 random variables (in this specific case, variables here are the reading that came from sensors placed in several places inside a structure). The idea is that, given a target

variable, select n-number of other variables which have strong linear correlation to the target variable. We denote the subset of variables obtained using this approach as the correlation subset selection (c-ss).

This approach while seems to be promising, has a limitation. The pitfall of this method is that, in many real cases, large correlation coefficient can be due to 2 variables, for example variable x_i and x_j being basically the same variables i.e., they represent the measurement of (almost) the same physical variable. When this happens, using both x_i and x_j to predict the next value of x_i, or using just x_i, does not change much the prediction accuracy as the variance carried out by x_j is negligible with respect to the variance contained in x_i. This intuition, has motivated us to consider a different metric to perform the sensor selection based on information theory, and in particular on the concepts of entropy and information gain. We call this sensor selection approach, the entropy-based sensor selection (e-ss). The e-ss algorithm is a novel sensor selection methodology based on the concepts of entropy and information gain from information theory. This algorithm is able to drastically reduce the number of sensors without affecting (and, in our experimental case, even strongly improving) the predictive model accuracy.

In Chapter 4 we take inspiration over the Regression Tree (RT) algorithm presented in [17–19] to create a novel fault detection algorithm. The authors in [17–19] proposed a modification to the classical RT algorithm by assigning an AR model to each leaf node inside the tree. In general, they were able to show that this novel RT approach is able to predict the dynamics of a system with an accuracy above 90%. Given this outstanding performance, we try to elevate this algorithm to the next level, by endowing it with an ability to detect faults. The main reason to do so is driven by the fact that a precise model gives a close representation to the system. Thus, this model can somehow be exploited in case when faults occurred in the system, i.e. by comparing the dynamics of the structure before and after an attack. In the RT based fault detection, we transform the AR models in each leaf to its time varying form: which the damage presences can be detected by tracking the change of dynamics of the AR parameters. In

the end, we show that our novel RT-based approaches outperform the PCA-based models especially on more complex structures.

And finally, in Chapter 5, we discuss about the Poly-Exponential (PE) residual model, which is the continuation of the fault detection algorithms explained in chapter 3. In this chapter, we introduce the PE model, which is a nonlinear model, that is fitted into residuals generated from the observation data and the e-ss PCA model. Similarly with the RT model in the 2nd main contribution of this dissertation, we define a time varying PE and then learn its change of dynamics through Kalman filter. This algorithm is able to improve the model prediction accuracy of e-ss PCA and more sensitive in detecting faults.

We validated our methodology on the experimental dataset taken from the repository of the Engineering Institute at the Los Alamos National Laboratory (LANL) [20]. In this thesis, we consider 3 different structures data available in their repository, which are: (i) the bookshelf resembling structure, (ii) the Irvine bridge's column and (iii) the Sheraton hotel. Details regarding to this structure will be presented in Chapter 3.

1.2 State of the Art and Progress Beyond

As stated before, there are 3 contributions presented in this thesis: first, we derive a novel sensors selection based on PCA and information gain from information theory; next, we enhance the RT algorithm proposed by [17–19] such that it is able to detect faults; and finally, we explore residuals generated by PCA algorithm with a PE model.

The PCA algorithm has been used as the basis to perform the fault detection: In [13] a nonlinear PCA algorithm was introduced to detect faults. In [12] an idea to detect faults combining the concept of wavelet transformation were introduced. In [21], the authors introduced a new statistical parameter which enhanced the fault detection quality of PCA algorithm. In [10], the authors presented a PCA-based fault detection algorithm integrated with AR model. In [9], the authors have demonstrated the capabilities of PCA algorithm in conjunction with Time

Varying Auto-Regressive (TVAR) Modeling for continuous online damage detection of vibrating structures. In this method, accelerometers data was used to derive the proper principal component (PC) followed by fitting these PCs with TVAR model in order to check the change of dynamics from nominal to faulty case: which is the indication of the presence of damages in the structure. They claimed that this damage detection framework is well suitable to detect faults inside a structure even when the data is noisy.

In Chapter 3, we further explore the fault detection concept presented in [9] and provide a novel fault detection algorithm based on information theory. In particular, our work addresses a possible limitation of the PCA algorithm, which is the linearity assumption. As stated before, PCA will not be able to deliver an optimal result if this assumption is not satisfied. To tackle this issue, we propose an idea to split the data based on the concept of information theory such that we can preserve the linearity between variables. Furthermore, we try to improve the way the fault is declared. Instead of tracking the dynamics of AR term one by one as in [9], we are proposing an idea of tracking the dynamics of L_2 norm of all AR terms simultaneously. This way, we can diminish the probability of inducing false negative or false positive due to the insensitive changes of some AR terms.

The second contribution of this research revolves around the RT algorithm from the machine learning. The RT model itself has been proven by numerous numbers of research as one of the most precise predicitive model[17–19]. In [17–19] the authors proposed a new model identification technique to derive a switching AR/ARX model starting from historical data of a system, leveraging standard model identification methods and RT. In [17] the authors applied the RT algorithm in order to predict the power consumption of 8 buildings block in the university of Pennsylvania in which: they were able to predict the dynamics with accuracy up to 98.9 %. In [18] the modified RT algorithm was used to derive a data driven model predictive control which was able to precisely optimize the power consumption and thermal comfort inside a building. In [19], the authors presented an data driven

predictive control based on RT and random forests for receding horizon control in which, they showed that the RT can closely model the dynamics of a building.

Observing the capability of the RT algorithm proposed by [17–19] we are motivated to leverage this RT algorithm to predict faults. In particular, we took into consideration the fact that model parameters can be use to track the condition of a structure. This RT model, can be used to train a precise nominal model which then can deployed to detect faults, leveraging the Kalman filter. The details on this concept is provided in Section 4.1 with the result numerically validated in Section 4.2.1, 4.2.2 and 4.2.3.

Finally, in the last contribution we try to model residuals with PE model. In machine learning, residuals are the difference between observed and predicted data. Residuals can be considered as the part that is unexplained by the model, for example due to high noise present in the data. In [22] the rescaled residual was introduced to study the earthquake occurrence and magnitude. In [23] the authors proposed an idea of standardization of the conditional residual, which was useful to identify observation outliers and clusters. In [24] the authors presented a fault detection algorithm for permanent-magnet synchronous machines based on residual analysis.

For the PE instead, this model has been used in several fields of study: in [25] the basic properties of the PE are listed and some applications are given. In particular, it is shown that certain Mellin integrals can be evaluated by this function. In [26] the authors derived several identities connecting the PE functions and the degenerate Bell polynomials. In [27] the authors modeled plasma concentration *versus* time profiles with the PE model. The proposed model was able to capture the time-course of drug concentrations with high accuracy. In [28] the author applied the PE model to perform zero-noise extrapolation, where they were able to show error reductions of 18 to 24 fold over non-mitigated circuits.

The usage of PE model in the SHM field is actually still quite low. Thus, it has motivated us to further "dig" the potential of this model. In brief, we propose an idea to fit the PE model into residuals, and integrated it to the e-ss PCA model in order to enhance the model predictive accuracy and the fault detection quality. We

will show in Chapter 5, that the e-ss PCA + PE correction term can improve both the model predictive accuracy and the fault detection quality of the e-ss PCA model.

2

Theoretical Background

Contents

2.1 Correlation Matrix 10
2.2 Principal Component Analysis (PCA) 11
2.3 Entropy and Information Gain 13
2.3.1 Binary Entropy, Joint Entropy and Conditional Entropy 14
2.3.2 Mutual Information 15
2.4 Classification and Regression Tree (CART) Algorithm 16
2.4.1 The Regression Tree 17
2.5 Introduction of Levenberg-Marquardt Algorithm . . . 18
2.5.1 Steepest Descend Algorithm 19
2.5.2 Gauss-Newton Algorithm 20
2.5.3 Levenberg-Marquardt Algorithm 21
2.6 Kalman Filter 22
2.6.1 Extended Kalman Filter 23

In this chapter, we will discuss some useful notions that are used in this thesis, which are as follows: the correlation matrix, the PCA algorithm, the entropy and information gain, the classification and regression tree (CART) algorithm, the Levenberg-Marquardt (LM) algorithm, the Kalman filter and extended Kalman filter.

2.1 Correlation Matrix

In the multivariate analysis, the correlation matrix is a square and symmetric matrix, which each entry of this matrix tells the linear dependency of each column elements. This matrix holds a significant role in clustering algorithms, for example in the PCA and the factor analysis (FA) [29].

Consider a matrix $X = [\mathbf{x}_1\ \mathbf{x}_2 \cdots\ \mathbf{x}_n] \in \mathbb{R}^{m\times n}$ consist of n-vectors of measurement $\mathbf{x}_i = [x_i(1)\ \cdots\ x_i(m)]^\top \in \mathbb{R}^m,\ i = 1, \ldots, n$, where $x_i(k) \in \mathbb{R}$ denotes the measurement at instance k of the i^{th} variable, and m denotes the number of samples. Let $\Sigma \in \mathbb{R}^{n\times n}$ be the covariance matrix

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} \ldots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} \ldots & \sigma_{2n} \\ \vdots & \vdots \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} \ldots & \sigma_{nn} \end{bmatrix} \tag{2.1}$$

and $D_\Sigma = diag([\sigma_{11}\ \sigma_{22}\ \cdots\ \sigma_{nn}])$ be the diagonal matrix of Σ, where σ_{ij} denotes the covariance between vector x_i and x_j. Then the correlation matrix $R \in \mathbb{R}^{n\times n}$ of X is defined as

$$R = D_\Sigma^{-\frac{1}{2}} \Sigma D_\Sigma^{-\frac{1}{2}} = \begin{bmatrix} 1 & r_{12} \ldots & r_{1n} \\ r_{21} & 1 \ldots & r_{2n} \\ \vdots & \vdots \ddots & \vdots \\ r_{n1} & r_{n2} \ldots & 1 \end{bmatrix}. \tag{2.2}$$

Each entry of the correlation matrix, $r_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}$ is called the correlation coefficient. The correlation coefficient basically tells us how strong the linear correlation between 2 variables is. The value of $r_{ij} \in [-1, 1]$ where value -1 or 1 indicates there exist a perfectly negative or positive linear correlation between variable x_i and x_j, while 0 means that 2 variables are not linearly correlated at all. The closer the r_{ij} to 1, the stronger the linear correlation [30].

The square of correlation coefficient (i.e., r^2)is called the coefficient of determination. The coefficient of determination is a statistical unit which describe the portion of the variance of the dependent variable covered by explanatory variable. In another way, the r^2 can be considered as the goodness-of-fit of the model, or the uncertainty of predicting the dependent variable, given the explanatory variable [31].

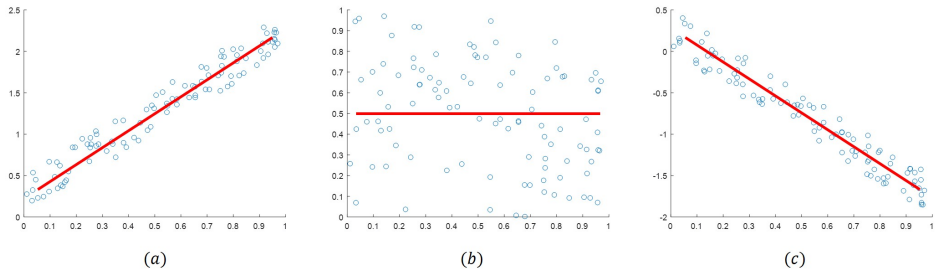


Figure 2.1: Example of the linear correlation between 2 variables: (a) positively correlated; (b) no correlation; (c) negatively correlated.

The value of the r^2 ranging from 0 to 1, and the closer the value of the r^2 is to 1, the greater is the proportion of the variance of the dependent variable that can be explained with the explanatory variable. For example, $r_{ij}^2 = 0.75$ can be interpreted as 75% of the variation of i can be explained by j, while there are still 25% of this variation remains undiscovered [32].

2.2 Principal Component Analysis (PCA)

The Principal Component Analysis or PCA is one of the most used algorithm in multivariate analysis and classification algorithm [7, 33]. In simplest way, PCA can be seen as an algorithm which project the original dataset to its orthogonal subspace such that the variance is maximized. Through this projection, a reduction of dimensions (that hold insignificant amount of the variance information) can be performed, which allows PCA to uncovered a much simpler pattern that can be extracted using Eigen Value Decomposition (EVD) via covariance or correlation matrix. This dimensional reduction can be achieved by transforming the original dataset to a new set of variables which is uncorrelated. This new set of variables are called the principal components [7, 34].

Let us consider again the matrix $X = [\mathbf{x}_1\ \mathbf{x}_2 \cdots\ \mathbf{x}_n] \in \mathbb{R}^{m \times n}$ and its covariance matrix Σ as in the previous section. Via the eigen decomposition, Σ can be rewritten in its component form as

$$\Sigma = V \Lambda V^\top, \tag{2.3}$$

where $V = [\mathbf{v}_1\ \mathbf{v}_2\ \cdots\ \mathbf{v}_n]$ is the orthogonal matrix of Σ, $\mathbf{v}_l$ is the l-th eigenvector of Σ and $\Lambda = diag([\lambda_1\ \lambda_2\ \cdots\ \lambda_n])$ is a diagonal matrix which contains the eigenvalue of matrix Σ. We can now define the principal components (PC) as orthonormal linear transformation of X:

$$\Psi = XV \tag{2.4}$$

where $\Psi = [\psi_1\ \psi_2\ \cdots\ \psi_n]$ is a matrix of PC with ψ_l is the l-th principal components. Notice that V is orthogonal matrix (i.e., $V^{-1} = V^\top$). Hence, the transformation to the original space can be performed by simply multiplying Ψ with $V^\top$ which results in $\Psi V^\top = XVV^\top = X$ [8, 16].

As mentioned before, the main idea of PCA algorithm is to perform dimensional reduction of the original dataset. To see how this reduction can be performed, let us recall the covariance matrix as defined in 2.3

$$\Sigma = V\Lambda V^\top$$

$$\begin{bmatrix} \sigma_{11} & \sigma_{12}\dots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22}\dots & \sigma_{2n} \\ \vdots & \vdots\ddots & \vdots \\ \sigma_{n1} & \sigma_{n2}\dots & \sigma_{nn} \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12}\dots & v_{1n} \\ v_{21} & v_{22}\dots & v_{2n} \\ \vdots & \vdots\ddots & \vdots \\ v_{n1} & v_{n2}\dots & v_{nn} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0\dots & 0 \\ 0 & \lambda_2\dots & 0 \\ \vdots & \vdots\ddots & \vdots \\ 0 & 0\dots & \lambda_n \end{bmatrix} \begin{bmatrix} v_{11} & v_{21}\dots & v_{n1} \\ v_{12} & v_{22}\dots & v_{n2} \\ \vdots & \vdots\ddots & \vdots \\ v_{1n} & v_{2n}\dots & v_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} \lambda_1 v_{11} & \lambda_2 v_{12}\dots & \lambda_n v_{1n} \\ \lambda_1 v_{21} & \lambda_2 v_{22}\dots & \lambda_n v_{2n} \\ \vdots & \vdots\ddots & \vdots \\ \lambda_1 v_{n1} & \lambda_2 v_{n2}\dots & \lambda_n v_{nn} \end{bmatrix} \begin{bmatrix} v_{11} & v_{21}\dots & v_{n1} \\ v_{12} & v_{22}\dots & v_{n2} \\ \vdots & \vdots\ddots & \vdots \\ v_{1n} & v_{2n}\dots & v_{nn} \end{bmatrix}$$

$$\sigma_{ij} = \sum_{k=1}^{n} \lambda_k v_{ik} v_{kj}; i, j = 1, 2, \cdots, n \tag{2.5}$$

implies diagonal element of Σ is given by:

$$\sigma_{ii} = var(\mathbf{x}_i) = \sum_{k=1}^{n} \lambda_k v_{ik} v_{ki} = \sum_{k=1}^{n} \lambda_k v_{ik}^2. \tag{2.6}$$

From (2.6) the variance of X depends on the eigenvalue. As the number of features n increases, the value of λ_k becomes tend to be smaller. If all elements of X are independent, then the PCs are exactly the original variables with unit variance. This means that, any PC with variance less than 1, contains less information than the original variables and thus, it is not worth retaining. Hence, it is possible to just

pick several number of PCs with eigenvalue greater than 1 and remove the other PCs which have variance less than 1. This selection is known as the Kaiser rule [33].

Both covariance and correlation matrix may be used to derive the PCs. The usage of correlation matrix tends to have advantages when the dataset X consist of measurement of different units (for example $\mathbf{x_1}$ is measured in inches while $\mathbf{x_2}$ in meters). In the case when all elements of X are measured in the same unit, using the covariance matrix to derive the PCs is much preferred, as the correlation matrix can be seen as the standardized version of covariance matrix. In case when all data are measured in the same unit, the standardization step is no longer required. Thus, the usage of the correlation matrix in this case will not provide any advantage over the covariance matrix.

2.3 Entropy and Information Gain

In information theory, the entropy is defined as the measure of uncertainty of a random variable. Let Y be a discrete random variable which takes value in the set $\{y_1, y_2, \cdots, y_n\}$. Then the entropy of Y is defined as:

$$H(Y) = -\sum_{i=1}^{n} \mathbb{P}(y_i)\ log_2\ \mathbb{P}(y_i), \tag{2.7}$$

where $\mathbb{P}(y_i)$ denotes the probability of Y to take the value of y_i. The entropy is expressed in bits and its value is non negative. The entropy is a function of the probability distribution of a random variable. Hence, it does not depend on the outcome values of a random variable, only its probabilities.

As stated before, the entropy is a measure of uncertainty of a random variable. The higher the entropy, the greater the uncertainty. For example, consider a case of fair coin toss. The possible outcome of this process are either "head" or "tail", In which, each of it has probability of 50% or $\frac{1}{2}$ to occur. Thus, from (2.7) we can calculate the entropy of fair coin toss, which is equal to 1. Then consider a case of fair die toss. For a die, there are 6 possible outcomes, which are number 1 to 6, with each numbers has $\frac{1}{6}$ chance to appear. In this case, based on (2.7) we can

compute the entropy of a fair die toss, which is 2.585. Observe that the entropy of a die toss is higher than coin, since a die has more possible outcomes than a coin [35].

Normally, the logarithmic function in (2.7) is to the base of 2. However, it is possible as well to define the entropy with other logarithmic base. For example, the entropy of the random variable Y with base of a is expressed as $H_a(Y)$. A special case is when the entropy is taken as a logarithmic function with the base of e or the natural number. In this case the entropy is defined as $H_e(Y)$ and it is measured in nats [36].

2.3.1 Binary Entropy, Joint Entropy and Conditional Entropy

In this section, we will further expand the notion of entropy as described previously. We will discuss several variants of entropy which will be useful later to define the information gain.

In information theory, the binary entropy is the entropy of a random variable which only takes 2 values (i.e., 0 and 1). Let Y be a random variable such that

$$Y = \begin{cases} 0, & \text{with probability } p \ , \\ 1, & \text{with probability } 1-p, \end{cases} \tag{2.8}$$

then, the binary entropy for Y is defined as:

$$\begin{aligned} H(Y) &= -\sum_{y \in Y} \mathbb{P}(Y=y) log_2(\mathbb{P}(Y=y)) \\ &= -[p.log_2(p) + (1-p).log_2(1-p)]. \end{aligned} \tag{2.9}$$

The maximum of binary entropy is attained when $p = 0.5$(see figure 2.2). This occur when both values have the same outcome likelihood [37].

Given 2 random variables, Y and Z, the joint entropy between Y and Z is defined as:

$$H(Y,Z) = -\sum_{y \in Y} \sum_{z \in Z} \mathbb{P}(y,z) \ log_2 \ \mathbb{P}(y,z) \tag{2.10}$$

Where $\mathbb{P}(y,z)$ joint distribution of Y and Z. Now, suppose that Z is known. We define the conditional entropy of Y given Z as

$$H(Y|Z) = \sum_{z \in Z} \mathbb{P}(Z=z) H(Y|Z=z)). \tag{2.11}$$

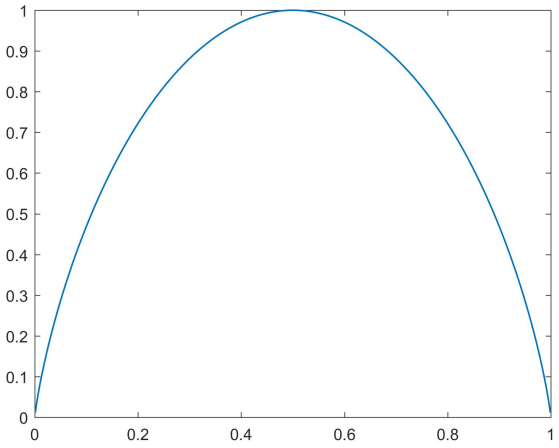


Figure 2.2: Plot of the binary entropy function $H(Y)$.

$H(Y|Z)$ represents the uncertainty of Y given the information of Z. In the case when $H(Y|X) = 0$, for each value taken by Z, we deterministically know the outcome of Y.

The joint entropy is closely related to the conditional entropy. The relation between both entropies is expressed in chain rule of entropy, i.e.,

$$\begin{aligned} H(Y,Z) &= -\sum_{y\in Y}\sum_{z\in Z}\mathbb{P}(y,z)\ log_2\ \mathbb{P}(y,z) \\ &= -\sum_{y\in Y}\sum_{z\in Z}\mathbb{P}(y,z)\ log_2\ \mathbb{P}(y)\mathbb{P}(z|y) \\ = -\sum_{y\in Y}\sum_{z\in Z}\mathbb{P}(y,z)\ log_2\ \mathbb{P}(y) &- \sum_{y\in Y}\sum_{z\in Z}\mathbb{P}(y,z)\ log_2\ \mathbb{P}(z|y) \\ = -\sum_{y\in Y}\sum_{z\in Z}\mathbb{P}(y)\ log_2\ \mathbb{P}(y) &- \sum_{y\in Y}\sum_{z\in Z}\mathbb{P}(y,z)\ log_2\ \mathbb{P}(z|y) \\ H(Y,Z) &= H(Y) + H(Z|Y). \end{aligned} \tag{2.12}$$

2.3.2 Mutual Information

If the entropy is the measure of uncertainty, the information gain or mutual information is the opposite of it. Information gain is a measure of the mutuality between 2 random variables. Information gain can been seen as the reduction in the uncertainty of a random variable due to the knowledge of the other random variable. Given 2 random variables Y and Z, the mutual information between

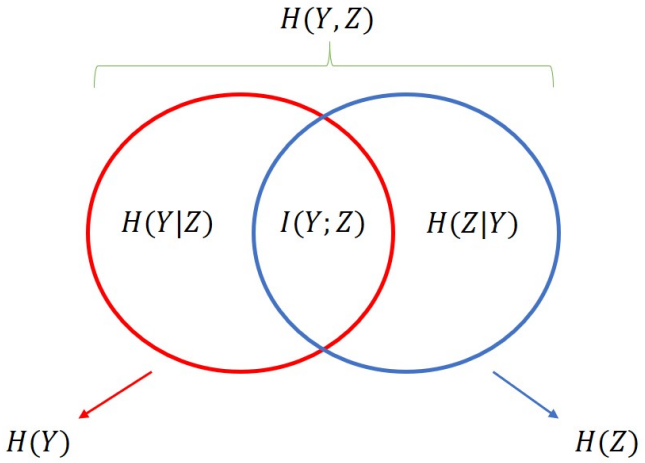


Figure 2.3: Relationship between entropy and mutual information for Y and Z.

them is defined as:

$$I(Y;Z) = H(Y) - H(Y|Z). \tag{2.13}$$

like the entropy, the value of the information gain is non-negative. This implies that $H(Y) \geq H(Y|X)$. When the value of information gain equal to 0, it means that two random variables are completely independent to each other [35]. The relation between entropy, joint entropy, conditional entropy and mutual information between 2 random variables Y and Z represented in Venn diagram is shown in Figure 2.3 [35].

2.4 Classification and Regression Tree (CART) Algorithm

The classification and regression tree (CART) algorithm is an algorithm that generates a tree like structure based on binary recursive partitioning [38]. The idea behind is: a dataset can be classified into smaller subsets based on their similarities. This partitioning process can be repeated multiple times until the dataset belongs to small clusters which can be managed easily. The CART algorithm can be summarized in 2 steps: tree building and optimal pruning process.

The tree building process starts with defining the "root" of the tree which is the whole dataset. At each step, the CART algorithm will pick the best splitting

conditions (i.e., based on the variance of samples or similarity of features) such that the dataset from the root level can be distributed in an optimal way to the each branches. This process is repeated until no more partitioning can be done. The position where no more splits is possible to be made is called the terminal node or leaf. During the tree building process, the CART algorithm assigns the predicted class for each branch, leaf and even the root. This is to determine which node will be the leaf after each partitioning process.

After no more can be done, the tree will start to prune the leaves. This is because the maximal tree tends to overfit the data. Hence, the tree tends to give a low prediction accuracy when exposed to the new input data. Leaves are pruned if the resulting change in the predicted misclassification cost is smaller with respect to the time needed to generate a more complex tree structure.

The CART algorithm has some advantages: first, it can handle highly skewed numerical and non-numerical data. Furthermore, this algorithm can also handle the missing point problem inside the data. When there exist missing data, the observation point which has a value will not be eliminated directly. Instead, the algorithm will try to find a relatively similar pattern for it, and assign it to the corresponding leaf. This is one of the main advantages compared to another analysis algorithm, as it ensures no data will be discarded during the process [39].

Based on its output, the CART algorithm is divided into 2 classes: the classification tree, which is the tree model where the outputs can take on discrete value and the regression tree which is the tree model that processes continues value such as real numbers as its output.

2.4.1 The Regression Tree

The regression tree (RT) algorithm is a part of the CART algorithm that specifically processes numbers as the input and a regression model is fitted to each node to give the prediction values. The RT algorithm generates a linear piece-wise model which is easy to interpret [38].

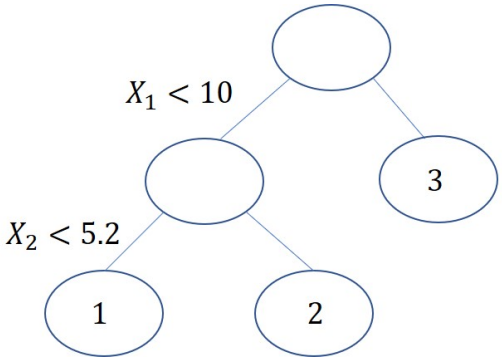


Figure 2.4: Example of regression tree structure for 2 variables and 3 partitions. Number 1, 2 and 3 are the leaf nodes.

The RT algorithm solves a least square problem by optimally choosing a variable (feature) to split on and a corresponding splitting point. After several steps the algorithm converges to the optimal solution, and the dataset X is partitioned into L hyper-rectangular sets R_ℓ such that $\bigcup_{\ell=1}^{L} R_\ell = X$: each of these sets is associated to a leaf $\ell = 1, 2, ..., L$ of the learned tree $\mathcal{T}$. Then, in each partition an estimate to the value of the variable we wish to predict is given as a constant taken from the average of the samples in the partition [40]. Figure 2.4 illustrates the example of RT when there are three partitions from 2 variables.

The main advantage for the RT algorithm is it can be applied to any number of variables. Furthermore, the RT algorithm does not require a dataset to be pre-processed (i.e., by normalization or standardization) before applied to this algorithm, also, it can handle the problem of missing data. However, the RT algorithm is considered to be computationally expensive, especially when the dataset is huge. In fact, the time complexity for the RT algorithm is the order of $O(m.n^2)$ where m and n are the number of samples and features of the dataset respectively [41].

2.5 Introduction of Levenberg-Marquardt Algorithm

Levenberg-Marquardt (LM) Algorithm is a non-linear curve fitting algorithm. This algorithm is able to provide an numerical solution to the non-linear function with a fast convergence rate of $\tilde{O}(\epsilon^{-2})$ or $O(\epsilon^{-2})$, where ϵ is the desired model

prediction error [42, 43]. The LM algorithm can be considered as the combination of the steepest descent and the Gauss-Newton algorithms (GNA). This algorithm gains the advantages of the convergence speed from the GNA and the stability of the solution from the steepest descent algorithm [43].

In the next subsection we will see how the LM algorithm is constructed as the combination between the steepest descent and the GNA. For the sake of convenience, let us define the indices that will be use for all of the following subsections: $X = [\mathbf{x_1}\ \mathbf{x_2}\ \cdots\ \mathbf{x_m}] \in \mathbb{R}^{p\times m}$ is the input dataset, $\mathbf{x_k} = [x_1(k)\ x_2(k)\ \cdots\ x_p(k)]^\intercal \in \mathbb{R}^p$ is the input vector at instance k; $F(\cdot) : \mathbb{R}^p \to \mathbb{R}$ is a non-linear function; $\boldsymbol{\alpha} = [\alpha_1\ \alpha_2\ \cdots\ \alpha_n] \in \mathbb{R}^n$ is a vector which contains the non linear model parameters of $F(\cdot)$, $\mathbf{y} = [y_1\ y_2\ \cdots\ y_m] \in \mathbb{R}^m$ is the output vector of m samples; $\hat{\mathbf{y}}$ be the non linear estimate of $\mathbf{y}$ such that $\hat{y_k} = F(\alpha_1, \alpha_2, \cdots, \alpha_n, \mathbf{x_k})$ and $\mathbf{e} \in \mathbb{R}^m = \mathbf{y} - \hat{\mathbf{y}} = [e_1\ e_2\ \cdots\ e_m]^\intercal$ is the residual vector.

2.5.1 Steepest Descend Algorithm

The steepest descent is a numerical minimization method which searches the local minimum of function via the first derivative of that function. The steepest descent algorithm converges well for the problem with a simple objective function [44]. To see how the gradient descend works, let us consider:

$$E(X, \alpha) = \frac{1}{2}\sum_{i=1}^{m} e_i^2 \tag{2.14}$$

be the sum squares of residual. On each iteration, the steepest descend algorithm will find the gradient of a function which is in this case, the first derivative of (2.14):

$$\nabla E(X, \boldsymbol{\alpha_k}) = \frac{\partial E(X, \boldsymbol{\alpha_k})}{\partial \boldsymbol{\alpha}} = \left[\frac{\partial E(X, \boldsymbol{\alpha_k})}{\partial \alpha_1} \quad \frac{\partial E(X, \boldsymbol{\alpha_k})}{\partial \alpha_2} \quad \cdots \quad \frac{\partial E(X, \boldsymbol{\alpha_k})}{\partial \alpha_n}\right]. \tag{2.15}$$

With the definition of the gradient in (2.15), the update condition for the non linear function parameters can be written as:

$$\alpha_{\mathbf{k+1}} = \alpha_{\mathbf{k}} - \mu \nabla E(\mathbf{x}, \boldsymbol{\alpha_k}), \tag{2.16}$$

where $\mu \in \mathbb{R}^+$ is a constant which determine the step of descent. Under the right choice of μ on each the iteration the error will become smaller until the algorithm converges to the small neighborhood of the local minimum [45].

2.5.2 Gauss-Newton Algorithm

The Gauss Newton algorithm (GNA) is an algorithm that can be used to solve the non-linear least squares problem by minimizing the sum of squares of the objective function [46]. Consider the sum squares of error as in (2.14). The optimal value for parameter α will be satisfied when the first derivative of (2.14) is equal to 0, i.e.,

$$\begin{aligned} \frac{\partial E(X, \boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} &= 0 \\ \frac{\partial}{\partial \boldsymbol{\alpha}} \frac{1}{2} \sum_{i=1}^{m} e_i^2 &= 0 \\ \frac{\partial}{\partial \boldsymbol{\alpha}} \frac{1}{2} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 &= 0 \\ J^\top (\mathbf{y} - \hat{\mathbf{y}}) &= 0 \end{aligned} \tag{2.17}$$

with $J = \begin{bmatrix} \frac{\partial \hat{\mathbf{y}}}{\partial \alpha_1} & \frac{\partial \hat{\mathbf{y}}}{\partial \alpha_2} & \cdots & \frac{\partial \hat{\mathbf{y}}}{\partial \alpha_n} \end{bmatrix}$ is the Jacobian matrix or the matrix of first derivative. Suppose that the optimal parameters are found by updating the current parameters iteratively as follows:

$$\alpha_{\mathbf{k+1}} = \alpha_{\mathbf{k}} + \Delta_\alpha, \tag{2.18}$$

where Δ_α is the update parameters. The goal now is to find what is the right estimate to Δ_α. Consider the first order Taylor expansion of $y(\alpha_{\mathbf{k+1}})$ at $\alpha_{\mathbf{k}}$:

$$\begin{aligned} \hat{y}(\alpha_{\mathbf{k+1}}) &= \hat{y}(\alpha_{\mathbf{k}}) + \frac{\partial \hat{y}(\alpha_{\mathbf{k}})}{\partial \boldsymbol{\alpha}} . (\alpha_{\mathbf{k+1}} - \alpha_{\mathbf{k}}) \\ y(\alpha_{\mathbf{k+1}}) &= y(\alpha_{\mathbf{k}}) + J\Delta_\alpha \end{aligned} \tag{2.19}$$

Assume that $\alpha_{\mathbf{k+1}}$ is the optimal parameters. Then, $\hat{y}(\alpha_{\mathbf{k+1}})$ should also satisfy (2.17). Substituting (2.19) to (2.17) yields:

$$\begin{aligned} J^\top(\mathbf{y} - \hat{\mathbf{y}}) &= 0 \\ \Rightarrow J^\top(\mathbf{y} - \hat{\mathbf{y}} + J\Delta_\alpha) &= 0 \\ \Rightarrow J^\top(\mathbf{y} - \hat{\mathbf{y}}) + J^\top J\Delta_\alpha &= 0 \\ \Rightarrow J^\top J\Delta_\alpha &= -J^\top(\mathbf{y} - \hat{\mathbf{y}}) \\ \Rightarrow \Delta_\alpha &= -(J^\top J)^{-1}J^\top(\mathbf{y} - \hat{\mathbf{y}}) \\ \Rightarrow \Delta_\alpha &= -(H)^{-1}J^\top(\mathbf{y} - \hat{\mathbf{y}}) \end{aligned} \tag{2.20}$$

where $H = J^\top J$ is the Hessian matrix, or the matrix of second order partial derivatives. Equation (2.20) is the update condition for the GNA. In general, the update rule of GNA for each iteration is defined as:

$$\alpha_{\mathbf{k+1}} = \alpha_{\mathbf{k}} - H^{-1}J^\top(\mathbf{y} - \hat{\mathbf{y}}) \tag{2.21}$$

For a moderate amount of data, the GNA typically converges with a rate that is faster than the steepest descent algorithm [47]. However, as good as it is, the GNA still has a convergence problem when the matrix $J^\top J$ is not invertible.

2.5.3 Levenberg-Marquardt Algorithm

The Levenberg-Marquardt (LM) algorithm tries to fix the "gap" left by GNA. This algorithm fixes the issue of invertibility in GNA by introducing an extra term in the Hessian matrix:

$$H = J^\top J + \mu I, \tag{2.22}$$

where: $\mu \in \mathbb{R}^+$ is the damping factor and I is the identity matrix. The extra term in (2.22) will ensure that the diagonal of the Hessian matrix is always greater than zero. Thus, it guarantees the Hessian matrix to be invertible [43].

Given (2.22) the update rule for the LM algorithm can be written as

$$\alpha_{\mathbf{k+1}} = \alpha_{\mathbf{k}} - (J^\top J + \mu I)^{-1}J^\top(\mathbf{y} - \hat{\mathbf{y}}). \tag{2.23}$$

The behavior of the LM algorithm is based on the selection of μ. When μ is much larger than $J^\top J$, the update rule of the LM algorithm in (2.23) will turn into the update rule of steepest gradient algorithm as in (2.16). Alternatively, when μ is smaller than $J^\top J$, says μ is close to zero, (2.23) will turn into (2.20), which is the update rule of GNA. Thus, by assigning μ to be a large number in the beginning and gradually reducing it when the solution is in the "path", the LM algorithm can solves the non linear least squares problem with the stability of steepest descent, and the convergence rate of the GNA [44].

2.6 Kalman Filter

The Kalman filter is an algorithm which provides an estimate of unknown variables from the measurements of another variables observed over time. This filter solves a set of mathematical equations by implementing an estimator which is optimal, such that it minimizes the estimated error covariance under right conditions [48]. This algorithm is able to estimate the past, present and future states given a model of the system dynamics.

Suppose that we want to estimate the evolution of the state $\mathbf{x} \in \mathbb{R}^n$ given a discrete-time process as follows:

$$\mathbf{x}(\mathbf{k}) = F\mathbf{x}(\mathbf{k}-\mathbf{1}) + B\mathbf{u}(\mathbf{k}-\mathbf{1}) + \mathbf{w}(\mathbf{k}-\mathbf{1}), \tag{2.24}$$

where: k is the time indexes, $F \in \mathbb{R}^{n\times n}$ is the state transition matrix, $B \in \mathbb{R}^{n\times p}$ is the control-input matrix, $\mathbf{u} \in \mathbb{R}^p$ is the control input and $w \in \mathbb{R}^n$ is the zero mean gaussian process noise with a the variance of σ_w^2. Furthermore, suppose that (2.24) is paired with the measurement vector, $\mathbf{z} \in \mathbb{R}^m$ as following:

$$\mathbf{z}(\mathbf{k}) = G\mathbf{x}(\mathbf{k}) + \mathbf{v}(\mathbf{k}). \tag{2.25}$$

Here, $G \in \mathbb{R}^{m\times n}$ is the measurement matrix and $\mathbf{v} \in \mathbb{R}^n$ is the measurement noise of zero mean and variance of σ_v^2. Then, on each step, the Kalman filter will provide

a solution in 2 stages: prediction and update as follows:

Prediction :	
Predicted state estimate	$\hat{\mathbf{x}}(\mathbf{k}\|\mathbf{k-1}) = F\hat{\mathbf{x}}(\mathbf{k-1}\|\mathbf{k-1}) + B\hat{\mathbf{u}}(\mathbf{k-1})$
Predicted error covariance	$\mathbf{P}(\mathbf{k}\|\mathbf{k-1}) = F\mathbf{P}(\mathbf{k-1}\|\mathbf{k-1})F^\top + \sigma_w^2(l)$
Update :	
Measured residual	$\mathbf{r}(\mathbf{k}) = \mathbf{z}(\mathbf{k}) - G\hat{\mathbf{x}}(\mathbf{k}\|\mathbf{k-1})$
Kalman gain	$K(k) = \mathbf{P}(\mathbf{k}\|\mathbf{k-1})G^\top[\sigma_v^2(k) + G\mathbf{P}(\mathbf{k}\|\mathbf{k-1})G^\top]^{-1}$
Updated state estimate	$\hat{\mathbf{x}}(\mathbf{k}\|\mathbf{k}) = \hat{\mathbf{x}}(\mathbf{k}\|\mathbf{k-1}) + K(k)\mathbf{r}(\mathbf{k})$
Updated covariance estimate	$\mathbf{P}(\mathbf{k}\|\mathbf{k}) = (I - K(k)G)\mathbf{P}(\mathbf{k}\|\mathbf{k-1})$

Where: $\hat{\mathbf{x}}(\mathbf{k}|\mathbf{k-1})$ and $\hat{\mathbf{x}}(\mathbf{k}|\mathbf{k})$ are the *a priori* and *a posteriori* estimate of $\mathbf{x}$, $\mathbf{P}(\mathbf{k}|\mathbf{k-1})$ and $\mathbf{P}(\mathbf{k}|\mathbf{k})$ are the *a priori* and *a posteriori* estimate of the covariance, $\mathbf{r}(\mathbf{k})$ and $K(k)$ are the prediction residual and the Kalman gain at instance k, respectively. On the initialization stage, the Kalman filter needs initial values on the stage estimate $\hat{\mathbf{x}}(\mathbf{k-1}|\mathbf{k-1})$ and the covariance $\mathbf{P}(\mathbf{k-1}|\mathbf{k-1})$. For these values, there is a rule so called *"initial ignorance"* which suggests to pick a large $\mathbf{P}(\mathbf{k-1}|\mathbf{k-1})$ in order to converge rapidly [49].

Notice that the Kalman filter is constructed under the assumption that the process and measurement models are linear. Thus, this algorithm will not be able to deliver a satisfying result if this assumption is not satisfied. If the models are non-linear, then we need to consider the extended Kalman filter, which is an adaptation of Kalman filter algorithm for non-linear model detailed in the next subsection.

2.6.1 Extended Kalman Filter

The Kalman filter solution for a nonlinear differentiable equation is similar to the its solution in linear system. The main difference is located in the definition of the state transition and observation matrix.

Following the definition from Section 2.6, consider a non-linear dynamical

system as follows:

$$\begin{aligned} \mathbf{x}(\mathbf{k}) &= f(\mathbf{x}(\mathbf{k-1}), \mathbf{u}(\mathbf{k-1})) + \mathbf{w}(\mathbf{k}), \\ \mathbf{z}(\mathbf{k}) &= g(\mathbf{x}(\mathbf{k})) + \mathbf{v}(\mathbf{k}), \end{aligned} \tag{2.26}$$

where: $f(\cdot) : \mathbb{R}^{n+p} \rightarrow \mathbb{R}^n$ and $g(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^m$ are non-linear differentiable function. Let $\tilde{\mathbf{F}}(\mathbf{k})$ and $\tilde{\mathbf{G}}(\mathbf{k})$ be the Jacobian vector of $f(\cdot)$ and $g(\cdot)$, respectively, an is given by:

$$\begin{aligned} \tilde{\mathbf{F}}(\mathbf{k}) &= \frac{\partial f}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}(\mathbf{k-1}|\mathbf{k-1}),\mathbf{u}(\mathbf{k-1})}, \\ \tilde{\mathbf{G}}(\mathbf{k}) &= \frac{\partial g}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}(\mathbf{k}|\mathbf{k-1})}. \end{aligned} \tag{2.27}$$

Then, the extended Kalman filter solution for (2.6.1) is described by the following 2 steps:

Prediction :	
Predicted state estimate	$\hat{\mathbf{x}}(\mathbf{k}\|\mathbf{k-1}) = f(\hat{\mathbf{x}}(\mathbf{k-1}\|\mathbf{k-1}), \mathbf{u}(\mathbf{k-1}))$
Predicted error covariance	$\mathbf{P}(\mathbf{k}\|\mathbf{k-1}) = \tilde{\mathbf{F}}(\mathbf{k})\mathbf{P}(\mathbf{k-1}\|\mathbf{k-1})\tilde{\mathbf{F}}(\mathbf{k})^\top + \sigma_w^2(l)$
Update :	
Measured residual	$\mathbf{r}(\mathbf{k}) = \mathbf{z}(\mathbf{k}) - \tilde{\mathbf{G}}(\mathbf{k})\hat{\mathbf{x}}(\mathbf{k}\|\mathbf{k-1})$
Kalman gain	$K(k) = \mathbf{P}(\mathbf{k}\|\mathbf{k-1})\tilde{\mathbf{G}}(\mathbf{k})^\top[\sigma_v^2(k) + \tilde{\mathbf{G}}(\mathbf{k})\mathbf{P}(\mathbf{k}\|\mathbf{k-1})\tilde{\mathbf{G}}(\mathbf{k})^\top]^{-\mathbf{1}}$
Updated state estimate	$\hat{\mathbf{x}}(\mathbf{k}\|\mathbf{k}) = \hat{\mathbf{x}}(\mathbf{k}\|\mathbf{k-1}) + K(k)\mathbf{r}(\mathbf{k})$
Updated covariance estimate	$\mathbf{P}(\mathbf{k}\|\mathbf{k}) = (I - K(k)\tilde{\mathbf{G}}(\mathbf{k}))\mathbf{P}(\mathbf{k}\|\mathbf{k-1})$

3

An Entropy Based Sensor Selection Algorithm for Fault Detection

Contents

3.1 PCA-based Damage Detection 26
3.2 Entropy-based Sensor Selection 28
3.3 Results and Numerical Simulations 33
3.3.1 Damage Detection Thresholds 33
3.3.2 Bookshelf Resembling Structure 34
3.3.3 UCI Bridge Column 39
3.3.4 Sheraton Hotel 43
3.4 Discussion 45

Some of the results in this section are also contained in [50] which has been presented in International Conference on Automation Science and Engineering 2020 (20-21 August 2020, online virtual meeting) and published by IEEE.

In this chapter, we will discuss how we tackle the issue of linearity that strongly affects the performance of PCA algorithm by introducing the entropy based sensor selection (e-ss) algorithm. The e-ss algorithm is an algorithm based on the concept of entropy and information gain. This algorithm is able to split a set of data into smaller subsets in order to maintain the linearity between variables. We will show

that the e-ss PCA improves the performance of PCA algorithm proposed in [9] both in model predictive accuracy and fault detection quality.

3.1 PCA-based Damage Detection

In this section, we recall the main concept of fault detection presented in [9] where they performed a PCA based fault detection. Their method consists of 2 main steps: (1) data projection onto an orthogonal subspace and AR terms identification, and (2) Kalman-based dynamic estimation of the corresponding AR coefficients. Let us define $X = [\mathbf{x}_1\ \mathbf{x}_2 \cdots \mathbf{x}_n] \in \mathbb{R}^{m\times n}$ as the matrix of sensor measurements consisting of vectors $\mathbf{x}_i = [x_i(1)\ \cdots\ x_i(m)]^\top \in \mathbb{R}^m,\ i = 1, \ldots, n$, where $x_i(k) \in \mathbb{R}$ denotes the measurement at time k of the i^{th} sensor, n denotes the number of sensors (features), and m denotes the number of samples measured from each sensor. Denote by $R \in \mathbb{R}^{n\times n}$ the correlation matrix of X, and by $V \in \mathbb{R}^{n\times n}$ the matrix of eigenvectors of R sorted so that the corresponding eigenvalues of V are decreasing from the left to the right columns.

Step 1. The principal orthogonal component (POC) $\Psi \in \mathbb{R}^{m\times n}$ can be defined as:

$$\Psi = XV, \tag{3.1}$$

and computed using the collected data. Then, for each j^{th} principal component (PC) ψ_j, i.e., the column wise elements of Ψ, an AR model can be described

$$\psi_j(k) = \sum_{i=1}^{\tau} \alpha_{ij}^0 \psi_j(k-i), \tag{3.2}$$

where τ denotes the Auto Regressive (AR) order, and k is the discrete time variable evolving according to the sampling time. Coefficients α_{ij}^0 are computed via the classical least squares method commonly used to identify AR models [9].

Step 2. Once the AR model (3.2) is obtained, we can use it to compute prediction, followed by defining a residuals to be used in the context of faults detection. Note that the AR coefficients in (3.2) refer to the system's nominal dynamics. The idea is to assume that the AR terms are subject to changes when for example a damage occurs. Thus, to estimate non-nominal changes of such

AR terms, a state variable vector $\boldsymbol{\alpha}_j(k)$ for each component j of our system is defined, together with the associated dynamics

$$\begin{aligned} \boldsymbol{\alpha}_j(k+1) &= I\boldsymbol{\alpha}_j(k) + \omega(k) \\ ,\psi_j(k) &= \bar{\boldsymbol{\psi}}_j(k)\boldsymbol{\alpha}_j(k) + \nu(k), \end{aligned} \tag{3.3}$$

where $\boldsymbol{\alpha}_j(k) = [\alpha_{1j}(k), ..., \alpha_{\tau j}(k)]$ is a state vector with initial condition $\boldsymbol{\alpha}_j(0) = [\alpha^0_{1j}, ..., \alpha^0_{\tau j}]$, $\bar{\boldsymbol{\psi}}_j(k) = [\psi_j(k-1), ..., \psi_j(k-\tau)]$ is a time varying observation matrix, $\omega(k)$ and $\nu(k)$ are uncorrelated white noise signals with variance σ^2_ω and σ^2_ν respectively, and I is the identity matrix of appropriate dimension. Since $\bar{\boldsymbol{\psi}}_j(k)$ and $\boldsymbol{\psi}_j(k)$ are given by sensor measurements transformed via (3.1), they are known at each time step k. In [9] a Kalman filter can be easily constructed leveraging [51–53] to estimate the time evolution of the AR parameters. In particular, from Section 2.6 for each time step k, the Kalman filter solution for the model above is computed as

$$\begin{aligned} &\boldsymbol{\alpha}_j(k|k-1) = \boldsymbol{\alpha}_j(k-1|k-1), \\ &\boldsymbol{P}_{\boldsymbol{\alpha}}(k|k-1) = \boldsymbol{P}_{\boldsymbol{\alpha}}(k-1|k-1) + I.\sigma^2_\omega, \\ &\psi_j(k|k-1) = \bar{\boldsymbol{\psi}}_j(k)\boldsymbol{\alpha}_j(k|k-1), \\ &S(k) = \bar{\boldsymbol{\psi}}_j(k)\boldsymbol{P}_{\boldsymbol{\alpha}}(k|k-1)\bar{\boldsymbol{\psi}}_j(k)^\top + \sigma^2_\nu, \\ &\tilde{R}(k) = \psi_j(k) - \psi_j(k|k-1), \\ &K_G(k) = \boldsymbol{P}_{\boldsymbol{\alpha}}(k|k-1)\bar{\boldsymbol{\psi}}_j(k)^\top S(k)^{-1}, \\ &\boldsymbol{\alpha}_j(k|k) = \boldsymbol{\alpha}_j(k|k-1) - K_G(k)\tilde{R}(k), \\ &\boldsymbol{P}_{\boldsymbol{\alpha}}(k|k) = [1 - K_G(k)\bar{\boldsymbol{\psi}}_j(k)]\boldsymbol{P}_{\boldsymbol{\alpha}}(k|k-1), \end{aligned} \tag{3.4}$$

where $\boldsymbol{\alpha}_j(k|k-1)$ and $\boldsymbol{\alpha}_j(k|k)$ are the a priori and a posteriori estimates of $\boldsymbol{\alpha}_j$, $\boldsymbol{P}_{\boldsymbol{\alpha}}(k|k-1)$ and $\boldsymbol{P}_{\boldsymbol{\alpha}}(k|k)$ are the a priori and a posteriori estimates of the covariance, and $S(k)$, $\tilde{R}(k)$, $K_G(k)$ are respectively the pre-fit residual covariance, the prediction residual and the Kalman gain at instance k. The Kalman filter above provides an estimate of $\boldsymbol{\alpha}_j(k)$ in run-time: we will use such estimate to detect whether a damage/fault is running or not in the system, as will be explained in Section 3.3.1.

3.2 Entropy-based Sensor Selection

Let us consider as above a dataset described by the matrix $X \in \mathbb{R}^{m \times n}$ consisting of vector measurements $[\mathbf{x}_1\ \mathbf{x}_2 \cdots \mathbf{x}_n]$, and let $X_1, ..., X_n$ be random variables corresponding to each component (sensor). Namely, each vector $\mathbf{x}_i = [x_i(1) \cdots x_i(m)]^\top$ is a sequence of stochastic processes due to time dependency of X_i. We will denote by $\mathcal{X}_i = \mathbf{x}_i \in \mathbb{R}^{m \times 1}$ the sequence of measurements of variable i and, with slight abuse of notation, by $\mathcal{X}_{ij} = [\mathbf{x}_i\ \mathbf{x}_j] \in \mathbb{R}^{m \times 2}$ the sequences of measurements of variables i and j.

The e-ss algorithm illustrated in this section will be based on the notions of entropy and information gain. If we assume that we wish to define a subset of sensors to predict the component i, and define a binary random variable Z_{ij}, $\forall j \neq i$, as

$$Z_{ij} = \begin{cases} 1 & \text{if } |X_i - \beta_{ij} X_j| < \theta \cdot \sigma(X_i) \\ 0 & \text{if otherwise} \end{cases}, \tag{3.5}$$

where $\theta \in [1, +\infty) \subset \mathbb{R}$, $\sigma(X_i)$ denotes the standard deviation of X_i and $\beta_{ij} = (\mathcal{X}_{ij}^\top \mathcal{X}_{ij})^{-1} \mathcal{X}_{ij}^\top \mathcal{X}_i$ is the solution of the least squares method applied on a dataset consisting of independent and dependent variable observations given by $\mathcal{X}_{ij}$ and $\mathcal{X}_i$, respectively. Let $p_{ij} = \mathbb{P}[Z_{ij} = 1]$. At this point, it is possible to define the information gain of Z_{ij} given β_{ij} as

$$I(Z_{ij}; \beta_{ij}) = H(Z_{ij}) - H(Z_{ij}|\beta_{ij}), \tag{3.6}$$

where $I(Z_{ij}; \beta_{ij})$ denotes the information gain of Z_{ij} given β_{ij}(i.e., given the datasets $\mathcal{X}_i$ and $\mathcal{X}_j$).

The main idea is to estimate the information gain between each pair $i, j, j \neq i$, and chose the component j^* such that the information gain $I(Z_{ij}; \beta_{ij})$ is maximized. This is equivalent to choosing j^* that minimizes the conditional entropy $H(Z_{ij}|\beta_{ij})$. However, it is very difficult in general to compute exactly $H(Z_{ij}|\beta_{ij})$. As the main contribution of this section, we derive an upper bound $\hat{H}(Z_{ij}|\beta_{ij})$ of $H(Z_{ij}|\beta_{ij})$ as a function of the coefficient of determination r_{ij}^2, which can be computed easily. As a consequence, when we compute $\hat{j}^*$ that minimizes $\hat{H}(Z_{ij}|\beta_{ij})$, we are basically minimizing an upper bound of the conditional entropy, and thus maximizing a lower bound of the information gain.

In the following, with abuse of notation, we emphasize that the residual error and the coefficient of determination depend on the number of samples m using the notations $e_{ij}(m)$ and $r_{ij}(m)$.

Lemma 1. *Given Z_{ij} the following holds:*

$$p_{ij} \geq \lim_{m\to\infty}(1 - \theta^{-2} + \theta^{-2} \cdot r_{ij}^2(m)). \tag{3.7}$$

Proof. We recall that, for $m \to \infty$, $e_{ij}(m) = X_i - \beta_{ij}(m)X_j$ is a random variable representing the residual error and $\sigma(X_i)$ is the standard deviation of X_i. By the Chebyshev's inequality, i.e.,

$$\lim_{m\to\infty} \mathbb{P}(|e_{ij}(m) - \mu(e_{ij}(m))| \geq c\sigma(e_{ij}(m))) \leq \frac{1}{c^2}, \tag{3.8}$$

where $\mu(\cdot) : \mathbb{R} \to \mathbb{R}$ and $\sigma(\cdot) : \mathbb{R} \to \mathbb{R}$ respectively denote the expectation and the standard deviation, and c is any real positive number. We now show that, under the assumption of random sampling, $\mu(e_{ij}(m))$ and $\sigma(e_{ij}(m))$ can be well estimated from X_i when the number of samples m goes to infinity. Let $\mu_m(\cdot)$ and $\sigma_m(\cdot)$ respectively denote the sample mean and the sample standard deviation of the set $\mathcal{X}_i$ of m samples, with:

$$\lim_{m\to\infty} \mu_m(\mathcal{X}_i) = \mu(X_i) \text{ and } \lim_{m\to\infty} \sigma_m(\mathcal{X}_i) = \sigma(X_i). \tag{3.9}$$

We first note that $\mu_m(e_{ij}(m)) = 0$, $\forall m$ as a direct consequence of defining $\beta_{ij}(m)$ via the least squares algorithm. Then, the sample standard deviation of $e_{ij}(m)$ can be defined as

$$\sigma_m(e_{ij}(m)) = \sqrt{\frac{\sum_{k=1}^{n}(x_i(k) - \beta_{ij}(m)x_j(k))^2}{m-d}}, \tag{3.10}$$

where d is the degree of freedom. The coefficient of determination obtained from the datasets $\mathcal{X}_i$ and $\mathcal{X}_j$, each consisting of m samples, is defined as

$$r_{ij}^2(m) = 1 - \frac{\sum_{k=1}^{n}(x_i(k) - \beta_{ij}(m)x_j(k))^2}{\sum_{k=1}^{n}(x_i(k) - \mu_m(\mathcal{X}_i))^2}. \tag{3.11}$$

By simple manipulations on (3.10) and (3.11) it follows that

$$\sigma_m(e_{ij}(m)) = \sigma_m(\mathcal{X}_i)\sqrt{1 - r_{ij}^2(m)}. \tag{3.12}$$

We can now rewrite (3.8) as

$$\lim_{m\to\infty} \mathbb{P}\left(|e_{ij}(m)| \geq c\sigma_m(\mathcal{X}_i)\sqrt{1-r_{ij}^2(m)}\ \right) \leq \frac{1}{c^2}. \tag{3.13}$$

By imposing $c = \frac{\theta}{\sqrt{1-r_{ij}^2(m)}} > 0$, we obtain

$$\lim_{m\to\infty} \mathbb{P}(|e_{ij}(m)| \geq \theta \cdot \sigma_m(\mathcal{X}_i)) \leq \lim_{m\to\infty}(\theta^{-2} - \theta^{-2} \cdot r_{ij}^2(m)). \tag{3.14}$$

By (3.9) if follows that

$$\lim_{m\to\infty} \mathbb{P}(|e_{ij}(m)| < \theta \cdot \sigma(X_i)) = p_{ij} \geq \lim_{m\to\infty}(1 - \theta^{-2} + \theta^{-2} \cdot r_{ij}^2(m)). \tag{3.15}$$

This concludes the proof.

Theorem 1. *Let*

$$\begin{aligned}
&\hat{H}(Z_{ij}|\beta_{ij}) \doteq \\
&- [(1 - \theta^{-2} + \theta^{-2} \cdot r_{ij}^2(m))log_2(1 - \theta^{-2} + \theta^{-2} \cdot r_{ij}^2(m)) \\
&+ (\theta^{-2} - \theta^{-2} \cdot r_{ij}^2(m))log_2(\theta^{-2} - \theta^{-2} \cdot r_{ij}^2(m))].
\end{aligned}$$

Then, for $m \to \infty$ and $r_{ij}^2 \in [1 - \frac{\theta^2}{2}, 1]$,

$$H(Z_{ij}|\beta_{ij}) \leq \hat{H}(Z_{ij}|\beta_{ij}).$$

Proof. $Z_{ij}|\beta_{ij}$ is a binary random variable, therefore we obtain,

$$H(Z_{ij}|\beta_{ij}) = H(p_{ij}) = -[p_{ij}log_2(p_{ij}) + (1 - p_{ij})log_2(1 - p_{ij})]. \tag{3.16}$$

Since by Lemma 1 $p_{ij} \geq \lim_{m\to\infty}(1-\theta^{-2}+\theta^{-2}\cdot r_{ij}^2(m))$, and since the binary entropy function is strictly decreasing function in the interval of [0.5, 1] (see Figure 2.2), it directly follows that $H(Z_{ij}|\beta_{ij}) \leq \hat{H}(Z_{ij}|\beta_{ij})$ for $(1 - \theta^{-2} + \theta^{-2} \cdot r_{ij}^2(m)) \in [0.5, 1]$, which directly implies the thesis since $\theta > 1$.

Remark 1. *Note that when $r_{ij}^2 \in [0, 1 - \frac{\theta^2}{2})$ we cannot state any relation between $H(Z_{ij}|\beta_{ij})$ and $\hat{H}(Z_{ij}|\beta_{ij})$, namely we can just imply the trivial inequality $H(Z_{ij}|\beta_{ij}) < 1$ that holds by definition.*

Table 3.1: Numerical test spanning r_{ij}^2 for $\theta = 1, 1.05, 1.1$.

$\theta = 1$			
r_{ij}^2	p_{ij}	$\hat{H}(Z_{ij}\|\beta_{ij})$	$H(Z_{ij}\|\beta_{ij})$
0.9613	1	0.2361	0
0.6051	0.913	0.9679	0.4264
0.4029	0.7514	0.9726	0.8091
0.3304	0.7089	0.9153	0.8701
0.3178*	0.698	0.9019	0.8837
0.2732	0.681	0.8461	0.9033
0.1639	0.6319	0.6436	0.9492
0.1409	0.6229	0.5865	0.956
0.1066	0.6102	0.4895	0.9647
$\theta = 1.05$			
r_{ij}^2	p_{ij}	$\hat{H}(Z_{ij}\|\beta_{ij})$	$H(Z_{ij}\|\beta_{ij})$
0.9405	1	0.303	0
0.6355	0.9978	0.9156	0.0226
0.4941	0.8514	0.9951	0.6063
0.3011	0.722	0.9476	0.8527
0.2397*	0.6918	0.8936	0.8911
0.1938	0.6753	0.8397	0.9094
0.1117	0.6436	0.7103	0.9397
0.0952	0.6376	0.6786	0.9447
0.0712	0.6313	0.6285	0.9497
$\theta = 1.1$			
r_{ij}^2	p_{ij}	$\hat{H}(Z_{ij}\|\beta_{ij})$	$H(Z_{ij}\|\beta_{ij})$
0.8782	1	0.4711	0
0.3124	0.7668	0.9865	0.7835
0.2257	0.7217	0.9428	0.8531
0.1687*	0.6966	0.8966	0.8854
0.1301	0.6804	0.8569	0.9039
0.103	0.6701	0.8247	0.9148
0.0834	0.6624	0.7991	0.9225
0.049	0.6495	0.7492	0.9345
0.0422	0.6467	0.7385	0.937

* denotes the largest experimental r_{ij}^2 such that $\hat{H}(Z_{ij}|\beta_{ij}) \geq H(Z_{ij}|\beta_{ij})$ is still satisfied.

To provide numerical experimental results we consider some simulations with a numerical test we generated 2 random vectors, namely x_i and g of 10^4 components, from normally distributed random variables X_i and G via the Matlab Random Number Generation (randn) function. Then, we defined the random variable $X_j = 2X_i + \xi G$. Thus, x_j is endowed with a linear correlation to x_i perturbed by ξg: the larger ξg the more independent x_j from x_i, and consequently the smaller r_{ij}^2. In particular, for the test we selected $\xi \in [0.75, 3.5]$. Starting from x_j and x_i, we calculated p_{ij} from (3.5) using Monte Carlo simulations, and $H(Z_{ij}|\beta_{ij})$ and $\hat{H}(Z_{ij}|\beta_{ij})$ as illustrated above in order to span from (almost) perfect linear correlation to (almost) no correlation at all. The results of the numerical test are shown in Table 3.1.

Using the above results, we propose Algorithm 1 to choose, given a dataset described by the matrix X consisting of m traces of n sensors, and a desired number of sensors $n^* < n$, a sub-optimal sensor selection from the point of view of information gain maximization. Given any integer $n \in \mathbb{N}$, denote by $\mathbf{n} = \{1, 2, \ldots, n\}$ the set of integers up to n.

Algorithm 1 Entropy-based Subset Selection

Inputs: Dataset $\mathcal{X} \in \mathbb{R}^{m \times n}$; Variable to predict $i \in \mathbf{n}$; Desired number of sensors $n^* < n$; Parameter θ.
Output: Set of sensors indices $S \subset \mathbf{n}$, $|S| = n^*$.
Initialization: $S := \{i\}$
for $k = 1, \ldots, n^* - 1$ **do**
  $j^* := \underset{j \in \mathbf{n} \setminus S}{argmin}\ \hat{H}(Z_{ij}|\beta_{ij})$
  $S := S \cup \{j^*\}$
end for

Applying the e-ss algorithm to our experimental use cases in Sections 3.3.2, 3.3.3, 3.3.4 evidences the following trade-off: when θ is chosen very close to 1 then clearly the e-ss and c-ss algorithms provide the same sensor selection in the range $r_{ij}^2 \in [1 - \frac{\theta^2}{2}, 1]$, while when θ is chosen much larger than 1 e-ss does not provide any benefit. In all our experimental datasets, the improvement of e-ss with respect to the c-ss occurs for values of θ around 1.05, namely when the random variable Z_{ij}

represents the event that the prediction error on X_i given measurements of X_i, X_j does not exceed 5% of the variance of X_i. Given a specific dataset, the problem of solving the above tradeoff and choosing the optimal value of θ such that e-ss outperforms c-ss is an interesting venue for future work.

3.3 Results and Numerical Simulations

In this section, we first validate the prediction accuracy using PCA based approaches in 3 cases: (1) using all sensor components; (2) using sensors selected via c-ss; and (3) using sensors selected via e-ss. Also, we will apply a combinatorial search over all subsets of sensors to obtain the set that provides the best achievable prediction accuracy (opt-ss). In our setup, this search can be done in a reasonable time: in general, this combinatorial search is impractical and only simpler sensor selection methods, such as c-ss and e-ss, can be used. We will show that the e-ss outperforms the c-ss.

The prediction accuracy will be measured using the Normalized Root Mean Square Error (NRMSE). We will validate our damage detection and sensor selection methodologies on 3 experimental datasets provided from the repository of the Engineering Institute at the Los Alamos National Laboraratory (LANL) [20], and show that it outperforms the results obtained using a standard PCA algorithm proposed in [9] over the same dataset.

3.3.1 Damage Detection Thresholds

To detect damages one can check whether the run-time trajectories of the $\boldsymbol{\alpha}_j(k)$ vectors in (3.3) change with respect to the nominal initial conditions(i.e., whether the AR coefficients change due to a change in the physics of the system). However, the choice of a threshold to decide whether the noisy change of $\boldsymbol{\alpha}(k)$ corresponds to a faulty event is in general application-dependent [7, 54]. In this case, we define a scalar residual vector

$$\gamma(k) = ||\boldsymbol{\alpha}(k)||_2, \tag{3.17}$$

with $\boldsymbol{\alpha}(k) = [\boldsymbol{\alpha}_1(k), \ldots, \boldsymbol{\alpha}_n(k)]$, and bounds:

$$\gamma_u = (1+b)||\gamma(0)||_2, \quad \gamma_l = (1-b)||\gamma(0)||_2, \tag{3.18}$$

where $b \in [0, 1]$. We consider a system nominal when $\gamma(k)$ does not exceed the nominal value of more than $b\%$

$$\gamma_l \leq \gamma(k) \leq \gamma_u, \tag{3.19}$$

and faulty otherwise. In particular, we also define the *escape time* t_e as the minimum time instant such that the residual signal $\gamma(k)$ leaves the *nominal zone* bounded by $\gamma_l and \gamma_u$, and never returns back again to a nominal value, i.e.,

$$t_e \doteq \min\{t^* \geq 0 : (\forall t \geq t^*, \gamma(t) \notin [\gamma_l, \gamma_u])\}.$$

Notice that it is possible for t_e to exit the bound and then returned into it after a certain amount of time. In this case, we categorize this behavior as there exist a change of dynamics in the system, however, it is not yet declared as a fault.

The choice of $b\%$ purely depends on the application, and often heuristic methods are used in practice taking into account that large values of $b\%$ can avoid false alarms, but can either increase the time to detect a damage or even fail to detect it.

3.3.2 Bookshelf Resembling Structure

This accelerometric dataset is taken from the repository of the Engineering Institute at the Los Alamos National Laboraratory (LANL)[1, 20]. The frame structure, which resembles a 3-tiers bookshelf, was built from unistrut columns and aluminum plates. Each of the shelves, which is 1.3 cm thick aluminum plate, is mounted on the unistrut column with 2-bolts connection brackets. In each tier, 8 accelerometers (2 per joint) were installed on the plate and column. Figure 3.1 illustrates the structure and accelerometers placement for each tier. To generate the data, the structure was vibrated with frequency ranging in 0-3 KHz for 8 seconds, producing 4096 data samples. Several experiments in the nominal configuration were repeated, generating nominal (or baseline) datasets. Then, 3 different types of

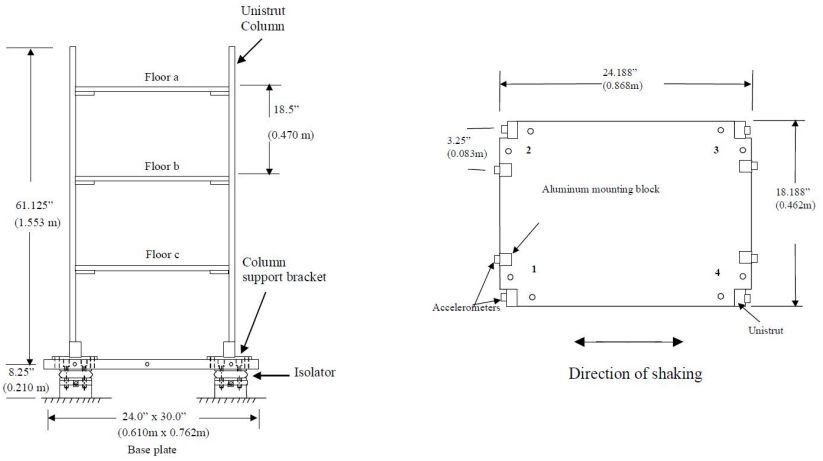


Figure 3.1: Assembled LANL test structure similar to a 3-tier bookshelf.

damage were induced in the structure. Each of them is characterized by a different location of the damage. Specifically, in case 1, a damage was induced on joint 2a; in case 2, a damage was induced on joint 4b and in case 3, a damage was induced both on joints 2a and 4b. To verify our fault detection algorithms capability to detect damages, we will consider all 3 cases of damage provided by LANL. In particular, we will train our models with 3000 samples of the nominal behavior. We will use 1000 samples of the nominal behavior and 4000 samples of each of the 3 damages to validate respectively the prediction accuracy, and the damage detection sensitivity of our algorithms, comparing them with previous work on the same dataset [9].

Predictive Model Accuracy

Table 3.2 summarizes the accuracy of the PCA-based and RT-based predictive models respectively, where the i-th row of each table indicates (1 - NRMSE) $\times 100\%$ for predicting sensor component i using respectively all sensors, the sensor subset obtained using the c-ss, the sensor subset obtained using the e-ss and the optimal subset obtained via permutation. The last row indicates the average prediction accuracy among all 24 components. For this case, we considered a selection of sets of sensors with cardinality $n^* = 5$ as it happens to be the best trade-off between prediction accuracy and computational complexity. Of course, this choice can be adjusted to suit specific application needs.

Table 3.2: (1 - NRMSE) ×100% and sensor selection sets, PCA-based model

all PCA	c-ss PCA	e-ss PCA	opt-ss PCA
91.88	95.36*	96.35*	97.87
91.03	92.65	95.07*	97.08
89.82	92.50	95.04*	97.21
91.18	93.15	95.95*	97.10
92.20	94.71	94.94*	97.78
90.87	92.53	94.87*	97.12
89.90	92.61	95.20*	97.11
90.48	92.17	95.29*	96.46
93.25	95.60	96.50*	96.93
94.96	96.69*	96.42	98.22
91.94	94.46	95.55*	96.24
93.15	93.97*	92.18	97.20
92.67	95.15	96.46*	97.41
95.35	96.37*	96.01	98.38
91.44	94.24	94.99*	95.71
93.59	94.14*	92.61	97.31
90.40	94.04	95.24*	95.65
91.16	93.02	95.09*	96.57
95.62	95.94	95.98*	96.79
96.18	96.73*	96.21	97.35
92.36	93.98	95.18*	95.88
90.99	94.75	94.88*	96.65
95.42	95.65	96.22*	96.74
95.80	96.16*	95.47	97.33
92.57	**94.44**	**95.32***	**96.96**

*best accuracy among all, c-ss and e-ss

Table 3.2 shows that both c-ss and e-ss PCA algorithms always achieve better prediction accuracy with respect to the use of all 24 sensors. This result is somehow related to the linearity problem in PCA: indeed, using PCA each principal component needs to be written as a linear combination of the original variables, but if the linearity between original variables does not exist PCA will not work properly. By introducing sensor selection into the standard PCA algorithm the linearity in the model is maintained by choosing the more linearly correlated components, thus increasing the prediction accuracy. In particular, the accuracy of e-ss PCA is larger than that of c-ss in almost all cases (up to +3% in accuracy) and in the average

Table 3.3: Escape time for different damages and techniques.

Escape time t_e	all PCA	c-ss PCA	e-ss PCA
Damage 1	3.68	3.51	2.93
Damage 2	3.74	4.16	3.19
Damage 3	5.73	2.41	2.37

(+1% in accuracy, reducing the optimality gap of 50%): as widely discussed in the introduction, our interpretation is that the e-ss is indeed able to select the components that add more information for the predictive model. With respect to the computational time, e-ss PCA is characterized by more steps compared to the standard PCA and c-ss PCA. However these extra steps, which consist of the calculation of the approximate entropy followed by finding the minimum of it, are solvable within time complexity of $O(k \cdot n)$, where $k = n^* - 1$ [55]: therefore, e-ss improves the prediction accuracy and does not increase significantly the computational complexity.

Damage Detection Sensitivity

In this section, we compare the sensitivity of damage detection algorithms using the predictive models introduced in previous section. As discussed above, we consider 3 datasets provided by LANL, each consisting of 4000 samples, corresponding to 3 different damages. Following the methodology described in Section 3.1, we first derive models on the nominal dataset and then apply the Kalman filter to generate a residual signal $\gamma(k)$: when such signal violates some predefined bounds, we raise a damage detection. In our simulations we have chosen $b = 0.5\%$ for all methods after a trial-and-error calibration process.

Figures 3.2 plots the trajectories of $\gamma(k)$ in nominal and faulty cases using respectively PCA-based predictive models as described above, and simulations show that damages are detected by all methods.

Table 3.3 clearly shows that e-ss PCA performs better than c-ss. More precisely, e-ss PCA reduces the escape time w.r.t. c-ss PCA up to 23%.

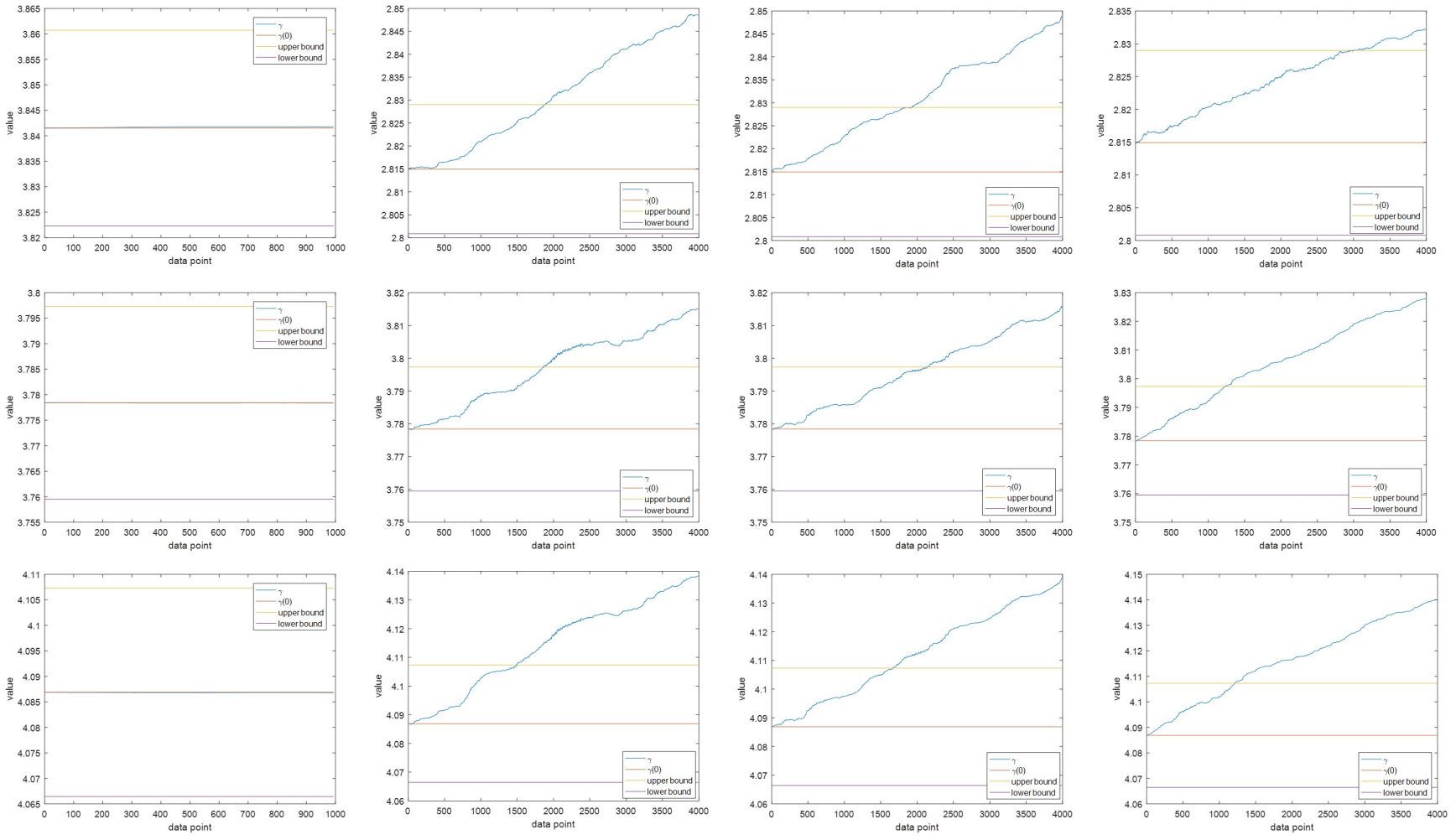


Figure 3.2: Trajectories of $\gamma(k)$: 1st column: nominal case, 2nd-4th column: faulty cases $1-3$; 1st row: all PCA, 2nd row: c-ss PCA, 3rd Row: e-ss PCA.

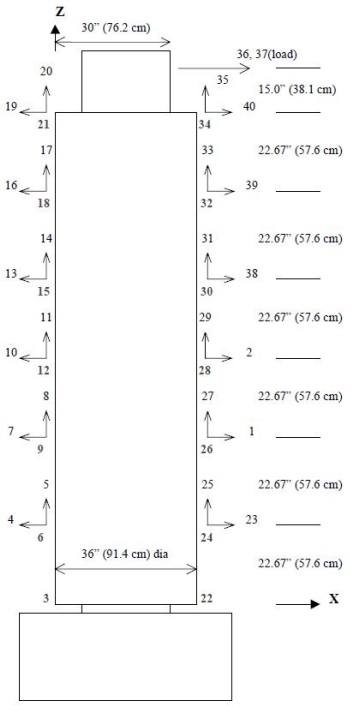


Figure 3.3: Accelerometers location on UCI bridge column.

3.3.3 UCI Bridge Column

This dataset is taken from an experimental setup performed by University of California, Irvine (UCI) on a bridge column [20, 56]. The structure consists of two concrete bridge columns with diameter of 0.61 m retrofitted to columns of diameter 0.91 m. Both columns are 3.45 m in length and cast on the top of concrete foundations 0.635 m tall, and 40 accelerometers are assigned to the structures to measure the dynamics of the columns. Figure 3.3 illustrates the structure and the accelerometers placement on the bridge's column. The structure was vibrated by a shaker with frequency between 0-400 Hz for 8-s, and produced 2048 samples. The experiment was repeated twice, thus creating 2 nominal datasets. Faulty datasets, containing data collected from the damaged structure, were generated with the same procedure as above. More precisely, an hydraulic actuator was used to apply lateral loads to the top of the column before the vibration process. The loads were applied in a force-controlled manner to produce lateral deformations at the top of the column corresponding to $0.15\Delta y, 0.25\Delta y, 0.4\Delta y$ and $0.7\Delta y$, where

Δy denotes the deformation with respect to the y axis of the column. These experiments were repeated multiple times generating 8 faulty datasets containing 2048 samples each. To verify our fault detection algorithms capability to detect damages we considered 3 cases of damage provided by LANL: they consist on $0.25\Delta y, 0.4\Delta y$ and $0.7\Delta y$ lateral deformation as in [56]. We trained our models using 2000 samples of the nominal behavior (1000 samples from each nominal dataset). We used the remaining 2096 samples from the nominal behavior (1048 samples from each dataset) to validate the prediction accuracy of our models. Finally, for the damage detection sensitivity of our algorithms we used 2048 samples from each of the 3 faulty datasets. We considered the datasets with $0.25\Delta y$ as damage level 1, which is the lowest, $0.4\Delta y$ as damage level 2, and $0.7\Delta y$ as damage level 3 (the higher the level, the stronger the damage).

Predictive Model Accuracy

We proceed as illustrated above, considering a selection of sets of sensors with cardinality $n^* = 5$. Tables 3.4 summarizes the accuracy of the PCA-based predictive models respectively. As above, the last row indicates the average prediction accuracy among all 40 components.

Table 3.4 shows that both c-ss and e-ss PCA algorithms always achieve better prediction accuracy with respect to using all 40 sensors. Also in this case, the accuracy of e-ss PCA is larger or almost equal to c-ss in almost all cases (up to +14% in accuracy) and in the average (+3.5% in accuracy, reducing the optimality gap of 50%).

Damage Detection Sensitivity

Figure 3.4 plots the trajectories of $\gamma(k)$ with $b = 0.5\%$ for all methods in nominal and faulty cases using PCA-based predictive model as described above, and simulations show that damages are detected by almost all methods.

Table 3.5 clearly shows that, for damages 2 and 3 (the strongest damages), all methods work well, without substantial differences. Regarding Damage 1 (the weakest damage and, therefore, the most difficult to be detected), c-ss PCA fails to

Table 3.4: 100 - NRMSE% and sensor selection sets, PCA-based model

all PCA	c-ss PCA	e-ss PCA	opt-ss PCA
86.04	94.21*	92.75	96.02
90.12*	90.5*	90.16*	93.14
83.23	87.07	90.38*	94.57
74.31	93.01*	92.63*	95.2
81.38	82.99*	82.31*	93.19
81.95	93.38	94.5*	95.62
90.08	94.39*	94.99*	96.31
18.09	74.94	81.82*	91.84
91.84	96.09*	95.84*	96.12
88.6	95.82*	94.81	96.68
20.46	82.04	89.28*	92.64
94.64	96.77*	93.65	97.15
89.51	95.96*	95.11*	96.89
63.95	85.26	86.42*	91.77
83.84	95.64*	95.74*	97.04
91.45	94.66	95.45*	96.6
55.5	75.48	89.56*	92.02
93.9	95.24	96.68*	97.15
86.08	94.17*	93.55*	96.4
34.87	76.76	88.3*	89.67
91.79	93.81*	92.69	97.09
77.86	86.5*	86.5*	94.47
77.34	92.5*	92.13*	94.53
90.63	92.79*	91.56	95.11
24.44	78.58*	77.74*	92.4
92.65	94.77*	94.77*	96.21
63.29	87.88*	87.76*	93.25
92.5	96.25*	96.27*	97.01
70.63	92.04*	85.98	94.26
96.08	97.35*	95.75	97.52
84.57	93.4*	92.65*	93.93
94.89*	95.99*	93.72	97.15
70	91.42*	89.22	93.99
93.41*	93.63*	93.63*	96.46
40.68	86.66	92.38*	94.23
86.34	90.38*	89.79*	92.93
91.95*	89.9	90.89*	93.83
85.95	95.37*	95.01*	96.74
85.58	88.24*	86.6	91.27
87.95	88.3*	88.66*	92.14
77.46	**87.50**	**91.05**	**94.76**

*best accuracy among all, c-ss and e-ss

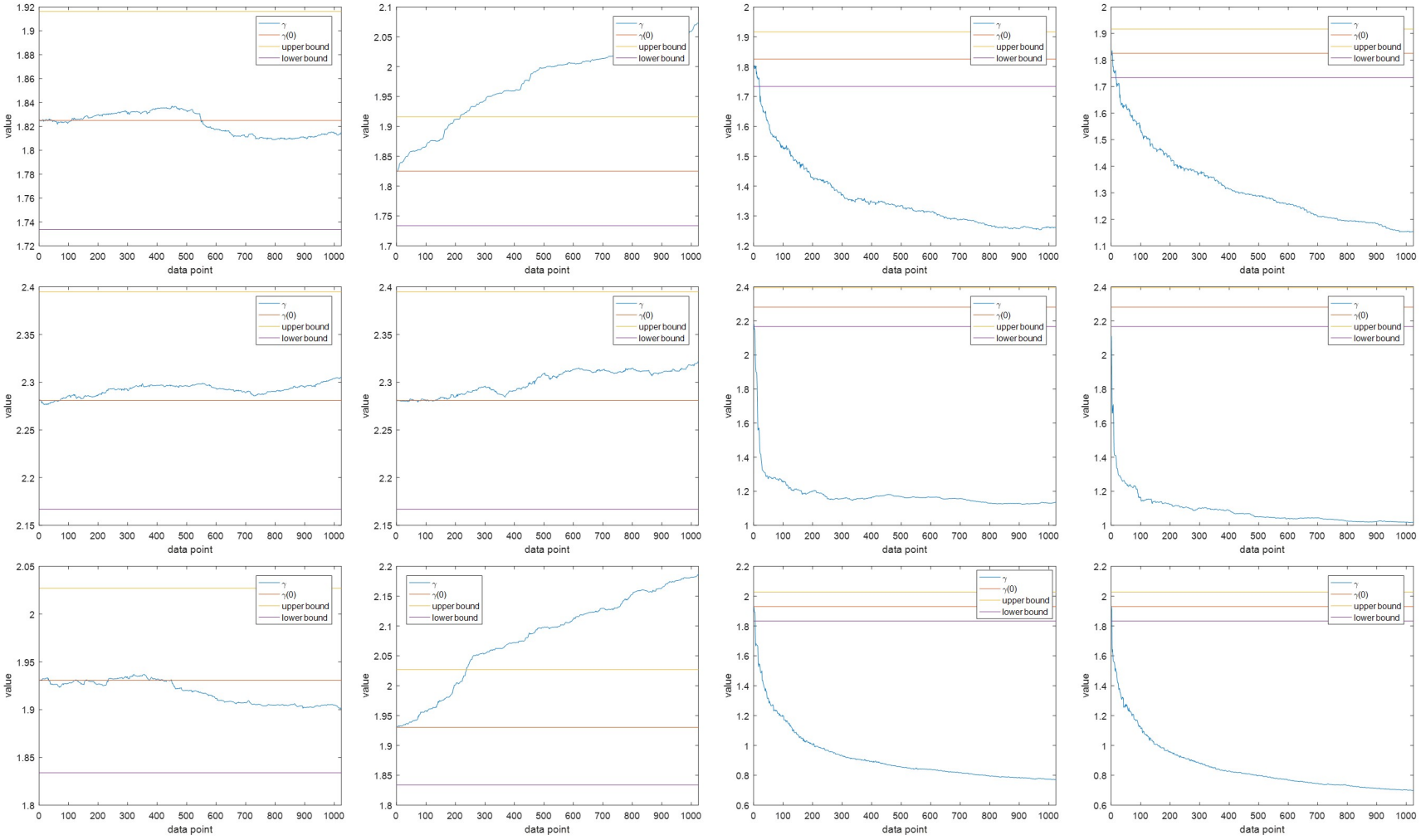


Figure 3.4: Trajectories of $\gamma(k)$: 1^{st} column: nominal case, 2^{nd}-4^{th} column: faulty cases $1-3$; 1^{st} row: all PCA, 2^{nd} row: c-ss PCA, 3^{rd} Row: e-ss PCA.

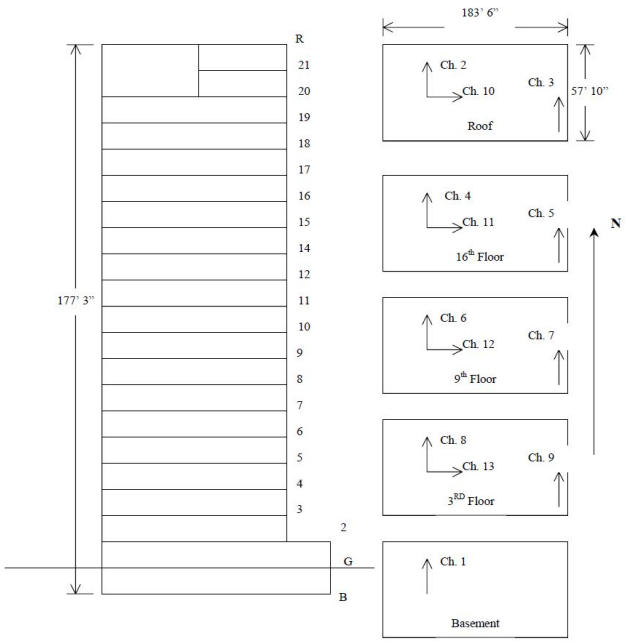


Figure 3.5: Accelerometers Placement Inside Sheraton Hotel.

detect the damage, while all PCA and e-ss PCA are characterized by a similar and small escape time. Since e-ss PCA uses only 5 sensors and all PCA uses 40 sensors, we can state that the e-ss PCA outperforms all other methods in detecting the weak Damage 1 for this specific case. This is crucial as, very often, strong damages in structures start from weak damages: being able to detect subtle damages is a huge advantage to prevent the occurrence of strong damages and catastrophic events.

3.3.4 Sheraton Hotel

The dataset is obtained from LANL [20] teams who measured the vibrational responses on the Sheraton Hotel at Universal City in North Hollywood, California.

Table 3.5: Escape time for different damages and techniques.

Escape time t_e	all PCA	c-ss PCA	e-ss PCA
Damage 1	0.88	-	0.91
Damage 2	0.04	0.04	0.03
Damage 3	0.02	0.01	0.01

Table 3.6: 100 - NRMSE% and sensor selection sets, PCA-based model

all PCA	c-ss PCA	e-ss PCA	opt-PCA
13.07	84.12*	74.1	88.38
87.67*	85.18	84.36	88.75
88.24	89.56*	90*	94.64
85.76*	62.7	83.35	88.31
80.49*	76.64	77.41	89.86
85.48*	84.36	84.41	87.6
86.83*	87.29*	87.29*	88.6
82.23	84.79*	84.79*	87.48
87.18	88.77*	88.77*	89.34
85.59	88.04*	88.04*	88.13
83.43	67.86	87.28*	88.17
86.78	89.72*	89.72*	91.41
87.67*	87.53*	87.45*	89.25
79.30	**79.74**	**81.85***	**89.22**

*best accuracy among all, c-ss and e-ss

The hotel is a twenty-one story ductile moment-resisting concrete frame structure that has permanent instrumentation installed on it. There are 13 accelerometers mounted on various locations inside the building, as shown in Figure 3.5. The accelerometers read the signal caused by background vibration sources such as wind, traffic, and mechanical equipment in the structure, with frequency recorded ranging from 0 to 25 Hz for various amount of times (32, 64, 128, 256, and 512 seconds). Such measurements generated 10 different datasets, each consisting of 4096 samples. We will consider 2 of the 10 datasets, i.e. the ones vibrated with frequency range of 0-6.25 Hz for 256 seconds, and create a unique dataset of 8192 samples. We will use 6144 samples to train the model, and use the remaining 2048 samples for the validation process. Since in this experimental setup there is no availability of datasets corresponding to a damage, we only validate the predictive model accuracy. We did it under the consideration of understanding the model predictive quality of the proposed algorithm on a real building dataset.

Predictive Model Accuracy

We proceed as illustrated above, considering a selection of sets of sensors with cardinality $n^* = 3$ due to the limited number of available accelerometers. Tables 3.6 shows the accuracy of the PCA-based predictive models. As above, the last row indicates the average prediction accuracy among all 13 components.

Table 3.6 shows that both c-ss and e-ss PCA algorithms always achieve better prediction accuracy with respect to all PCA. Also, in this case, the accuracy of e-ss PCA is larger or almost equal to c-ss in almost all cases and in the average. For the prediction of each specific sensor, the optimal sensor selection, which is combinatorial and thus impractical when the number of sensors is very large, largely outperforms e-ss and c-ss. Our interpretation is that e-ss and c-ss assume that the relation of the vibrational dynamics recorded among each pair of sensors is linear. That said, they seem to fail in the Sheraton Hotel dataset because, being it much more complex than the previous structures, the relation of the vibrational dynamics among some subsets of sensors is nonlinear.

3.4 Discussion

We considered experimental setups in structural damage detection, and provided a novel algorithm based on information theory to select a subset of sensors in order to improve the predictive model accuracy. We demonstrated on an experimental dataset that the combination of our algorithms outperforms other approaches based on PCA and c-ss, obtaining a model prediction accuracy improvement from 3% up to 7%, improving the damage detection sensitivity up to 59%, and reducing the number of accelerometers up to 80%.

4

Regression Trees-based identification for structural damage detection

Contents

4.1 RT-based Damage Detection 48
4.2 Results and Numerical Simulations 49
4.2.1 Bookshelf Resembling Structure 49
4.2.2 UCI Bridge Column 51
4.2.3 Sheraton Hotel 53
4.3 Discussion 57

This section is the extension of Chapter 3 in which, the results presented in this chapter are currently under preparation for journal submission.

A good model is a good representation to the system. In order to develop a sensitive fault detection algorithm, it is necessary to have a model which can explain the dynamic of a system with little to no error. There are numerous algorithms which are able to generate precise models to represent the system, one among them, is the Regression Tree (RT) algorithm. In this chapter, we will leverage the RT algorithm proposed by [17–19] to perform the task of faults detection. The fault detection itself will follow a similar formulation as described in Chapter 3. However this time, the model which needs to be evaluated will be a piecewise linear model.

We will show that the RT based approach can also detect faults with high accuracy and even outperforms the PCA in some cases.

4.1 RT-based Damage Detection

In this section, we present an idea on how to perform faults detection leveraging the RT algorithm presented in [17–19]. To apply such identification technique in our context, we have to extend the dataset identified by the matrix X into an extended dataset $\mathcal{D}_x$ by adding the regressive terms of the measurements as features, i.e.,

$$\mathcal{D}_x = \{(x(k)^\top, x(k-1)^\top, \dots, x(k-\tau)^\top)\}_{k=1,\dots,m}, \tag{4.1}$$

where $x(k) = [x_1(k), \dots, x_n(k)]^\top \in \mathbb{R}^n$.

In the method proposed in [17–19], differently from CART, an AR model is fitted in each leaf ℓ using only the samples belonging to the corresponding hyper-rectangle R_ℓ:

$$x_j(k) = \sum_{i=1}^{\tau} \alpha^0_{ij,\ell(\bar{\mathbf{x}}(k))} x_j(k-i), \tag{4.2}$$

where $\bar{\mathbf{x}}(k) = [x(k-1)^\top \ \cdots \ x(k-\tau)^\top]$, and $\ell(\bar{\mathbf{x}}(k)) = \{\ell : \bar{\mathbf{x}}(k) \in R_\ell\}$ assigns to the state measurements $\bar{\mathbf{x}}(k)$ the corresponding leaf in $\mathcal{T}$.

We can now leverage this modeling to extend the previous section, defining for each component j a dynamical model of the AR terms:

$$\begin{aligned} \boldsymbol{\alpha}_j(k+1) &= I\boldsymbol{\alpha}_j(k) + \omega(k) \\ x_j(k) &= C(\bar{\mathbf{x}}(k))\boldsymbol{\alpha}_j(k) + \nu(k), \end{aligned} \tag{4.3}$$

where:

$$\boldsymbol{\alpha}_j(k) = [\alpha_{1j,1}(k) \ \dots \ \alpha_{\tau j,1}(k) \ \dots \ \alpha_{1j,L}(k) \ \dots \ \alpha_{\tau j,L}(k)]^\top$$

is a state vector with initial condition

$$\boldsymbol{\alpha}_j(0) = [\alpha^0_{1j,1} \ \dots \ \alpha^0_{\tau j,1} \ \dots \ \alpha^0_{1j,L} \ \dots \ \alpha^0_{\tau j,L}],$$

$C(\bar{\mathbf{x}}(k))$ is a time varying observation matrix defined by:

$$C(\bar{\mathbf{x}}(k)) = [\bar{\mathbf{x}}(k)\delta_{1,\ell(\bar{\mathbf{x}}(k))} \ \cdots \ \bar{\mathbf{x}}(k)\delta_{N,\ell(\bar{\mathbf{x}}(k))}],$$

with $\delta_{.,.}$ the Kronecker delta, $\omega(k)$ and $\nu(k)$ uncorrelated white noise signals with variance σ_ω and σ_ν respectively, L is the number of leaves and I the identity matrix of appropriate dimension. Note that in this case we are not using the PCA transformation as in (3.1), and $\bar{\mathbf{x}}(k)$ is directly measurable. With few modifications with respect to the previous section, a Kalman filter can be easily constructed following the explanation in Section 2.6 to estimate the time evolution of the AR parameters. In particular, the Kalman filter solution for (4.3) is given as a series of computation as following:

$$
\begin{aligned}
&\boldsymbol{\alpha}_j(k|k-1) = \boldsymbol{\alpha}_j(k-1|k-1),\\
&\boldsymbol{P}_{\boldsymbol{\alpha}}(k|k-1) = \boldsymbol{P}_{\boldsymbol{\alpha}}(k-1|k-1) + I.\sigma_\omega^2,\\
&x_j(k|k-1) = C(\bar{\mathbf{x}}(k))\boldsymbol{\alpha}_j(k|k-1),\\
&S(k) = C(\bar{\mathbf{x}}(k))\boldsymbol{P}_{\boldsymbol{\alpha}}(k|k-1)\bar{\mathbf{x}}(k))^\top + \sigma_\nu^2,\\
&\tilde{R}(k) = x_j(k) - x_j(k|k-1),\\
&K_G(k) = \boldsymbol{P}_{\boldsymbol{\alpha}}(k|k-1)C(\bar{\mathbf{x}}(k))^\top S(k)^{-1},\\
&\boldsymbol{\alpha}_j(k|k) = \boldsymbol{\alpha}_j(k|k-1) - K_G(k)\tilde{R}(k),\\
&\boldsymbol{P}_{\boldsymbol{\alpha}}(k|k) = [1 - K_G(k)C(\bar{\mathbf{x}}(k))]\boldsymbol{P}_{\boldsymbol{\alpha}}(k|k-1),
\end{aligned}
\tag{4.4}
$$

4.2 Results and Numerical Simulations

In this section, in a similar manner as we have done in Section 3.3, we also validate the prediction accuracy using RT based approaches for 3 cases: (1) using all sensor components; (2) using sensors selected via c-ss; and (3) using sensors selected via e-ss. The prediction accuracy will also be measured using the Normalized Root Mean Square Error (NRMSE), and we will validate our damage detection and sensor selection methodologies on same experimental datasets as in 3.3.2, 3.3.3 and 3.3.4.

4.2.1 Bookshelf Resembling Structure

As in Section 3.3.2, we compare the model prediction accuracy and sensitivity of damage detection algorithms using the predictive models introduced in Section 4.1.

Following the methodology in Section 4.1, we first derive models on the nominal dataset and then apply the Kalman filter to generate a residual signal $\gamma(k)$: when such signal violates some predefined bounds, we raise a damage detection. In this simulation, we also choose $\beta = 0.5\%$, to have a fair comparison with the PCA based approach.

Predictive Model Accuracy

Table 4.1: (1 - NRMSE)×100% for the RT-based model (sensor selection sets are same as in Table 3.2)

all RT	c-ss RT	e-ss RT
92.03*	91.77	92.06
93.16*	92.6	91.08
91.37*	91.26	90.28
92.37*	92.29	90.76
92.8*	92.08	91.58
92.89	92.6*	91.11
91.99*	91.34	90.92
92.16*	92.02	91.13
91.58	91.67	92.07*
91.69	92.18*	91.7*
90.54	90.69*	90.32
91.78	91.34	92.07*
92.39	91.64	92.48*
92.46*	91.6	92.24
90.21	90.62	90.8*
91.21	91.4*	91.13
90.51*	90.15	90.43
91.13	91.31*	91.17
91.79*	91.79*	92.43
91.43	91.49	91.73*
90.81*	90.26	90.5
91.43	91.56*	90.91
91.93	92.02*	91.26
91.82	91.61	91.86*
91.73*	**91.55**	**91.33**

*best accuracy among all, c-ss and e-ss

Table 4.1 shows that there is no significant improvement of the prediction accuracy using e-ss RT and c-ss RT with respect to all-RT. However, both e-ss

Table 4.2: Escape time for different damages and techniques.

Escape time t_e	all RT	c-ss RT	e-ss RT
Damage 1	3.02	6.31	1.23
Damage 2	1.96	4.90	0.51
Damage 3	1.28	2.17	0.50

RT and c-ss RT use less sensors (5 instead of 24), yet able to reach a similar level of accuracy. This is an interesting result as in a real life problems, this can be a reference on where a set of accelerometers needs to be placed. For example, in this case, one may choose 2 subsets and assign a maximum of 10 accelerometers inside the building, instead of 24, which will be more cost efficient (i.e., removes the other 14 sensors after collecting the data in order to reduce the maintenance cost).

Damage Detection Sensitivity

Figure 4.1 plots the trajectories of $\gamma(k)$ in nominal and faulty cases using RT-based predictive model as described above, and simulations show that damages are detected by all methods.

Table 4.2 suggests that e-ss RT-based models perform better than the c-ss RT. It can be seen that e-ss RT reduces the escape time w.r.t. c-ss RT up to 88%. It is interesting to note that e-ss RT, whose prediction accuracy is smaller than the PCA-based methods (as in Section 3.3.2), outperforms all other methods in terms of Damage Detection Sensitivity (see table 3.3) our interpretation of this fact is that the nonlinear dynamics of the RT-based model is less accurate but more responsive to sudden changes in the structure.

4.2.2 UCI Bridge Column

The dataset used in this section is the same as in Section 3.3.3. In this section, we consider a selection of sets of sensors with cardinality $n^* = 5$ and the same bound with the PCA-based technique which is 0.5%.

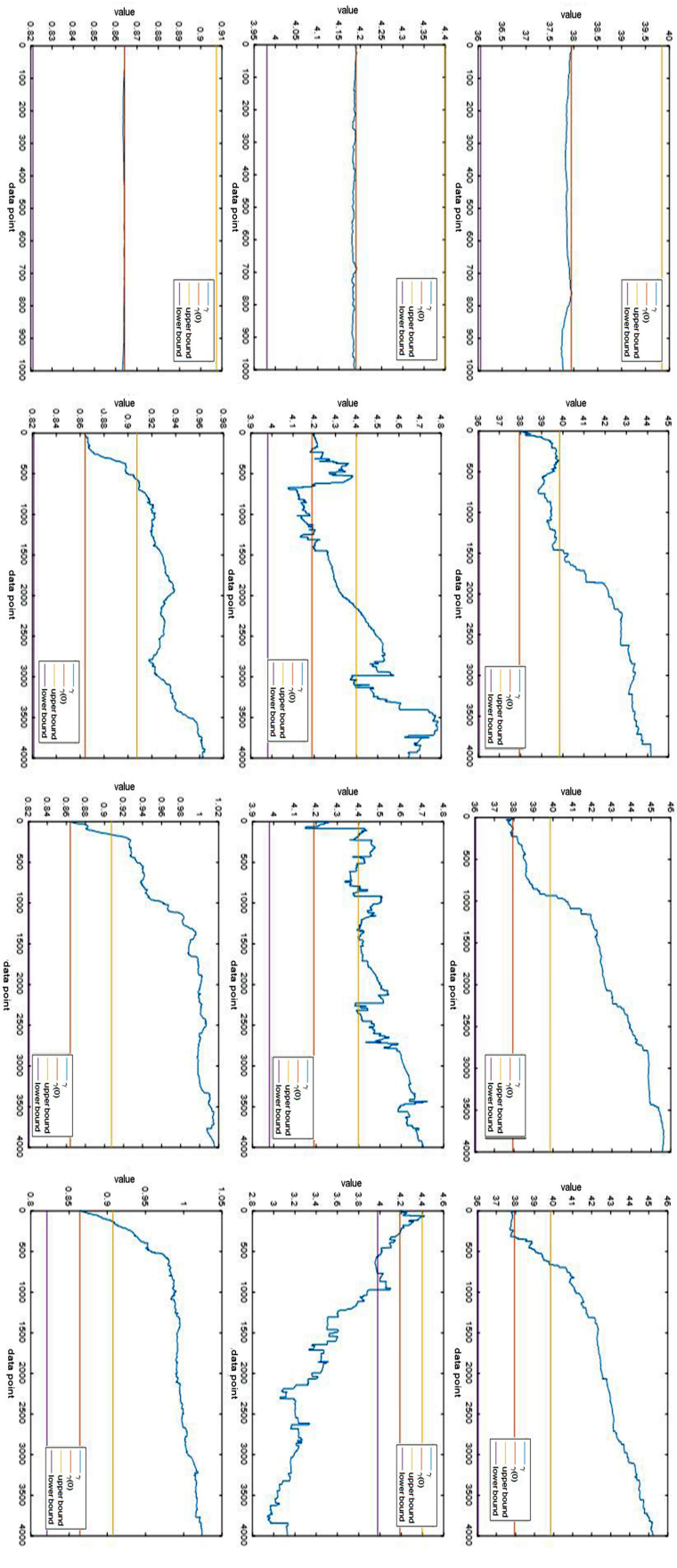


Figure 4.1: Trajectories of $\gamma(k)$: 1st column: nominal case, 2nd–4th column: faulty cases $1-3$; 1st row: all RT, 2nd row: e-ss RT, 3rd Row: e-ss RT.

Predictive Model Accuracy

Table 4.3 shows the model predictive accuracy of RT algorithm in predicting the dynamics of accelerometers inside the structure. As in previous section, the last row indicates the average prediction accuracy among all 40 components. Table 4.3 reveals that, the RT-based nonlinear models perform better than the PCA-based linear models (except the opt-ss) as shown in Section 3.3.3. Furthermore, the all-RT has higher prediction accuracies in average w.r.t the all-PCA, while it performs worse than c-ss and e-ss RT. This result was anticipated, as the linearity was never a problem for RT algorithm.

Damage Detection Sensitivity

Figure 4.2 plots the trajectories of $\gamma(k)$ in nominal and faulty cases using and RT-based predictive models as described above, and simulations show that damages are detected by almost all methods.

Table 4.4 suggests that, all methods work, without substantial differences. In this case, the all RT is characterized by a very large escape time for damage 1, while e-ss RT are characterized by a similar and small escape time. Since e-ss RT uses only 5 sensors and all RT uses 40 sensors, it is fair to say that e-ss RT outperform all other methods in detecting the weak damage 1.

4.2.3 Sheraton Hotel

The dataset used in this section is the same dataset as in Section 3.3.4. Similarly with its PCA counterpart, we consider a selection of sets of sensors with cardinality $n^* = 3$ and threshold bound of 0.5%. As before, only nominal dataset existed in the repository. Thus, the analysis in this subsection will be limited in just the predictive model accuracy.

Predictive Model Accuracy

Table 4.5 shows the accuracy of the RT-based predictive model. As above, the last row indicates the average prediction accuracy among all 13 components.

Table 4.3: (1 - NRMSE)×100% for the RT-based model (sensor selection sets are same as in Table 3.4)

all RT	c-ss RT	e-ss RT
87.77	91.9*	91.42*
89.92	92.28*	92.27*
87.41	90.97*	90.97*
87.04	90.94*	90.75*
88.23	91.37*	91.92*
87.84	91.78*	91.36*
88.7	92.88*	92.83*
87.96	90.64*	90.97*
88.77	91.93*	91.84*
88.62	93.39*	93.66*
89.17	91.74*	91.74*
89.58	92.8*	92.8*
89.45	94.56*	94.47*
87.81	90.14*	89.9*
89.2	93.02*	93.42*
89.23	94.57*	94.64*
87.97	90.49*	90.25*
88.49	92.7*	93.54*
90.79	94.28*	94.06*
86.01	88.54*	87.71*
88.48	92.61*	92.7*
87.43	91.09*	91.09*
87.92	91.35*	90.8*
87.31	91.11*	90.65*
88.11	90.69*	90.7*
85.95	90.86*	90.86*
88.14	91.02*	90.97*
89.07	91.37*	91.7*
87.77	91.32*	91.61*
89.43	92.45*	92.41*
88.87	91.84*	92.4*
88.58	92.36*	92.36*
88.97	92.11*	92.2*
88.64	92.81*	92.81*
90.15	92.28*	92.81*
88.03	91.35*	91.43*
88.87	91.84*	91.84*
89.09	94.58*	94.47*
87.68	90.25*	90.26*
89.03	91.52*	91.47*
88.43	**91.83***	**91.70***

*best accuracy among all, c-ss and e-ss.

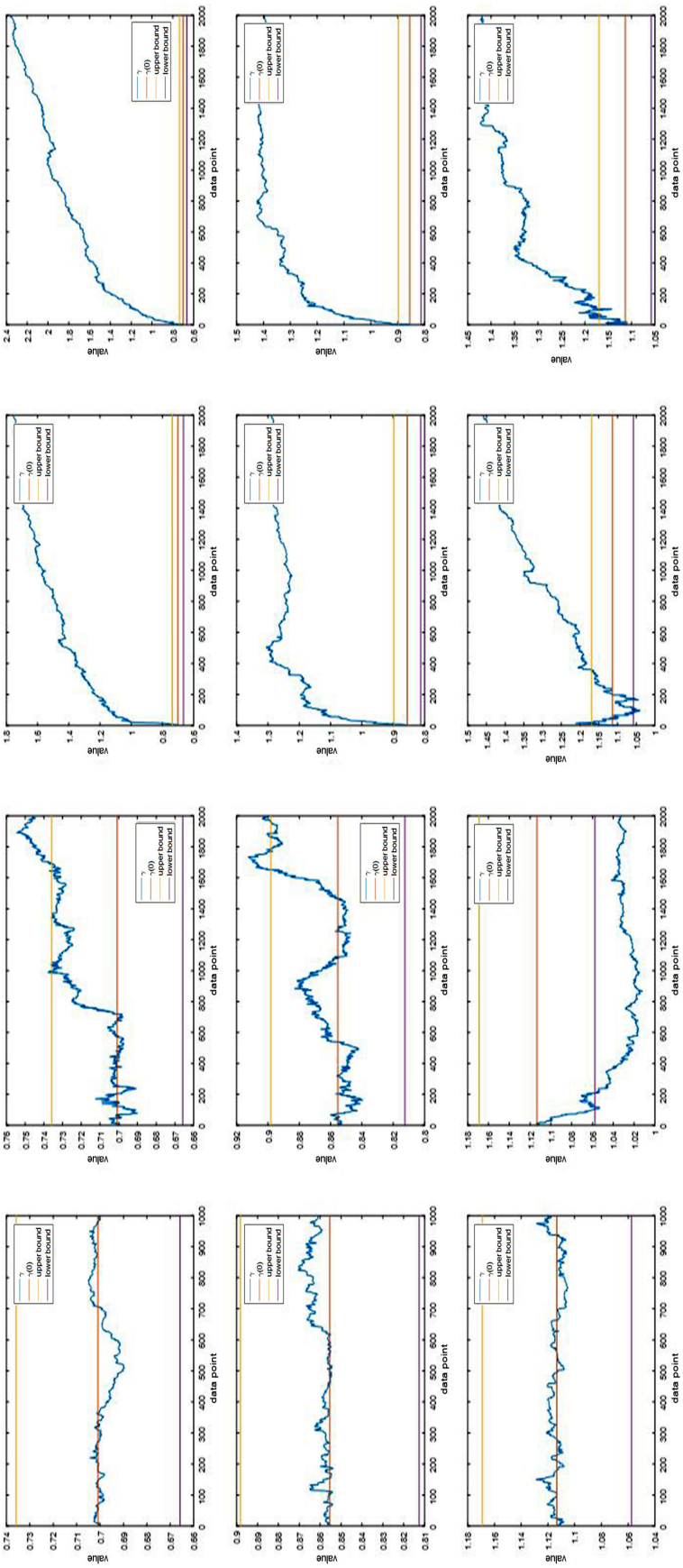


Figure 4.2: Trajectories of $\gamma(k)$: 1$^{\text{st}}$ column: nominal case, 2$^{\text{nd}}$-4$^{\text{th}}$ column: faulty cases $1-3$. 1$^{\text{st}}$ row: all RT, 2$^{\text{nd}}$ row: c-ss RT, 3$^{\text{rd}}$ Row: e-ss RT.

Table 4.4: Escape time for different damages and techniques.

Escape time t_e	all RT	c-ss RT	e-ss RT
Damage 1	6.76	6.52	0.82
Damage 2	0.05	0.05	1.55
Damage 3	0.02	0.02	0.26

Table 4.5: (1 - NRMSE)×100% for the RT-based model (sensor selection sets are same as in Table 3.6)

all RT	c-ss RT	e-ss RT	opt-RT
87.96*	88.12*	88.08*	88.58
90.01*	88.85	88.22	90.36
93.22*	93.64*	93.62*	93.83
88.67*	86.42	85.9	88.92
88.67*	87.91*	87.85*	89.12
90.01*	86.22	86.55	89.13
89.19*	88.59*	88.59*	88.84
88.74*	86.27	86.27	88.22
89.87*	89.93*	89.93*	90.44
86.76*	86.92*	86.92*	87.49
88.34*	87.07	86.53	88.99
89.96*	88.95	88.95	90.19
88.87*	87.85	86.74	89.54
89.56*	**87.63**	**87.94**	**89.51**

*best accuracy among all, c-ss and e-ss.

Interestingly, Table 4.5 shows that in the RT case all the 3 algorithms seem to perform very similarly, and this result outperforms the PCA-based models (+10% in average accuracy) as shown in Section 3.3.4. It is important to notice that, in terms of number of sensors, c-ss RT and e-ss RT use less sensors with respect all-RT (3 against 13), yet still able to perform up to the level of all-RT. In conclusion, the RT-based models perform better than PCA-models when the dataset presents nonlinearities, which can be very well captured by the nonlinear models derived exploiting the RT-based methodology in [17–19] with respect to the linear models obtained via PCA.

4.3 Discussion

We addressed the problem of structural damage detection, and proposed a novel damage detection methodology based on the combination of Kalman filtering with a model identification algorithm based on AR system identification and Regression Trees. All experimental setups show the capability of our novel RT-based approaches in predicting faults, which also outperform PCA-based models one. The damage detection validation in Sections 4.2.1 and 4.2.2 shows that e-ss RT performs better than c-ss and all-RT, in the prediction accuracy and the damage detection sensitivity while significantly reduce the number of sensors.

5

Poly-Exponential Based Fault Detection Identification Technique

Contents

5.1 Poly-Exponential Correction Term 60
5.2 Results and Numerical Simulations 61
5.2.1 Bookshelf Resembling Structure 62
5.2.2 UCI Bridge Column 65
5.2.3 Sheraton Hotel 68
5.3 Discussion 68

This section is the extension of Chapter 3 in which, the results presented in this chapter are currently under preparation for journal submission.

In this chapter, we will introduce the notion of Poly-Exponential (PE) correction term. The PE correction term is an extension to the e-ss PCA algorithm proposed in Chapter 3, which serves as the correction to the residuals generated by the PCA algorithm. In a simple way, the Poly-Exponential function can be considered as a polynomial function with base of natural numbers e. We will show that the e-ss PCA + PE correction term improves the performance of the e-ss PCA in both model predictive accuracy and fault detection.

5.1 Poly-Exponential Correction Term

In this section, we introduce a novel fault detection technique based on PE model. Let $X = [\mathbf{x}_1\ \mathbf{x}_2 \cdots \mathbf{x}_n] \in \mathbb{R}^{m\times n}$ be the matrix of sensor measurements consisting of vectors $\mathbf{x}_i = [x_i(1)\ \cdots\ x_i(m)]^\top \in \mathbb{R}^m,\ i = 1, \ldots, n$, where $x_i(k) \in \mathbb{R}$ is the measurement at instance k of the i^{th} sensor, n is the number of sensors (features), and m is the number of samples measured from each sensor. Furthermore, let $\hat{X} = [\hat{\mathbf{x}}_1\ \hat{\mathbf{x}}_2 \cdots \hat{\mathbf{x}}_n]$ be the estimate of matrix X obtained from e-ss PCA algorithm, $\Gamma = X - \hat{X} = [\mathbf{r}_1\ \mathbf{r}_2 \cdots \mathbf{r}_n]$ be the matrix of residual, with $\mathbf{r}_j = \mathbf{x}_j - \hat{\mathbf{x}}_j$, with $j = 1, 2, \cdots, n$, are the residual vectors.

Suppose that we fit a PE model for each of residual vectors:

$$\hat{r}_j(k) = \sum_{i=1}^{p} b_{ij} e^{l_j(\hat{x}_j(k) - C_j)^i} + D_j, \tag{5.1}$$

where p is the order of Poly-Exponential, e is the natural number, $b_{1j}, b_{2j}, ..., b_{pj}, l_j, C_j$ and D_j are the model parameters to be estimated. These model parameters i.e. can be estimated by using the LM algorithm as described in Section 2.5. Given (5.1), the new estimate to $\mathbf{x}_j$ namely $\tilde{\mathbf{x}}_\mathbf{j}$ is defined as

$$\tilde{x}_j = \hat{x}_j + \sum_{i=1}^{p} b_{ij} e^{l_j(\hat{x}_j(k) - C_j)^i} + D_j. \tag{5.2}$$

We can now extend (5.1) as in previous section to detect faults. As before, we assume that the nonlinear regressive terms in (5.1) remain constant in nominal condition. Let $\boldsymbol{\beta_j} = [b_{1j}, b_{2j}, ..., b_{pj}, l_j, C_j, D_j]$ be the state vector. Then the dynamical model for (5.1) is given by

$$\begin{aligned} \boldsymbol{\beta}_j(k+1) &= I\boldsymbol{\beta}_j(k) + \omega(k), \\ r_j(k) &= \sum_{i=1}^{p} b_{ij}(k) e^{l_j(k)(\hat{x}_j(k) - C_j(k))^i} + D_j(k) + \nu(k), \end{aligned} \tag{5.3}$$

where $\omega(k)$ and $\nu(k)$ are uncorrelated white noise signals with variance σ_ω^2 and σ_ν^2 respectively. Similar to Section 3 and 4, the time evolution of these nonlinear parameters in (5.3) can be estimated leveraging the extended Kalman filter described

in Section 2.6.1. In particular, the extended Kalman filter solution for (5.3) yields the series of equations as follows:

$$\begin{aligned}
\boldsymbol{\beta}_j(k|k-1) &= \boldsymbol{\beta}_j(k-1|k-1),\\
\boldsymbol{P}_{\boldsymbol{\beta}}(k|k-1) &= \boldsymbol{P}_{\boldsymbol{\beta}}(k-1|k-1) + I.\sigma_w^2,\\
r(k|k-1) &= \sum_{i=1}^{p} \beta_{ij}(k|k-1) e^{l_j(k|k-1)(\hat{x}_j(k)-C_j(k|k-1)^i)} + D(k|k-1),\\
\boldsymbol{J}(k) &= \frac{\partial \sum_{i=1}^{p} b_{ij} e^{l(\hat{x}_j(k)-C)^i} + D}{\partial \boldsymbol{\beta}_j}\Big|_{\beta(k|k-1)},\\
S(k) &= \boldsymbol{J}(k)\boldsymbol{P}_{\boldsymbol{\beta}}(k|k-1)\boldsymbol{J}(k)^\top + \sigma_v^2,\\
\tilde{R}(k) &= r(k) - r(k|k-1),\\
K_G(k) &= \boldsymbol{P}_{\boldsymbol{\beta}}(k|k-1)\boldsymbol{J}(k)^\top S(k)^{-1},\\
\boldsymbol{\beta}_j(k|k) &= \boldsymbol{\beta}_j(k|k-1) - K_G(k)\tilde{R}(k),\\
\boldsymbol{P}_{\boldsymbol{\beta}}(k|k) &= [1 - K_G(k)\boldsymbol{J}(k)]P_{\beta}(k|k-1),
\end{aligned} \tag{5.4}$$

where $\boldsymbol{\beta}_j(k|k-1)$ and $\boldsymbol{\beta}_j(k|k)$ are the a priori and a posteriori estimates of $\boldsymbol{\beta}_j$, $\boldsymbol{P}_{\boldsymbol{\beta}}(k|k-1)$ and $\boldsymbol{P}_{\boldsymbol{\beta}}(k|k)$ are the a priori and a posteriori estimates of the covariance, and $S(k)$, $\tilde{R}(k)$, $K_G(k), \mathbf{J}(k)$ are the pre-fit residual covariance, the prediction residual, the Kalman gain and the Jacobian vector, respectively. The extended Kalman filter solution above gives an estimate of $\boldsymbol{\beta}_j(k)$ in the run-time. Thus, we can use such estimate to detect whether a fault existed or not in the system, as in Section 3.3.1.

5.2 Results and Numerical Simulations

In this section, we will validate the model prediction accuracy and the capability of PE algorithm in detecting fault, on the same dataset defined in Section 3.3.2, 3.3.3 and 3.3.4. As a comparison, we consider only the e-ss PCA, as it has been shown in previous section that the e-ss PCA outperforms the c-ss and all-PCA.

For the fault detection step, the Kalman Filter will be applied only to the PE model for the ess-PCA + PE model, such that we can understand how the nonlinear counterpart of this algorithm works. Furthermore, we will consider the sensitivity of

both models in detecting faults by observing the escape time (as defined in Section 3.3.1) of all accelerometers. The shorter the escape time, the more sensitive the algorithm in detecting faults for that particular accelerometer, and vice versa. We will show that the e-ss PCA + PE model in general exceed the e-ss PCA model both in model prediction accuracy and fault detection performance.

5.2.1 Bookshelf Resembling Structure

For the damage detection sensitivity, as in the previous section, we consider 3 datasets provided by LANL, each consisting of 4000 samples, corresponding to 3 different damages. From the 4000 samples of nominal data, we use 3000 samples to train the PCA model, use another 300 samples to train the PE models and take the remaining 700 samples for the validating the model predictive accuracy. The fault detection steps will follow the methodology explained in Section 3.3.1 with the tolerance bound $\beta = 5\%$ and $n^* = 5$. We purposely choose a bound which is 10 times larger than the bound used in Section 3.3.2, in order to understand which algorithm is more sensitive to detect fault.

Model Predictive Accuracy

Table 5.1 summarizes the accuracy of the e-ss PCA and e-ss PCA + PE predictive models. As before, the i-th row of each table indicates (1 - NRMSE) $\times 100\%$ for predicting sensor component i using the sensor subset obtained from the e-ss algorithm. The last row indicates the average prediction accuracy among all 24 components.

Table 5.1 shows that the e-ss PCA + PE correction term algorithm performs better if not in the similar range as the ess-PCA. In particular, the e-ss PCA + PE model increases the model prediction accuracy by a big margin in several sensors (sensor 2, 5, 6, 8, 22 and 24) and by around 0.4% in average with respect to the e-ss PCA. For the time complexity, the e-ss PCA + PE have additional steps compared to the e-ss PCA, which is the computation of the LM algorithm. However, this calculation converges rapidly, as it is solvable between the time

Table 5.1: Model predictive accuracy (1 - NRMSE) ×100% for e-ss PCA and e-ss PCA + PE correction term

Sensor N^o	e-ss PCA	e-ss PCA + PE
1	96.35	96.34
2	95.14	96.12*
3	94.84	94.78
4	95.18	95.5
5	94.88	95.95*
6	94.95	96.16*
7	95.02	94.95
8	94.41	95.34*
9	96.49	96.49
10	96.11	96.33
11	95.63	95.58
12	90.95	91.09
13	96.41	96.82
14	95.5	95.91
15	95.05	95.41
16	91.73	91.56
17	95.45	95.47
18	95.19	95.94
19	95.88	96.39
20	96.19	96.51
21	95.29	95.27
22	94.95	95.91*
23	96.24	96.53
24	95.42	96.55*
Average	95.14	95.54

*improvement by around 1%

complexity of $\tilde{O}(\epsilon^{-2})$ or $O(\epsilon^{-2})$, where ϵ is the desired model prediction error [42] (in our use case, the algorithm converges at most in 10 iteration steps). Hence, this additional term does not increase significantly the complexity of PCA algorithm, while able to improve the prediction accuracy.

Damage Detection Sensitivity

Table 5.2 shows the escape time for all 24 sensors inside the structure for both e-ss PCA model and the PE correction term model. Each row indicates the escape time (measured in seconds), and represents the time needed such that the combined

Table 5.2: Escape time for PE correction term model and e-ss PCA for 3 different damage cases.

Sensor N^o	e-ss PCA			PE Correction term		
	Damage 1	Damage 2	Damage 3	Damage 1	Damage 2	Damage 3
1	2.988	3.341	2.429	0.820	0.008	1.062
2	3.351	4.229	2.657	1.306	0.008	1.224
3	2.048	2.555	1.022	0.008	0.006	0.790
4	1.022	0.271	0.110	0.542	0.302	0.130
5	4.096	6.024	2.675	0.430	-	2.697
6	3.351	4.229	1.080	-	0.006	0.582
7	2.048	2.555	1.022	0.006	7.694	3.807
8	1.022	0.271	0.110	0.010	-	0.418
9	-	-	2.481	0.006	0.006	1.092
10	-	-	2.677	6.854	0.006	1.272
11	-	-	4.846	0.006	0.010	0.008
12	-	-	3.593	0.010	0.074	1.274
13	-	-	2.481	0.008	0.008	0.006
14	-	-	2.677	0.008	0.008	1.694
15	-	-	4.846	1.504	0.008	0.428
16	-	-	3.593	0.862	0.556	1.234
17	4.305	6.415	3.439	0.470	0.010	0.704
18	4.281	-	1.860	0.386	-	1.706
19	-	-	-	0.006	0.008	0.008
20	-	-	-	0.006	0.008	0.006
21	-	6.653	1.589	0.010	0.008	0.008
22	-	-	1.860	0.012	0.432	0.008
23	-	7.726	-	0.008	0.006	0.008
24	-	-	-	0.008	0.010	0.008

$^{-}$ indication of false positive

model parameter dynamics leave the bound and never re-enter it again. The (-) indicates the inability of the algorithm to detect a fault within specific accelerometer readings, given the tolerance bound of 5%. Table 5.2 indicates that the PE model is more sensitive than the e-ss PCA in detecting faults. It can be seen clearly that the PE model has less cases of false positive than the e-ss PCA. Also, notice that the PE correction term model is able to raise alarm faster (characterized through shorter escape time) than the e-ss PCA even on a larger bound. Being able to detect faults faster on a larger bound is an advantage, as it allows to minimize the risk of inducing false negative and false positive simultaneously.

5.2.2 UCI Bridge Column

In this experiment, from 4000 sample of nominal samples, we use 3000 samples to train the e-ss PCA model, take another 300 samples to train the PE model while the remaining 700 samples are use to validate the model prediction accuracy. The bound in this case is set to be 1%, which is twice of the bound used in Section 3.3.3.

Model Predictive Accuracy

Table 5.3 shows the model predictive accuracy of the e-ss PCA and the e-ss PCA + PE correction term as in the previous section.

From Table 5.3, we notice that the e-ss PCA + PE model is able to improve the model predictive accuracy w.r.t. the e-ss PCA up to 9% (see, for instance, accelerometer 8). Given the complexity of the bridge's column structure, it is not surprising that using only the linear model is not sufficient to represent the system. Thus, in this case, the nonlinear term introduced by PE model aids the e-ss PCA model to better match the actual system.

Damage Detection Sensitivity

Table 5.4 shows the escape time for all 40 sensors installed on the bridge's column for both the e-ss PCA model and the PE correction term model. As in the previous section, the $(-)$ indicates that the algorithm failed to detect a fault on that specific accelerometer. Notice that, when the damages are strong enough (e.g., damage 2 and damage 3), the PE model can detect the fault dynamics from the reading of all accelerometers. However, both algorithms exhibit a large number of false positive cases, when the damage is weak (damage 1). In particular, the PE model induces more false positive cases than the e-ss PCA model. However, it is important to underline that the escape times for e-ss PCA + PE model in Table 5.4 are calculated only with the PE model. Furthermore, notice that the PE model is able to detect the presence of faults on the several accelerometers, that are undetected by e-ss PCA (see accelerometer number: 2, 7, 10 and 37). In a real

Table 5.3: Model predictive accuracy (1 - NRMSE) ×100% for e-ss PCA and e-ss PCA + PE correction term

Sensor N^o	e-ss PCA	e-ss PCA + PE
1	84.88	85.89*
2	86.27	85.91
3	87.38	90.75*
4	91.48	92.6
5	80.61	87.27*
6	93.33	94.05
7	93.81	92.85
8	77.52	87.44*
9	92.9	92.93
10	94.5	94.51
11	75.08	77.53*
12	91.45	93.22*
13	94.82	94.89
14	78.54	83.95*
15	91.47	91.33
16	95.35	94.74
17	86.32	86.85
18	79.6	79.39
19	92.51	93.77*
20	86.59	86.68
21	91.09	92.93*
22	81.38	87.46*
23	90.87	89.76
24	89.32	93.21*
25	71.5	85.22*
26	92.79	94.53*
27	78.93	85.22*
28	94.69	94.56
29	76.12	83.39*
30	94.41	95.31
31	84.93	85.64
32	91.73	93.89*
33	79.81	79.72
34	91.85	92.73
35	89.47	89.37
36	87.59	88.32
37	87.4	91.01*
38	94.57	93.74
39	87.44	87.51
40	84.93	84.42
Average	87.38	89.36*

*improvement by around 1% or more

Table 5.4: Escape time for PE correction term model and e-ss PCA for 3 different damage cases.

Sensor N^o	e-ss PCA			PE Correction term		
	Damage 1	Damage 2	Damage 3	Damage 1	Damage 2	Damage 3
1	6.1255	-	0.0157	-	0.0664	0.0274
2	-	0.2784	0.2902	0.5315	0.1368	0.1212
3	0.8863	0.3412	0.251	-	0.1446	0.1759
4	3.9765	0.0275	0.0118	-	0.2775	0.2384
5	0.9255	0.0196	0.0078	-	0.2228	0.1954
6	3.4314	0.1412	0.4	-	0.8324	0.6527
7	-	-	0.0118	7.871	0.0899	0.0664
8	2.7373	0.0039	0.0235	-	0.3009	0.3009
9	1.3176	0.0157	0.0196	-	0.1485	0.0743
10	-	-	0.0118	7.9961	0.0664	0.0469
11	6.3725	0.0039	0.0118	-	0.254	0.2189
12	3.4314	0.1412	0.4	-	0.2736	0.2423
13	-	-	0.0039	-	0.1133	0.0821
14	-	-	0.0157	-	2.1612	1.5164
15	1.4627	0.4667	0.3961	-	0.1602	0.1251
16	-	0.0314	0.0118	-	0.1212	0.086
17	5.9922	0.8039	0.4	-	0.3596	0.3127
18	0.7451	0.3529	0.7059	-	0.1524	0.1915
19	-	0.0078	0	-	0.1212	0.0821
20	5.9922	0.8039	0.4	-	0.0977	0.0899
21	-	0.0196	0.0118	-	0.7777	1.0904
22	0.4353	2.902	1.6784	-	0.3244	0.9692
23	4.0627	0.0039	0.0118	7.7968	1.0708	0.8676
24	1.8196	2.5255	1.9569	-	1.4617	1.704
25	-	-	0.0118	-	0.0703	0.043
26	-	0.0941	0.2471	-	4.85	4.9946
27	4.1529	-	0.0078	-	0.6527	0.3517
28	-	0.1451	0.4039	-	1.5476	2.1729
29	1.7059	0.0275	0.0196	-	0.6448	0.5159
30	-	0.0941	0.2471	-	0.8481	0.8403
31	-	-	0.0118	-	0.0899	0.0625
32	-	0.0941	0.2471	-	0.5042	0.6175
33	2.7961	0.5451	0.4157	-	0.7308	0.555
34	-	0.2196	0.4863	-	0.4064	0.512
35	1.8627	0.0431	0.0314	7.9805	6.0342	5.1861
36	-	0.0196	0.0118	-	0.1172	0.1133
37	-	-	-	0.3361	1.7509	3.0132
38	-	-	0.0039	-	0.1172	0.086
39	-	-	6.7373	0.3596	0.3439	0.4963
40	-	0.2784	0.2902	0.0352	0.0743	0.043

$^{-}$ indication of false positive

life application, the fault detection tracking must be done in parallel for both the e-ss PCA and PE Such that, false positive cases can be minimized.

For the stronger damage cases instead, it can be seen that the PE model are able to detect the fault presence in each accelerometers, with respect to the e-ss PCA. This is an evidence that PE correction term does improves the damage detection sensitivity of the e-ss PCA algorithm.

5.2.3 Sheraton Hotel

There are only 2 nominal datasets for this structure, each containing 4096 samples. We first train the e-ss PCA model with 6144 samples. Then, we take another 1024 samples to train the PE model and finally, we use the remaining 1024 samples to validate the model predictive accuracy. Unfortunately, there is no faulty dataset available for this structure. Thus, the comparison between these two algorithms will be done only on the model predictive accuracy.

Model Predictive Accuracy

Table 5.5 summarizes the accuracy of the e-ss PCA and e-ss PCA + PE predictive models. The i-th row of each table indicates the value (1 - NRMSE) $\times 100\%$ used for predicting the output of the sensor component i using the sensor subset obtained via the e-ss algorithm.

Table 5.5 shows that the e-ss PCA + PE model improves the accuracy of e-ss PCA for several sensors by around 1% in average and up to 11% (see. sensor-5). This is another proof that e-ss PCA + PE model can predict the accelerometers dynamics better than the e-ss PCA algorithm.

5.3 Discussion

In this section, we presented a new fault detection identification technique based on the Poly-Exponential (PE) correction term. The numerical simulation results show that e-ss PCA + PE correction term is able to improve the model predictive accuracy w.r.t the ess-PCA up to 9% and 2% on average as demonstrated in three

Table 5.5: Model predictive accuracy (1 - NRMSE) ×100% for e-ss PCA and e-ss PCA + PE correction term

Sensor N^o	e-ss PCA	e-ss PCA + PE
1	70.4	70.53
2	83.99	88.49*
3	88.83	92.74*
4	82.25	82.29
5	77.5	89.03*
6	83.95	83.98
7	86.84	86.84
8	85.25	85.29
9	88.19	88.21
10	88.43	88.52
11	85.93	85.94
12	89.29	89.33
13	86.42	86.5
Average	84.40	85.97*

*improvement by around 1%

different structures. Furthermore, the proposed algorithm is capable of detecting faults in a shorter time with a larger bound compared to the e-ss PCA, while also potentially reduce the number of false positives.

6
Conclusions and Future Work

The field of structural health monitoring (SHM) holds a significant role in preventing unfortunate events that occurred due to a bad maintenance procedure inside a structure. In this thesis, we provide novel fault detection identification techniques which are precise and cost efficient.

We addressed the problem of structural damage detection and exploit it in order to enhance the performance of existing fault detection algorithm. In specific this problem has been tackled in 3 different ways:

- First, we proposed a novel entropy based subset selection algorithm to solve the linearity issue of PCA.
- Second, we derived a novel fault detection identification technique based on the modified RT algorithm.
- Third, we proposed an idea to enhance the model prediction accuracy and fault detection quality of PCA with Poly-Exponential correction term.

In Chapter 3, we discussed the weakness of PCA algorithm, and introduced the notion of e-ss algorithm which is a novel subset selection algorithm based on entropy and information gain to solve this issue. It is shown that the e-ss PCA provides a higher model predictive accuracy and better fault detection quality with

respect to the standard PCA algorithm. Furthermore, this subset selection is able to reduce the number of accelerometers significantly, which ultimately leads to the reduction of maintenance cost inside the building.

In Chapter 4, we introduced a novel fault detection algorithm based on Regression tree (RT). A direct comparison shows that the RT based model outperforms the PCA in detecting faults, especially in the more complex structure. We are able to show as well that the e-ss RT gains advantage over standard RT, in model predictive accuracy and fault detection while using significantly less number of sensors.

In Chapter 5, we enhanced both model predictive accuracy and fault detection quality by integrating the Poly-Exponential (PE) correction term into e-ss PCA. We are able to show that e-ss PCA + PE model can detects faults quicker than e-ss PCA even on a larger tolerance bound. These results are quite interesting, as using the e-ss PCA + PE model, we can reduce the risk of inducing false positive and false negative simultaneously.

In the future, we will consider to exploit more of the PE and RT based model in detecting faults (i.e., by assigning a better Poly-Exponential model or integrating RT based model with PE correction term). In particular, we will explore another notions from information gain, i.e., the cross entropy or the binary cross entropy to derive the e-ss subset. And finally, we will further explore the behavior of the proposed e-ss subset in case when the linear correlations among all variables are low (e.g., when the correlation is less than 0.5).

References

[1] Timothy R. Fasel, Hoon Sohn, and Charles R. Farrar. "Application of frequency domain ARX models and extreme value statistics to damage detection". In: *Smart Structures and Materials 2003: Smart Systems and Nondestructive Evaluation for Civil Infrastructures*. Vol. 5057. SPIE, 2003, pp. 145 –156.

[2] Luciana Balsamo and Raimondo Betti. "Data-based structural health monitoring using small training data sets". In: *Structural Control and Health Monitoring* 22.10 (2015), pp. 1240–1264.

[3] Jean-Bernard Bodeux and Jean-Claude Golinval. "Application of ARMAV models to the identification and damage detection of mechanical and civil engineering structures". In: *Smart materials and structures* 10.3 (2001), p. 479.

[4] Francesco Smarra, Giovanni D Di Girolamo, Vittorio De Iuliis, Achin Jain, Rahul Mangharam, and Alessandro D'Innocenzo. "Data-driven switching modeling for MPC using Regression Trees and Random Forests". In: *Nonlinear Analysis: Hybrid Systems* 36 (2020), p. 100882.

[5] Giovanni D Di Girolamo, Francesco Smarra, Vincenzo Gattulli, Francesco Potenza, Fabio Graziosi, and Alessandro D'Innocenzo. "Data-driven optimal predictive control of seismic induced vibrations in frame structures". In: *Structural Control and Health Monitoring* 27.4 (2020), pp. 1–23.

[6] Francesco Smarra, Giovanni D Di Girolamo, Vincenzo Gattulli, Fabio Graziosi, and Alessandro D'Innocenzo. "Learning Models for Seismic-Induced Vibrations Optimal Control in Structures via Random Forests". In: *Journal of Optimization Theory and Applications* (2020).

[7] Rolf Isermann. *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer Science & Business Media, 2006.

[8] J Edward Jackson. *A user's guide to principal components*. John Wiley & Sons, 2005.

[9] Manu Krishnan, Basuraj Bhowmik, Ashwani K Tiwari, and Budhaditya Hazra. "Online damage detection using recursive principal component analysis and recursive condition indicators". In: *Smart Materials and Structures* 26.8 (2017), p. 085017.

[10] K Lakshmi and A Rama Mohan Rao. "A robust damage-detection technique with environmental variability combining time-series models with principal components". In: *Nondestructive Testing and Evaluation* 29.4 (2014), pp. 357–376.

[11] Ricardo Dunia, S Joe Qin, Thomas F Edgar, and Thomas J McAvoy. "Identification of faulty sensors using principal component analysis". In: *AIChE Journal* 42.10 (1996), pp. 2797–2812.

[12] Krishnan N Kesavan and Anne S Kiremidjian. "A wavelet-based damage diagnosis algorithm using principal component analysis". In: *Structural Control and Health Monitoring* 19.8 (2012), pp. 672–685.

[13] Viet H Nguyen and Jean-Claude Golinval. "Fault detection based on kernel principal component analysis". In: *Engineering Structures* 32.11 (2010), pp. 3683–3691.

[14] Victor Y Pan and Zhao Q Chen. "The complexity of the matrix eigenproblem". In: *Proceedings of the thirty-first annual ACM symposium on Theory of computing*. 1999, pp. 507–516.

[15] Young K Lee, Eun R Lee, and Byeong U Park. "Principal component analysis in very high-dimensional spaces". In: *Statistica Sinica* (2012), pp. 933–956.

[16] Iain M Johnstone and Debashis Paul. "Pca in high dimensions: An orientation". In: *Proceedings of the IEEE* 106.8 (2018), pp. 1277–1292.

[17] Madhur Behl, Francesco Smarra, and Rahul Mangharam. "DR-Advisor: A data-driven demand response recommender system". In: *Applied Energy* 170 (2016), pp. 30–46.

[18] Achin Jain, Francesco Smarra, Madhur Behl, and Rahul Mangharam. "Data-driven model predictive control with regression trees—An application to building energy management". In: *ACM Transactions on Cyber-Physical Systems* 2.1 (2018), pp. 1–21.

[19] Achin Jain, Francesco Smarra, and Rahul Mangharam. "Data predictive control using regression trees and ensemble learning". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2017, pp. 4446–4451.

[20] LANL. *SHM datasets and software*. 2017. URL: `https://www.lanl.gov/projects/national-security-education-center/engineering/software/shm-data-sets-and-software.php`.

[21] S Ding, P Zhang, E Ding, S Yin, A Naik, P Deng, and W Gui. "On the Application of PCA Technique to Fault Diagnosis". In: *Tsinghua Science & Technology* 15.2 (2010), pp. 138–144.

[22] Frederic Paik Schoenberg. "Multidimensional residual analysis of point process models for earthquake occurrences". In: *Journal of the American Statistical Association* 98.464 (2003), pp. 789–795.

[23] Juvencio Santos Nobre and Julio da Motta Singer. "Residual analysis for linear mixed models". In: *Biometrical Journal: Journal of Mathematical Methods in Biosciences* 49.6 (2007), pp. 863–875.

[24] Nicolas Leboeuf, Thierry Boileau, Babak Nahid-Mobarakeh, Guy Clerc, and Farid Meibody-Tabar. "Real-time detection of interturn faults in PM drives using back-EMF estimation and residual analysis". In: *IEEE Transactions on Industry Applications* 47.6 (2011), pp. 2402–2412.

[25] Khristo N Boyadzhiev. "Polyexponentials". In: *arXiv preprint arXiv:0710.1332* (2007).

[26] Taekyun Kim and Dae San Kim. "Degenerate polyexponential functions and degenerate Bell polynomials". In: *Journal of Mathematical Analysis and Applications* 487.2 (2020), p. 124017.

[27] Donald E Mager and William J Jusko. "General pharmacokinetic model for drugs exhibiting target-mediated drug disposition". In: *Journal of pharmacokinetics and pharmacodynamics* 28.6 (2001), pp. 507–532.

[28] Tudor Giurgica-Tiron, Yousef Hindy, Ryan LaRose, Andrea Mari, and William J Zeng. "Digital zero noise extrapolation for quantum error mitigation". In: *IEEE International Conference on Quantum Computing and Engineering*. IEEE. 2020, pp. 306–316.

[29] A. G. Asuero, A. Sayago, and A. G. González. "The Correlation Coefficient: An Overview". In: *Critical Reviews in Analytical Chemistry* 36.1 (2006), pp. 41–59. eprint: `https://doi.org/10.1080/10408340500526766`. URL: `https://doi.org/10.1080/10408340500526766`.

[30] Thu Pham-Gia and Vartan Choulakian. "Distribution of the sample correlation matrix and applications". In: *Open Journal of Statistics* 2014 (2014).

[31] Daniel J Ozer. "Correlation and the coefficient of determination." In: *Psychological bulletin* 97.2 (1985), p. 307.

[32] Mark A. Hall. *Correlation-based Feature Subset Selection for Machine Learning*. University of Waikato, 1999.

[33] Ian T Jolliffe. *Principal component analysis*. Springer, 2002.

[34] Hervé Abdi and Lynne J Williams. "Principal component analysis". In: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459.

[35] Kevin P Murphy. *Machine learning: a probabilistic perspective*. Cambridge, MA, 2012.

[36] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[37] Søren Asmussen, Dirk P Kroese, and Reuven Y Rubinstein. "Heavy tails, importance sampling and cross–entropy". In: *Stochastic Models* 21.1 (2005), pp. 57–76.

[38] Wei-Yin Loh. "Classification and regression trees". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1.1 (2011), pp. 14–23.

[39] Roger J Lewis. "An introduction to classification and regression tree (CART) analysis". In: *Annual meeting of the society for academic emergency medicine in San Francisco, California*. Vol. 14. 2000.

[40] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[41] Jiang Su and Harry Zhang. "A Fast Decision Tree Learning Algorithm". In: *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*. AAAI'06. Boston, Massachusetts: AAAI Press, 2006, 500–505.

[42] N. Yamashita and M. Fukushima. “On the Rate of Convergence of the Levenberg-Marquardt Method”. In: *Topics in Numerical Analysis*. Vienna: Springer Vienna, 2001, pp. 239–249.

[43] Hao Yu and Bogdan Wilamowski. *Levenberg Marquardt Training*. Feb. 2011, pp. 1–16.

[44] Henri Gavin. “The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems”. In: *Department of Civil and Environmental Engineering, Duke University* 28 (2011), pp. 1–5.

[45] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[46] Kenneth Levenberg. “A method for the solution of certain non-linear problems in least squares”. In: *Quarterly of applied mathematics* 2.2 (1944), pp. 164–168.

[47] Donald W Marquardt. “An algorithm for least-squares estimation of nonlinear parameters”. In: *Journal of the society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441.

[48] Greg Welch and Gary Bishop. “An Introduction to the Kalman Filter”. In: *Proc. Siggraph Course* 8 (2006).

[49] Youngjoo Kim and Hyochoong Bang. “Introduction to Kalman Filter and Its Applications”. In: 2018.

[50] Jimmy Tjen, Francesco Smarra, and Alessandro D’Innocenzo. “An entropy-based sensor selection algorithm for structural damage detection”. In: *IEEE 16th International Conference on Automation Science and Engineering*. 2020, pp. 1566–1571.

[51] Rajamani Doraiswami, Maryhelen Stevenson, and Chris Diduch. *Identification of physical systems: Applications to condition monitoring, fault diagnosis, soft sensor and controller design*. John Wiley & Sons, 2014.

[52] Arthur Gelb. *Applied optimal estimation*. MIT press, 1974.

[53] Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[54] Alan S Willsky. “A survey of design methods for failure detection in dynamic systems”. In: *Automatica* 12 (1976), pp. 601–611.

[55] Michael Sipser. *Introduction to the Theory of Computation*. 1st. International Thomson Publishing, 1996.

[56] Charles R Farrar, David A Nix, Thomas A Duffey, Phillip J Cornwell, and Gerard C Pardoen. “Damage Identification With Linear Discriminant Operators”. In: *17th International Modal Analysis Conference*. 1999.