DYNAMIC DATA DRIVEN SENSOR PLACEMENT FOR ENABLING CAPABILITY ESTIMATION OF SELF - AWARE AEROSPACE VEHICLES

A Thesis

by

ARJUN SINGH

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Dr. Douglas Allaire
Committee Members,	Dr. Vinayak Krishnamurthy
	Dr. Ulisses Braga-Neto
Head of Department,	Dr. Bryan Rasmussen

December 2020

Major Subject: Mechanical Engineering

ABSTRACT

Unmanned aerial vehicles (UAV) are utilized in numerous industries and in recent years with the advent of learning techniques, the focus is now on developing Self – aware UAVs that rely on an array of environmental sensors to replace a pilots awareness of the structural capability of the UAV and provide time critical analysis of the sensor data to make complex decisions in real time. Hence a self-aware UAV is capable of dynamically and autonomously sense its structural state and act accordingly to perform the required task. In this thesis we propose a data driven approach to producing estimates of capability of a self – aware UAV and using that to optimize the sensor location is presented. This process involves using high physics-based models such as ASWING, an aerodynamic, structural, and control response analysis software in tandem with Akselos modeler to produce an offline library that comprises of damage states along with capabilities corresponding to different kinematic states of a UAV. Further this generated information is used to create a classification model which is used to predict the capability for the real time data. The classification model serves as an enabler for the optimization algorithm to measure the error value between the true and the predicted capability of the UAV. The objective of the study is to provide evidence that optimizing sensor locations will yield to better decision making and extended life – cycle of the UAV. We demonstrate this through a performance comparison between optimum placement and standard placement of sensors. We have also provided evidence of proof of concept of how dynamic sampling of information can improve the process of capability estimation in self – aware aerospace vehicles.

ACKNOWLEDGMENTS

I would like to express my gratitude to the committee chair, Dr. Douglas Allaire and my committee members, Dr. Vinayak Krishnamurthy and Dr. Ulisses Braga-Neto for their continuous guidance and support throughout the course of this research project.

I would also like to thank my colleagues, friends, department faculty and staff for making my time at Texas A&M University a memorable experience.

Lastly, I would like to sincerely thank my family for their continuous encouragement, love and support.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by the thesis committee consisting of Associate. Prof. Dr. Douglas Allaire (advisor) and Assistant. Prof. Dr. Vinayak Krishnamurthy of the Department of Mechanical Engineering and Professor Dr. Ulisses Braga-Neto of the Department of Electrical and Computer Engineering.

All other work conducted for the thesis was completed independently.

Funding Sources

This work was supported by the Air Force Office of Scientific Research (AFOSR) under award numbers FA9550-15-1-0038 (program manager Fariba Fahroo), and FA9550-16-1-0108 (program manager Erik Blasch), Opinions expressed in this paper are of the authors and do not necessarily reflect the views of the funding sources.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

1. INTRODUCTION

1.1 Motivation

In the current society complex engineered systems such as wind turbines, pipelines, aircraft, ships, among others, are valuable assets to the society and enrich the lives of the people[1]. It is this dependence that has led to the increase demand for safety and reliability of these systems. Continuous use of any machinery leads to deterioration of its health and factors such as humidity, temperature variations, collisions, etc. are major contributors, hence it is necessary to evaluate the health of the machines over a period of time to plan maintenance actions for a longer useful life.

For each structure or system there are various ways in which the inspections can be carried out, For example, after the first industrial revolution in the 18th century the handling of machinery was fairly simple i.e. utilize it until it fails, this type of methodology is called corrective maintenance. Before the great depression, during mid to late 19th century with the advent of new discoveries and innovations in manufacturing, maintenance became more dynamic and used a basic time based maintenance technique which required replacement of parts at regular intervals to prolong the life of the machinery. Still the process was not lean as it required high downtimes and led to loss of funds. Post World War II during the industrial rebuilding of Japan the concept of preventive maintenance was conceived where checks and repairs were scheduled at specified intervals to prolong the life of equipment. In the current generation manufacturers have opted to incorporate artificial intelligence and machine learning in the complex systems to provide high resolution information about the equipment's health allowing them to plan repairs when required rather than at specific interval, which in turn leads to less downtime and much less fiscal losses and this technique is referred to as Structural Health Monitoring (SHM). Figure 1.1 illustrates the evolution of maintenance techniques as described above and Figure 1.2 summarizes how the data flows through the SHM process to give out accurate predictions for remaining useful life or capability.

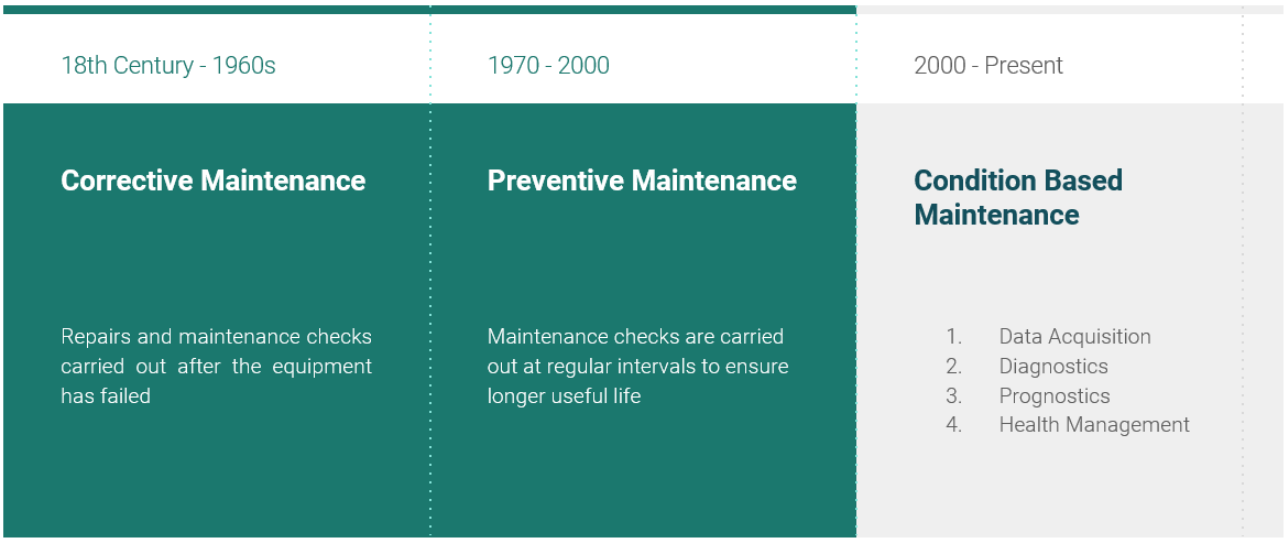


Figure 1.1: Evolution of maintenance strategies

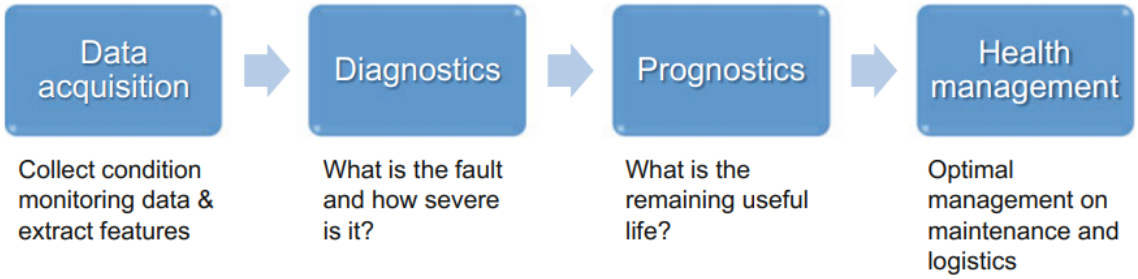


Figure 1.2: Flow of information for Structural Health Monitoring

Based on the plot between maintenance strategies versus cost in [2]. If the number of failures in the equipment is very small, it is advisable to use corrective maintenance because as it will be cost effective and the number of parts required will be less for maintenance. If the number of failed parts is large, it would be more beneficial to perform preventive maintenance as it is much more efficient. However, in most engineering application, the method opted is condition - based maintenance due to its cost effectiveness and ubiquitous use.

SHM is defined as examining the structure through performance evaluation while in - service. [1].

The SHM system should fulfill certain requirements including:

- Lower Cost
- Continuous assessment
- Sensitive to local of damages
- Insensitive to measurement noise
- Insensitive to environmental conditions

Over a period of time several techniques have been proposed for health monitoring, common approaches include strain monitoring, vibration based monitoring, elastic waves – based monitoring and so on [1].

Generally for complex engineered systems such as Gas turbines, aircraft, unmanned aerial vehicles, the damage usually occurs at a local level hence it is advisable to use strain sensors as they are sensitive to small scale damages and the emergence of fiber optics (FO) has added a further stimulus to the use of strain sensors for structural health monitoring and has expanded their relevance due to the possibility of inserting them within the structure [3]. The vibration and strain monitoring techniques are mostly implemented in the passive domain in order to reduce the effect on the operation of the engineering system. These techniques measure the effect of ambient loads on the structure, without imparting energy to the system [1].

The SHM framework currently used in the industry (shown in figure 1.2) uses sampled sensor information followed by extracting damage sensitive features from this information to perform statistical analysis which in turn will determine the current state of the system [4]. Therefore the SHM process requires data fusion, filters for data cleaning, data management systems for feature extraction and post – processing and algorithms for the analysis of the data.

Since the SHM approach is highly dependent on the sensor information, tremendous research has been carried out with respect to sensor technology over the past few years. Figure 1.3 lists out what types of measurement techniques can be used for which specific application depending on requirement and end results. The development of the FO sensors, PZT transducers, and the smart films, is a result of this hunt for better sensors and sensing abilities. Unfortunately, the performance of these sensors is limited and in order to achieve further improvement, this has led to the motivation behind sensor placement optimization.

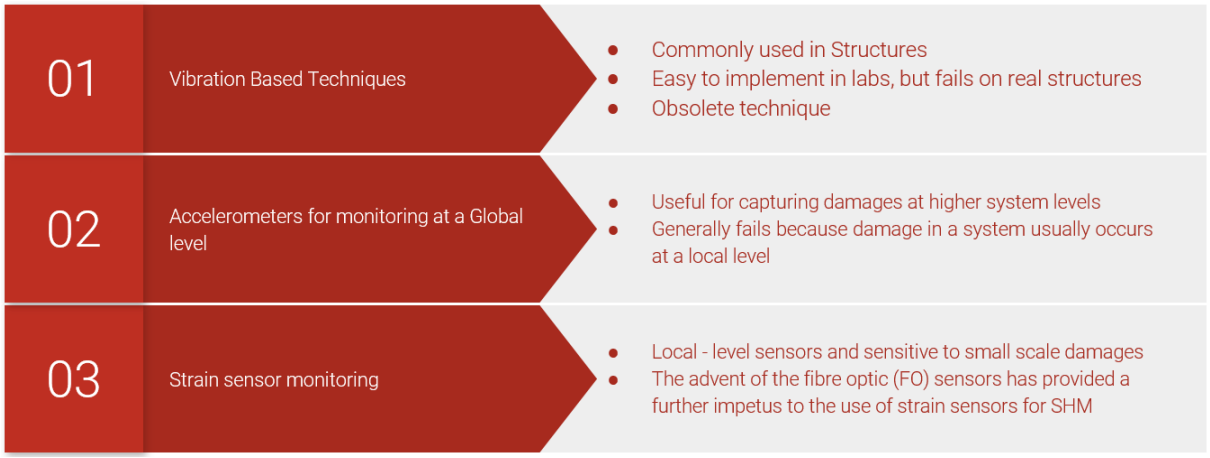


Figure 1.3: Sensor techniques for detecting damage

In [5] they propose an approach for optimal sensor placement starting with a general formulation of Bayes risk and deriving global optimality criterion within a detection theory framework. They make the argument that since for an SHM system the fundamental step is to detect the damage correctly and it is advisable to develop an approach rooted in theory of detection. They use genetic algorithm with a time varying mutation rate to search the design space to present a flexible actuator/sensor placement strategy. In [6] they design a three phase sensor placement strategy which aims to achieve the following objectives: A high quality placement of an array of sensors which ensures effective communication and low placement complexity whilst reducing the probability of false detection in a wireless sensor network. Apart from this, their model enables sensor

nodes to develop "connectivity trees" to maintain structural health state and network connectivity. In [7] they propose a modification on the gradient descent algorithm where they use it in combination with a model that considers both the topography of the environment and a set of sensors with probabilistic sensing. In [8] they use a performance index based on damage detection and use genetic algorithm to determine optimum sensor location and they also present improved strategies for genetic algorithm to provide an optimal sensor placement and compare it with penalty function method and forced mutation method. The central idea behind all the approaches is to provide an optimal sensor location through minimizing false detection error or through maximum area coverage (MAC). In a recent study by Department of Civil & Environmental Engineering, MIT on Optimal sensor placement for Structural health monitoring using discrete optimization, they propose a discrete optimization scheme using the artificial bee colony algorithm with sensor location as design variable constraint on the number of sensors. A MAC oriented objective function is explored to compute the usefulness of a sensor layout in the optimization framework based on the modal characteristics of a reduced order model. The reduced order model is obtained using an iterated improved reduced system technique [9]. Another example in this field include Structural health monitoring sensor placement optimization under uncertainty [10] where a methodology is developed to find the optimal layout of sensor arrays of structural health monitoring systems. The approach utilizes probabilistic finite element analysis, structural damage detection algorithms in tandem with reliability-based optimization. In our approach we employ Dynamic data driven application system (DDDAS) paradigm to dynamically incorporate additional data into an executing application, and in reverse the ability of an application to dynamically steer the measurement process. We use offline physics based models specifically a Digital twin of the UAV in combinations with an aerodynamic, structural, and control-response analysis software to create data that emulates real time data from sensors on a self-aware UAV and incorporate certain amount of noise such that our classification model is more robust to the actual noise in flight. We then use elitist genetic algorithm, a Meta heuristic algorithm to explore the design space and prevent the loss of information to output an optimum sensor placement which predicts the damage state of the Self-aware

UAV and estimates its current capability.

1.2 Dynamic Data Driven Application Systems (DDDAS)

DDDAS paradigm was pioneered by Frederica Darema , according to whom "in DDDAS instrumentation, information and the running application models of the engineered systems become a feedback loop, where information is included into the running model of the system dynamically, to improve the prediction accuracy of the overall model (or simulation), or to improve the speed of the simulation, and in reverse the executing model controls the instrumentation process to guide the simulation process for more accurate measurements. DDDAS is a framework that allows us to create new capabilities through more analysis, precise interpretation, and forecasting the functioning of complex engineered systems, which can be engineered, societal or natural, and to create decision methods which can have the precision of an all - inclusive simulation. DDDAS unifies the computational and measurement aspects of an engineered system, it also stretches the idea of Big Computing to span from the high-end to the real-time data acquisition and control, and it's a primary methodology in managing and smartly exploiting Big Data" [11]. A Self – aware unmanned aerial vehicle (UAV) can sense, plan and act in real time based on the sensor information it receives and can appreciably improve the UAV performance over its life cycle [12]. To achieve this a UAV must be able to predict its own capability during flight through "learning" algorithms and a combination of physics based models and sensor data as shown in figure 1.4 to amass relevant information about its environment. Such a vehicle manifests the dynamic data driven application systems (DDDAS) paradigm by means of this any measurement, data or simulations becomes a feedback to the system.

Several advances have been made to improve structural sensing and vehicle health monitoring systems which includes Integrated Vehicle Health Management (IVHM) framework created by NASA [13] and the Dept. of Defense [14]. These approaches take operational data, physics-based models, and forecasting techniques to build an architecture that improves the mission reliability [12]. Reference [15], [16] presents a methodology to acquire data through a model that incorporates an aircraft model of a medium altitude, long endurance (MALE) UAV, an aero/structural

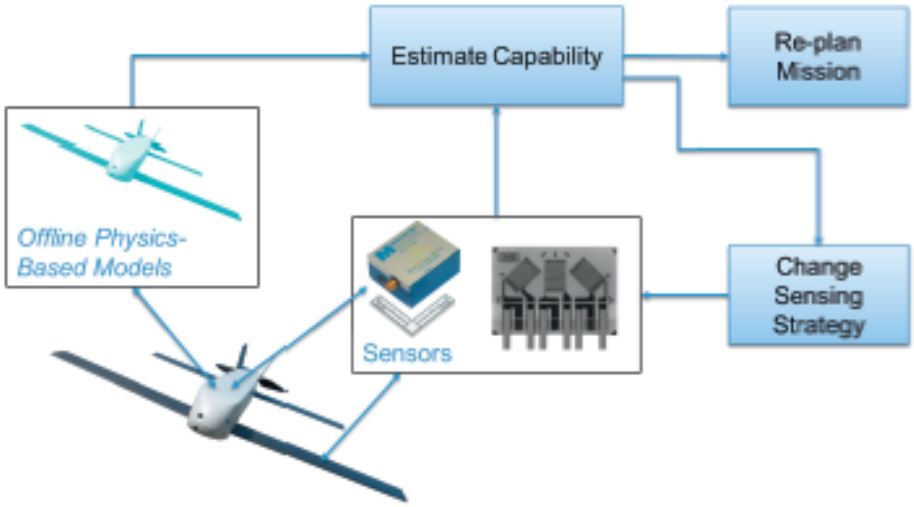


Figure 1.4: To calculate the capability for a self - aware UAV, we use a combination of physics based model and real - timw sensor data to achieve dynamically updated estimates of capability

Reprinted from [12]

multi-physics model, and a structural solver. The information generated through the simulations is stored in an offline library and this information is used to provide estimates of capability in real – time. The study also presents ways to mitigate sensor noise using James – stein estimator and preventing classification through information gained from sensors over different maneuvers. The availability of real - time information along with increased on-board computing power provides an opportunity to extrapolate the digital twin framework further to bolster real-time and on-board decision-making. To achieve this we require DDDAS modeling approaches and algorithms.

1.2.1 Physics based model combination for our research study

Figure 1.5 shows our approach that integrates offline simulations with online sensor information to provide an approximation of the current capability of the UAV. Specifically, we combine data from physics-based models, simulated offline to create an offline library comprising of different kinematic states with damage cases which corresponds to different UAV capability, together with dynamic sensor data. We use dynamic data to make our estimates of capability more robust

to the uncertainty experienced in real time by a self-aware UAV.

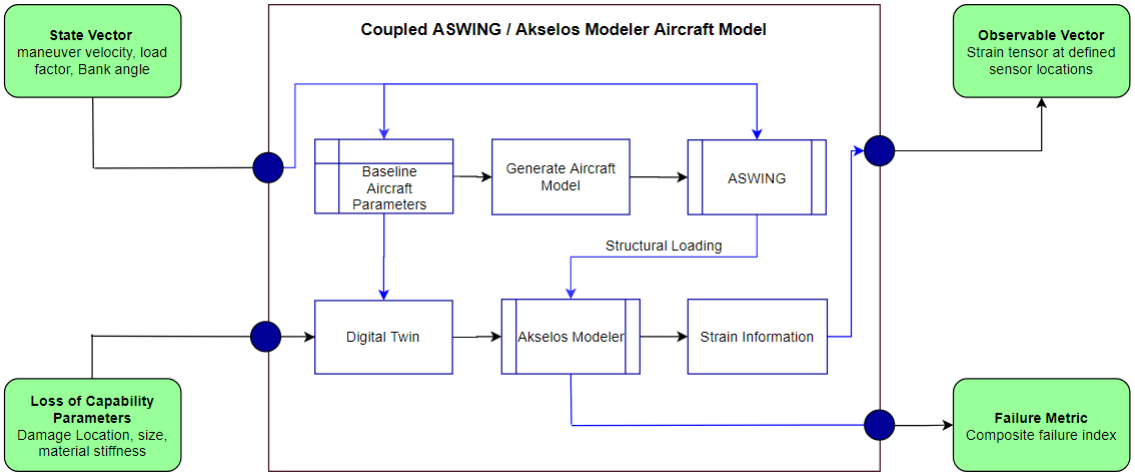


Figure 1.5: Information flow through physics based models in the offline phase to generate damage library for online prediction

1.3 Hardware Test-bed

The physical entity used in our study is a fixed-wing UAV with a wingspan of 12 feet. The fuselage is from an off-the-shelf airplane, while the wings are built with a plywood and carbon fiber construction. The wings are interchangeable, so that data can be collected with both pristine and damaged wings. The upper surface of the wing is fitted with 24 uni-axial strain sensors distributed span-wise between 25% and 75% span. The electric motor and avionics are also a custom installation. Images of the aircraft are shown in figure 1.6. Offline, we build physics-based models of the pristine vehicle using finite element structural models, vortex lattice aerodynamic models, and other models of key aircraft systems. We run simulations of different damage scenarios using these models and create reduced-order models and damage libraries. Then online (in flight), we will acquire sensor data, perform a classification task, and then rapidly access the appropriate pre-computed reduced models and damage libraries. The sensor data will also be used to update the reduced-order models in the face of changing conditions.

Figure 1.6: Physical custom built hardware test-bed whose digital twin is used for this research

1.4 Digital Twin

A digital twin is defined as a digital clone of the physical entity which integrates IoT, artificial intelligence, machine learning techniques, and software analytics [17] to create live computational simulation models that update and change as the physical entities. Akselos Inc have developed a component based reduced order model for the physical UAV using Akselos Integra modelling software, which contains an implementation of Static - Condensation Reduced-Basis-Element (SCRBE) method called as RB – FEA which has the ability to rapidly adapt different damage states and provide the UAV capability for different load factors and wing loading.

In [18] they develop Static-Condensation Reduced-Basis-Element (SCRBE) method, a reduced order modelling approach which addresses the challenges faced by reduction techniques such as reduced-basis (RB) methods, the key constraint in these techniques is that the complete system-level problem is quite large for complex engineering systems and hence there is a need for digital twins. So much, that even a single solution of the full FE system is often intractable. Even if the complete model could be solved, to express the digital twin model we require many parameters, each with large domains and erratic effect on the solution [19].

Akselos software provides us with an extensive model library, the library contains 15 components as shown in figure 1.7, components selected for the model takes into consideration various factors which include spatial distribution of parameters, mesh size, etc. 15 components in Akselos

provides a good balance between complexity and flexibility for a system of this scale [19].

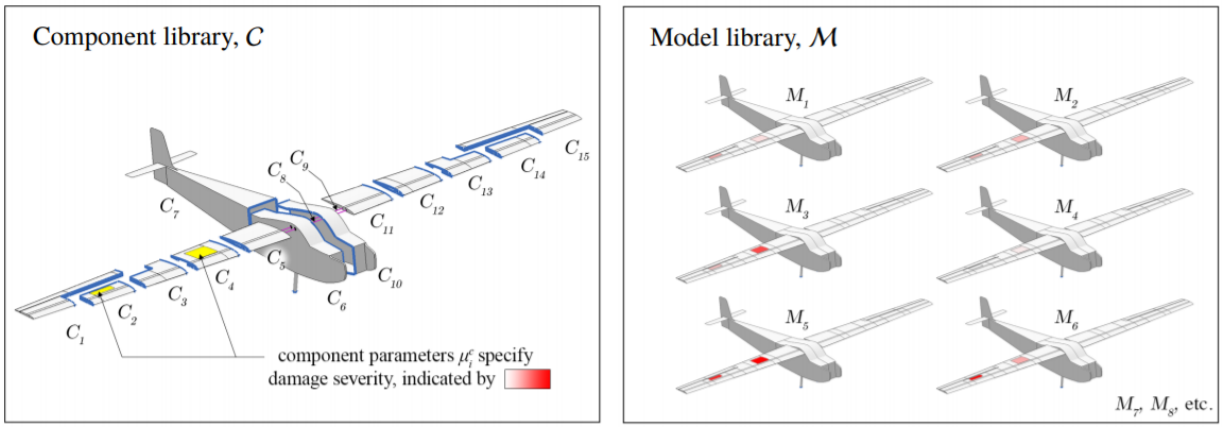


Figure 1.7: Component and model library in Akselos to represent a damaged UAV asset. Every model in the library has a distinct parameter setting

Reprinted from [20]

1.4.1 Damage case definition in Akselos Modeler

In order for the digital twin (UAV) to be capable of replicating and estimating capability of the physical asset it must emulate the damage states such as cracking, delamination, impact damage, or loss of material, at all possible degrees of severity. But since that is difficult, Akselos considers reducing the material stiffness to define a damage state. This reduction in stiffness is a notional model that acts as a proxy for more complex damage occurring in the damage region [19]. Due to presence of 15 components and different damage locations as shown in figure 1.7 and 1.9, Akselos allows us to create variegated damage states through combination of different damage locations and percentage material stiffness at that location. The idea behind defining damage states is to study the response of the digital twin to damages and it will be help develop a methodology to determine the course of action for the UAV in real time.

Figure 1.8 illustrates the different regions in red numbered from 1 to 14 which allow us to represent a damage by reducing the material stiffness value for those regions and figure 1.9 shows

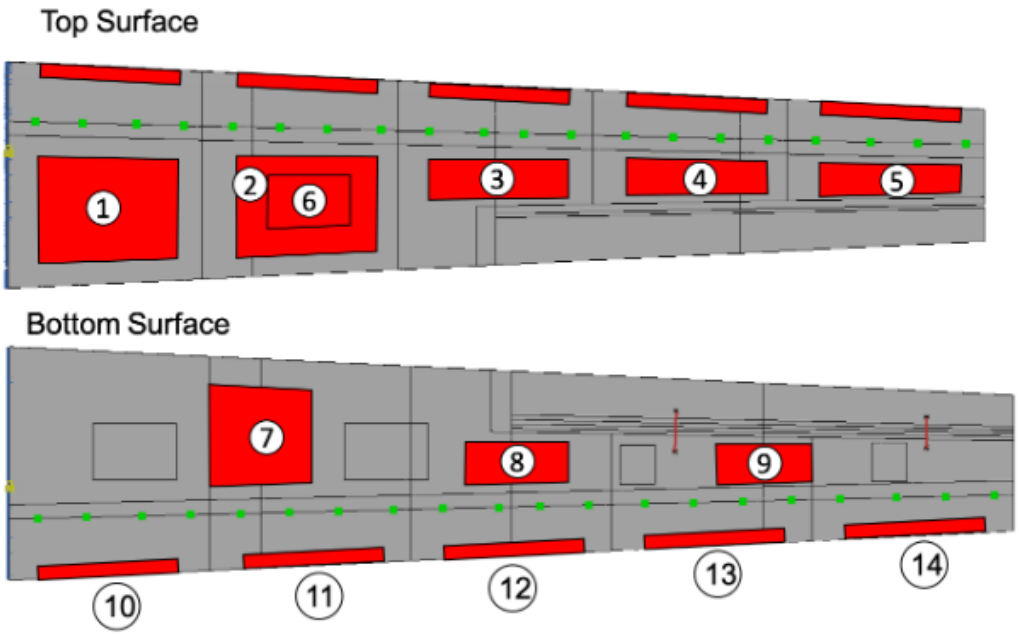


Figure 1.8: Locations defined in akselos to change the material stiffness values to emulate a damage case

how severity of damage increases with reducing material stiffness. Therefore if we were to select defect regions 2, 3, and 4, and change the material stiffness from 100% for each material to 30% then dr02_30, dr03_30, and dr04_30 would constitute as one damage case. Figure 1.10 details the structure of the wing, as represented in our model along with the material used for each component

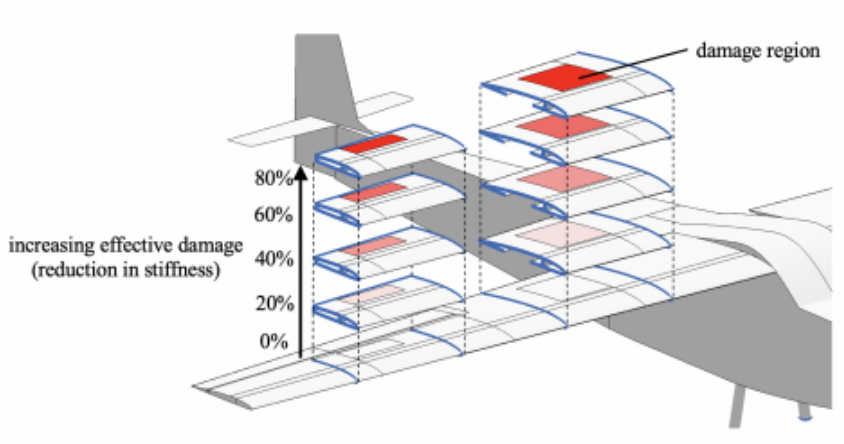
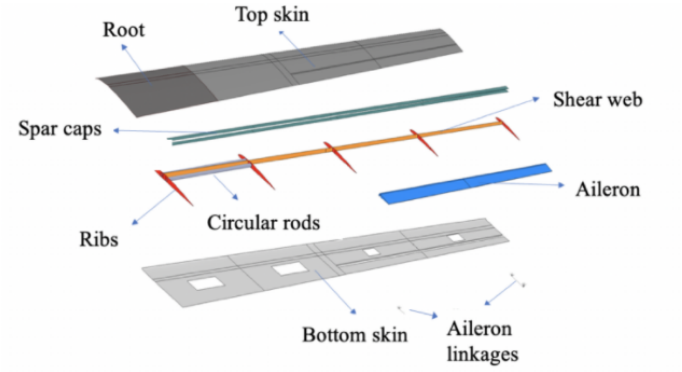


Figure 1.9: An illustration of how the effectiveness of damage increases with reduced material stiffness

Reprinted from [20]



Component	Material	Material Type	Element Type	No. Layers
Root	Carbon Fabric	2D Orthotropic	Quad4	4
Top skin	Carbon Fabric	2D Orthotropic	Quad4	3
Bottom skin	Carbon Fabric	2D Orthotropic	Quad4	3
Spar caps	Carbon Uni	2D Orthotropic	Quad4	8
Shear web	Carbon Fabric	2D Orthotropic	Quad4	3
Ribs	Plywood	Isotropic	Quad4	1
Circular rods	Carbon Iso-Tube	Isotropic	Beam2	-
Ailerons	Foam (HiLoad60)	Isotropic	Hex8	-
Aileron Linkages	-	-	3dof Spring, Rigid	-

Figure 1.10: Representation of the internal structure of the wing in Akselos

Reprinted from [20]

1.5 ASWING formulation

Developed by MIT, ASWING is an aerodynamic, control response analysis software for aircraft with high to moderate aspect ratio and flexible wings. The main use of ASWING software is to enable rapid and comprehensive investigation of the V-n flight envelope to identify potential failure scenarios. It is an application used to forecast the quasi-static and static loads and contortion of aircraft with flexible high aspect ratio surfaces and fuselage. The central characteristics of the overall structural/aerodynamic model are illustrated in figure 1.11. A fully nonlinear Bernoulli-Euler beam representation is used for all the surface and fuselage structures, and an enhanced lifting-line representation is used to model the aerodynamic surface characteristics. The lifting-line model employs wind-aligned trailing vorticity, a Prandtl-Glauert compressibility transformation, and local-stall lift coefficient limiting to economically predict extreme flight situations with reasonable accuracy [21].

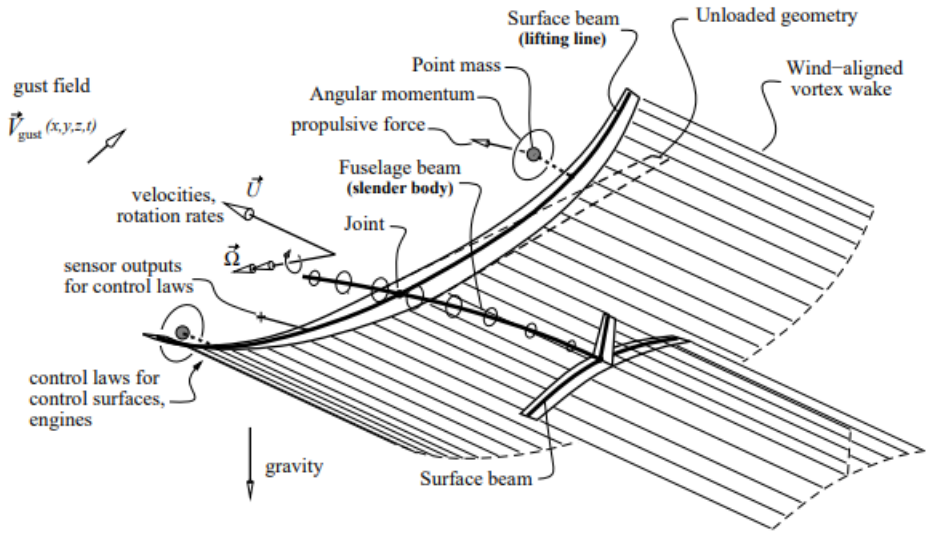


Figure 1.11: ASWING configuration representation

1.6 Optimum sensor placement

The work on Optimization of sensor placement for structural health monitoring is fairly limited, hence there is need for more investigation in order for improved application of health monitoring techniques. In terms of influence, an optimum position of sensors can have a direct and indirect impact on the maintenance process, in terms of direct impact it will reduce the cost of instrumentation by optimizing the number of sensors along with position. As for indirect impact, OSP will reduce the risk of false positive and improve the SHM process. The incentive for OSP differs for different applications, for example, in case of self – aware aerial vehicles, due to weight constraints it is advisable to use lesser sensors to reduce the weight and therefore we must only take the information that contributes towards prediction of capability or state of the vehicle, for large structures such as bridges and dams, the number of sensors are considerably large, therefore the data management and the energy requirement of these large number of sensors is a challenge. The data management costs incorporates the costs of collection and storage of data, and even with the use of signal processing tools and innovative solutions such as compressed sensing [17], [22] the repeating nature of the costs makes this problem important and significant. Hence, there is high monetary motivation in investigating optimum sensor placement for structural health monitoring applications. Based on the description above it is clear that the framework for optimizing the sensor placement differs for each application as shown in figure 1.12.

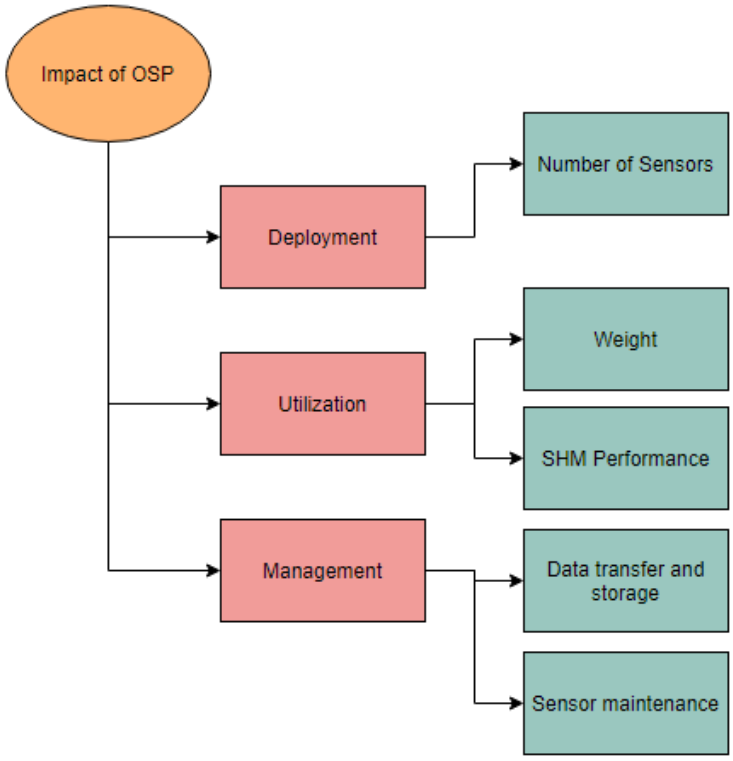


Figure 1.12: Impact of optimizing the sensor placement on different aspects of the SHM process

Reprinted from [1]

1.7 Offline Damage Library

The offline library was created using the models and methods described above, we start by defining input conditions for a maneuver in ASWING which include control conditions, bank angle, velocity, load factor, altitude and sensing parameters. This simulation provides us with the beam loads and bending moments acting on the wing which is an input to Akselos modeler which gives us the strain readings at specified sensor locations. The offline library contains many defined kinematic states from ASWING which are input to Akselos modeler to be simulated at different load factors. As described in section 1.4.1, we create 50 unique damage cases and measure the strain values at defined sensor grid for different load factors and tabulate them in a tree structure as shown in figure 1.13

As we can see from the tree structure that each damage case has an attached capability value

with it, In Akselos software there are various solution fields that calculated, one such is the composite failure index, described in detail in section 3.2.1, value is calculated as the ratio of maximum stress occurring in the wing divided by the yield stress, hence when the ratio equals 1 the maximum stress is equal to the yield stress and we can safely assume that is when the UAV fails. The load factor at which the failure index equals 1 will be different for different damage cases but not necessarily. To determine the load value at which the damage case fails we employ a root finding algorithm where we first increase AVL load case by a certain amount until the value of failure index exceeds 1 and then we interpolate between the AVL load case corresponding to failure index > 1 and the AVL load case corresponding to failure index < 1 to determine the AVL load case corresponding to failure index $= 1$ and this is noted as the capability corresponding to a damage case. We apply this root finding algorithm to all the damage cases and note the capability for the offline library.

The illustration in figure 1.13 is not a general interpretation of a damage library, this framework is adopted as it best represents the dataset used for our research project and allows for easy interpretation,

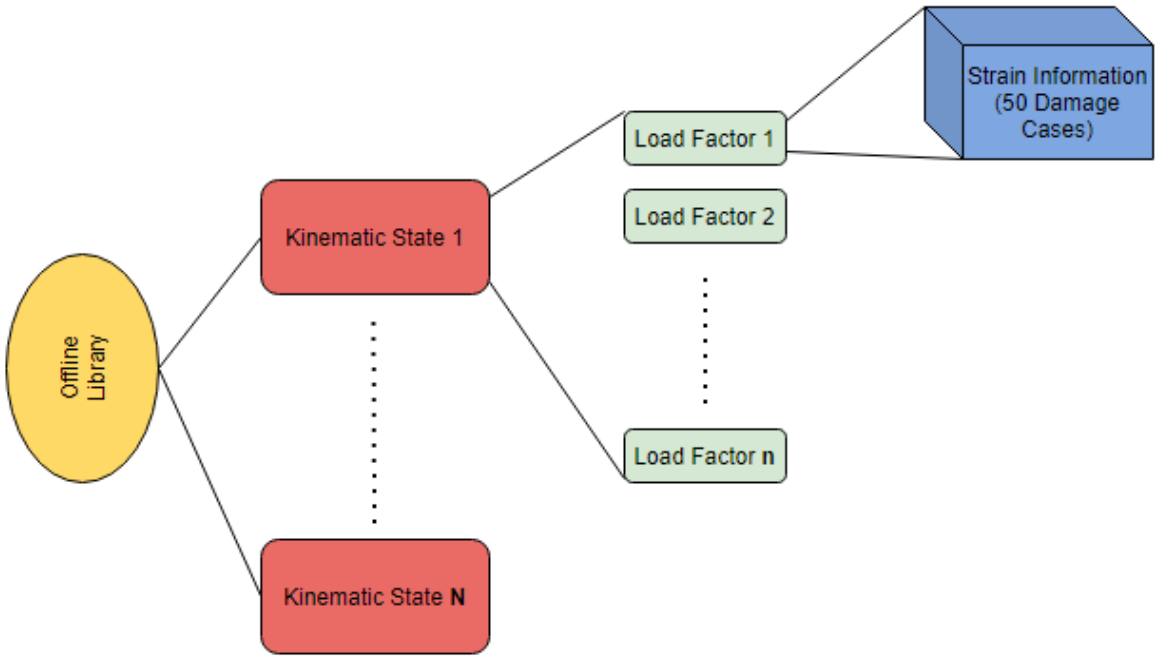


Figure 1.13: Offline library representation used in our research study

2. BACKGROUND

2.1 Introduction to Machine Learning

Machine learning in general can be defined as mathematical and statistical methods using past events to improve the performance of the model to make more accurate future predictions. Here, past events refers to the prior information available to the learning model, which typically takes the form of electronic data collected and made available for analysis. "Arthur Samuel" coined the term Machine Learning in 1959 and since then there have been several advances in the field. Currently learning techniques are an integral part of our daily lifestyle which includes personal assistants on our mobile devices, Traffic predictions on our GPS, email span and malware filtering and so on. All of this involves the computer discovering to perform tasks without being specifically programmed to do so. The subject of machine learning utilizes different approaches to make computers accomplish tasks where no deterministic algorithm is present. In cases where a high number of possible response variable exist, one approach is to label those answers as true labels and use this as training data for the computer to improve the algorithm(s) to predict correct answers for a future case [23]. For our study we are using learning algorithms to make predictions about the health and capability of the self-aware UAV through training the model on strain information generated by Akselos.

2.1.1 Terminology

A dataset is defined as m x n matrix, where m represents the number of data points we have for each class and n as the number of features corresponding based on the problem. In our study we create a vector with all the strain information, hence we have a dataset with 50 data points, one corresponding to each class and the dimension of the vector depends on the number of sensors selected. Since we have a single data point for each class we do not opt for the conventional train test split approach coupled with cross validation to determine the best model, instead we train on the whole dataset and test on dataset that emulates real time strain information tensor.

2.1.2 Types of Learning Algorithms

Machine learning algorithms are generally classified into two categories, supervised learning and unsupervised learning.

2.1.2.1 Supervised learning

Supervised learning algorithms perform the task of forming an equation or function that can map an input to a specific class based on the training dataset. Therefore, if we have a set of N training samples of the form $\{(x_1, y_1), ..., (x_N,\ y_N)\}$ such that x_i is the feature vector of the ith example and y_i is its label (i.e. class), then a supervised learning algorithm would seek a function $g : X \rightarrow Y$, where X is the input space and Y is the output space or the class that correspond to training samples. The function g is called the hypothesis, and similar functions exist in the hypothesis space. General approach adopted to choose the optimum function from the space include empirical risk minimization and structural risk minimization [24], where empirical risk minimization aims to find the most reasonable fit to the training data and the structural risk minimization includes a penalty function to control the bias / variance balance. The supervised learning problems can be divided into regression and classification problems. In a classification problem, the output is generally a categorical variable whereas in a regression problem the output is a continuous value.

2.1.2.2 Unsupervised learning

As opposed to supervised learning algorithms, unsupervised learning algorithms only have the X component of the training data and no corresponding labels. Essentially these algorithms look for undetected patterns in the dataset and form probability densities over inputs [25]. Two of the major techniques used in unsupervised learning are PCA and cluster analysis. Cluster analysis is a technique that forms factions of the data that has not been labelled or classified. Hence, cluster analysis identifies similarities in the data and responds based on the presence or absence of such similarities in each new chunk of information. This approach has been helpful in detecting anomalous data points that do not fit into either group. Principal component analysis is generally used for dimensionality reduction by projecting each data point onto its principal component to

prevent loss of information and make the data more tractable.

For our study we have opted to study the performance of only supervised learning algorithms. We specifically compared the performance of K – nearest neighbors, Support vector machines, and Bernoulli naïve bayes. The implementations of each algorithm are given below.

2.2 Support Vector Machines

2.2.1 History

The original SVM algorithm was invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963. In 1992, Bernhard Boser, Isabelle Guyon and Vladimir Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes [26].But the model in use as of now can be attributed to Cortes and Vapnik in 1992.[27]

2.2.2 Introduction

The support vector machines(SVM) is a more general formulation of a simple classifier called the maximal margin classifier, which aims to find a hyperplane in an n – dimensional space. A hyperplane can be defined as a flat affine subspace of dimension n – 1, for example in two dimension, a hyperplane is a one – dimensional subspace or a line as shown in figure 2.1. A hyperplane in an n - dimensional feature space can be represented by the following equation –

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b = \sum_{i=1}^{n} x_i w_i + b = 0 \tag{2.1}$$

Dividing by $||\mathbf{w}||$, we get

$$\frac{\mathbf{x}^T\mathbf{w}}{||\mathbf{w}||} = P_{\mathbf{w}}(\mathbf{x}) = -\frac{b}{||\mathbf{w}||} \tag{2.2}$$

suggesting that the projection of any point $\mathbf{x}$ on the plane onto the vector $\mathbf{w}$ is always $-b/||\mathbf{w}||$, i.e., $\mathbf{w}$ in the normal direction of the plane, and $|b|/||\mathbf{w}||$ is the distance from the origin to the plane. Note that the equation of the hyper plane is not unique. $c\, f(\mathbf{x}) = 0$ represents the same plane for any c.

The n - dimensional space is divided into two different regions by the plane. Specifically, we

define a mapping function $y = sign(f(\mathbf{x})) \in \{1, -1\}$,

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b = \begin{cases} > 0, & y = sign(f(\mathbf{x})) = 1,\ \mathbf{x} \in P \\ < 0, & y = sign(f(\mathbf{x})) = -1,\ \mathbf{x} \in N \end{cases} \tag{2.3}$$

Any point $\mathbf{x} \in P$ on the positive side of the plane is mapped to 1, while any point $\mathbf{x} \in N$ on the negative side is mapped to -1. A point $\mathbf{x}$ of unknown class will be classified to P if $f(\mathbf{x}) > 0$, or N if $f(\mathbf{x}) < 0$.

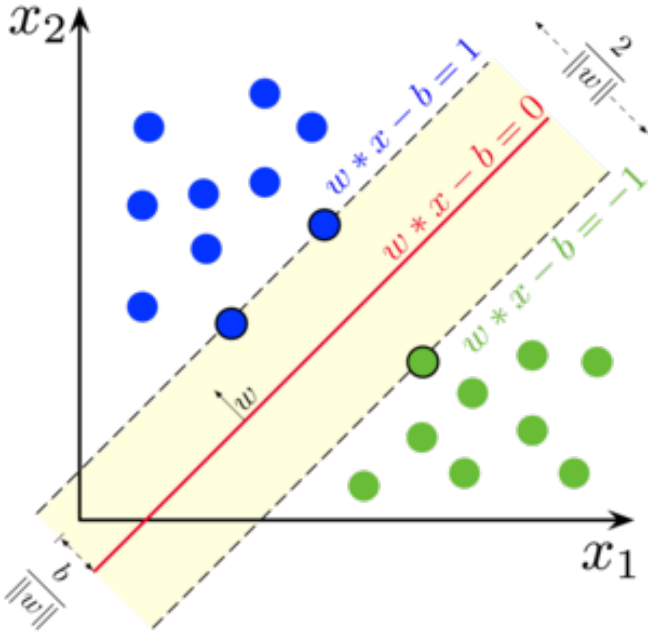


Figure 2.1: Representation of hyperplane identified by support vector machines

Reprinted from en.wikipedia.org

2.2.3 Linear formulation

For a decision hyper-plane $\mathbf{x}^T\mathbf{w} + b = 0$ to separate the two classes $P = \{(\mathbf{x}_i, 1)\}$ and $N = \{(\mathbf{x}_i, -1)\}$, it has to satisfy

$$y_i(\mathbf{x}_i^T\mathbf{w} + b) \geq 0 \tag{2.4}$$

for both $\mathbf{x}_i \in P$ and $\mathbf{x}_i \in N$. Amongst all different planes satisfying the above mentioned conditions, we aim to find the optimal one H_0 that divides the region for the two classes with the maximal distance between the decision plane and the closest sample points

The optimal plane must be in the middle of the two classes, so that the distance measured from the plane to the closest sample is equal on either sides. We define two additional planes H_+ and H_- that are parallel to H_0 and go through the point closest to the plane on either side:

$$\mathbf{x}^T\mathbf{w} + b = 1, \quad \text{and} \quad \mathbf{x}^T\mathbf{w} + b = -1 \tag{2.5}$$

All points $\mathbf{x}_i \in P$ on the positive side should satisfy

$$\mathbf{x}_i^T\mathbf{w} + b \geq 1, \quad y_i = 1$$

and all points $\mathbf{x}_i \in N$ on the negative side should satisfy

$$\mathbf{x}_i^T\mathbf{w} + b \leq -1, \quad y_i = -1$$

These can be combined into one inequality:

$$y_i(\mathbf{x}_i^T\mathbf{w} + b) \geq 1, \quad (i = 1, \cdots, m) \tag{2.6}$$

The equality holds for those points on the planes H_+ or H_-. These points or samples are defined as *support vectors*, for which

$$\mathbf{x}_i^T\mathbf{w} + b = y_i \tag{2.7}$$

i.e., the following holds for all support vectors:

$$b = y_i - \mathbf{x}_i^T\mathbf{w} = y_i - \sum_{j=1}^{m} \alpha_j y_j (\mathbf{x}_i^T\mathbf{x}_j) \tag{2.8}$$

Our aim is to maximize this distance, or, to minimize the norm $||\mathbf{w}||$. Therefore, the problem of finding the optimal hyperplane plane in terms of $\mathbf{w}$ and b can be formulated as:

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} = \frac{1}{2}||\mathbf{w}||^2 \qquad \text{(objective function)} \\ &\text{subject to} \quad y_i(\mathbf{x}_i^T\mathbf{w} + b) \geq 1, \quad \text{or} \quad 1 - y_i(\mathbf{x}_i^T\mathbf{w} + b) \leq 0, \qquad (i = 1, \cdots, m) \end{aligned}$$

2.2.3.1 SVM Formulation

As we have seen above, the linear view of support vector machines is used to find an optimal plane that divides the two classes equally. whereas in the general formulation, SVM finds a hyperplane in a different space than the input space x. This space will be induced by a Kernel K, in this case, it is a dot product between the feature vectors of the input space. The Kernel method induces hypothesis space which is now a set of different hyperplanes that divide the feature space.

The hypothesis space of the SVM is a subset of a set of hyperplanes defined in an RKHS which can be described as

$$\{f : ||f||_k^2 < \infty\} \tag{2.9}$$

where k is the kernel that defines the RKHS and $\|f\|_k^2$ is the RKHS norm of the function [28]. In the linear case, Kernel $K(x_1, x_2) = x_1.x_2$ is the dot product and the functions considered are of the form described above in the hyperplane equation for the linear one and RKHS norm is simply $\|f\|_k^2 = \|w\|^2$, norm of w. The main objective of the support vector machines here is to determine a solution that gives an optimal RKHS norm.

Therefore, support vector machines work towards minimizing the generalized error while taking the hypothesis space into consideration by minimizing the RKHS norm. During implementation we have the flexibility to perform a trade-off between error and the complexity of the hypothesis space through tuning hyperparameter such as C in equation 2.10. The SVM classification can be represented by the below optimization problem:

$$\min_{f} \|f\|_k^2 + C \sum_{i=1}^{j} |1 - yf(x)| \tag{2.10}$$

2.2.4 Implementation

Support vector machines algorithm requires answering the optimization mentioned in equation 2.10 which can be converted to a Quadratic programming problem. The quadratic programming formulation for support vector machine is given as:

$$\begin{aligned} \min_{f} \quad & \|f\|_k^2 + C \sum_{i=1}^{j} \xi_i \\ \text{s.t.} \quad & y_i f(x_i) \geq 1 - \xi_i \text{ for all i,} \\ & \xi \geq 0 \end{aligned} \tag{2.11}$$

Variables ξ_i are the slack variables which measure the error made at a point (x_i, y_i). In the above formulation, the number of constraints is equal to the number of observations in the training data. There have been several proposals to speed up the SVM training and one such method was the primal-dual optimization method which can be referred at [28]. SVM dual formulation :

$$\begin{aligned} \min_{\alpha_i} \quad & \sum_{i=1}^{j} \alpha_i - \frac{1}{2} \sum_{i=1}^{j} \sum_{l=1}^{j} \alpha_i \alpha_l y_i y_l K(x_i, x_l) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \text{for all i,} \\ & \sum_{i=1}^{j} \alpha_i y_i = 0 \end{aligned} \tag{2.12}$$

Here, K represents the Kernel that was utilized. This is generally mentioned as the kernel trick, this is used because it is intractable to project the sample space onto a higher dimensional space. So special kernel functions that operate on lower-dimensional vectors x_i and x_l to yield a value equivalent to the dot product of higher dimensional vectors.

2.3 K - Nearest Neighbors

2.3.1 Introduction

KNN is a non - parametric supervised learning algorithm and when we say non-parametric, it means that the algorithm does not make any presumptions on how the underlying data is distributed. The kNN method does not explicitly have a training stage and therefore during prediction stage we calculate the distance between the test data and all the training data to obtain its nearest neighbors and then conducting kNN classification where the algorithm assigns labels to the test data by majority rule on the basis of n - nearest neighbors to that test sample. Since kNN is simple to implement and provides significant classification performance, this method is very popular during data mining and in statistics [29].

2.3.2 Distance metric

In general, each feature vector can be considered as points or vector within an n- D space where each value of the n-dimensional vector corresponds to one of the features that describe the instance of the model. The position of these sample points in the space is not as important as the relative distance between them. This relative distance is determined by using a distance metric. Preferably we select the distance metric that minimizes the distance between two similarly classified instances and maximizes the distance between instances of different classes. There are many distance metrics available, the most notable ones are mentioned in figure 2.2.

2.3.3 Parameter Selection

To choose the best value of K we consult the dataset itself, generally a large value of k reduces the impact of noise on the classification model, but this can lead to boundaries with more overlap between different classes and hence it is advisable to test the performance of the classification model for different values of the hyper parameter through cross validation and then fixate on a single value. For our study, the dataset contain only a single point for each damage case, therefore we have fixed as equal to 1

Minkowsky: $D(x,y)=\left(\sum_{i=1}^{m}\left\|x_i-y_i\right\|^r\right)^{1/r}$
Manhattan: $D(x,y)=\sum_{i=1}^{m}\left\|x_i-y_i\right\|$
Chebychev: $D(x,y)=\max_{i=1}^{m}\left\|x_i-y_i\right\|$
Euclidean: $D(x,y)=\left(\sum_{i=1}^{m}\left\|x_i-y_i\right\|^2\right)^{1/2}$
Camberra: $D(x,y)=\sum_{i=1}^{m}\frac{\left\|x_i-y_i\right\|}{\left\|x_i+y_i\right\|}$
Kendall's Rank Correlation: $D(x,y)=1-\frac{2}{m(m-1)}\sum_{i=j}^{m}\sum_{j=1}^{i-1}sign(x_i-x_j)sign(y_i-y_j)$

Figure 2.2: Distance metrics used in KNN to calculate the similarity between two different feature vectors

2.4 Naive Bayes

2.4.1 Introduction

Naïve Bayes is another supervised learning algorithm considered for our research project. This algorithm is based on application of Bayes' theorem with the assumption of conditional independence between every pair of sample space given the value of the predictor variables. Bayes' theorem states the following relationship, given class variable y and dependent feature vector x_1 through x_n.

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)} \tag{2.13}$$

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y), \tag{2.14}$$

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)} \tag{2.15}$$

Since $P(x_1, \ldots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$
$$\Downarrow \tag{2.16}$$
$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

And we can use maximum a posteriori (MAP) estimation to estimate $P(y)$ and $P(x_i \mid y)$; the former is then the relative frequency of class y in the training set. The various naive Bayes classifiers differ mainly by the assumption on $P(x_i \mid y)$ is distributed. Naive Bayes algorithm generally is extremely fast compared to more complex methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn reduces the problems raised by the curse of dimensionality.

2.4.2 Bernoulli Naive Bayes

Bernoulli Naïve Bayes is generally implemented for data that is distributed according to a multivariate Bernoulli distribution i.e. there may be multiple features but many of them are assumed to be binary values. The algorithm implemented in Scikit learn 0.23.2, if given an input that is not binary the algorithm will binarize its input (depending on the binarize parameter).

The decision rule is given as:

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i) \tag{2.17}$$

In Bernoulli Naive Bayes as mentioned earlier, each feature is treated independently with binary values only, it explicitly gives penalty to the model for non-occurrence of any of the features which are necessary for predicting the output y. And the other multinomial variant of Naive Bayes ignores this features instead of penalizing.

2.5 Optimization Algorithms

Optimization algorithms are generally classified as stochastic algorithms or deterministic algorithms. Deterministic algorithms such as gradient based algorithms are best used with smooth, convex functions which encompass many local minima. These techniques are useful for deriving optimum values when we know the functional formulation of the problem. Unfortunately, in most real world problems, the function is complex and does not have a continuous formulation [1]. Hence we make use of stochastic algorithms for such problems, stochastic algorithms can be further classified based on the inspiration for the optimization, but for our study we have limited ourselves to Genetic algorithm, which is a bio – inspired optimization algorithm based on the Darwinian principle of evolution.

2.5.1 Genetic Algorithm

2.5.1.1 Introduction

As mentioned before Genetic algorithm is inspired from the Darwinian principle of evolution, i.e. based on the concept of natural selection between individual based on reproduction which causes the spread of their genes. Genetic algorithm works on a population which constitutes as several solutions, each solution is an individual and based on how a human is encoded, and each individual has a chromosome. The chromosome is represented as a set of parameters (features) that define the individual and each gene in turn is represent as combination of binary values. Now to differentiate between individuals, GA assigns a fitness value to each individual, selection based on their quality is applied to generate what is called the mating pool where the individual with higher fitness value has higher probability of being selected for mating. The individuals in mating pool are referred as Parents, and two are selected to generate offspring. By mating it is expected that the parents with high fitness value will generate offspring with even better fitness value. But there is a possibility that the generated offspring has lower fitness value, hence GA offers other operators such as crossover and mutation to generate completely new offspring and thus more chance of higher fitness value.

2.5.1.2 Selection

Selection operator is used to select which offspring will survive to the next generation. Tournament selection is a common operator used which randomly sorts individuals in different blocks and chooses the best ones from each block. Another operator used is the Truncation operator which simply chooses the fittest individuals from the parent and the offspring population.

2.5.1.3 Crossover

Crossover operator generates new generations the same way as natural mutation. By mutating the old generation parents, the offspring comes with genes from both parents. The amount of genes is random. Most crossover operators convert the individual into binary representation to perform the operations. One – point crossover crosses the binary digits at a random location of the two parents to create two new individuals.

2.5.1.4 Mutation

Mutation operator ensures genetic diversity amongst the generations and ensures non - convergence to a solution Mutation alters the gene values in a chromosome from its initial state. There is a possibility in which the solution generated through mutation may change entirely from the previous solution. Hence genetic algorithms are expected to arrive at a better solution through mutation. Mutation occurs during evolution according to a user-definable mutation probability. This probability should be set low. If the value is too high, the search will turn into a random search.

In conclusion to decide influence of these operators on the optimization model we employ an iterative approach and choose the parameters that allows extensive search of design space, is computationally inexpensive and finds an optimum location for which the error between predicted value and true value is low.

3. METHODOLOGY

3.1 Aim

The aim of our research project is to develop a model which can output an optimum set of locations for sensors to be placed on an unmanned aerial vehicle for accurate prediction of capability using Dynamic data driven applications paradigm, machine learning techniques, and heuristic optimization.

3.1.1 Objectives

- To create an offline library which will comprise of different kinematic states of the vehicle corresponding to damage states and capability
- Build a classification model using the offline damage library to predict aircraft capability for real – time data
- Implement an optimization algorithm on a dataset that emulates a real time strain information vector to find an optimum set of locations for the sensor for a specific aircraft mission.
- Explore the capability estimation and damage prediction through dynamic sampling of sensor information in real time

3.2 Combination of ASWING and Akselos modeler to simulate damage cases

As first step in our research approach we start by creating the offline damage library as mentioned before. Firstly we define a pull up and a turn maneuver in ASWING and note the structural loading of the wing along its span. The results for the simulation with different kinematic states can be seen in figure 3.1. The structural loading is an input to Akselos modeler to simulate the maneuver with the damage states to output strain information at the sensor locations.

Once we have both the software working in conjunction we next define a grid of sensors on the wing, the aim here is to maximize the area covered by the sensors and hence we define 1000

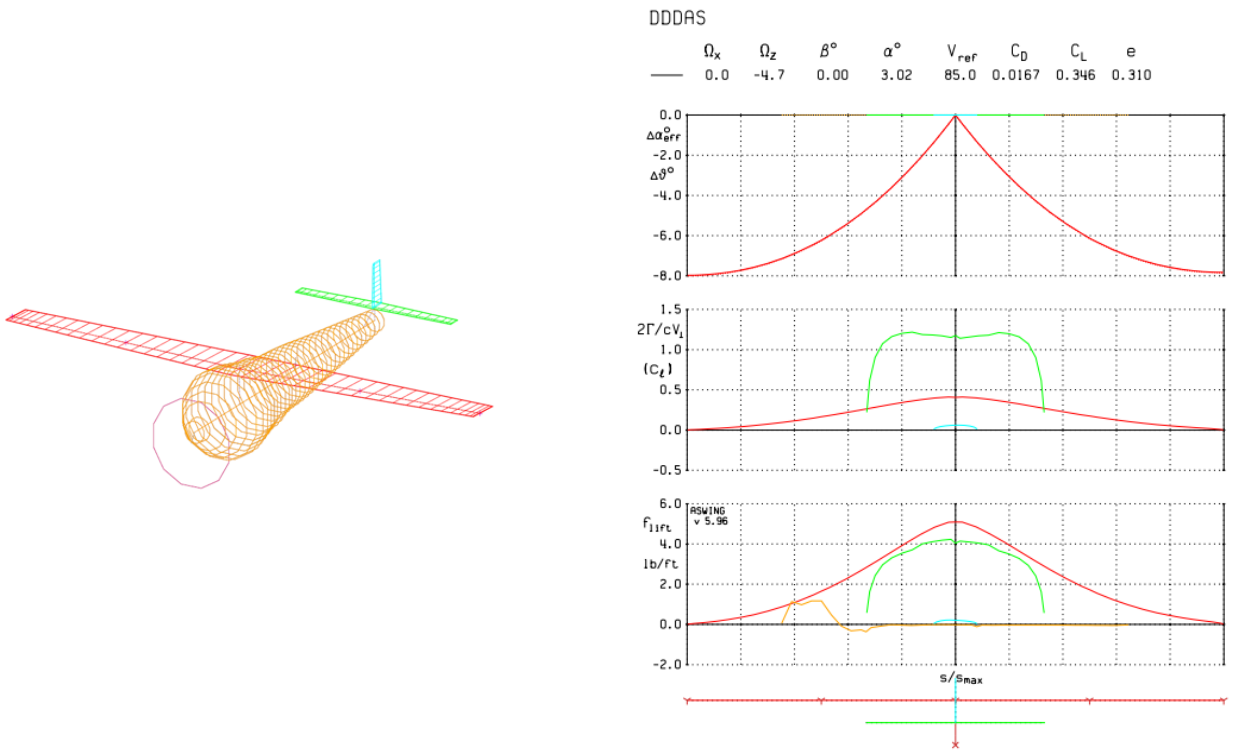


Figure 3.1: Our UAV represented in ASWING on the left. Plots on the right display the trim solution for a turn maneuver computed by ASWING at a specific velocity and load factor value

sensor locations on the top and the bottom of the wing to sample the strain information as shown in figure 3.2. Since the model uses shell elements, which are essentially 2 – dimensional elements (zero thickness) living in a 3 dimensional space, Akselos is able to calculate the in-plane strain tensor for the top, middle and the bottom of the shell element, even though the element technically has zero thickness. For our study we note the bottom strain and the top strain values along xx, xy and the yy axis because it is representative of the true strain values.

Next, we simulate the damage cases and note the strain readings as described above at the sensor locations. The values returned by Akselos are in micro strain and hence some of the values are of lower order, for simplicity of understanding and for the classification model we scale the values by 10^6.

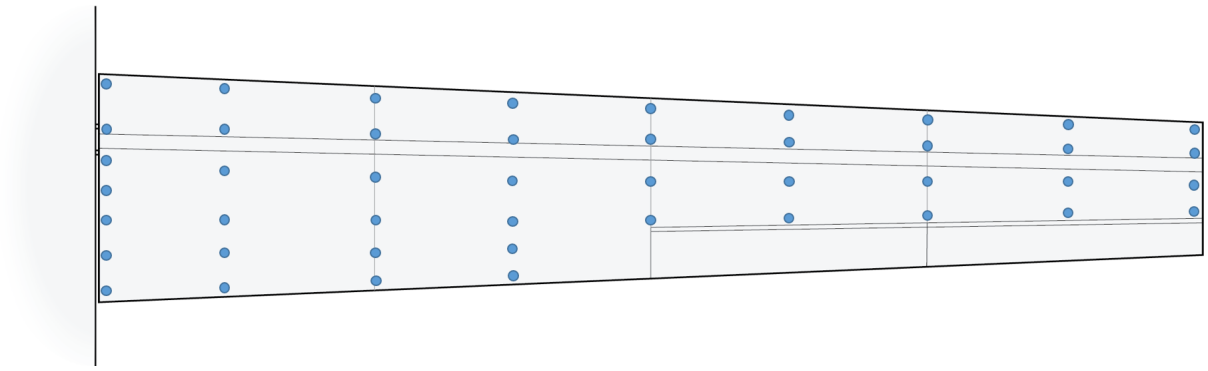


Figure 3.2: Sensor grid representation

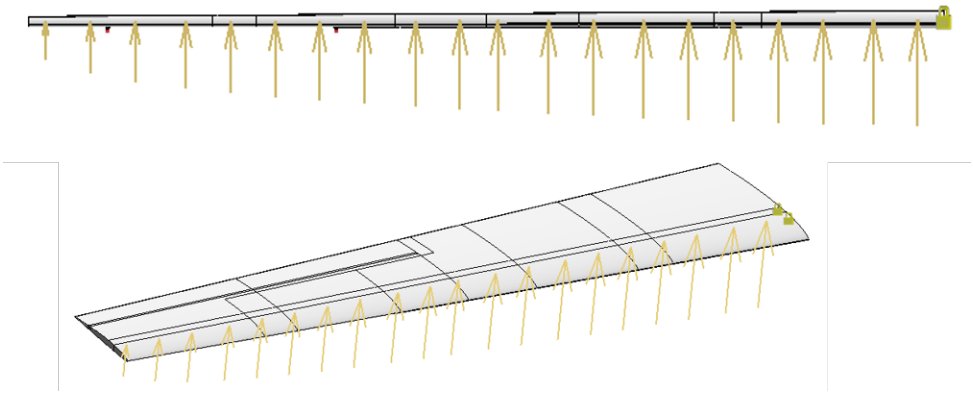


Figure 3.3: Representation of load values imported to Akselos modeler from ASWING

3.2.1 Capability calculation

After creating the framework described above, to complete the offline library we also need to estimate the true capability of each damage case. Akselos amongst its various solution fields has a parameter defined as the composite failure index, which is an indicator of the structural condition that is translated into a scaling factor for maneuver parameters. Failure index (FI) is defined as the ratio between the experienced stress σ and the maximum allowable stress σ_{max} (i.e., the compression/tension/shear strengths that are characteristic properties of the material) for each element in the FEM mesh. For each ply, five main orthotropic axes of the ply (C_{n1} and C_{n2}), tension along the main orthotropic axes of the ply (T_{n1} and T_{n2}), and the shear on ply plane (S_{s12}).

The application of these criteria to the stress field, obtained by running the FEM model of the panel, provides a set of FI(x) values for each discrete plate element (e = 1,. . . , E) for each ply (ply = 1,2,. . . 6) and for each failure mode (mode = $C_{n1}, C_{n2}, T_{n1}, T_{n2}, S_{s12}$) considered [30] :

$$FI(e, ply, x) = \frac{\sigma(e, ply, x)}{\sigma_{max}} \tag{3.1}$$

As mentioned in section 1.7, through the root finding algorithm we tabulate the capability values for all damage cases in the library. In terms of effectiveness of the metric, since load factor values are quite small and hence does not provide much intuition between the dissimilarity of the damage cases. Therefore we calculate the turn radius using the load factor values from equation 3.2 because the turn radius is a more robust metric for capability estimation.

$$TurnRadius = \frac{V^2}{g \times \sqrt{n^2 - 1}} \tag{3.2}$$

Where: V = Indicated air speed in (m/s)

g = Gravitational constant in (m/s^2)

n = Load factor

3.3 Building the Classification Model

We have established our approach to create the offline library and now we build our classification model which can predict the capability of a damage case through real time or dynamic strain information. The flowchart in figure 3.4 is a representation of the general scheme that we have followed to build our classification model. The dataset returned by Akselos contains the strain information corresponding to the sensor locations and we have a strain tensor for each sensor location with the top skin and bottom skin in – plane strain values. But to emulate the dataset that we receive in real time we convert the Akselos dataset into a single vector in which we stack the sensor information one after the other thus creating a high dimensional feature vector corresponding to a damage case, where the damage case represents the class in the training data. Due to this way of

data wrangling, the dataset contains 50 classes and a single feature vector comprising of the strain information corresponding to each class. Because of this we opt against the traditional train and test split approach with cross validation to estimate the model performance, instead we train our model on the whole dataset and estimate its performance on a "noisy" dataset which emulates a real time dataset. We model noise through Monte Carlo sampling, where we define a Gaussian distribution for each value in the feature vector with the current value as the mean and noise level of (5%, 10%, and 15%) as the standard deviation. Next we randomly sample from the defined Gaussian distribution and replace the current value with the sampled value to create the noisy dataset. This estimate of the standard deviation is obtained from published data that high-accuracy strain gages often have a 2–5% accuracy range when properly calibrated [31]. But for our model we have considered much higher noise levels to create a robust models to account for unexpected environmental conditions and higher noise levels due to damage in the sensors.

After data wrangling, we build the classification model as described in section 2 and perform a comparison study to select the model that gives maximum prediction accuracy and gives more information about the health of the system. In our study the noisy dataset is a representation of the real time dataset, but since the noise that we add is randomly sampled from a Gaussian distribution, hence there is a possibility that the classification model performs well for particular noise and performs poorly for another case. Therefore to maneuver around this issue we test the classification model on 50 different noisy data sets to prevent the model from over-fitting. We do this for all levels of noise selected for our study. In the comparison study we also incorporate the effect of number of sensors on the classification model, where we sample a certain number of sensors in each iteration to build and test the model accuracy.

Figure 3.5, 3.6, 3.7 shows the plot with percentage prediction accuracy and the number of sensors for different levels of noise. We can note from the plots that KNN and SVM model have very similar performance in terms of the classification accuracy and Bernoulli naïve bayes clearly outperforms KNN and SVM as percentage classification accuracy is much higher for 800 to 50 sensors. Another thing to note is that the performance is quite similar for lower number of sensors.

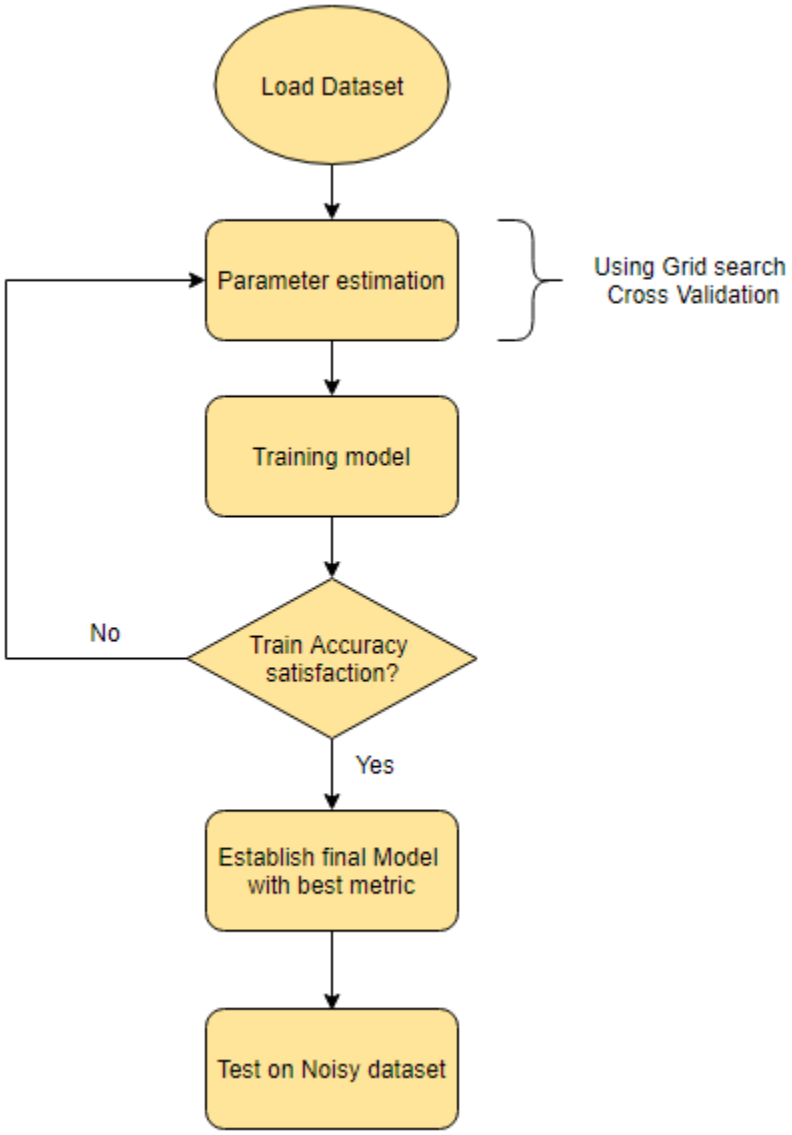


Figure 3.4: Classification model building scheme followed for our research study

This can attributed to the fact that our design space is vast because the number of combinations of choosing 5 sensors out of 1000 is quite large and hence it is not feasible to explore them in only 50 runs and hence on an average we expect a lower prediction accuracy.

Based on this comparison study we selected naïve bayes classifier for the optimization model to help predict the damage case for the strain information corresponding to different locations in each iteration.

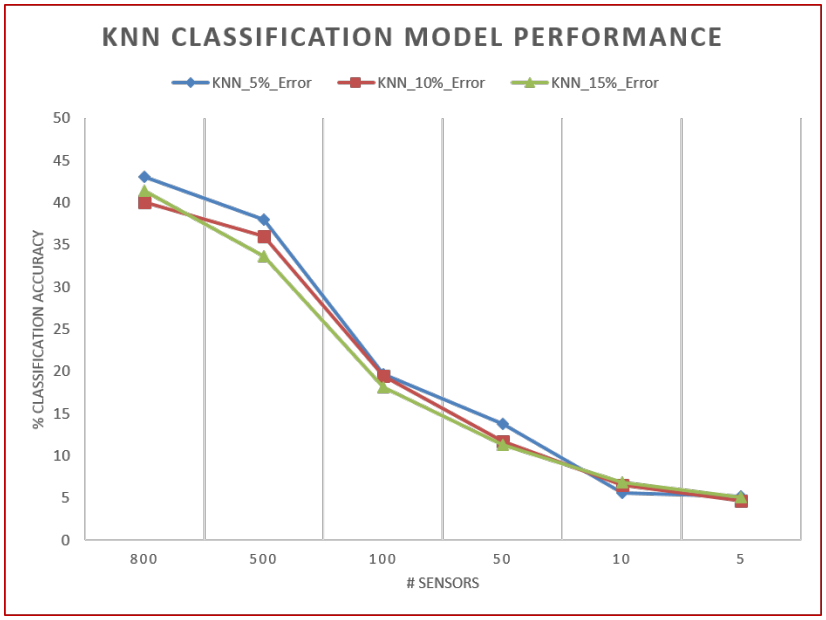


Figure 3.5: Comparison between classification accuracy and number of sensors for three levels of noise using K - Nearest Neighbors classifier

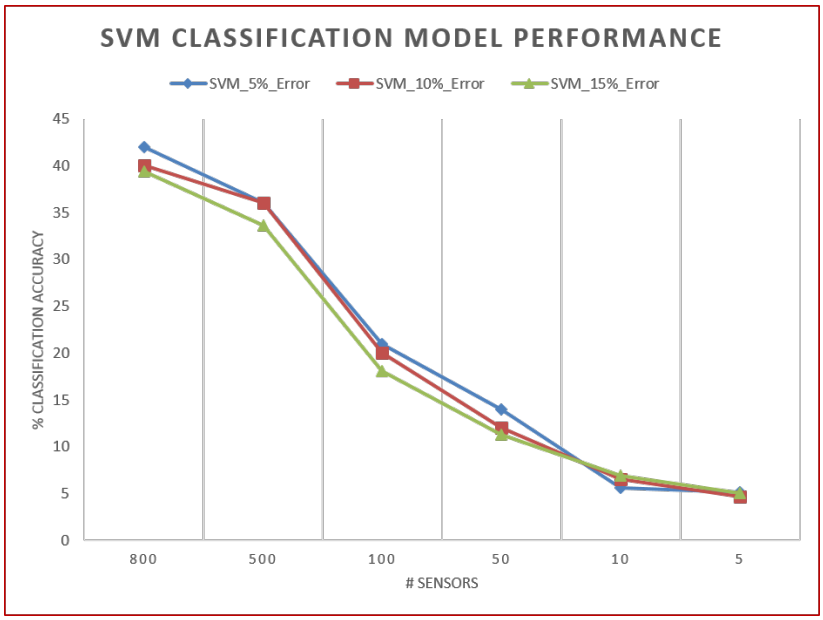


Figure 3.6: Comparison between classification accuracy and number of sensors for three levels of noise using support vector machines classifier

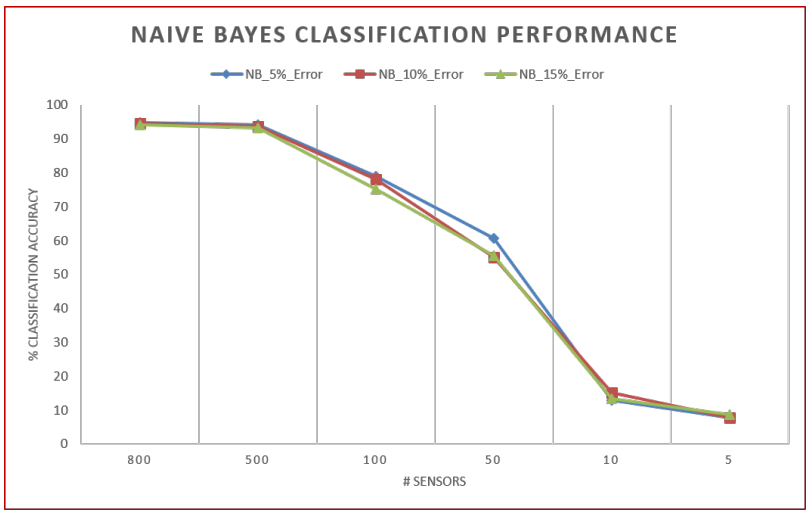


Figure 3.7: Comparison between classification accuracy and number of sensors for three levels of noise using Naive Bayes classifier

3.4 Optimization Methodology

For our problem statement we selected a Meta heuristic algorithm in Elitist genetic algorithm to obtain the optimum sensor location, figure 3.9 shows the implementation of the algorithm in a flowchart. Our optimization problem is defined as:

Objective Function:

$$MSE = \frac{1}{m}\sum_{j=1}^{m}(y_{true,j} - y_{predicted,j})^2 \tag{3.3}$$

Where:

m = Represents the number of damage case in the offline library

y_{true} = Represents the set of true capability values

$y_{predicted}$ = Represents the set of predicted capability value

Design Variables: Sensor locations and we can see in figure 3.8 a sample grid of sensors defined along the span of the self aware UAV

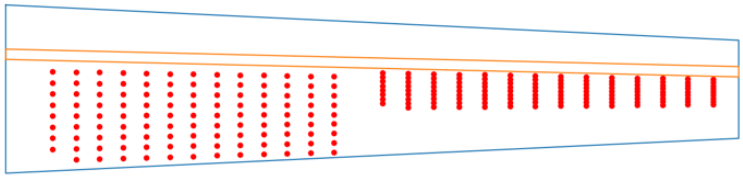


Figure 3.8: Representation of sensor locations defined on the wing of our UAV

Design Parameters: Amount of sensors defined on the surface of the win

The objective equation can be formulated as the mean squared error between the true and predicted value of capability. For example if our aim was to optimize for a single damage case then in a single iteration the optimizer would sample the data corresponding to a certain location from the true dataset, train the classification model, sample the data corresponding to the same locations from the noisy dataset and make use of the classification model to predict a damage case. The objective function returns the mean squared error between the true capability and the predicted capability value. Similarly if we were to optimize for let's say 10 damage cases, the objective function would return the maximum mean squared error between the true and predicted capability. This simply means that if we find the optimum location corresponding to the maximum error, the error for the damage cases will automatically be lower for that optimum sensor location. We selected elitist genetic algorithm for our optimization model due to three reasons, firstly because our design space is discrete, as we have a specific set of locations for the sensors, secondly, because the elitist implementation of genetic algorithm prevents loss of information as best members form the current iteration are passed onto next one without any mutation or crossover. Thirdly, since the design space is extremely large, since we are optimizing for a fixed number of sensors (5 or 10) the number of combinations for selecting 5 sensors out of a 1000 locations is quite large and hence it is better to explore the design space through a population of solutions rather than individual

solutions.

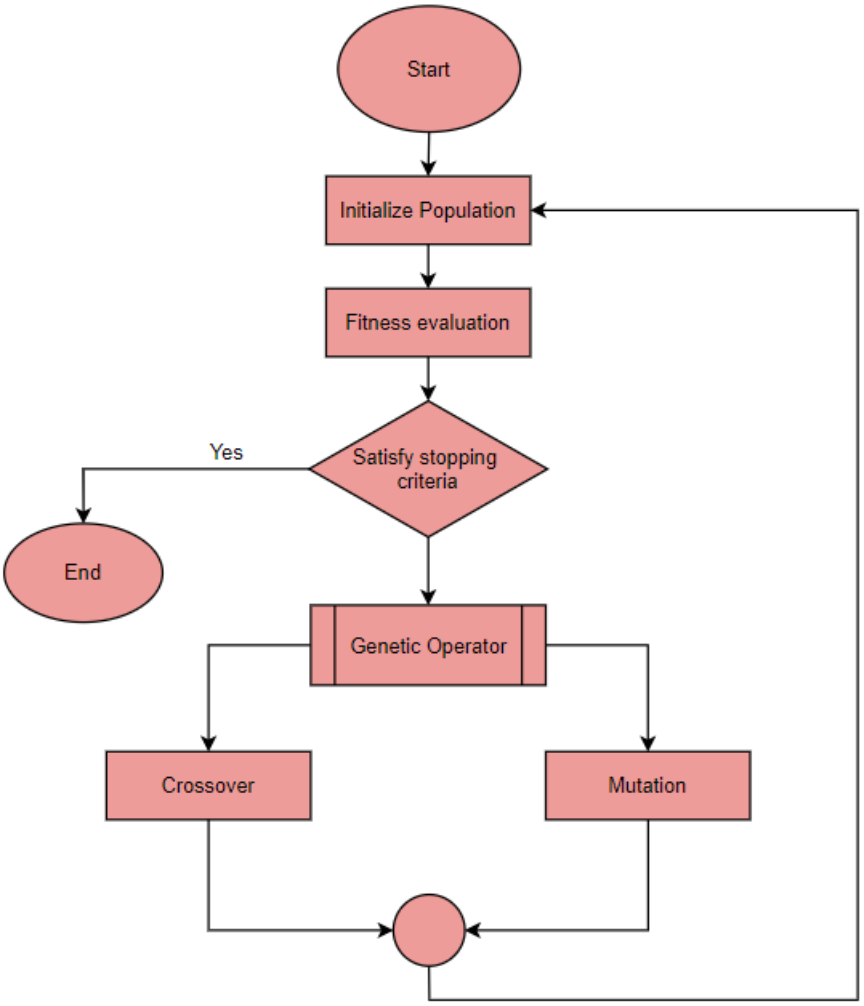


Figure 3.9: Implementation of Elitist genetic algorithm followed for our research study

3.5 Dynamic Sampling of Information in real time from optimum location of sensors

One of the objective as mentioned earlier was to study the how dynamic sampling of information would affect the prediction process. With the advent of smart strain sensing skins, it is possible to sample the information from anywhere on the wing of the self – aware UAV, Therefore an argument can be made that why not sample the data from everywhere on the wing, the issue with sampling strain information from all the nodes is that it will lead to a high amount of redundant

data due to neighboring nodes having similar values of strain and also there will be a requirement of high computational power for data processing such a large dataset, hence it is more reasonable to sample data from nodes that can give us a more clear intuition about the health and capability of the self – aware UAV. To implement this we created an algorithm as given below, where we first find the optimum location for each damage case. So if we assume a damage case from the offline library has occurred in real time then the algorithm would start at the initial optimum position based on offline calculations and predictions, then it would classify the real time strain information received from the strain sensors and update the location to the optimum location corresponding to the predicted damage case. This algorithm will run until a damage case classifies into itself or if there are oscillation where different damage cases are predicting each other, then the algorithm stops when a certain damage case has been visited certain number of times. This method is considered to be an efficient way of predicting the damage case because instead of randomly searching the grid of sensors we are selecting information from optimized locations which gives us more accurate prediction about the health of the UAV and its capability.

3.5.1 Algorithm for dynamic sampling information

The pseudo code for algorithm 1 is given below as implemented in Python. The efficiency of the algorithm can be improved through a better second stopping condition by adding more damage case and noting the oscillation between different damage cases while the algorithm determines next optimum location.

Algorithm 1 Pseudo - code for dynamic sampling of information from optimum locations

```
Load dictionary with all damage cases and corresponding optimum locations
Prepare noisy dataset that emulates real time strain information
Set sensor location to the optimum placement for all damage cases
Set a variable Flag with Boolean value as True
Initialize an empty dictionary
Initialize an empty list
while Flag is True do
    Sample data at the initialized location from the dataset with no noise (True dataset)
    Train Bernoulli Naive Bayes classifier using true dataset
    Sample data at the initialized location from the noisy dataset
    Use the learning model to predict a damage case
    Keep record of the damage cases predicted by the learning model and its count in dictionary

    Stopping Condition 1
    if a damage cases has been visited a certain number of times then
        break
    end if

    if list is empty then
        Append the predicted damage name to list
        Update location to optimum location corresponding to the predicted damage case
    else
        if Damage case exists in list then
            Change Flag value to False
        else
            Update list with new damage case
            Update location corresponding to predicted damage case
        end if
    end if
end while
```

4.1 Illustrative Results

In this section we have shown sample simulation on a damage case in figure 4.1 and a run of our optimization model in figure 4.2. For our study we have not used the graphic user interface that Akselos offers, because it is much slower and does not allow us to simulate more damage cases together. Instead we use the python scripts to allow for much faster simulations. Similar simulations have been carried out for all the damage cases in the offline library to compile the strain data and capability values.

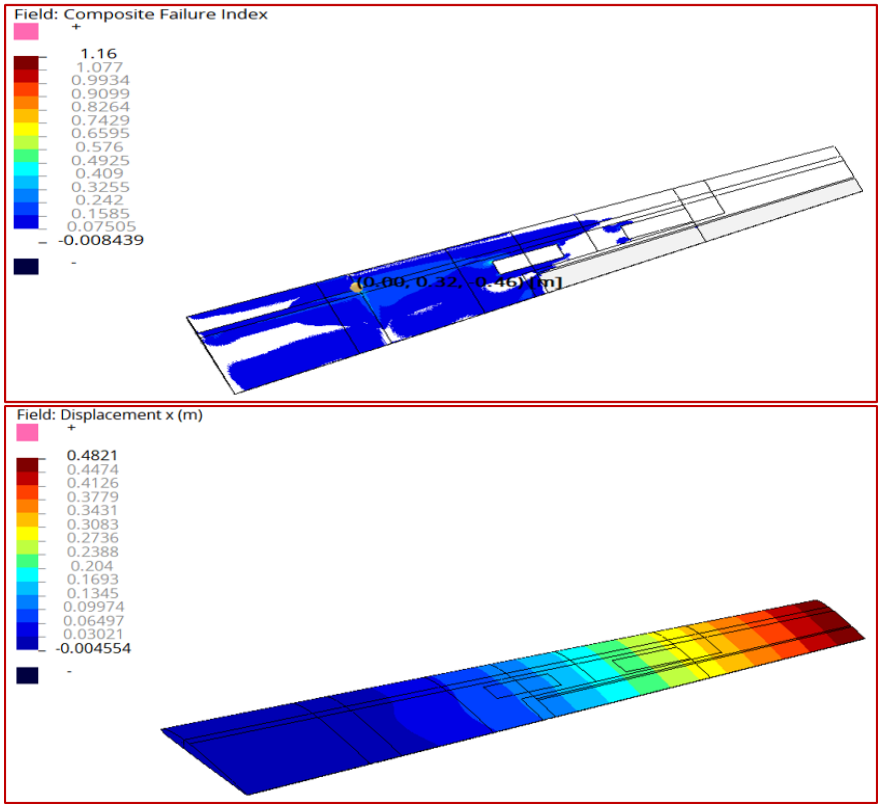


Figure 4.1: Composite failure index (top) and displacement along x axis (bottom) simulation representation on a damage case in Akselos

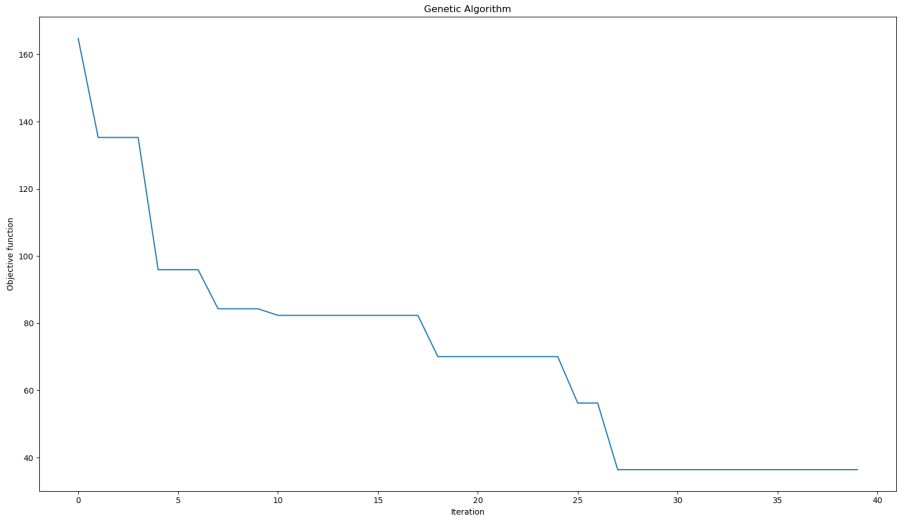


Figure 4.2: Objective function history plot using genetic algorithm

4.2 Comparison study between the performance of optimum sensor location and standard placement of sensors

To establish the improvement in prediction accuracy through optimum sensor placement we perform a comparison study between the optimum location and the standard placement of sensors opted by aurora flight sciences for their unmanned aerial vehicle. Figure 4.3 shows the standard placement with 13 sensors along the wing span. Figure 4.4 is another variation of the standard placement of sensors where we have shifted all the 13 sensors towards the trailing edge of the wing.

In our study since the noise is added through random sampling as described before, we perform multiple iterations with different random samples to estimate the average mean squared error to estimate the performance of the standard placement of sensors.

Firstly to find a single optimum location that offers low prediction error or low mean squared error between the true and the predicted value of capability, we use the optimization model to find the sensor placement for different data sets with same levels of noise i.e. For 5%, 10%, and 15%

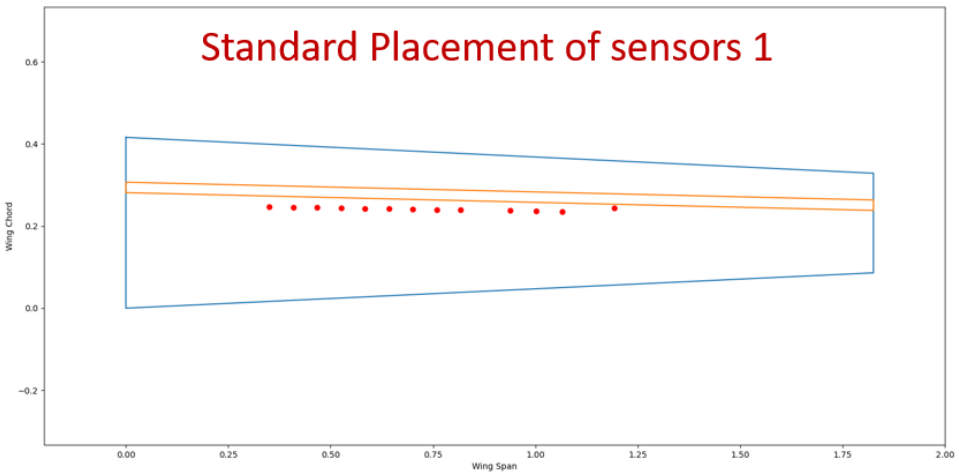


Figure 4.3: Standard placement of sensor type 1

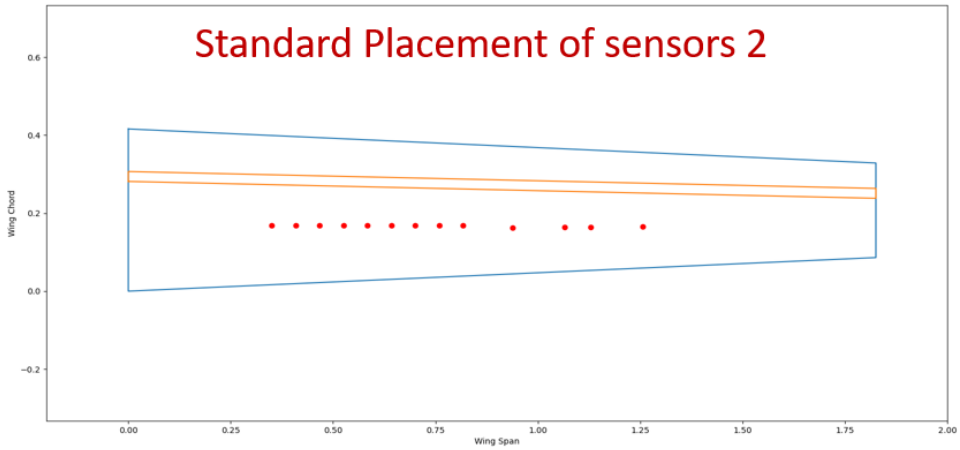


Figure 4.4: Standard placement of sensor type 2

we optimize on 7 different data sets each to find the optimum sensor placement for each case. Through this we end up with a set 21 potential locations and therefore to select 1 out of 21 we test all the locations on 20 different data sets with varying noise levels from 5 – 15% and select the one location that gives us minimum average mean squared error. figure 4.5, 4.6, and 4.7 shows a comparison between the average error for optimum sensor placement and standard placement of sensors for different levels of noise. We can note from figure 4.5, 4.6, and 4.7 that the estimated error is much lower for the sensor placement suggested by the optimization model as compared to

standard placement 1 and 2. As the level of noise increases we can see that the prediction error does not vary much for the standard locations, but for the optimum location the error increases with increase in noise, this rise is higher for the sensor array with 5 sensors as compared to 10 sensors, which is in concurrence with the earlier comparison between the accuracy of prediction and the number of sensors. Hence it is recommended to use the optimum sensor placement as compared to the standard placement of sensors.

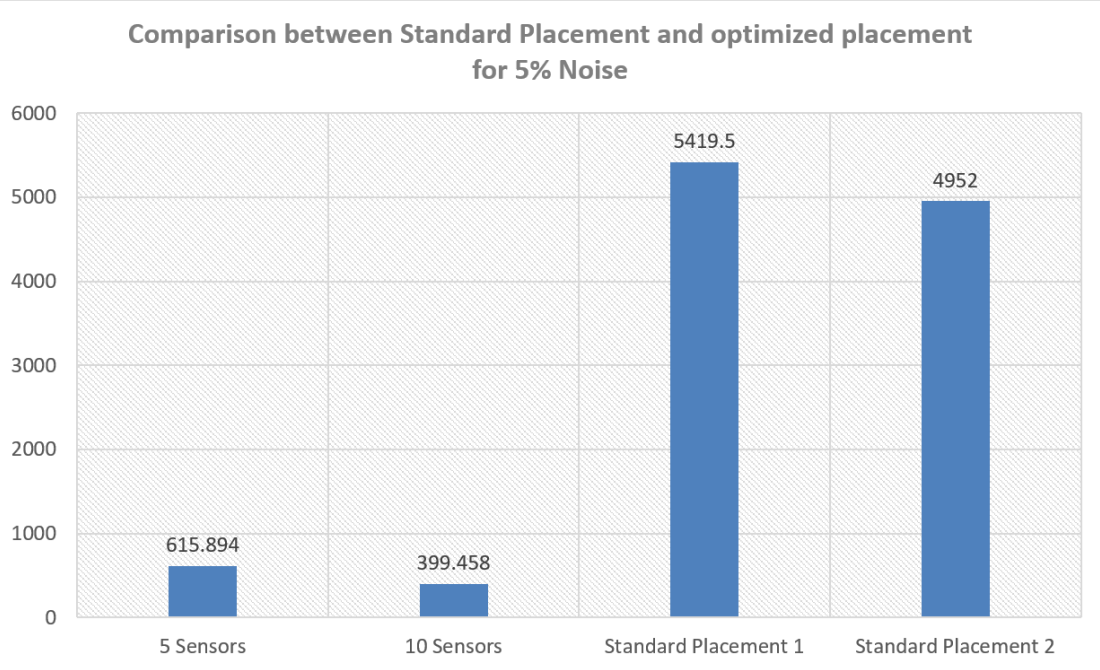


Figure 4.5: Comparison in performance between optimum location and standard placement with 5% noise level

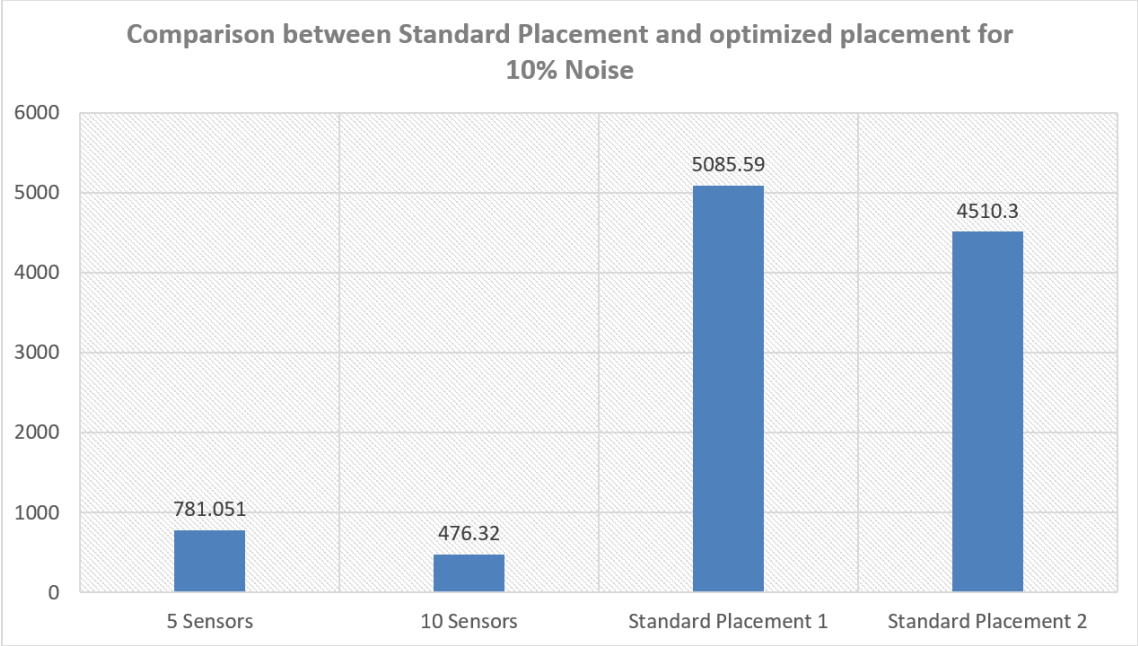


Figure 4.6: Comparison in performance between optimum location and standard placement with 10% noise level

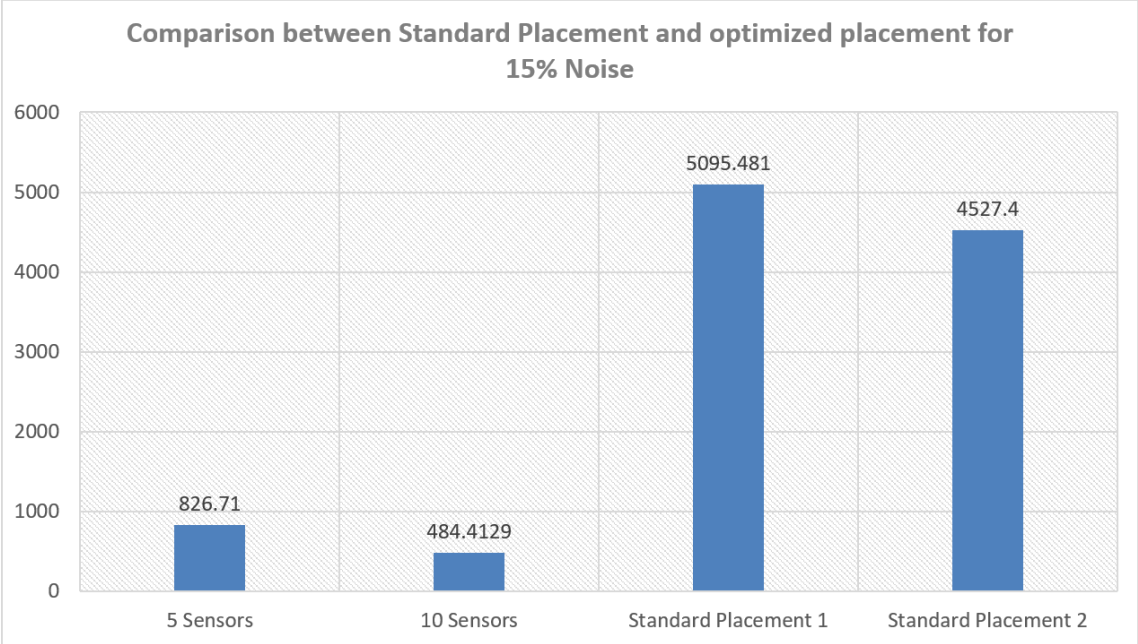


Figure 4.7: Comparison in performance between optimum location and standard placement with 15% noise level

4.2.1 Resultant Optimum Location from comparison study

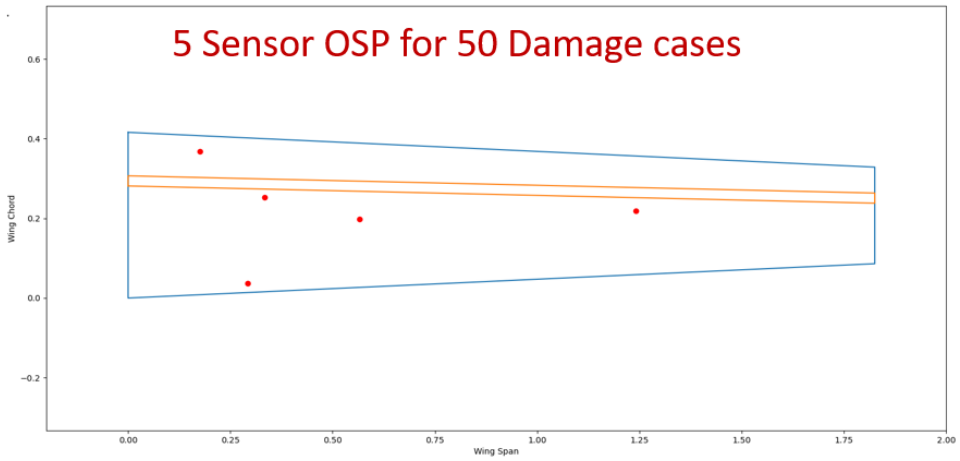


Figure 4.8: Optimum sensor placement for 5 sensors that outputs minimum prediction error for all damage cases

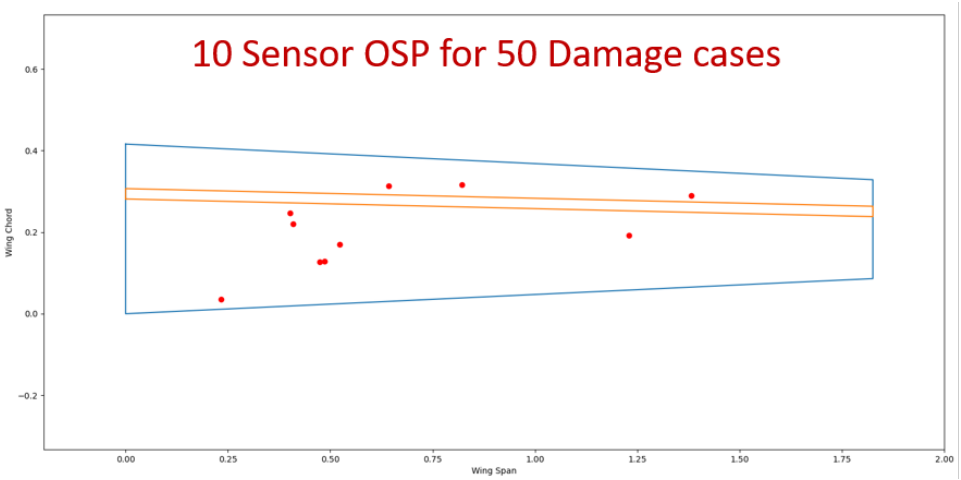


Figure 4.9: Optimum sensor placement for 10 sensors that outputs minimum prediction error for all damage cases

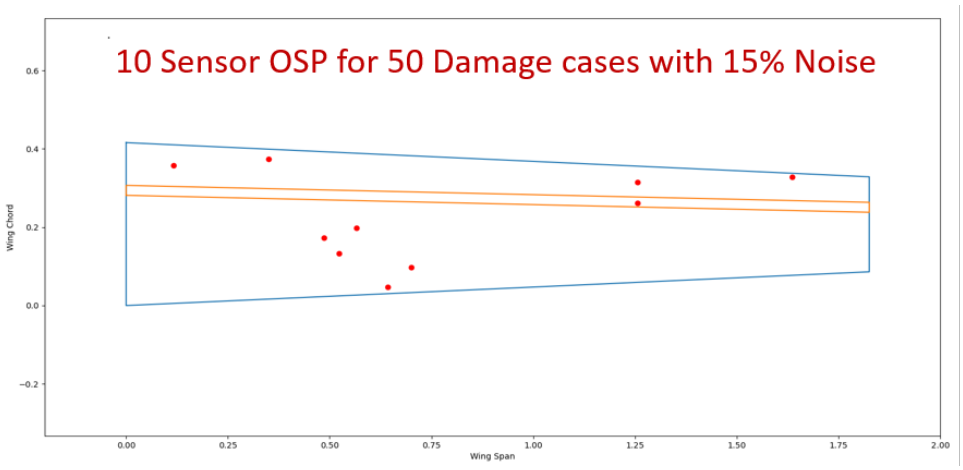


Figure 4.10: Optimum sensor placement for 10 sensors that outputs minimum prediction error for all damage cases with 15% noise level

4.3 Comparison of performance of sensor placement based on occurrence of predicted case versus random damage case

4.3.1 Offline library with 5 damage cases

The comparison presented above makes the underlying assumption that the damage case that will occur in real time will also be present in the offline library, hence there is a possibility that the real time damage case is not present in the offline library, to test out this scenario we performed two different studies, in first case we determined the optimum sensor placement for a specific set of 5 damage cases and tested the performance of that location on a set of 5 different damage case. In figure 4.11 we present the expected performance of the optimum sensor location if the damage case that occurs in real time is present in the offline library, as expected the performance on average is noticeably better than the standard placement of sensors. In figure 4.12 we present the comparison between the average error and the sensor placements if the damage case that occurred in real time was not present in the offline library. Due to an inaccurate assumption of expected damage case, we can see that standard placement 1 (in Fig 4.3) outperforms the optimum location and this asserts that if our initial assumption is incorrect or we find the optimum location for a small set of damage cases then the prediction accuracy will suffer and we will not gain any useful information about

the health of our system.

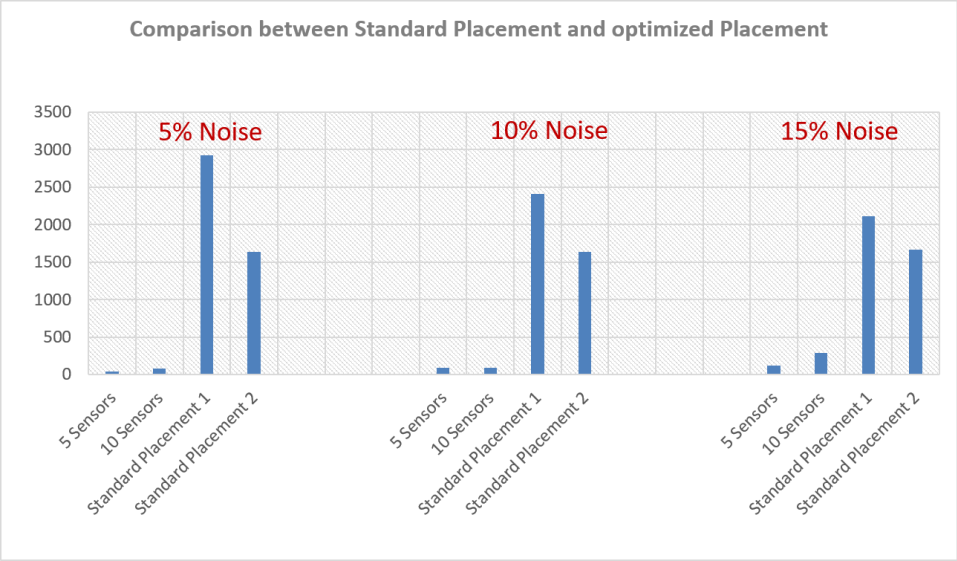


Figure 4.11: Expected performance of optimum sensor placement if initial assumption of damage library is correct

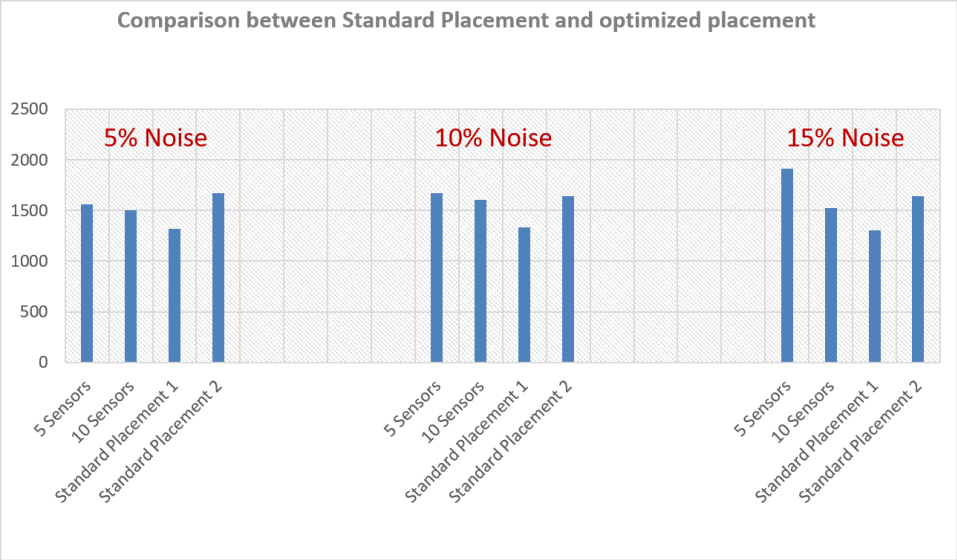


Figure 4.12: Achieved performance through optimum placement if initial assumption of expected damage case is incorrect

4.3.2 Offline library with 15 damage cases

In the second study we have increased the number of damage cases in the offline library to 15, hence we are assuming that one of the these 15 damage cases will occur in real time. Fig 4.13 illustrates the expected performance of optimum placement and standard placement of sensors, we can note that if our assumption of expected damage cases is correct, then optimum sensor location outperforms the standard sensor locations. Fig 4.14 illustrates the performance of optimum location and standard location if our initial assumption is incorrect and random damage cases occur. In previous section, from fig 4.12 we noted that standard placement 1 gave better results when tested on a random set of damage cases. But we can note from fig 4.14 that optimum placement with 10 sensors performs better in general even on a random dataset, which implies that if we were to improve our damage library with more damage cases, the optimum sensor placement would perform reasonably well even on damage cases not seen in the offline library.

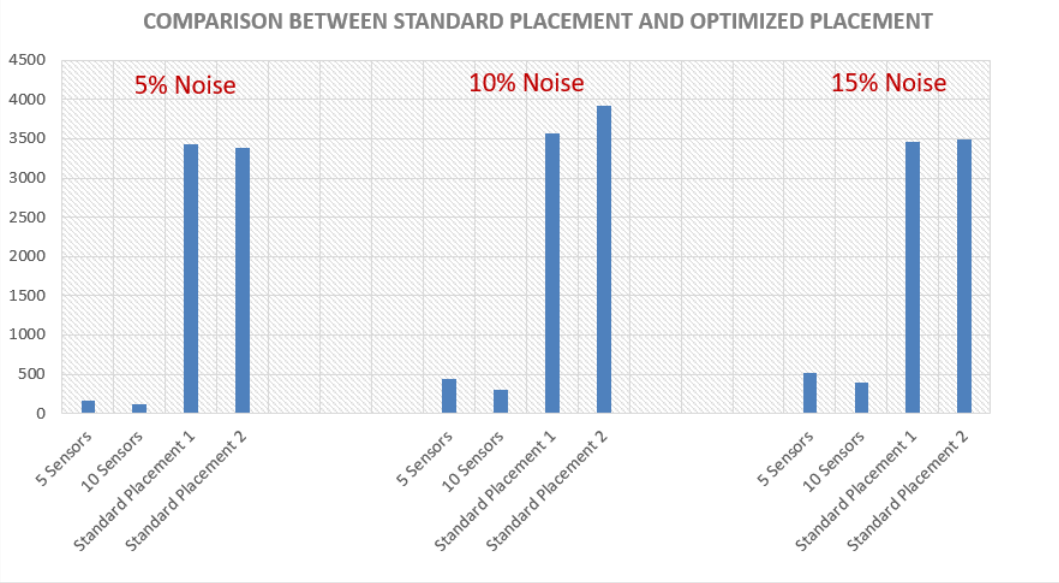


Figure 4.13: Expected performance of optimum sensor placement if initial assumption of damage library is correct

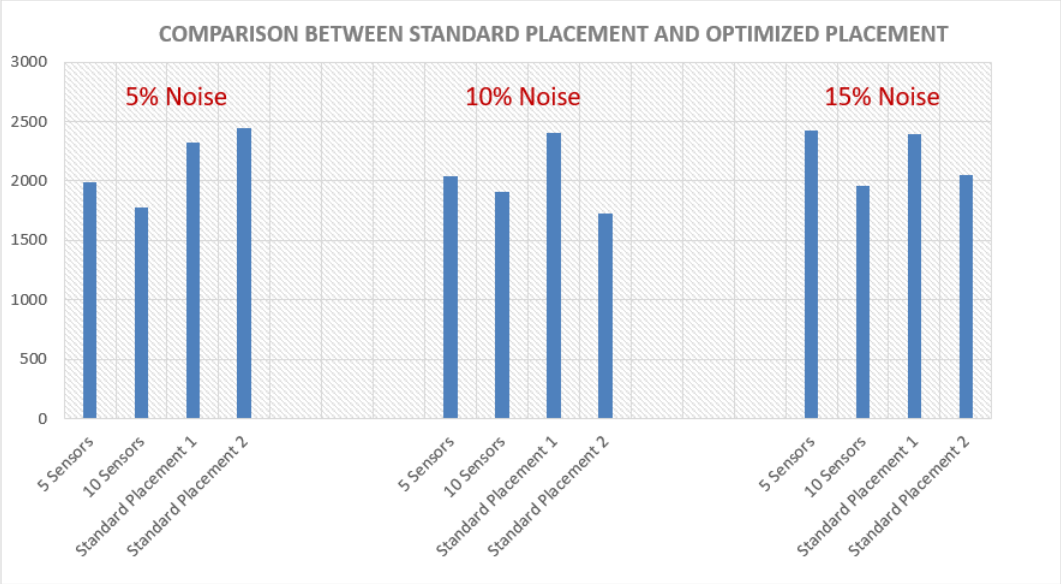


Figure 4.14: Achieved performance through optimum placement if initial assumption of expected damage case is incorrect

4.4 Prediction results through dynamic sampling of information with 5 & 10 sensors

As per the algorithm described in section 3 to find the damage case through dynamic sampling of information from different locations on the wing, we tested the performance and accuracy for optimum location with 5 and 10 sensors. For 5 sensors out of 50 damage cases the dynamic sampling approach was able to predict 23 of the correctly, out of the 27 that were not predicted correctly, the true capability value was close the predicted value for 25 of them, as for the last 2 cases the predicted value was far from the true capability. As for 10 sensors 33 out of 50 cases were predicted correctly and for the other 17 cases the predicted capability was very close to the true capability value and there were no outlier cases for 10 sensors.

We have presented the initial and final location of sensors as suggested by the algorithm for different cases that can be encountered during dynamic sampling of information. **Case 1**. as shown below depicts the case where we the algorithm is able to find the correct damage case through dynamic sampling.

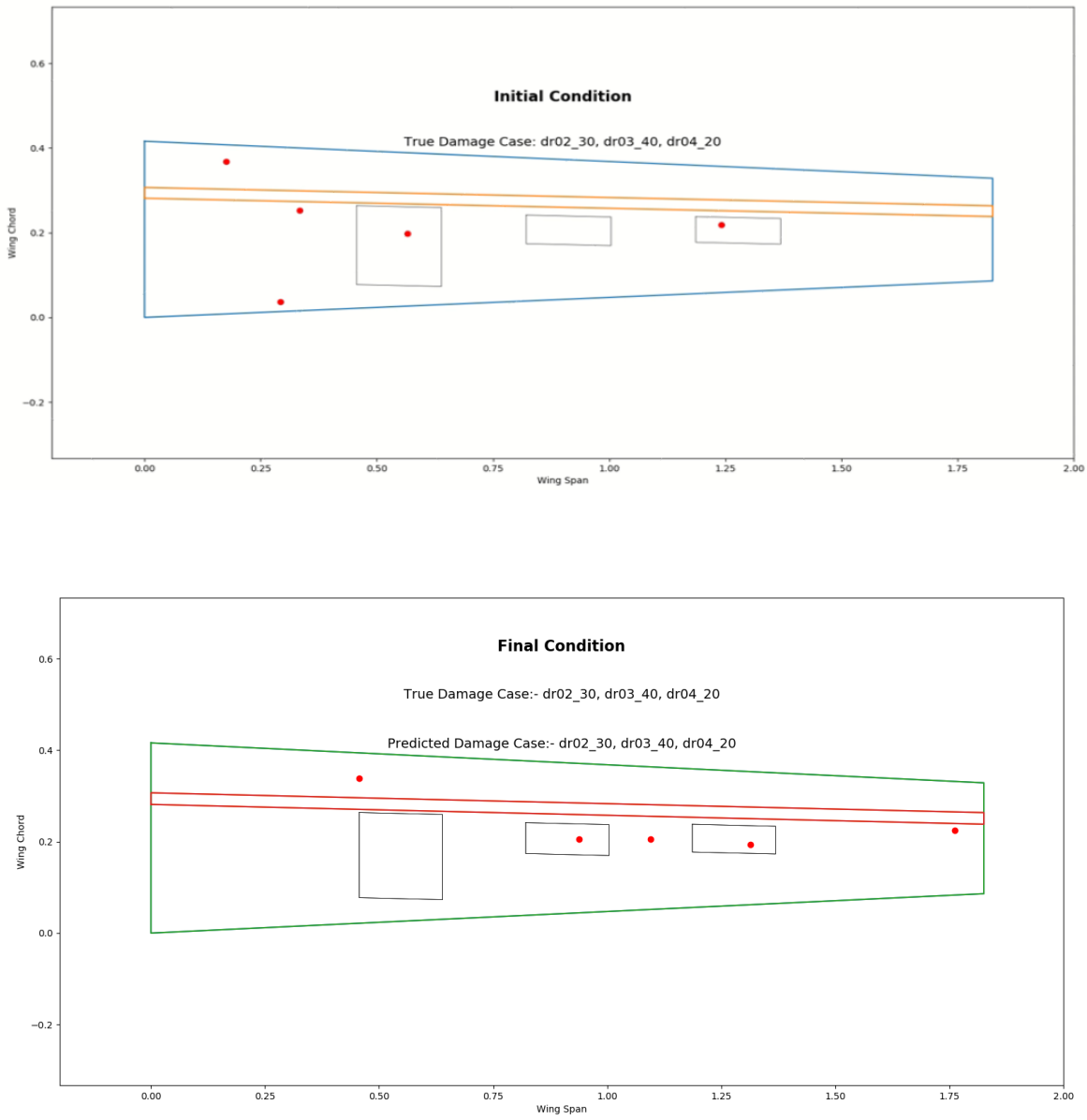


Figure 4.15: Figure represents the initial and final placement of sensor using dynamic sampling of information method where the algorithm is successfully able to identify the true damage case

Therefore the algorithm is suggesting that it is better to sample from the sensor placement shown in bottom in the fig 4.15 because that location provides more useful information about the health of the self-aware UAV.

Case 2. Presents a case where the predicted damage case is incorrect, but the capability value is close to the true capability value. As we can see from fig 4.16, the true damage case is dr07_20,

dr03_70, and dr09_30 (notation described earlier in section 1.4) and the predicted damage case is dr07_20, dr03_50, and dr04_20. In terms of damage location the algorithm predicts two correctly but there is still some difference in terms of material stiffness value but more importantly capability value is close to the true capability value as listed below.

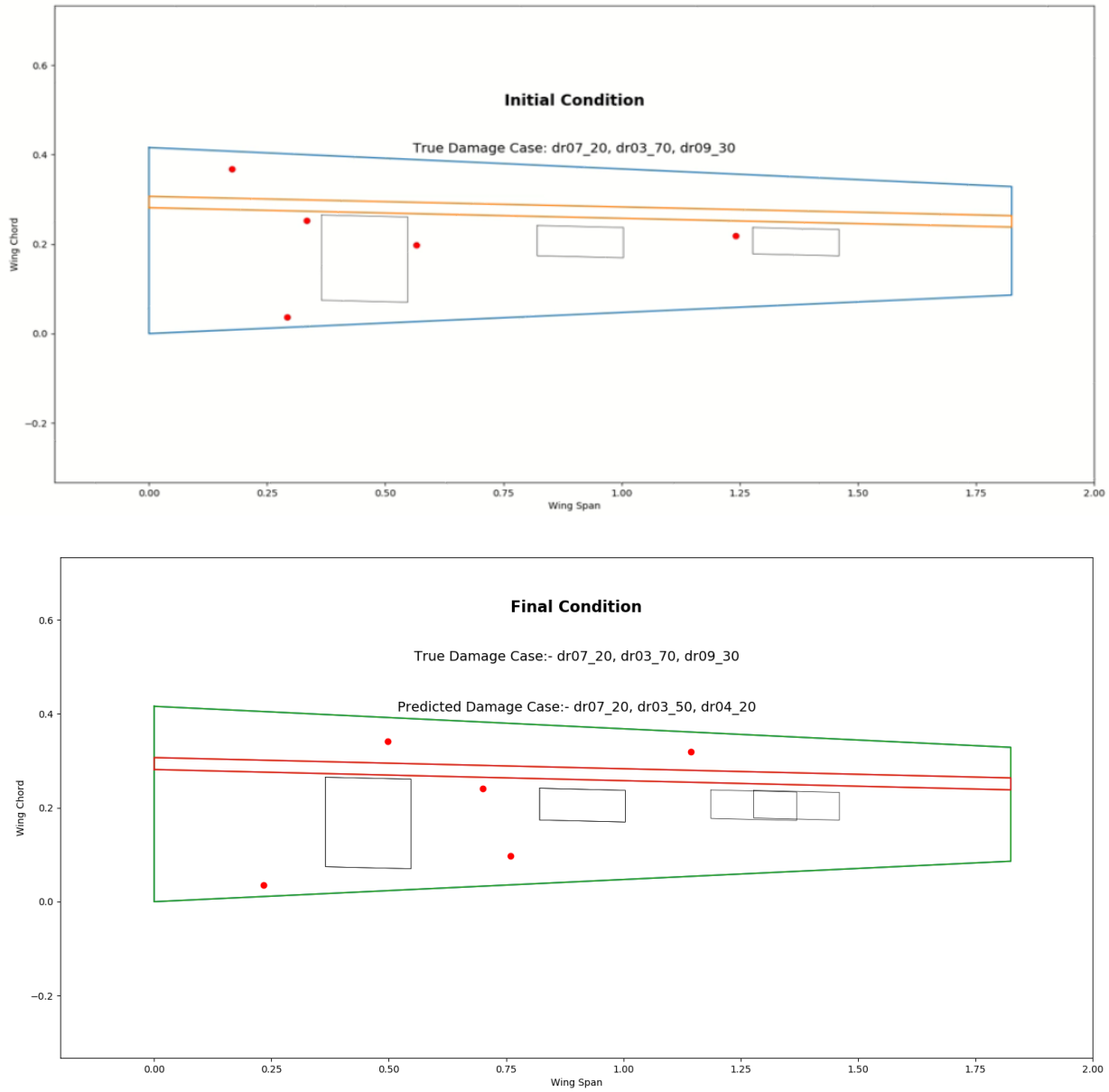


Figure 4.16: Figure represents the initial and final placement of sensor using dynamic sampling of information method where the algorithm is not able to identify the true damage case but the error between true capability and predicted capability is low

Case 3. Presents a case where the predicted damage case is incorrect and the capability value is far from the true capability value. This can be attributed to the difference between the material stiffness and locations of the true damage and the predicted damage.

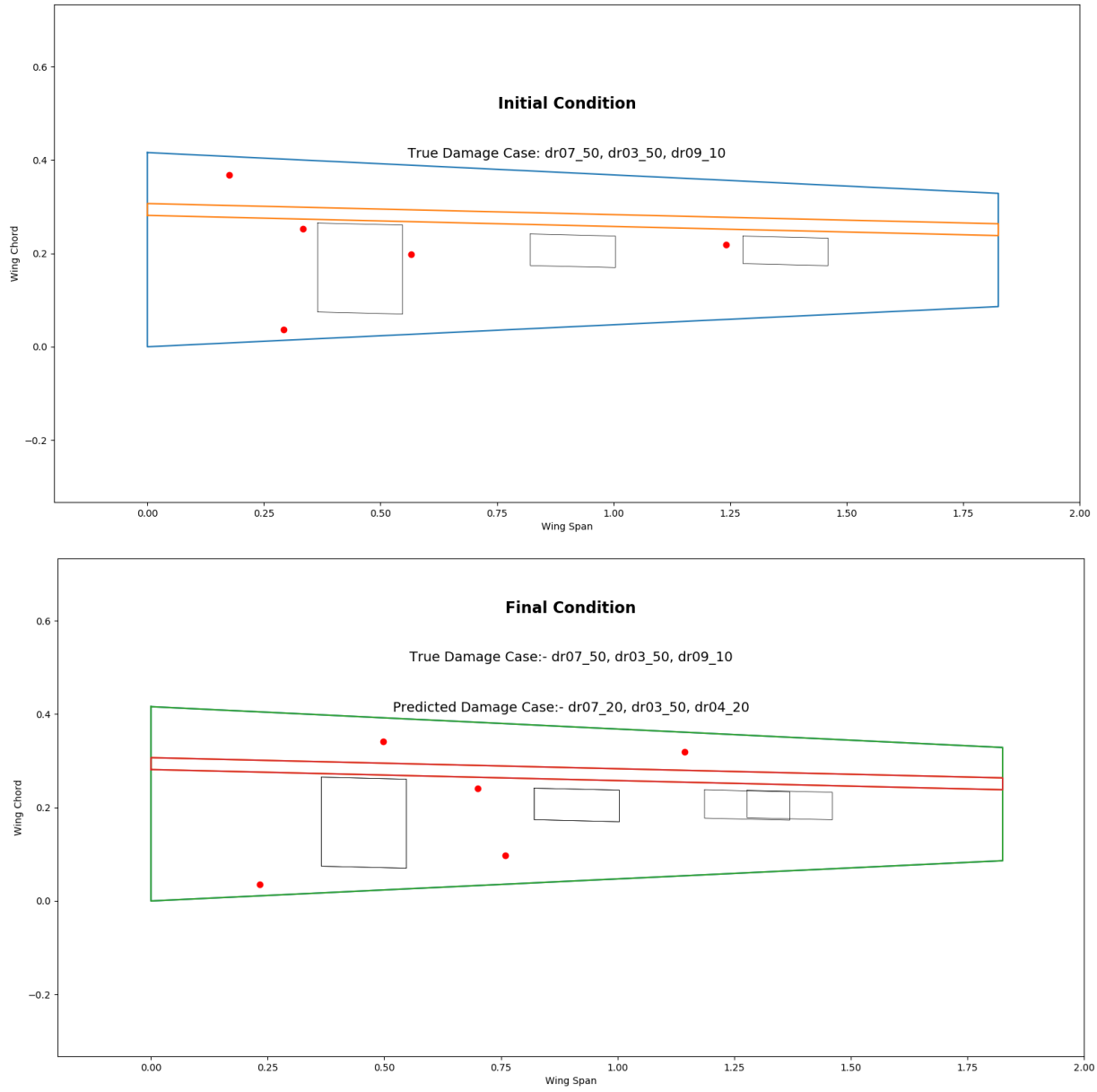


Figure 4.17: Figure represents the initial and final placement of sensor using dynamic sampling of information method where the algorithm is not able to identify the true damage case and the error between true capability and predicted capability is high

Thus we have demonstrated that our model in general performs better than the standard placement of sensors in predicting the capability of a self - aware UAV.

5. SUMMARY AND CONCLUSIONS

The Model developed shows great promise and since it uses a combination of physics based models, learning techniques and optimization algorithm, there is room for exploration in terms of how other algorithms will affect the performance, but the results observed with the current framework are very encouraging. This technique of capability estimation and optimum sensor placement is highly relevant to the industries that use strain monitoring to gather information about the health of their engineering system and in particular to industries that use self – aware unmanned aerial vehicles.

5.1 Contribution

We have successfully created a framework through combination of different models to predict the optimum placement of sensors for a library of damage cases defined in Akselos for a specific kinematic state. We have demonstrated through comparison how the optimum placement is superior in predicting the capability of the UAV as compared to the standard placement of sensors. We have shown why the framework needs to be used carefully and with a large offline library comprising of high number of damage cases in order to avoid a high prediction error for unexpected damage cases. We have established evidence of proof of concept of how dynamic sampling of information from optimum locations can lead to better predictions about the damage state and the capability of the self – aware aerospace vehicles.

5.2 Conclusions and Future work

As mentioned earlier the results are encouraging and the model is able to predict the capability of the self-aware UAV even with higher noise in the dataset and hence to solidify the results we will expand the offline library to more maneuvers and damage states, and perform greater number of iterations to establish the optimum location with the least mean squared error between the true and predicted capability value. As of now we have considered the number of sensors as a parameter, in future we intend to find an optimum value or a threshold value after which the improvement in per-

formance in terms of prediction accuracy is low. It will also be beneficial to include other metrics of capability other than turn radius to further improve the offline library. To improve the overall performance of the capability estimation process of our study we also to plan to explore other sophisticated learning techniques and noise reduction techniques such as James Stein estimator for better health prediction of self aware aerospace vehicles.

REFERENCES

[1] R. S. Wieslaw Ostachowicz and P. Malinowski, "Optimization of sensor placement for structural health monitoring: a review," 2019.

[2] J. H. C. Nam Ho Kim, Dawn An, *Prognostics and Health management of engineering systems*. 2016.

[3] T. M. G. W. Yeager, M, "Assessment of embedded fiber bragg gratings for structural health monitoring of composites," *SAGE*, vol. 16, no. 3, pp. 262–275, 2017.

[4] M. P. M. K. Soman, R, "Kalman filter based neutral axis tracking in composites under varying temperature conditions," vol. 110, pp. 485 – 498, 2018.

[5] M. D. Eric B.Flynn, "A bayesian approach to optimal sensor placement for structural health monitoring with application to active sensing," *ScienceDirect*, vol. 24, pp. 891–903, 2010.

[6] J. C. Md Zakirul Alam Bhuiyan, Guojun Wang and J. Wu, "Sensor placement with multiple objectives for structural health monitoring," vol. 68, no. 45, 2014.

[7] C. G. Vahab Akbarzadeh, Julien-Charles Lévesque and M. Parizeau, "Efficient sensor placement optimization using gradient descent and probabilistic coverage," *MPDI*, vol. 8, 2014.

[8] L. L. Z. H Y Guo, L Zhang and J. X. Zhou, "Optimal placement of sensors for structural health monitoring using improved genetic algorithms," vol. 13, 2004.

[9] H. Sun and C. M. . U. Oral Büyüköztürk Department of Civil Environmental Engineering, MIT, "Optimal sensor placement in structural health monitoring using discrete optimization," *IOP Science*, vol. 24, p. 16, 2015.

[10] C. P. Robert Guratzsch, Sankaran Mahadevan and M. Derriso, "Sensor placement optimization for shm systems under uncertainty," *Aerspace research central*, 2012.

[11] F. Darema, "Dynamic data driven applications systems (dddas) – a transformative paradigm," *Springer Link*, 2008.

[12] D. A. Brian J. Burrows, Benson Isaac, "A dynamic data-driven approach to multiple task capability estimation for self-aware aerospace vehicles," *AIAA*, 2016.

[13] M. R. Srivastava, A. and M. Claudia, "Integrated vehicle health management," 2009.

[14] G. R. T. E. Wieslaw Staszewski (Editor), C. Boller (Editor), *Health Monitoring of Aerospace Structures: Smart Sensor Technologies and Signal Processing*. 2004.

[15] A. D. Lecerf, M. and K. Willcox, "Methodology for dynamic data-driven online flight capability estimation," *AIAA*, vol. 53, no. 10, pp. 3073 – 3087, 2015.

[16] M. A. Lecerf, "A data-driven approach to online flight capability estimation," 2014.

[17] J. T. C. F. David Mascareñas, Alessandro Cattaneo, "Compressed sensing techniques for detecting damage in structures," *SAGE Journals*, vol. 12, pp. 325–338, 2013.

[18] H. D. K. D. R. E. Eftang, J. and A. Patera, "Adaptive port reduction in static condensation," *IFAC*, vol. 45, pp. 695 – 699, 2012.

[19] D. J. P. A. T. Phuong Huynh, Dinh Bao; Knezevic, "A static condensation reduced basis element method : approximation and a posteriori error estimation," *ESAIM*, vol. 47, pp. 213 – 251, 2013.

[20] K. E. W. Michael G. Kapteyn, David J. Knezevic, "Toward predictive digital twins via component-based reduced-order models and interpretable machine learning," *AIAA*, 2020.

[21] M. Drela, "Technical description — steady formulation," p. 57, 2015.

[22] H. L. Yuequan Bao, James L Beck, "Compressive sampling for accelerometer signals in structural health monitoring," *SAGE Journals*, vol. 10, pp. 235–246, 2010.

[23] E. Alpaydin, *Introduction to Machine Learning*. 2020.

[24] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer, 2000.

[25] T. Hinton, Geoffrey; Sejnowski, *Unsupervised Learning: Foundations of Neural Computation*. 1999.

[26] I. M. V. V. N. P. o. t. f. a. w. o. C. l. t. Boser, Bernhard E.; Guyon, *A training algorithm for optimal margin classifiers*. 1992.

[27] V. N. Cortes, Corinna; Vapnik, "Support-vector networks," *Springer Link*, 1995.

[28] T. Trafalis, "Primal-dual optimization methods in neural networks and support vector machines training," *ACAI*, vol. 5, pp. 459 – 478, 1999.

[29] S. Zhang, "Nearest neighbor selection for iteratively knn imputation," vol. 85, pp. 2541–2552, 2012.

[30] L. Mainini and K. Willcox, "Surrogate modeling approach to support real-time structural assessment and decision making," *ARC*, 2015.

[31] V. P. Group, "Strain gage selection: Criteria, procedures, and recommendations," 2014.