University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Offshore Technology/ Industrial Asset Management	Spring semester, 2015 Open / ~~Restricted access~~
Writer: Tawan Poungkrajorn	…… (Writer's signature)
Faculty supervisor: Srividya Ajit External supervisor(s): Helle A. Botnen	
Thesis title: Assuring asset integrity through improving the accuracy of leakage source identification of a permanently installed subsea leak detection system using artificial neural networks	
Credits (ECTS): 30	
Key words: Inverse problem Subsea leak detection system Source identification Artificial neural networks (ANNs) Computational fluid dynamics (CFD)	Pages: 96 + enclosure: 46 + CD Stavanger, 12 June 2015 Date/year

ABSTRACT

Environmental concerns and regulatory controls for oil and gas exploration and production activities have been increasing with the prospecting of deep-water fields and sensitive areas, such as the artic seas. To stop any incidents developing into critical events, subsea leak detection systems are required for a fast, cost-effective, and reasonable accurate method to not only detect the leakage substance (in this case methane), but also to identify its source and location. This thesis evaluates approaches to extend the capabilities of such systems deploying methane sniffers (pinpoint sensors) in locating the leakage sources by combining their sensory information with advanced data analytics. It will assess the potential role of artificial neural networks (ANNs) in improving the accuracy of leak source identification of permanently installed subsea leak detection systems.

The study reviews the advantages and disadvantages of four different modeling techniques that have been chosen to support this task: the analytical approach, the optimization approach, the probabilistic approach and the direct inverse approach. In the evaluation phase, the optimization approach, which underlies the working principles of artificial neural networks, was identified as the approach providing the highest accuracy, shortest time to run and with the lightest demands on resources in order to identify the location of methane leakage source in subsea condition. In addition, a computational fluid dynamics (CFD) module is also introduced to generate the data that is essential for the ANN training and testing process.

This thesis contains five main experiments. The first experiment provides the use of CFD to simulate different methane leakage source locations and its area of dispersion in steady state. The next two experiments are creating and training the artificial neural network architecture in order to maximize its performance. The last two experiments demonstrate the performance of ANNs using unseen data in the presence of noise-free and noisy data sets.

The overall results lead to the conclusion that the combined approach (CFD and ANN) is a promising tool for supporting pinpoint sensors used in subsea leak detection systems to increase the efficiency of identifying leakages in calm condition. Moreover this combined approach can also tolerate contaminated data up to approximately 4% of noise.

ACKNOWLEDGEMENTS

The subject of this thesis was proposed by Stinger Technology AS. I would like to to say thank you to Bjarte Langeland for giving me a great opportunity to tackle this incredible challenging and exciting task, as well as allowing me to freely select any methodologies used in this thesis. In addition, I would like also to thank Helle A Botnen who became my mentor at the company, for the time she spent on me providing stimulating discussions, consistently useful suggestion, and technical information.

For UiS, I am grateful to Bjørn Helge Hjertager for providing such a wonderful CFD course. His teachings and insights are very extremely helpful. It enabled me to go from having no previous CFD experience to a solid understanding of the subject matter in a short period of time. Thank you also to my university supervisor Srividya Ajit for the feedback and guidance.

Moreover, many thanks to my supervisor and professor in my bachelor degree, Yuttana Kitjaidure who taught and inspired me to get engaged with artificial intelligence and understand some of its potential. My family that always supported me over the two years of studies in Norway, as well as Peer Hackmann for the positive attitude and encouragement to take me out of my comfort zone and boosting my confident to write this thesis.

Stavanger, June 2015.

Tawan Poungkrajorn

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ANN	Artificial neural network
BFG	BFGS quasi-Newton
BFGS	Broyden–Fletcher–Goldfarb–Shanno
BMC	Bayesian Monte Carlo
BMCMC	Bayesian Markov Chain Monte Carlo
BP	Backpropagation
CDR	Convection-diffusion reaction
CFD	Computational fluid dynamics
CGB	Powell-Beale conjugate gradient
CGF	Fletcher-Powell conjugate gradient
CGP	Polak-Ribiere conjugate gradient
CTMs	Chemical Transport Models
FNN	Feed forward Neural Network
FOAM	Field, Operation And Manipulations
GA	Generic Algorithm
GD	Gradient descendent
GDA	Gradient descendent with adaptive linear rate
GDM	Gradient descendent with momentum
GDX	Gradient descendent with momentum and adaptive linear
GUI	Graphical User Interface
IS	Important Sampling
LES	Large Eddy Simulations
LM	Levenberg-Marquard
MC	Monte Carlo
MLP	Multilayer Perceptron
MSE	Mean Square Errors
NCS	Norwegian Continental Shelf
NDIR	Non-Dispersive Infrared Spectrometry
NE	Normalized Error
NN	Neural Network
OIVS	Optimal initial value setting
OSS	one-step secant
PDE	Partial Differential Equation
pdf	probability density function
PID	Proportional-Integral-Derivative
PISO	Pressure Implicit with Splitting of Operator
RAND	Random
RANS	Reynolds Average Navier–Stokes
RNN	Recurrent Neural Network
ROV	Remotely Operated Vehicle
RP	Resilient backpropagation

SA	Simulated Annealing
SCAWI	Statically controlled activation weight initialization
SCG	Scale conjugate gradient
SOM	Self-Organizing Map

CHAPTER 1 INTRODUCTION

1. Introduction

1.1. Background and Purpose

Subsea oil and gas leakages and spills are not only of concern to the oil & gas industry, but pose a substantial risk to societies as a whole. With the exploration of increasingly environmentally sensitive areas such as the arctic seas, environmental concerns are on the rise. This in turn has created new demands from regulators and authorities around the world, especially on the Norwegian Continental Shelf, to install leak detection systems for new field developments (as well as existing ones) in order to avoid the most severe consequences from malfunctions of offshore oil and gas activities.

For this reason, DNV GL; the organization that establishes and maintains technical standards for the construction and operation in maritime oil and gas, introduced the Joint Industry Project (JIP) on Offshore Leak Detection, with more than twenty participants ranging from regulators, operators, integrators, and subsea supplier who are working together to integrate different technologies into a leak detection system usable in real-world applications. In addition, JIP has also established the relevant functional requirements and general specification for subsea leak detection systems as well as *the recommended practice DNV-RP-F302*[1]*: selection and use of subsea leak detection systems*. This document is a guide for companies and organizations to provide reliable monitoring systems that will minimize the impact to human life, property and the environment from major accidents in oil and gas activities both subsea and on the surface (Decomworld, 2014).

It has been a few years now since leak detection systems became first available on the market. Manufacturers of such systems have been continuously releasing new products and methodologies to fulfil the safety requirements from new regulations. However, current solutions often address very specific parts of an overall incident scenario, such as high sensitivity sensors, but are not applicable to others, such as wide-area monitoring, for example (DNV GL, n.d.). The overarching aim of this master thesis is to integrate any technologies or methodologies into subsea leak detection systems, specifically in methane sniffers, that have the potential to enhance their performance when it comes to identifying leakage sources.

[1] The recommended practice DNV-RP-F302 is not the final version, as the JIP is upgrading this document at the moment.

1.2. Motivation

Stinger Technology AS, one of the JIP's participant members, offered me the opportunity to write this master thesis on improving the accuracy of leakage source identification of a permanently installed subsea leak detection system. As an innovation solution provider focusing on subsea systems, Stinger Technology has been developing permanently installed subsea leak detection systems since 2013 that have been deployed in the Norwegian Continental Shelf (NCS) to provide early warning of conditions which may develop into critical events.

Stinger's subsea leak detection system is designed to combine two different type of sensor technology based on JIP's guidelines. These are Sonar and Methane Detectors (Methane sniffers). Each type of sensor can compensate for the weaknesses of the other and in combination result in a vastly improved system performance compared to single sensor type deployment. For example, the Methane sniffers are extremely sensitive to low concentration of gases, but are limited to the small local area where the sensors are mounted; for wider area coverage and longer distances to the leak source Sonar can compensate for this shortcoming (Coley, 2013). In combination this supports Stinger's Subsea Leak Detection capabilities to detect concentrations of dissolved gas from very small leaks to plumes of bubbles within wider coverage areas.

However, the most challenging part is to provide an effective way to identify the exact source of a leak once the system issues an alert. Even though Remotely Operated Vehicles (ROV) can be used to pinpoint the exact location of the leakage subsea, this is a complex, time-consuming and costly process with its success impacted by the location, environment and experience of ROV pilots, as well as the sea depth, currents, and so on. Some areas may not be accessible by ROV at all. These issues suggest that there is a significant opportunity for a quicker, cheaper and yet still useably accurate method and system to identify the source of leaks in subsea conditions to strengthen the defence against major incidents.

Leakage source identification is not a new feature and sonars have been used in the past to pinpoint leakage locations. In order to improve the performance of the leakage detection system Stinger Technology wants to enable the use of point sensors (methane sniffers) to identify the leakage source, since they are more sensitive than the sonars; performing well even with low concentrations of methane dissolved in the water and able to detect even very small leakages and provide early warning to operators.

Unfortunately, the measurement of methane sniffers is limited to the very localized area in which they are mounted. Hence, this study will focus on concepts that aim to extend the ability of the methane sniffers (for leakage source identifications) in terms

coverage in particular and source location accuracy, as well as improve factors such as time-to –action, speed of detection and overall resource requirements.

1.3. Scope of work

The scope of work for this master thesis contains two main parts. These are:

1.3.1. Gathering the related information and techniques

Generally, leakage source identification can be characterized as a reconstruction problem: identifying which input (unknown causes) has led to what output (known consequences) based on given system parameters (Bady, 2013). This means that observed measurements (concentration of dissolved methane in the sea water) must be interpreted to investigate the cause of this occurrence (methane leakage location). In modeling processes this is called inverse modeling.

Therefore, the first part of this thesis collects any related information and case studies from a variety of sources relating to the use of inverse modeling techniques in source identification. Bady (2013) divides inverse modelling techniques into four main approaches. Zhang 's (2007) and Zheng and Chen 's (2010) research provides a definition of each approach summarised below:

- *The Analytical Approach* – This uses an analytical solution of the distributions of known outputs to inversely solve the casual characteristics. Because this approach is only effective for very simple problems, its applications are very limited.
- *The Optimization Approach* – For this approach, source determination would be treated as an optimization problem. It tries to identify which optimization methods, such as pattern search methods or genetic algorithms, can be used to find out the optimal solution. In addition, this approach uses forward modelling to create the effectual data based on all possible causal characteristics.
- *The Probabilistic Approach* – This approach is almost the same as the optimization approach, but it uses probability concepts such as Bayesian inference or stochastic Monte Carlo to express possible causal characteristics instead.
- *The Direct Inverse Approach* – Generally for this approach, the governing equations that describe cause-effect relations would be reversed to solve the reversed governing equation.

Each approach and its advantages and disadvantages will be further described in the literature review sections, using case studies from a variety of industries,. In addition, only a methodology that has a reasonable possibility to be useful and operationally deployable with point sensors for identifying the leakage locations in a wider coverage area will be selected for further investigation.

1.3.2. Examine the possibility of the use of the selected methodology to improve the performance of methane sniffers

In case of a serious incident, real-time prediction is one of the most important tools for successful crisis management. The optimization approach has been chosen for this master thesis as it provides fast calculation times that matches the requirements and characteristics encountered during real-world incidents. However, this approach involves a large amount of forward modelling to build up an extensive database, which requires various combinations of parameters to cover as many real incidents as possible. In addition, to keep resource and time requirements at manageable and cost-effective levels, artificial neural networks (one of several methods in the optimization approach) may offer an alternative approach to reduce the size of the database and still be able to provide relevant results. This will make leak source detection easier to deploy, faster to calculate and more accurate.

Therefore, the second part of this study is about setting up the experiments to examine the ability of artificial neural networks (ANNs) and computational fluid dynamics (CFD) to support the use of point sensors to pinpoint the leakage source in subsea environments. As artificial neural networks require the data for the database to generate outputs, the simulation part in this thesis consists of two major steps: creating the database and setting up the artificial neural network to solve the task at hand.

For the step of creating the database, I describe the data requirements and the importance of data for ANNs, as well as demonstrating how to acquire these data sets. CFD is introduced to simulate the release of methane from different known source locations (representing real leakage points), in order to identify the dissolved methane concentration levels at six locations (representing the reading value from six methane sniffers). CFD has been used as the collection of data in the real-world environment is very costly and time consuming. In addition, CFD offers an alternative to generate as many data sets for this study as the artificial neural network requires.

The architecture, selected from a variety of options, underpinning the artificial neural network is a backpropagation neural network. The backpropagation neural network will be examined and I will describe how to set up such a network from scratch and get it ready for use. This also includes the optimisation of each related parameter in order to enable the network to generate the best outcomes. Lastly, this optimised

network will be trained to try to validate my hypothesis that artificial neural networks have potential to improve the performance of methane sniffers and can be used for subsea leak source identification. As the use of both CFD and artificial neural networks in combination with point sensors is relatively new in subsea conditions in the oil and gas industry, the results of this experiment should provide interesting topics for discussion and may provide the focus for future, more detailed studies into this subject.

1.4. Outline of the thesis

The readers of this thesis will have a range of subject knowledge, motivations and interests so I deemed it useful to provide an outline of the thesis because not all readers require the same level of detail. The structure of the thesis consists of five main chapters:

Chapter 1: Introduction

This is intended to provide the context in which both oil and gas companies and society as a whole are increasingly concerned about the severe potential (and actual) consequences from offshore oil and gas activities, especially with regards to environmental impacts. This has led to calls (and in some cases implementation) of much stricter rules by regulators and authorities. As part of a wider range of measures, this thesis concerns itself specifically with the potential performance improvement of current subsea leak detection systems.

Chapter 2: Literature review

The literature review will provide readers with background information, based on summaries of all useful and related research and analysis, which support this work from a variety of different viewpoints and application areas. The background information and analysis can be classified as:

1.) Leak detection technology – reviews the advantages and disadvantages of current technologies. The reader will gain a better understanding of areas that need improving.
2.) Inverse modelling method – deals with four modelling approaches to overcome the limitations of pinpoint sensors in a variety of industries and applications. The modelling approach with the highest potential will be selected for further investigation.
3.) Testing tools – evaluation of a variety of tools that can support the modelling approach for subsea applications.

Chapter 3: Experimental design

I have set up five different experiments to support my research. The experiments range from how to set up the network and collect the data from scratch to getting the network ready for use. This will be of specific interest to readers who want to obtain knowledge on modelling with neural networks. The experiments are applicable to other areas and also several useful techniques are provided in each experiment.

Within the variation of difficulties and methodology of all experiments, I have applied the same structure to all experiments to eliminate complexity and potential confusion. Experiments will start with stating my assumptions and a basic introduction, followed by the objectives and will conclude with the methodology used for conducting the experiment.

Chapter 4: Experimental results and analysis

All experimental results are shown in this chapter. This chapter contains five results from five experiments and one summary and discussion part. For all five parts I present the result using graphics such as a table or graph, along with the results of the analysis.

At the end of the chapter, a summary and discussion are provided. The statement here delivers a summary of the useful information from all five experiments in brief: why we needed to conduct this experiment, what are the final outcomes, and how each experiment connects to each other. Moreover, I also indicate the potential for improvements as well as possible alternative solutions to improve the performance of the selected model, so it can be used in real world environments.

Chapter 5: Conclusion and suggestion.

In the concluding part, all research work and experiments conducted for this master thesis are summarised and I finalise my conviction case to answer the key question: do the use of ANNs and CFD have potential to improve the performance of methane sniffers in sub-sea environments. In addition, I also assess if it is reasonable to compare results from other industries and application areas within the same category in order to demonstrate supporting evidence from related research; this is to help strengthen the case of this study.

As this master thesis is only an initial feasibility study, the recommendation and future scope of study are also provided. This will help to define what application users (businesses, organisations, institutions, etc.) actually need in terms of skills or knowledge in order to enable this model to perform in a useful capacity in real world scenarios.

CHAPTER 2 LITERATURE STUDY

2. Literature study

The literature review is an important part of any mater thesis, as it provides useful and relevant information to the reader. This applies to and is in support of all parts of the thesis: defining the problem, establishing a hypothesis or thesis's question, designing an experiment, analysing the results, making a conclusion, and providing suggestions for improvement. Therefore, all the information contained here has been collected from respectable sources recommended by the university, such as Springer, ScienceDirect, IEEE, and Wiley. This approach aims to strengthen the credibility of the research and conclusions of this thesis.

As this part of the chapter contains material from several areas of expertise, I only outline the basic theory of each concept and its application based on case studies. The main reason for this approach is to enable the reader to fully understand each concept as well as to connect each concept with subsea leak detection systems. Moreover, I have tried to avoid using technical terms and phrases as much as possible in order to eliminate complexity. This should make this thesis more readable and its findings easier to digest. Hopefully both experienced and non-experienced readers will find some of the concepts stimulating and thought-provoking.

The relevant literature has been separated into three different groups based on three different topics:

1.) Leak detection technology – As this thesis focuses on methane sniffer solutions for leak detection, I will review current technology concepts, properties, advantages, limitations and area coverage, as well as briefly discussing what the industry is excepting from leak detection systems in the near future.

2.) Inverse modelling method – This is the algorithm, which can be used to solve an inverse problem. There are a number of different methodologies used with inverse modelling. In this study four main approaches, which are widely used for a variety of applications and in a number of industries, will be reviewed. The principles of reviewing are based on both technical and business criteria. The advantages and disadvantages of each of the approaches for both implementations in leak detection systems, and more specifically, leaks source identification, will be summarised. Finally, only the approach with the highest potential for these applications will be used to run a feasibility study for this specific purpose.

3.) ANNs and CFD concepts and tools – This reviews the underlying concept and functionality of ANNs and CFD and tries to establish its relationships for the questions under review in this thesis. Metaphorically speaking, this constitutes the "brain" of this exercise. In contrast the OpenFOAM and Artificial Neural Network toolbox from Matlab are the tool-sets, or the "hands", of the exercise. All of them will be briefly described in terms of characteristics and functionality, as well as relevance for the core topic.

2.1. Leak detection technology

Environmental concerns and regulatory pressures are the main factors in making the oil and gas industry focus more on effective and reliable monitoring systems to prevent any leakage of oil and gas and its resulting contamination. This applies especially to subsea production systems located on the seabed in depth of 100m to 1000m; these pose a significant challenge for operators. Apart from human safety and the avoidance of environmental impacts, high installation and maintenance costs are also a concern. Therefore, many methods and principles are currently introduced to provide high reliability and efficient subsea leak monitoring systems in order to achieve early warning and allow for immediate corrective action.

As mentioned, earlier, the focus is on the methane sniffer method as one of the leading detection technologies in the field. It is one of the technologies that has already been deployed as part of subsea leakage detection systems. The aim of this thesis is to identify possible solutions to enhance the detection system performance by using the data collected from methane sniffer to quickly identify the leakage location.

2.1.1. Methane sniffer method

Sniffers are sensors that detect and measure even small concentration of carbon molecules in water. Generally two principle measuring technologies are deployed with these sniffers. The first principle is a semi-conductor system, which is based on the conductivity of the component (coated with tin-dioxide layer) inside the sensor chamber. Theoretically, the hydrocarbon molecules react with the oxygen on the surface of the component. This releases free electrons in the layer and increase the conductivity. The changing of conductivity is converted to a voltage signal which is then digitized in order to provide readable data to users (Neptune Oceanographics, n.d.). The second principle is associated with the optical non-dispersive infrared spectrometry (NDIR) method; it basically uses a degree of absorption of infrared light to determine the concentration of methane (Cole, 2013). In addition, regardless of the measurement technology deployed, both are based on the diffusion of hydrocarbons from the water across the specific membrane into the sensor chamber behind. The

water stays outside. Generally, sniffer methods can be applied to detect the higher order chains of hydrocarbons (hydrocarbon sniffer) or just methane (CH_4); which is the substance of interest (methane sniffer). A picture of a methane sniffer and its dimensions is shown in Figure 1 and more specific details are shown in Appendix A.

Methane sniffers provide a number of advantages for subsea leak detection system. These are (Neptune Oceanographics, n.d.),(Neptune Oceanographics, n.d.),(Esser, 2008):

- Being able to detect almost any hydrocarbon leakage sources, as CH_4 is the smallest molecule of almost all hydrocarbons including oil and natural gas. Any other gaseous substances in the environment such as H_2S cannot be detected, and are of no consideration for our purposes.
- Very high sensitivity (about 50nM – 10uM for standard setting) meaning that methane sniffer still performs well even with low concentrations of methane dissolved in the water; able to detect even very small leakages.
- Response time within a few seconds (immediate detection).
- Eliminates the use of dyes (fluorescent marker) in the field makes the system more environmentally friendly.
- Low failure rate and power consumption due to no internal moving parts or pumps.

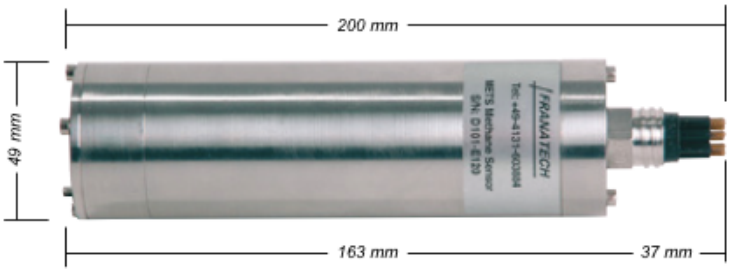

Figure 1 The methane sniffer

On the other hand, the methane sniffer also has some limitations because they are point sensors by design. The main disadvantage is that the measurement is limited to a very localized area from which the gas is diffused into a chamber for analysis. In addition, the environmental conditions might also interfere with the measurement performance; for example sensors would not be able to detect any methane dissolved in water if sea currents drive the leaking medium in the opposite direction away from where the sensor is located. This could make detection slow and unreliable. In order to overcome these challenges, leakage detection system tends to use an array of sensors that include a sufficient number of methane sniffers. That extends the detection area coverage as well as reduces traveling times of methane from leakage

source to the chamber of the sensors; allowing the system faster and near real-time detection (Esser, 2008).

In addition, there are other drawbacks of methane sniffers that still exist and cannot be eliminated easily. Those are the difficulties of quantification of leakage levels and the limitation of identifying a leak source location (Cole, 2013). In order to overcome both drawbacks the combination of several types of sensor can compensate in these circumstances, but would often prove difficult and costly.

Based on recommended practice DNV-RP-F302 on the challenge posed by the development of subsea leak detection systems, there are four capabilities the industry would like to see most in leakage detection system (DNV, 2010). Those are the technologies, which can provide quantification, identification, localization, and classification of a leakage. It would be of specific benefit to the industry if these four operations could be integrated into only one system or technology. This is, therefore, one of the main starting points that underlies the objective of this master thesis. Specifically, it is to identify any solutions or algorithms, which can provide extra functionality (identification) for the methane sniffers, in order to decrease the high complexity of combining several sensors as well as reduce related costs.

2.2. Inverse modeling method

Before going into detail about inverse modeling methods, we should start with a brief introduction of what inverse theory and an inverse problem is and what it does.

An inverse problem is a general framework to find unknown causes based on known consequences. The inverse problem methodology is principally used for two different types of problems: the reconstruction problem; model and output are used to identify which input has led to this output.

The other problem is the identification problem which, with given model parameters and observation data, identifies the model (system) of the relationship between inputs and outputs. While the forward theory is typically focused on cause-effect sequences such as the forward-time problem, which with given model parameters and model, finding out the output of the model (Bady, 2013). Figure 2 show the system of inverse problem and forward problem (Richardson and Zandt, 2009).

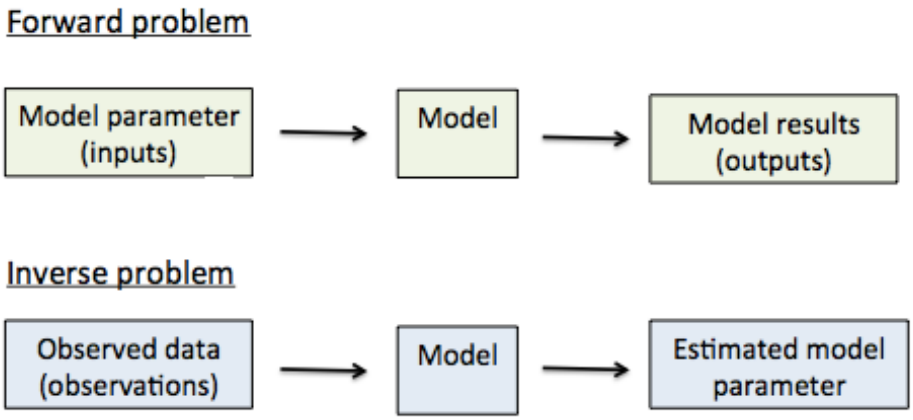

Figure 2 The system of inverse problem and forward problem

Where:

The model parameters: They are the parameters that characterize a model or define a particular system. It can be either the numerical quantitative values for the forward problem or unknown values that we want to estimate in case of the inverse problem.

Model: This describes the mathematical relationship between model parameters as well as other related information and the output data. This relationship may be a simple linear model or something inherently more difficult such as a non-linear model.

Model results: This is the data which is observed or measured in each circumstance and are inferred by the model parameters that characterize the system.

As the definition of the inverse problem is completely different from the forward problem, using the normal forward method in order to solve the problem might not be appropriate or can yield complex solutions. Therefore, the inverse modeling method is deemed to be more suitable, as it can estimate the model parameters based on observed data and physical understanding of the model characteristics.

In this study, the inverse modeling method is used to investigate the methane leakage source on the seabed around the oil platform, as an incident might develop into a severe oil or gas spill with potentially catastrophic consequences.

However, using inverse modeling methods in this specific application requires some base data inputs. This is provided by the methane concentration measured by methane sniffers at six different locations and the seawater current flow in the considered area. They are used as both the observed data and the model respectively. The estimated

numerical value derived from the model will represent the methane leakage locations (estimated model parameter).

In addition, inverse modeling is a discipline that applies any mathematical techniques to combine measurements and models to solve the inverse problem (ETH Zurich, 2008). Unfortunately, there exist no inverse modeling (mathematical) techniques that have been employed in subsea conditions as in the present study. Therefore, each mathematical technique reviewed in this chapter will be based on techniques used in applications with some proximity to this specific problem (subsea leakage source identification) such as the detection of pollutant sources in air or ground water.

As each approach (inverse modeling method) provides several types of concepts to solve the specific application, only one concept for source identification will be used for each approach demonstrated in the following sections. This allows the reader to see the big picture of each approach and total area of inverse modeling method in particular: how they work and also the differences between each approach regard to solving a problem process. In addition, the four approaches (mathematical techniques) will be discussed highlighting each advantages, and disadvantages, as well as possibility to implement it as a leakage source identification feature in subsea condition.

2.2.1. The analytical approach

This approach is a very simple method, which most people have probably already used, but do not necessarily know. For example y = 3 + x, where y = number of total seats for the dinner tonight, 3 is the number of hosts. If the total seats are 5, we know that we have to invite 2 more people to fill all seats (x). It seem that the results from the analytical approach is solved by fitting the model parameter (x) to the observed data (y).

$$x = G(y)$$

where, G(y) is a linear/non linear operator

However, the system is more complex in relation to this thesis; it consists of a large diversity of elements connected together and the operator can be much more complicated. Therefore the methods to solve the problem need to be more advanced.

In the case of air pollution in urban areas, Islam and Roy's (2002) and Islam's (1999) research demonstrated a solution to identify the emission source by using an analytical approach. The methodology of using this approach for inverse problems is shown in following steps.

Firstly, the analytical approach typically requires a starting equation, which has to be able to describe the distributions of airflow and pollutant concentration from source. Therefore, the Gaussian plume equation, one of the best known diffusion equations and the most commonly used in several research studies, was deployed in Islam, (1999) work. The Gaussian plume equation is (Turner, 1994)(L. C. Thomson et al., 2007) :

$$C(x,y,z,H) = \frac{Q}{2\pi u\sigma_y\sigma_z} \cdot \exp\left(-\frac{y^2}{2\sigma_y^2}\right) \cdot \left\{\exp\left[-\frac{(z-H)^2}{2\sigma_z^2}\right] + \exp\left[-\frac{(z+H)^2}{2\sigma_z^2}\right]\right\}$$

with

$$\sigma_y = ax^b$$
$$\sigma_Z = cx^d + f$$

where:

C = the concentration of the pollutant (kg/m^3)
Q = the source injection rate (g/s)
u = the average wind speed (m/s)
x = the distance downwind from the stack (m.)
y = the crosswind distance from plum centerline (m.)
z = vertical distance from ground level (m.)
H = the stack height; sum of stack height and plume rise (m.)
σ_y, σ_Z = the standard deviations of the concentration in y and z axis (m.)
a,c,d, and f = the indices of the downwind position x

Based on the equation earlier, it consists of a large diversity of elements connected together. As a result researchers have to apply more advanced mathematical techniques to solve this equation.

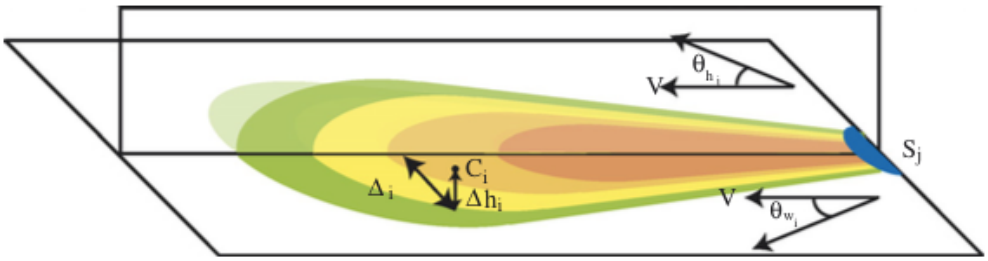

Figure 3 The Gaussian plume model (Thomson et al., 2007).

Secondly, as the origin of the emission source is unknown, most of the parameters in this equation still remain unknown such as coordinate x and y as well as the emission rate of the source (Q) that may vary with time. Therefore, apart from trying to directly solve these unknown parameters by the different types of equation, Islam, (1999) decided to use data from two different locations of sampling sites in order to create the ratio of the concentration. This will eliminate parameter Q. The ratio of the concentrations between two sampling site is:

$$\frac{C_1}{C_2} = \exp - \frac{(y_1^2 - y_2^2)}{2\sigma_y^2} \qquad \text{Equation 2-1}$$

that is equal to

$$y_1 = \frac{-2\sigma_y^2 \ln\left(\frac{C_1}{C_2}\right) + r_{12}^2}{2r_{12}} \qquad \text{Equation 2-2}$$

where, r_{12} is the distance between two sampling sites, and $\sigma_y = ax^b$

To solve the problem, one of the techniques which can be implemented is a graphical solution; defining the value of x then calculate y from the Equation 2-2, repeat it with several x values, and finally plot and draw a line into the coordinate x - y graph. The emission source location will lie on the curve somewhere. Fortunately, it is possible to be more specific about the location of the emission source by adding the new set of data from another pair of sampling sites and solve it employing the methods mentioned earlier. Consequently, the intersection of both curves denotes the location of the emission source as shown in Figure 4.

This is one example of several articles that have been published on this topic. It implies, in order to determine an emission source location, there are many different type of models (which are created to predict the concentration of gas in the atmosphere), and also many mathematical techniques to solve the unknown parameters of a model. Kathirgamanathan et al., (2012) research created the source term from an advection-diffusion equation and also implemented a non-linear least squares regression as a methodology for identifying the source location. In addition, even though there are several concepts that can be used in this analytical approach, they all have one thing in common no matter how complex the question are, is the requirement of a starting equation (for example the Gaussian plume equation).

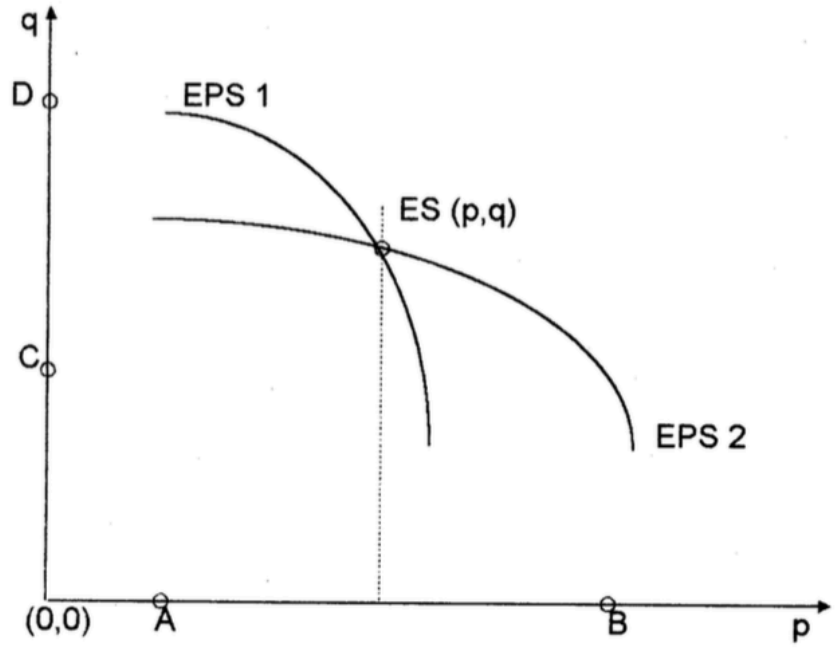

Figure 4 Equipotential pollution source curves (EPS1 and EPS2) from two pair of sampling sites (A,B) and (C,D). The emission source locates in location (p,q) (Islam, 1999).

Finding a real starting equation (model), which is able to completely describe the dispersion of pollutants can be very difficult. Even the Gaussian dispersion equation which has been widely used for many years, has many limitations such as the emission rate and horizontal meteorological conditions being constant, no wind shear, and the inability to take historical data into an account (Durrenberger, 2014). All of the major and minor constraints have significant impact on the prediction accuracy of the model (How well the model can represent the real environments). Low accuracy models can lead to differences between estimated and actual concentration that might directly affect the prediction process and the performance may deteriorate further.

This drawback applies especially to subsea conditions. Therefore, the disadvantage of using the analytical approach for subsea leakage source identification can be summarized as follows:

1.) At present there are no theoretical models to fully support the characteristic or behavior of methane dispersed in the seawater, meaning that it is more difficult to estimate the related parameter as well as the methane concentration at various locations.
2.) The dispersion of methane in subsea condition can be a very complex process. There are many related parameters, which could interfere with the behavior of methane dispersion such as temperature, location, timing, and the total volume

of release. Also, the water current path does not remain static; the water current velocity and direction is changing over time and specific events such as surge currents may occur. This constitutes a dynamic system. All this makes it difficult to build a model that can take account of all these variances.

3.) After the model has been built, it may consist of many related parameters, meaning that in practical terms the total system has to combine many types of sensor to measure and identify the value of all related parameters in order to solve the model. In addition, the mathematical concepts used to solve the equation have to be more advanced as the complexity of the model grows. This may result in increased calculation time, more resource to solve the equation and a final answer that is very much an approximation.

4.) As this approach is based on mathematical equations any errors, which are added during the measurement process, would introduce additional uncertainty into the prediction. The implication is that this approach is unlikely to tolerate any noise (such as the error from reading a sensor, malfunction of sensors, etc.) from even a small percentage of the data.

On the other hand, the analytical approach has been validated by Alifanov's (1983) research for specific applications. He focuses on solving inverse heat-conduction problems with the analytical approach. The results show that this approach can provide very high accuracy and efficient methodologies for this specific problem.

However, one dimensional heat conduction problems are simple problems compared to 2D or 3D methane dispersion in subsea condition. Therefore, using the analytical approach can provide a usable result only for a case of a simple problem, whereas for high complexity problems such as subsea leakage source identification, good results are unlikely to be produced due to the many limitations mentioned earlier.

2.2.2. The Probabilistic Approach

The underlying principle of the probabilistic approach is the use of probability theory to quantify variation and ambiguity of information used in the model. The implication is that in the modelling process the value of parameters are typically represented in distributed form, instead of the fixed value as with the other three approaches (Ghahramani, 2011). The benefit of distribution enable each source parameter to describe a range of possible values and also to show which value is most likely to occur. In addition, not only the source parameter can be shown as a distribution, but also the results of the model. Therefore, the probabilistic approach can express all aspects of uncertainty in the model; both input and output. That may lead to performance improvements of the model in terms of forecasting or decision making respectively, due to the full range of possible outcomes, which will be taken into account.

The probabilistic modelling methods are mainly based on the Bayesian inference, especially when the probabilistic modelling is used to solve inverse problems. Using the Bayesian framework in contamination source identification was originally developed in the late 90's by Neupauer and Wilson in the area of groundwater pollutant source identification (Zhai et al., 2011). This approach has been developed further to enhance performance by the introduction of the adjoint equation and dynamic inversion amongst others. This has increased the efficiency of the probability modelling method, as well as making it more adaptive respectively (Zheng and Chen, 2011). However, the most common use of this approach is to combine the Bayesian inference with stochastic Monte Carlo or Markov Chain Monte Carlo. Both are used as sampling tools to estimate the outcome.

Using the Bayesian approach in contamination source identification (in air conditions) can be divided into two main stages: the pre-event or simulation stage and the data interpretation stage (Sohn et al., 2002).

The pre-event or simulation stage

Sreedharan et al., (2006) briefly concludes that the major task of the first stage"… consists of developing a library of hypothetical contaminant transport simulations spanning the set of all plausible pollutant release and internal airflow conditions.". Typically a library is created by applying the fundamental principles of Bayesian inference or Bayes' rule.

Bayes' rule provides a statistical method to calculate the output (a posterior probability) based on unknown parameters including its probability distribution, and the likelihood of observation given unknown parameters. The Bayes' rule is shown as the following equation (Walpole et al., 2011).

$$p(Y|O) = \frac{p(Y)p(O|Y)}{p(O)} \propto p(Y)p(O|Y)$$

Where;

Y	= the unknown parameter
O	= the observations
$p(Y)$	= the prior distribution of the parameter
$p(Y\|O)$	= the posterior distribution of the parameter
$p(O\|Y)$	= the likelihood function
$p(O)$	= the marginal distribution of O

Basically, we can treat $p(O)$ as a constant. Then the posterior distribution can be expressed as

$$p(Y|O) \propto p(Y)p(O|Y)$$

where the symbol "∝" mean *is proportional to*

As we have seen from the earlier equation, the posterior is proportional to the product of the prior distribution and the likelihood function. Therefore, the first stage of the Bayesian modelling framework is to express all assumptions using the concept of probability, especially for both the prior distribution and the likelihood function. However, assignment of both the prior distribution and the likelihood is difficult due to the fact that it contains a high degree of unknown probability and also demands a huge amount of processing time.

It is a requirement of the Bayesian method that model creators define parameters prior to execute calculations. The prior probability can contain any information known about the unknown parameters before the event will occur. In addition, information is provided based on the fact that different configurations might lead to the same outcome and vice versa, some will be more probable than others (Keats et al., 2007).

In order to estimate uncertainty, the model should be based on the reliability of historic observation data in order to maximise its performance. For example, if historical data shows that an area with a high density of pipes is the main source of leakages, then the value of the prior probability will increase (it represent how likely the leakage will occur in this region). However, all possible parameter scenarios will have to be defined in order to cover all probabilistic ranges (location of source in x y z axis, strength of source, and source duration respectively). We can simply call this process the parameters uncertainty characterization.

The likelihood function describes the probability of the level of contamination, which is given by unknown parameters (i.e. location, strength, and duration). This probability of the likelihood function is basically obtained by solving the equation (Zeng et al., 2012). However, there are many other techniques and several types of equation to determine the probability, which can be used varying from application to application.

Once both the prior distribution and likelihood function are defined the model designer can generate a library by sampling the pool of the model parameters (unknown parameter) and predicting the output for each set of parameters. Each set of parameters and their outcomes represent one scenario that has a chance to occur in real environments. This sampling process is being repeated again and again with varying the parameters. Thus the resulting library of simulations may consist of more than a thousand different scenarios. In addition there are several types of sampling techniques, which can be applied here, such as Important Sampling (IS), Monte Carlo (MC), Bayesian Monte Carlo (BMC), and Bayesian Markov Chain Monte Carlo (BMCMC). The last two sampling techniques are performing well and are widely

used in many applications and in contamination source identification in particular. However, the benefits of selecting a well performing sampling method is not only to fully capture all ranges of possible condition, but also reducing the time used for the sampling process. Rajabalinejad, (2010) demonstrated the comparison between the results obtained by IS, MC, and BMC and showed that BMC can reduce the number of simulations by 3 and 30 times compared to MC and IS respectively. Also, BMCMC perform better in many cases because of its ability to improve on previous samplings to better approximate the next sample selection as it shown in Figure 5 (Zheng and Chen, 2011).

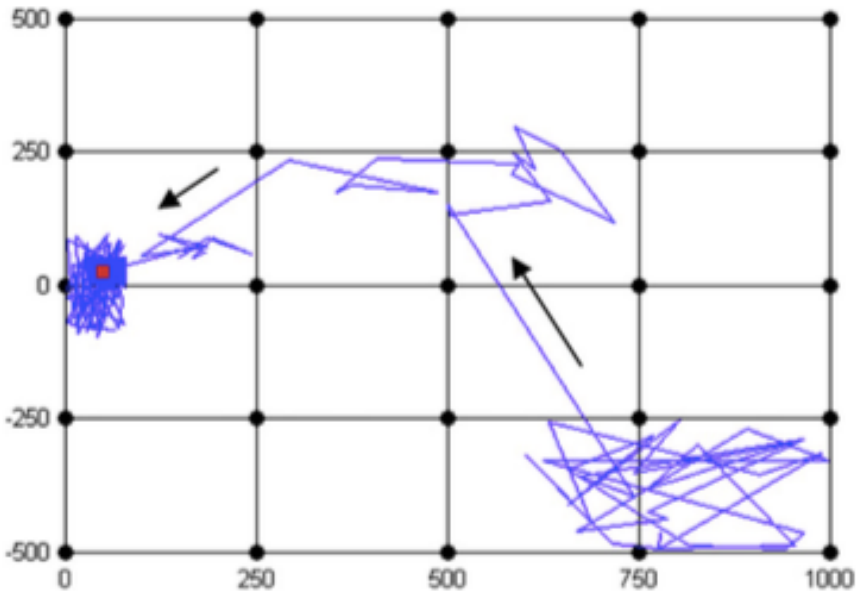

Figure 5 Sampling results of MCMC. Blue points are the sampling point and the arrows show the direction of the sampling process. The sampling point quickly arrived at the neighbourhood of the source, and gradually approximate to the source (Zheng and Chen, 2011).

The data interpretation stage

This stage always takes place during a release event. It is basically to gather the measurement value from several sensors installed in different locations within the considered area and send it to the monitoring-computer. All of the collected data will be interpreted by the algorithm, which will compare the sensoring data with the data in the library. The data with the best fit will become the output of the model. This stage is very simple and quickly executed due to no requirement to re-execute the time-consuming pre-event stage of the analysis (Sohn et al., 2002)

Based on the overview of the probabilistic approach, which was provided earlier, I have identified both the advantages and the disadvantages of this approach and have

applied it to subsea leakage source identification. This is summarized and shown below:

The advantage

1. The Bayesian modelling framework expresses all assumptions using the concept of probability. This means that not only one specific value will be assigned to a parameter, but also the value contains uncertainty of how likely this value can occur. This is called the probability density function (pdf). Therefore a posterior probability distribution of the parameters is obtained rather than a single solution. The example of the posterior probability densities of unknown parameter is shown in Figure 6 (Zhang et al., 2015). In practical terms, this form of output can add extra benefits to the user, such as the model forecasting the leakage source in x-axis at 3 meters from the reference point (based on posterior probability densities of x_s shown in Figure 6), which is not exactly the same location as the true leakage source. This means that once operators go to the site they will have a range of options and areas to investigate based on the ranking of high probability obtained by the posterior.

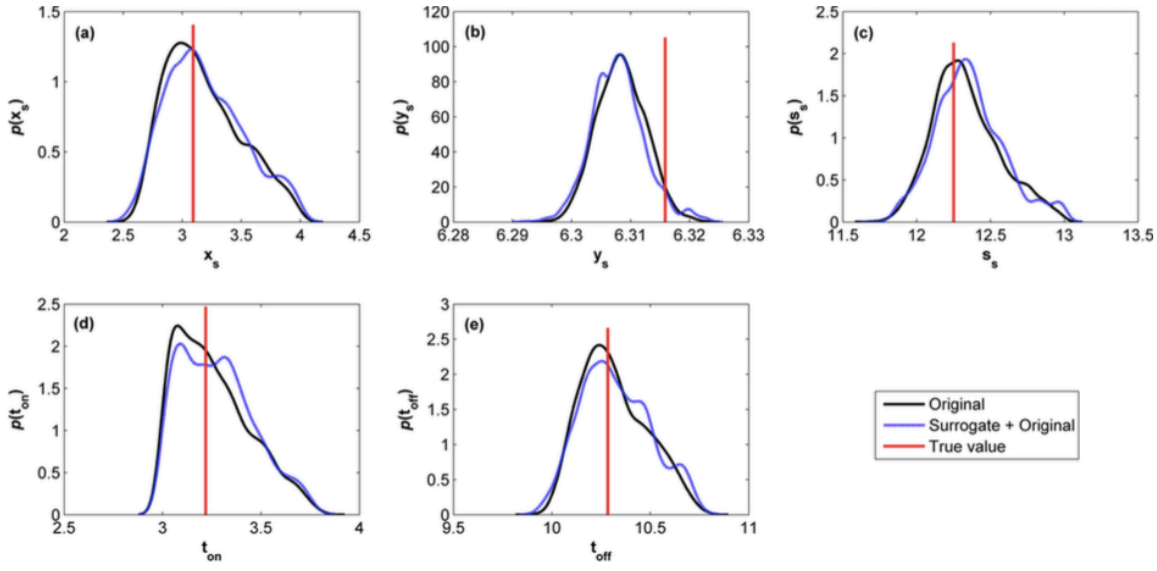

Figure 6 Comparison of the posterior probability densities of unknown parameter (a) x_s, (b)y_s, (c) S_s, (d) t_{on}, and (e) t_{off} between true value and prediction value from original BMCMC and improved BMCMC.

2. This approach allows the evaluation of data confidence by repeating the data interpretation stage again and again with the same value of all input parameters, 100 times for example. The model basically generated 100 slightly different posterior distribution of each unknown parameter. These can

be used to calculate a confident interval in order to ensure the accuracy of the outcome. The implication is that the uncertainty of the solution can be easily be evaluated.

3. Because of the libraries that were previously created, the probabilistic approach can perform very fast analysis of sensory data by cross referencing. Based on Sohn et al., (2002) research, the researched model could predict contaminant dispersion in buildings within under two minutes with high probability.

The disadvantage

1. One of the major limitations of the probability approaches, especially one with Bayesian inference, is to presuppose the probability density function of prior distribution of all model parameter (predict the distribution of the parameters) such as leakage source location, source strength, and source emission duration. This is a very difficult task and hard to achieve in a short period of time due to lack of the real operational information. However, uncertainty distribution that describes the probabilistic range of possible values can be assumed or estimated; wide distributions are generally used by the limited prior information. These cannot fully represent the characteristics of the considered parameters. This results in the difficulty to obtain an explicit scheme of the posterior probability distribution of the parameters. Moreover, even the high reliability of the observation data, which can be obtained in one specific site, cannot necessarily be fully applied to other locations.
2. Even though there are several techniques that can be applied for a sampling process to reduce the processing time, this process is still time consuming, especially when the many unknown parameters are taken into account. In addition, reducing the processing time may interfere with the sampling performance, thus, sampling technique have to be carefully selected.
3. Bayesian inversion is limited only to models with a few numbers of unknown parameters. However, whilst it is possible to use this approach for scenarios with a high number of unknown parameters by deploying the Wiener integration, this is difficult to be implemented as the level of equations used gets extremely complex (Snieder, 1998).

2.2.3. The optimisation approach

The process of the optimisation approach is typically based on trial and error of many possible values to optimise the objective functions or model parameters.

In case of subsea leakage identification for example, the optimisation approach focuses on interpreting the output or the consequence of the event (the level of methane dissolved in the water measured by sensors). It subsequently fits them to the

library (the database), which can contain all related information such as the water current flow model and the methane transport model. In addition, the 'fit', sometimes called 'match' or 'pair', is usually achieved by adjusting the model parameter until they identify the best or the most probable data (less errors). The final result will identify the possible source of contamination, specifically the leakage location.

Basically, each method of the optimisation approach can be separated into two main groups based on types of the final data which is generated and it characteristics. These are the indirect search method and the direct search method (Zheng and Chen, 2010). The basic concept and the limitation of both methods will be demonstrated based on case studies. The featured case studies mainly relate to the source identification problems.

The case studies of indirect search method are:

Based on Elbern et al., (2000) research, the four-dimensional variational assimilation (4D-var) method was deployed to identify the emission rates of a source (which it is an objective function of this case study) based on pollutant measurements such as SO_2 and NO in the environment. In Elbern's example the emission rates are subject to optimization. The 4D-var method will iteratively use any possible values of the emission rates to minimize the misfit between modeled concentration levels and measurements. Later the predicted emission rates of sources will be sent to an air pollution dispersion model, in order to inversely locate the pollutant source. Also Yumimoto and Uno's (2006) research demonstrated the upgraded 4D-var method which was applied to optimise CO_2 emissions based on the measured data from three main stations and then used CO_2 emission data to predict the concentration field based on chemical transport models (CTMs).

It seems that the output obtained by the indirect search method could not fully answer the optimal control problem, unless other post-processes were applied. Typically, the optimised values of objective functions are needed as input for other related models and equations, for generating the final results. However, the selected objective functions may create a major problem with this approach by introducing high levels of complexity. The more complex the objective functions are, the higher the calculating times will be and the greater the number of related equations that need to be taken into consideration will be. In addition, the indirect search is a method that requires partial derivatives of objective functions to search for an optimal point (also called descent method) unlike direct search, which does not require partial derivatives and can therefor be described as non-gradient or zero-order method (Kumar, n.d.).

On the other hand, the direct search method focuses on the overall system rather than each individual parameter. This group of algorithms are more suitable for pollutant contamination source location. Firstly, the implication is that a usable final outcome (also knows as optimal solution) will be generated when the given tolerance is

reached (such as errors less than 0.1). Secondly, usually the objective functions, which indirectly relate to the original optimal control problem can be ignored. The differences between both methods are shown in Figure 7, where the first optimises any parameters to produce a predicted leakage location and the second optimises the matching between input and output sources.

However, the same optimisation methods can be used for both direct and indirect methods.

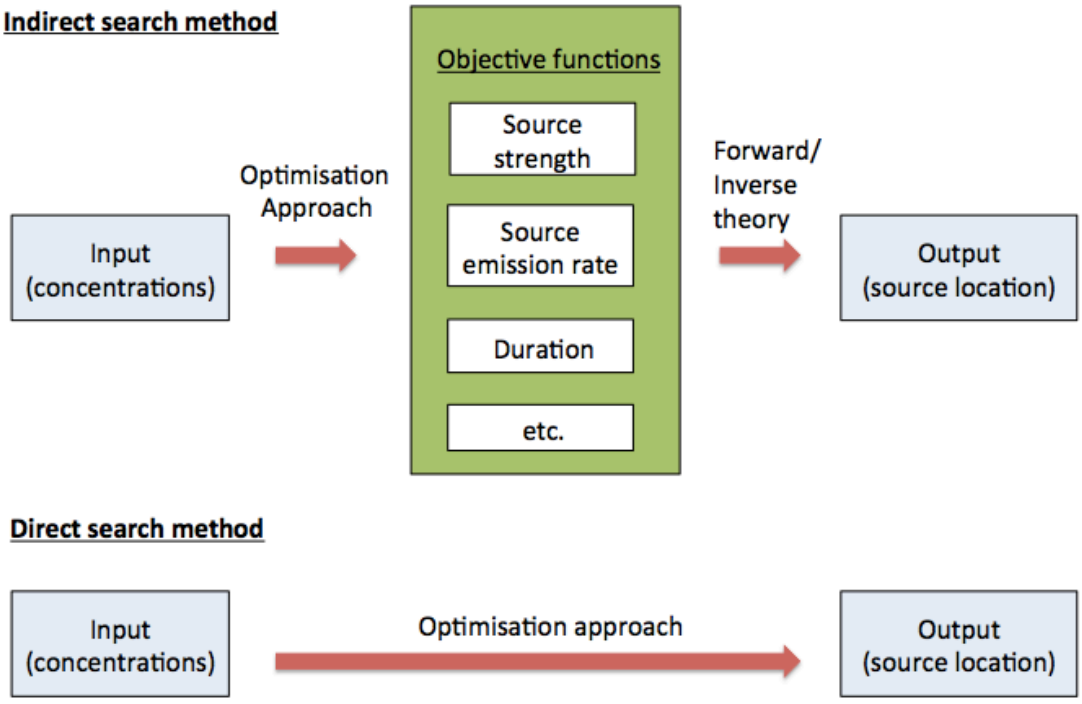

Figure 7 Flowcharts of indirect/direct search method

In the case study we will describe several concepts of each optimisation algorithms (using for direct search methods in source identification problem). The most widely used algorithms are:

2.2.3.1. The pattern search method

This algorithm is one of the basic optimisation methods. In terms of the searching process, the pattern search method consists of two basic steps. They are “the axis direction move” and “the pattern move”. Zheng and Chen's (2010) research has demonstrated the application of the pattern search method to identify the location and

strength of hazardous material releases. In this case, the location and also source strength were given as the unknown parameters. Also, the Gaussian plume model was used as the air dispersion model in order to calculate the concentration at each location.

The processes of the pattern search method can be summarised in the following steps. Step One defines the theoretical parameters (leakage location and source strength) and its initial values (the global or local optimum point depending on this initial value). In Step Two the model algorithm will vary each parameter by increasing or decreasing their values from current point applying a constant factor. It is called the axis direction move. Following on, we calculate the criteria value (the difference between calculated concentrations and measured concentration). If there are no such increase or decrease of the criteria value compared with the values of the previous points, the step size is halved (it is called the pattern move) and the process is repeated from the first step again and again until the termination criteria are reached (Davidon, 1991), (Zheng and Chen, 2010).

2.2.3.2. The simulated annealing

The method of Simulated Annealing (SA) was introduced by Kirkpatrick et al., (1983). It is based on an analogy with thermodynamics, specifically the process of heating and controlled cooling of a material in order to reduce any defects in the material. This process directly depends on thermodynamic energy (E).

Once applying this thermodynamic analogy to the optimization problem, the goal is to bring the system from initial state to a state in which the system uses minimum possible energy. The rule for accepting change in state is also based on the Boltzmann probability distribution as in the following equation (Laura C. Thomson et al., 2007).

$$R(0,1) < \exp\left(-\frac{E_n - E_{n-1}}{T_n}\right)$$

where:

R (0,1)	= A random number between zero to one
E_n	= The energy of the system at current point
E_{n-1}	= The energy of the system at last accepted point
T_n	= The temperature or cooling parameter.

Laura C. Thomson et al., (2007) research uses the simulated annealing to locate the gas source. The location was given as the unknown parameter and the Gaussian plume model was applied as the air dispersion model in order to calculate the concentration at each location. The processes of the simulated annealing method can be summarised as follows:

1.) Initialize control parameter T. The value of T is decreased after each iteration based on $T_n = T_{n-1}(1 - \varepsilon)$, where ε is the constant value (between 0 to 1) specifies the time used to reach the solution. The higher value of ε (close to 1), the smaller number of iterations using to find an output. However, this might results in missing an optimal solution.
2.) Randomise the initial value of gas source location.
3.) Model starts calculating the energy E_n based on the new value of unknown parameters. This E_n value represents the difference between calculated concentrations and measured concentrations.
4.) Comparison between E_n and E_{n-1}. The change is automatically accepted, if $E_n<E_{n-1}$. Whereas, in case of $E_n > E_{n-1}$, the change will only occur, if the results from the exponential was greater than the random number R.
5.) Adjust T
6.) Repeat the step 2, 3, 4, 5 and 6 until the value T becomes zero.

Because the new value of the unknown parameter is newly set at every iteration, this algorithm can search for the global optimum.

In addition, the simulated annealing was also deployed by Newman et al., (2005) to determine contaminant source zones in natural ground water, in which the flux plane conceptual model was treated as the groundwater flow model in order to calculate the concentration at each location.

2.2.3.3. The genetic algorithm

The genetic algorithm (GA) is classified as one of the artificial intelligent optimization methods. Similarly to most optimisation techniques, the genetic algorithm is based on iteration, but the most interesting and different part of GA is its operation to modify the parameter. This is inspired by the principle of biological genetics and natural selection. The process of GA can be shown as the following steps (Ushakov, 2013):

1.) Initialisation – Random selection of genes (population of solutions which contains the parameter values) from gene pool and determination of the number of selecting genes.
2.) Selection – each set of the parameters are evaluated by a fitness function (similar to the cost function) in order to measure the quality of the represented

solution. Genes with better quality than the requirement stay alive while others die off. The results are called first generation.

3.) Genetic operations – this step is all about generating a second generation population of solutions. The second generation basically contains a higher quality of genes than the earlier generation. In terms of breeding, the algorithm randomly selects the pairs of individuals and shares many of the characteristics of its individuals using genetic operators (the most widely use are crossover or mutation).
4.) This generational process (step 1, 2 and 3) keeps repeating until a termination condition has been reached.

The genetic algorithms has been studied in many research articles such as Khlaifi et al., (2009). These represent the combination of the Gaussian dispersion model with genetic algorithm in order to identify SO_2 emitting source. This study utilises real data obtained from three different locations in France over a 24 hour period on the 11th March 1992. It showed promising results. Likewise, the research from Haupt (2005) proved useful for using genetic algorithm in dispersion modelling specifically in source position identification applications. In addition, the main advantages of GA are that it can deal with a large number of parameters, it does not require initial values and its ability to utilise global search and provide more than a single solution (Goldberg, 1989).

Even though all three reference algorithms have their own specific advantages and limitations, they have one commonality; they all require the pollutant dispersion model. However, as I mention earlier in the section on the disadvantage of the analytical approach, there are no theoretical models to fully support the characteristic or behavior of pollutants dispersed in conditions such as air and ground water. Therefore, these kinds of support models may constitute the weakest link of the total system.

Fortunately there is one well-known optimization algorithm, which does not require the dispersion model. This is the artificial neural network. The research of Reich et al., (1999) and Rege and W.Tock, (1996) demonstrates this by using artificial neural networks for source identification of unknown air pollution and for estimating emission rates of hydrogen sulfide and ammonia respectively. Moreover, they also suggest that artificial neural networks are a promising method to deal with identification of patterns, specifically in the presence of noisy or ambiguous data (in non-linear systems). For this reason I have decided to select this algorithm to represent the optimization approach and to compare it with the other three approaches applied to subsea leakage source identification. The fundamental advantages and disadvantages are discussed below.

2.2.3.4. The artificial neural network

Artificial neural network (ANN) is a mathematical model that was designed to mimic the learning process of humans. Human learning processes are typically based on being exposed to a range of experiences. For example, a proofreader who works at a publisher for many years is likely to proofread faster and has a higher accuracy compared to a newly graduated one. This is due to the fact that a professional proofreader has had much greater exposure and familiarity with words, sentences and syntax. The underlying concept is that humans can learn to recognize and respond to patterns, which they have experienced and have been frequently exposed to.

Artificial neural networks (ANN) are designed using similar principles. They can be trained to recognize patterns in data as well as learning and understanding the relationships between inputs and outputs. This enables an ANN to make predictions based on the unseen inputs within a range of historical data used for training purposes (Rege and W.Tock, 1996). The process of ANN development can be divided into two main steps. Those are:

1. Initial stage – the ANN architecture as well as relevant coefficients, such as bias and weight, have to be carefully set up with the appropriate values.
2. Training stage – a significant amount of the data such as observed concentration (input) and leakage source location (output) are presented to the ANN. Then the model creates mapping of the input-output correlations in an underlying process. This is the key stage to determine the performance of the ANN in terms of prediction or forecasting.

The advantages and disadvantages of using ANNs for subsea leakage source identification are summarized and shown below:

<u>The advantages</u>

1.) An event or accident usually consists of three elements: These are cause, consequence, and system, which links cause to consequence. The other algorithms reviewed earlier all need the elements of consequence and system for inverse calculation to find the cause. This can sometimes create a problem due to a lack of models, which represent the 'real' environment and are usable. However, this problem is eliminated by ANNs, which do not require a methane dispersion model, but only concentration data and the location of leakages. This means this algorithm can easily be applied to a high-complexity and non-linear system.
2.) Most models of dispersion are based on estimated parameters such as dispersion coefficients in the horizontal and vertical directions (in case of the Gaussian model), which are difficult to incorporate into a predictive model. Moreover, in case of creating new air dispersion models, factors such as

atmospheric stability characteristics have to be fully understood and taken into consideration to generate a usable model. This can be very complex and time consuming. Fortunately, ANNs are not subject to those constraints by being a completely empirical model (using relationship rather than logical explanation) (Rege and W.Tock, 1996). This can reduce or eliminate the need for specific knowledge of related parameter such as turbulence or vortices occurring in the considered area needing to be taken into account.

3.) As ANNs are completely empirical models, as well as a reliable predictive tool, they have the ability to solve inverse problems that contain noisy or ambiguous data. This means that ANNs have greater tolerance of noise, which can improve the overall system performance. This is a very useful feature, as small errors from measurements can cause large oscillation in the results (Li and Niu, 2005).

4.) Many of the optimisation approach techniques require a much larger amount of time per iteration to solve the problem. With ANNs, the main time-investment occurs during the training process only. This means that subsequently ANNs can predict the output almost in real time in operational conditions.

The disadvantages

1.) As mentioned earlier, ANNs are a new approach to deal with complex and non-linear problems and are based on soft computing methodology. Therefore, to find the theoretical guidance that supports each specific application can be very difficult. Constructing a model is fundamentally based on trial and error of the different values and theories in order to determine and improve the performance of the ANN. This is a time consuming process and has its own inherent problems.

2.) The output of the optimisation approach is estimated data, which minimises data misfits between the database and calculated value. This implies that the optimal point should be presented in the lowest point (global minima) of the objective function. Normally, only linear problems can contain one minima (or maxima) point. Unlike the nonlinear problem, the objective function has a large number of local minima and maxima as shown in Figure 8. (Snieder, 1998). The fact that ANNs use a backpropagation algorithm as its local optimization method can lead to misidentification of a local minimum as the global minimum. In addition, the final estimates (or optimal solution) obtained by local optimization methods cannot always easily be verified if the result is located at the global minima or at one of the local minima.

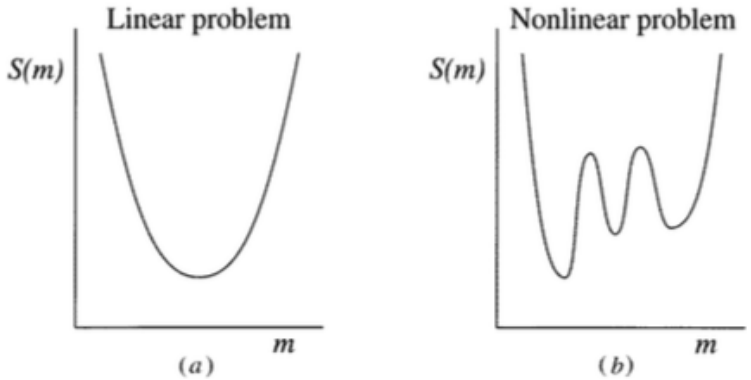

Figure 8 the example of the objective function (a) for linear problem which has one minima (b) for nonlinear problem which has a large number of minima and maxima (Snieder, 1998).

3.) In order to train an ANN a high number of data sets are required. This data has to be sufficiently scattered over a range of variables in order to capture all possible scenarios for a given process to increase the ability of prediction of an ANN. However, the vast sets of data often prove to be computationally and financially challenging as well as time consuming.

2.2.4. The Direct Inverse Approach

The direct inverse approach is the methodology that solves the inverse problem by reversing the governing equations, which can describe cause–effect relations. The implication is that this modelling method basically starts from the end result running simulations counting down time to zero in decremental steps in order to obtain the contaminant concentration evolution (from end point to starting point) (Liu and Zhai, 2007). The original techniques for direct inverse modelling are the regularization technique and the stabilization technique. These techniques can improve the solution accuracy by reducing the instability of reversed governing equations. In addition, Kato's et al., (2004) research, which focuses on atmospheric contaminant transport modelling, evaluates a contamination source by backward trajectory analysis of the flow. This technique reverses only the contaminant transport by convection. This is simple to implement and can approximate the contamination source. However, the method can cause calculation problems when the convection is weak.

The process of the direct inverse approach can be separated into two main steps: operator identification and numerical methods. The first steps is concerned with finding/adjusting a suitable operator to perform decremental time steps with the operator being still stable.

According to Zhang and Chen, (2007) and Bady, (2013) research, which uses direct inverse modelling to identify the contaminant source in enclosed environments and

urban areas respectively, the pollutant concentrations are based on the convection-diffusion equation (it can be in solid, liquid, and gas form). However, this original equation has to be slightly adjusted for the inverse calculation as shown below:

The original governing transport (without source) equation:

$$\frac{\partial[\phi(\tau)]}{\partial\tau} = -\frac{\partial}{\partial x_i}[u_i\phi(\tau)] + \frac{\partial}{\partial x_i}\left[\frac{\Gamma}{\rho}\frac{\partial\phi(\tau)}{\partial x_i}\right]$$

The adjusted equation:

$$\frac{\partial[\phi(\tau)]}{\partial\tau} = -\frac{\partial}{\partial x_i}[u_i\phi(\tau)] + \varepsilon\frac{\partial^2}{\partial x_i^2}\left[\frac{\partial^2\phi(\tau)}{\partial x_i^2}\right]$$

Based on one-dimensional flow as shown in Figure 9 and implicit format, the adjusted equation can be discretized into:

$$\phi_p(\tau+\Delta\tau) = \frac{1}{1+\frac{u_e\Delta\tau}{\Delta x}-\frac{6\varepsilon\Delta\tau}{(\Delta x)^4}}\cdot\phi_p(\tau) + \frac{-\frac{4\varepsilon\Delta\tau}{(\Delta x)^4}+\frac{u_w\Delta\tau}{\Delta x}}{1+\frac{u_e\Delta\tau}{\Delta x}-\frac{6\varepsilon\Delta\tau}{(\Delta x)^4}}\cdot\phi_w(\tau+\Delta\tau) + \frac{-\frac{4\varepsilon\Delta\tau}{(\Delta x)^4}}{1+\frac{u_e\Delta\tau}{\Delta x}-\frac{6\varepsilon\Delta\tau}{(\Delta x)^4}}\cdot\phi_e(\tau)$$

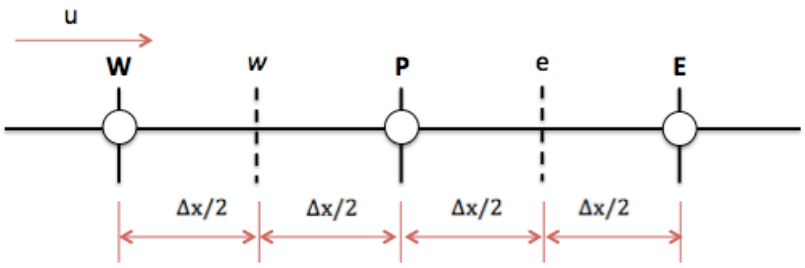

Figure 9 Typical control volume for the one-dimensional flow

This equation can be used to identify contaminant sources as long as the value of ε is larger than $u_e\Delta x^3/6$.

The second step of the direct inverse approach deals with numerical methods. Reversed equations and numerical schemes are forwarded to a simulation platform,

such as the CFD program, that helps answer the question based on the finite-volume discretization. For reverse simulation we assume that the event is taking place during the period between 0s and 200s. Therefore the distributions of airflow and transient pollutant concentration at t = 200s become an initial data point for the inverse CFD modelling. Later, CFD starts solving the transport equation from this moment using negative time steps until reaching t = 0s. At this point the pollutant source can be identified by the location that contains the highest concentration (Bady, 2013). Based on my review of this approach, it is similar to other approaches in that they all have both advantages and disadvantages for use in source identification. These are described below:

The advantages

1.) The first main advantage of the direct inverse approach is that it needs less contaminant source data compared to the other three approaches. Based on the Bady's (2013) case study, the adjusted equation requires only the flow field of air and the pollutant concentration at the last time step. These two different types of data can easily be extracted from sensors (i.e. airflow meter for flow rate and methane sniffer for contaminant concentration).
2.) The direct inverse approach seems to have greater ability to solve high complexity problems when compared to the analytical approach, for example. It also can be used in other areas and applications. For example Zhang and Chen, (2007) use this approach to identify the contaminant source in the office environments, which have many obstructions and specific air flow characteristics such as inlets located on the rear wall at the floor level and outlets is in the ceiling in the centre of the room. This kind of scenario might be too complex for the analytical approach but not for the direct inverse approach.

The disadvantages

1.) The direct inverse approach requires CFD with its high demand on processing resources to calculate the leakage source. The domain (considered area) will be discretized into a finite number of elements or cells; the higher number of cells, the more data will needed to be calculated. Therefor this method demands high calculation efforts (it is very common that CFD process require millions of calculation), which require often the use of supercomputers and long calculation times.
2.) From the case study of Bady (2013), the governing transport (which can be called the operator of the system) has to be adjusted in order to make an equation become stable in inverse modelling. This process sometime might be too complex because of the adjusted equation not being exactly the same as the original governing transport for contaminant as the solution of contaminant source might not be accurate, especially in a complex case as

show in Figure 10. The maximum concentration at t = 1s obtained by forward simulation is a spot next to the box, whereas the results from reverse simulation (c.) is more dispersive which makes it harder to pinpoint the real polluting source.

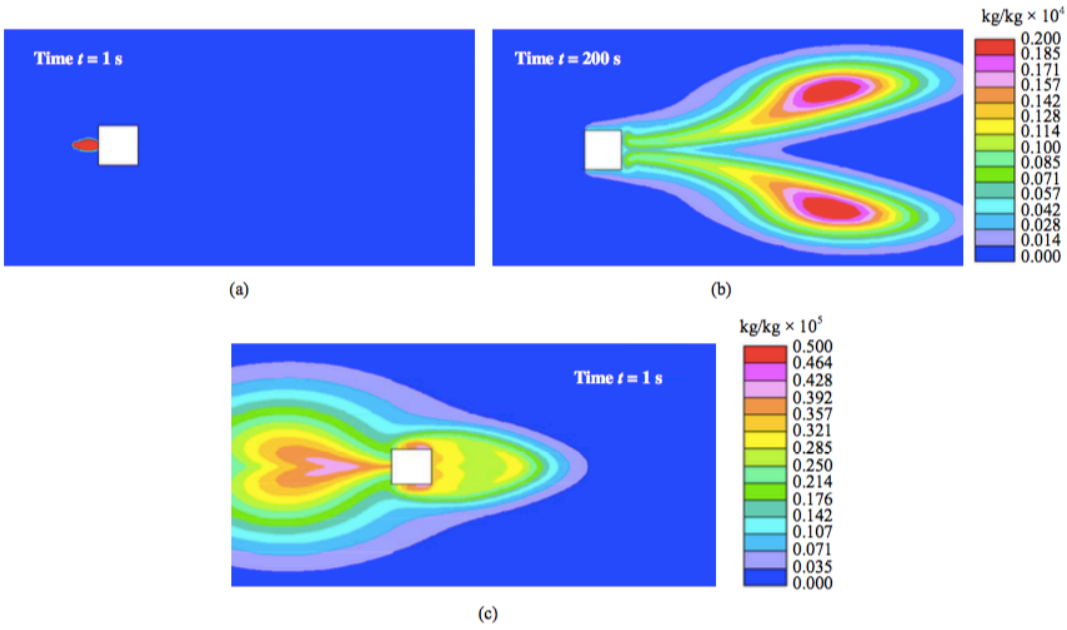

Figure 10 Concentration fields, (a) at t = 1 s obtained by forward time simulation; (b) at t = 200 s obtained by forward-time simulation; (c) at t = 1 s obtained by inverse simulation (Bady, 2013).

3.) This approach requires time between the initial leakage and the time it reaches the sensors, as well as the concentration. This can reduce the accuracy of the results if the simulation cannot stop at the right moment (contaminant strength will become more dispersive while the simulations keep running).

2.2.5. Summary

The four different methodologies used in inverse problem approaches all have their own advantages and disadvantages. The analytical approach does not provide sufficient information and is not able to simulate the complexity of subsea leak detection scenarios and therefore cannot be used in such systems. Equally, the probabilistic approach, whilst potentially yielding usable results, is not appropriate for subsea leak detection because of its need for large amounts of accurate historical data. This is often not practical in an operational environment. Therefor the probabilistic approach will not be considered further in this master thesis.

The direct inverse modelling approach is very interesting. It is a very straightforward method with only a few data points needed for calculation, and no requirements for obtaining large amounts of historical data. Similar to some of the other approaches, the main drawback is the time needed for running caluclations. The more accurate the results, the more resource is required. Whilst the long calculating times can be alleviated by using a super computer, this will definitely increase the capex cost of the system.

This leaves the optimization approach. This approach requires significant investment in building a database, which matches inputs and outputs based on historical data (training data) obtained by forward CFD modeling. This increases the time to simulate each scenario.

In case of a serious incident, real-time prediction is one of the most important tools for successful crisis management. The artificial neural network is the optimization approach method, which has been chosen for this study because it can run in almost real time and can easily deal with a non-linear system. Even though the optimization approach requires a large database of information to improve the accuracy and demands a large amount of forward CFD modeling, the database can be set up prior to any incident occurring. That means that long processes of acquiring information for the database would not interfere or create a huge load of calculation for the identification of the leakage source once the incident is occurring, meaning the system can work almost in real-time.

Vukovic and Srebric, (2007) mentions that for successful source determination, real incidental events have to fall within the scope of the database. That means a comprehensive database requires various combinations of parameters and covers as many real incidents as possible. Therefore, in order to keep resource and time requirements at manageable and cost-effective levels, the ANN can replace these pre-populated databases with the neural network's nonlinear mapping. This alternative approach can reduce the size of databases and is still able to provide relevant results. This will make leak source detection technology easier to deploy, faster to compute and produce more accurate predictions.

In addition, as the optimization approach is counted as regressive inverse modeling, Willmann et al., (2015) suggest three solutions that might help to improve the performance of regressive inverse modeling. Those are: 1.) Introduction of prior knowledge, 2.) Consideration of other data sources, and 3.) Leaning how to handle uncertainty. Fortunately, the architecture of ANNs is extremely flexible and is either suitable or unsuitable with the only decisive criteria to be considered its performance for the task in hand. Thus, all of the three features can be integrated easily by customizing the ANN architecture.

For all the reasons mentioned earlier the artificial neural network will be used in the present study to identify any potential for its use in leakage source identification, especially in the subsea environment where ANNs have not yet been used extensively or indeed at all.

2.3. ANN concepts and tools.

Using artificial neural networks for leakage source identification requires a well designed algorithm (the artificial neural network), which improves the accuracy of the outputs, as well a good database which is essential for the ANNs learning process. For this thesis, I am generating a database (set of patterns) based on CFD.

Therefore, information in this part is divided into two sections: the concepts of the artificial neural network (ANN) and the concept of computational fluid dynamics (CFD). In addition, the introductions of the applications or toolboxes, which are used to implement both ANN and CFD in this master thesis, are also briefly described.

2.3.1. Artificial neural networks (ANNs)

2.3.1.1. Introduction

In 1958, psychologist Frank Rosenblatt developed the first artificial neural network, which was called Perceptron (and is today referred to as artificial neuron). This artificial neuron was derived from observation of biological neurons, which include soma, dendrites and axon. Generally in the biological process of each neuron, information comes into the neuron via dendrites (receivers) first. Soma processes the information and then passes it to an axon (sender). Therefore, an artificial neuron is a mathematical model, which mimics the structure and functionalities of a biological neuron. The structures of both the artificial and the biological neurons are shown in Figure 11.

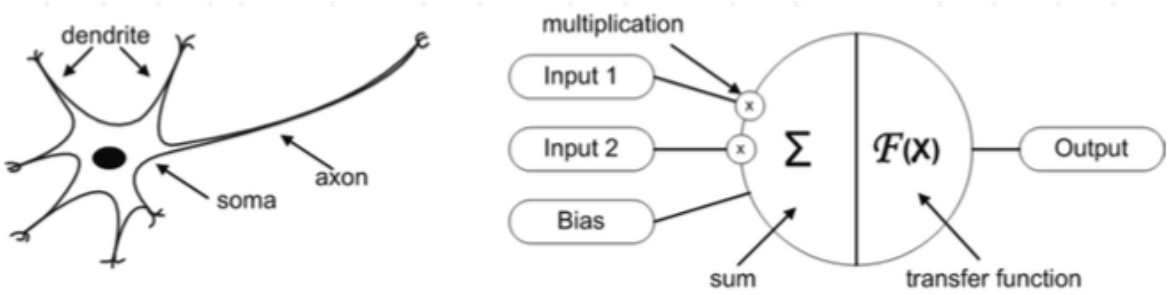

Figure 11 Biological and artificial neuron design (Krenker et al., 2011).

An artificial neuron has three simple sets of rules to process the information, which flows though the neuron. Those are the multiplication, summation and activation rules. The information comes into the body of the neuron via inputs and every input value is multiplied with individual weight and bias values. This section is called multiplication. The summation part summarizes all weighted inputs and biases. Finally this sum of weighted inputs is transferred into the activation function (also called transfer function). This process will generate the output(s), which are readable and understandable for a human operator. At the end of the system, the processed information is passed to users via output channels (Krenker et al., 2011). The process of an artificial neuron is shown in Figure 12.

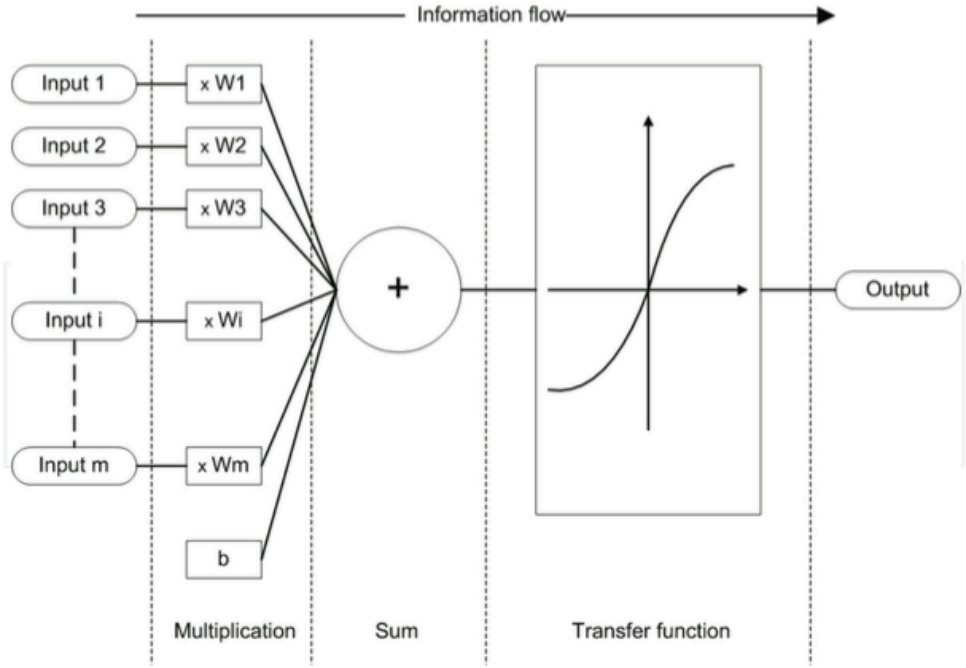

Figure 12 The process of artificial neuron (Krenker et al., 2011).

Based on the Figure 12, each artificial neuron can be shown in mathematical form as the following equation (Krenker et al., 2011).

$$y(k) = F(\sum_{i=0}^{m} w_i(k) \cdot x_i(k) + b)$$

where: $x_i(k)$ is input value, $w_i(k)$ is weight value, $y_i(k)$ is output value, b is bias, and F is transfer function.

The equation demonstrates the working principle of artificial neurons and it is simple and does not require specialist calculations. However, the human brain (biological neuron network) consists of more than a million neurons, which are connecting each other with both dendrites and axons in order to process visual data and learn to recognize objects, for example. This also applies to an artificial neural network. Therefore ANNs can become the solution, which can provide higher calculation power to address several types of high complexity problems. These are difficult or impossible to solve with conventional methods. The example of the artificial neural networks which consists of several single neurons is shown in Figure 13 (Almurib et al., 2011).

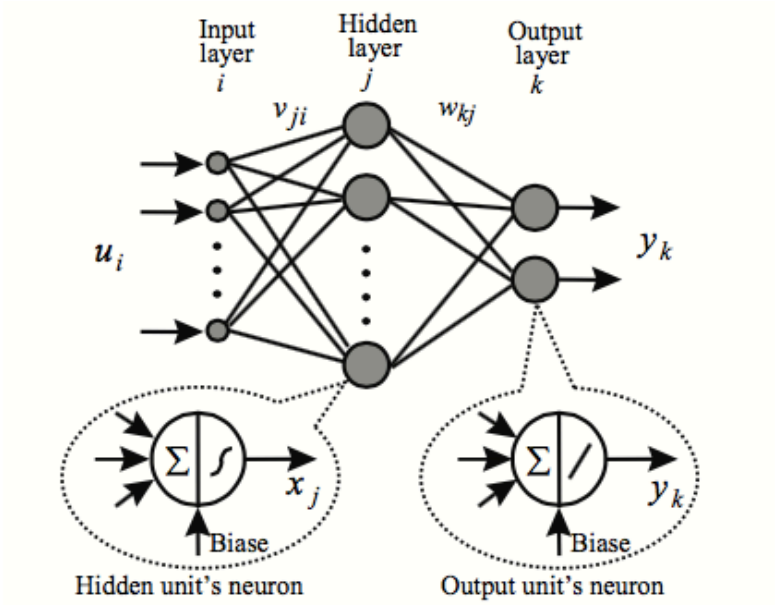

Figure 13 A two layer artificial neural network (Almurib et al., 2011)

In addition, to provide usable outcomes, the artificial neural network needs to be taught to solve the type of given problem before implementation. Every time the artificial neural network is learning something new, the weight values inside of each neuron are adjusted to respond to the new information. This is similar to human brain cognition (biological neural networks), which learns from the experiences on the bases of how humans interact with the environment, both past and present.

The artificial neural network is a mathematical model, which mimics the structure and functionalities of biological neural networks. That means it demands an understanding of basic mathematical concepts, which are used to describe and analyze neural network processing. This can help model designers to deliver results, which are comparable with those that the human brain can produce. In addition, each neural network has it own specification values and functionalities depending on the problem and the expected outcome. In a broader sense creating an artificial neural network comprises three principle elements. These are:

Neural network components – how basic components can influence the design, implementation and use of artificial neural networks.
Topology – how an artificial neural network is organized into layers and how the layers are connected.
Adaptation – how an artificial neural network is configured to deal with incoming information.

2.3.1.2. The artificial neural network components

The artificial neural network consists of many simple processing units (neurons), which communicate by sending weighted signals to each other. Therefore the main components of each neural network have to be combined with at least two structural components: connection weights and processing units. However, there are also some sets of basic factors that can contribute to the design, implementation and use of neural networks, in order to obtain the best results. Those are processing element activation functions and input/output patterns. Each of these elements are examined below:

1. Processing units

The main role of processing units can be separated into two main simple tasks. The first role of the processing unit is as the center of information flow. It receives the input from other processing units or from external sources, processing this information to generate an output signal, and then sends it to different destinations. As the center of information flow, the processing units can be sorted into three types based on functionality. Those are input units, which receive information from outside the network, output units which transfer information to the target outside of the neural network, and hidden units with input and output signals continuing inside the system.

All of them are laid out in input layer, output layer, and hidden layer respectively as shown in Figure 13. In addition, even though there are many units connected with each other, all of them carry out their computations at the same time (Alavala, 2008). Another function of processing units is the adjustment of the weights, which let neural networks become adaptive systems.

Building the number of processing units for a neural network has to be well defined as it directly affects the models performance. Too few processing units lead to suboptimal outputs, whereas too large a number can increase processing time, cost and also result in poor predictions due to overfitting. The concept of overfiting is described in Appendix C.

2. Connection between units (Network Weight)

Based on Figure 12, we can see that apart from exchanging the data signal between processing units, artificial neurons also extend their performance by including the weight value in each (input) connection. The main function of the weight value is to define the information flow through the network and also modify the amount of information transferring between processing units. Typically, the connection weights are automatically updated (either synchronously or asynchronously) to fit with the set of information during the training process. Connection weights with positive values are excitatory connections, whereas negative values are inhibitory connections (Eberhart and Shi, 2007).

3. Input and output pattern

Neural networks require good data sets for the training process in order to exploit their full learning potential. The network can be trained in two different ways: Training with single patterns (inputs) is defined as an autoassociative network, whereas networks that use pattern pairs (both inputs and outputs) are called heteroassociative network (Eberhart and Shi, 2007). In addition, the key issue is to determine what patterns should apply to the neural network. The data (input and output patterns) used for the training process should be able to fully represent all possible features of the system. This allows neural networks to deal with any type of unseen information. The better the training data, the higher the performance the model can achieve.

4. Processing Element Activation Function

The transfer function (F) is typically used to map a processing unit's domain (i.e. inputs, weights, and bias) to produce the new value of output signal. Also the value of each output signal is located within the pre-specified range for example -1 to 1, or 0 to 1 (Eberhart and Shi, 2007). This means that each output is standardized to easily connect with other processing units. In addition, transfer functions can be any mathematical function, depending on the problem to be solved. Examples could be that the linear function performs well with other linear functions, whereas the step

function, the ramp function, the sigmoid function, and the Gaussian function are best suited to non-linear functions. Five of the activation functions are shown in Figure 14.

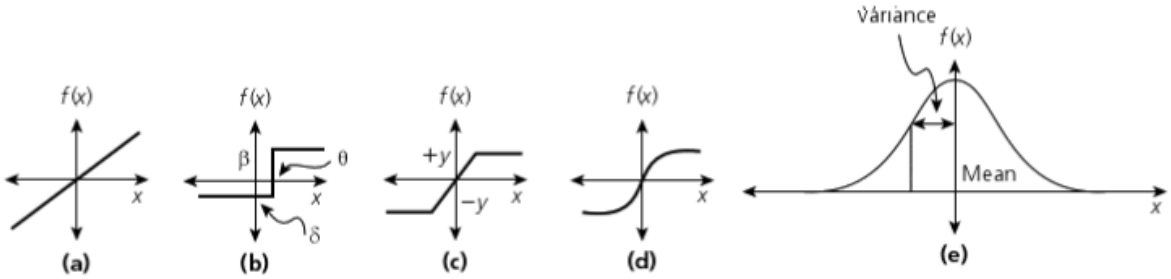

Figure 14 Five of the activation functions: a) the linear function, b) the step function, c) the ramp function, d) the sigmoid function, e) the Gaussian function (Eberhart and Shi, 2007).

The sigmoid function was deployed in this study because of its s-shaped function with a simple formula, which creates output values in the range of -1 to 1. That can be important when calculating weight updates in the artificial neural network, especially when we are mainly focusing on time and accuracy.

2.3.1.3. Topologies

Even though the neural network is created by interconnecting of individual artificial neurons, the random connection between each neuron may lead to high complexity and an unmanageable system. Therefore, researchers have defined and created several standardized topologies of artificial neural networks in order to solve the different types of problems with easier, faster and more efficient solutions. However, the topologies of the neural network can be clearly separated into two different types of networks based on patterns of connections (Krenker et al., 2011).

Feed forward artificial neural networks are those network in which the information must flow from input to output unit in only one direction without back-loops connections. A good example of feed forward networks is backpropagation, which will be discussed later. In contrast, recurrent artificial neural network allow the information to transmit backward. This feed back loop creates the internal stable state of the network, which allows the network to exhibit dynamic behavior.

The main distinctions between both networks: feed forward (FNN) and recurrent (RNN), are shown in Figure 15.

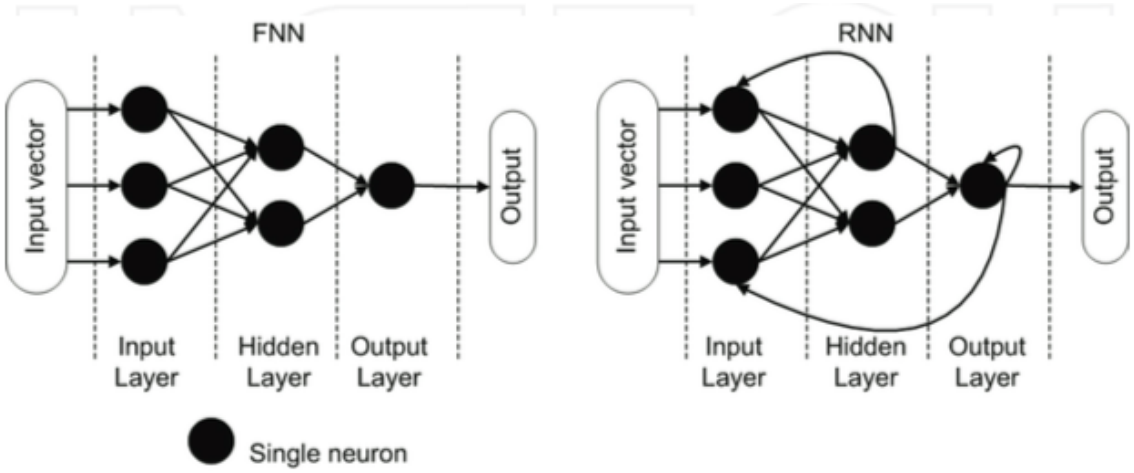

Figure 15 Feed forward neural network and recurrent neural network (Krenker et al., 2011).

2.3.1.4. Training of artificial neural networks

Irrespective of which topology is used or how big the model is, it will have to learn to respond to inputs given by the environment. This task can be achieved by the training process, which will enable an ANN to become a valuable solution tool. The way to train the network is to provide teaching patterns to the model, then letting it adjust the values of weight and biases (Alavala, 2008). The two major training paradigms are:

Supervised learning – the network is trained by providing the pairs of input and desired output values. The performance of the network is limited by the range and quality of training data. This paradigm is mostly used in the area of pattern recognition problems.

Unsupervised learning – this is the machine learning technique that allows the system to be trained and to develop its own performance. Every time the network (with unsupervised learning) corrects the new input, the model will learn and adapt itself automatically. This paradigm is mostly used in the area of estimation problems.

2.3.2. The backpropagation algorithm

There are hundreds of artificial neural network types that have been introduced over the last few years, such as the Hopfield network, the Bi-directional network, and the Self-Organizing Map (SOM). However, there is a small group of classic networks, which are well known and widely used in many applications. One of them is the feed forward backpropagation neural network or backpropagation network. This network will be explained here as it is deployed in this study.

The popularity of using backpropagation networks arises from its simple architecture combined with a basic learning process. The architecture of backpropagation is feed forward network and multilayer perceptron (MLP). The multilayer perceptron architecture has to consist of at least 2 layers: one hidden layer and one output layer. Apart from this criterion, a MLP can have any number of layers, units per layer, inputs and outputs. Actually, the name 'backpropagation' does not refer to the architecture with feed forward network, but it rather represents the learning or training algorithms. The backpropagation scheme is a supervised learning approach, which consists of two major steps: the forward activation and the backward error flows. These two steps will occur every time when a training pair is fed into the training process. (Moustafa et al., 2011).

The basic working process of backpropagation algorithm can be briefly described like this (The IDEAS Research Institute, n.d.):

1.) Providing the input and its corresponding target (output). Both of them are called a training pair. The training pair is a representative of what we want the network to perform.
2.) Once the training pair is provided, the network initializes all its weight values. Next, input from the training pair is used to calculate the output (this is called the forward pass). In addition, this initial output could be any value, as all weight values are randomly defined.
3.) Calculating the error of each neuron by solving the difference between an actual output and calculated output (target – actual output). This error is then processed by mathematical method to change the weights in order to reduce the error. This results in outputs of each neuron, which closely approximate outputs collected in the field (this part is called the backward error flows or backpropagate).
4.) Repeating step 2 to 4 until the error is minimal or reaching the stopping criteria.

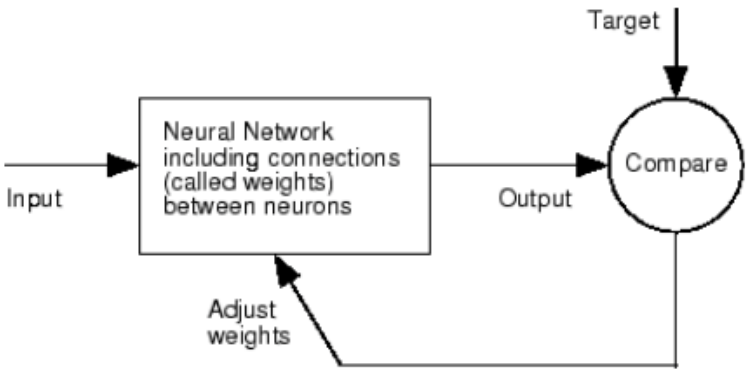

Figure 16 The working process of backpropagation neural network
(Demuth et al., 2013)

The process above indicates that a backpropagation algorithm corrects the mistakes made by the network in the training process as it will be sending the information backward through the network in order to correct weight value as is shown in Figure 16. In addition, a backpropagation algorithm use the concept of a gradient descent to reduce errors between the actual output and the desired output of the network (Moustafa et al., 2011).

2.3.3. Neural Network Toolbox

As artificial neural networks are mathematical models we can comprehend the concept behind the algorithms, which are relatively easy to calculate. However, this is only practical for problems that contain relatively short mathematical equations, such as the artificial neural network with two or three neurons. Therefore, in order to solve more complex problems, it is advantageous to use computers and software to build the model as well as optimize any parameters of the artificial neural network. An artificial neural network can be created from scratch based on computer language such as C/C++, Java, and Python. However, there are several specialized commercial programs or toolboxes that can support the artificial neural network development. One toolbox, which has been used here, is the "Neural Network Toolbox". It is a one of several extension toolboxes, which are provided by Matlab.

The Neural Network Toolbox is a useful tool to design, train, visualize, and simulate neural networks, as well as providing several features such as supervised and unsupervised learning, parallel computing to reduce calculating time, and Simulink block to connect neural networks with other systems or applications (Demuth et al., 2013a). This allows researchers to evaluate the neural network with minimal effort.

In addition, there are two major ways to use this toolbox depending on the levels of users from novice to expert. The easiest way to operate the program is through the Graphical User Interface (GUIs) as it provides a quick and easy way to solve the problem, although it is limited to only a few specific problems. A more advanced way to use the toolbox is through the command-line. It offers more flexibility than the GUIs and has the advantage of being able to create custom neural networks based on any of the functions contained in the toolbox. In addition, this way also offers the ability to modify every computational component contained in the toolbox, however Matlab code is required (Demuth et al., 2013a).

2.4. Computational fluid dynamics (CFD) concept and tool

2.4.1. Computational fluid dynamics (CFD)

The basic way to calculate fluid (gas and liquid) flows is by using governing equations or partial differential equations (PDE), which represent conservation laws for the mass, momentum, and energy. However, in certain circumstances the flow governing equations can be extremely complex for solving and validating the results. Therefore, computation fluid dynamics (CFD) is introduced to replace the governing partial differential equation systems with a set of algebraic equations. It is much easier to solve with computers and cheaper in terms of testing fluid flow systems. In addition, CFD provides the ability to run and test any hypothesis in severe conditions, which would otherwise be impossible or extremely difficult to set up, such as the measurement of complex flows in chemical reactors or furnaces (Kuzmin, n.d.).

Computational fluid dynamics (CFD) provides a qualitative and quantitative estimate of fluid flows by using three basic concepts of knowledge. Those are a mathematical modeling (partial differential equations), a numerical method (discretization) and software tools (solvers, pre- and post-processing utilities) (Versteeg and Malalasekera, 2007). That means CFD enables scientists and engineers to perform numerical experiments in a virtual flow laboratory based on coding.

The three main steps of CFD working process (which relate to coding) are:

1.) Pre-processor

After determining a problem statement, the mathematical model and all relevant parameters as well as its values have to be carefully defined (Kuzmin, n.d.). This could include:

- Selecting a suitable flow model and reference frame.
- Identifying any forces, which cause and influence the fluid motion.
- Defining the geometry (computational domain), which will be solved.
- Simplifying the governing equation to reduce the computational efforts.
- Determining the fluid properties as well as specify initial condition and boundary conditions.

In addition, grid size (or mesh size) in the area under consideration has to be well defined in this step. The finer the grid, the more accurate the solution obtained.

2.) Solver

CFD applies a numerical method to develop approximations of the governing equations. This is called the discretization process. This process starts with integrating the governing equations over each cell of the grid set. CFD solves flow field variables

at each cell to provide absolute solution (Bakker, 2002). The process of discretizing is the key element of CFD or Finite Element Method.

3.) Post-processor

The simulated results obtained from the discretization process are generated. The results can be presented in two formats:

- Raw data (number) from calculation of integral parameters such as lift force and drag force.
- Visualization, which represent numbers as images and animation. This can be shown in 1D, 2D and 3D, for example function values connected by straight lines, contour levels and cut planes respectively.

The CFD (simulation) method is today quickly replacing testing methods (e.g. sensory data capture) in several analysis applications, as it is far cheaper and faster to implement. However, results of a CFD simulation are subject to a certain amount of prediction or imprecision as the input data can bias the output (Sayma, 2009). To take these potential weaknesses into account, a good understanding of physical characteristics and systems is required in order to obtain a reliable result.

2.4.2. OpenFOAM

Most CFD packages used in industry and the academic environment are now largely commercial programs such as ANSYS and ABAQUS. The implication is that CFD packages might be costly, the implementations of models in these packages are limited, and their source codes are not provided. Potentially OpenFOAM or Open Source Field Operation and Manipulation, which are open source applications, could act as alternative high-performance CFD tools.

OpenFOAM is a free and open source software under the GNU General Public License, which operates for the Linux platform. It basically uses C++ toolbox for the development of customized numerical solvers, and pre-/post-processing utilities for the solution of continuum mechanics problems, including CFD (Hjertager, 2009). This offers several advantages for developers. The advantages of OpenFOAM are summarized below (Silva and Lage, 2011):

- It offers numerous pre-configured solvers, utilities, and libraries, especially to solve fluid flow problems with a large variety of turbulence models for RANS and LES integrated, similar to any commercial simulation packages. The OpenFOAM Structure is shown in Figure 17. It is supplied with solver, pre- and post-processing environments.
- OpenFOAM is open in terms of source code and its structure and hierarchical design. The new solvers, utilities and libraries can be created by users and shared with others to cover a wide range of problems and also

creates a range of new possibilities. This enables OpenFOAM to become a fully flexible and extensible program.

- This program is free of charge for all users. Hence, capex cost can be reduced.

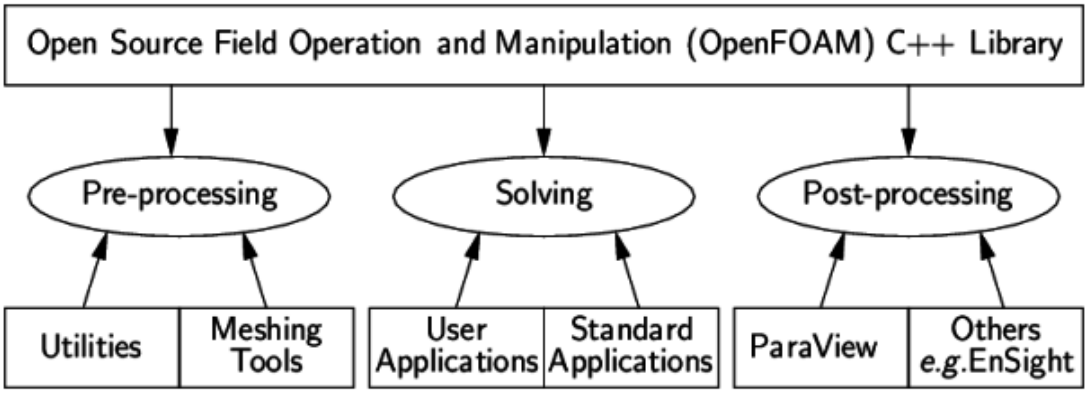

Figure 17 Overview of OpenFOAM structure (Hjertager, 2009)

2.4.2.1. OpenFOAM case structure

OpenFOAM is a program that is based on coding rather than graphical user interfaces (GUIs). Each case is saved as a directory on the computer. Therefore, to adjust, edit or set all parameters, users have to manually code before solving the equation.

The case directory includes as the minimum the following directories (Hjertager, 2009):

1.) *Constant* – which control mesh characteristics, material properties, and turbulence properties that correspond to each problem.
2.) *System* – which defines number of iterations, time step size, and solution controls.
3.) *0* – which determines initial flow fields and boundary conditions.

The case structure is shown in Figure 18.

Within the OpenFOAM toolbox, there are several different solvers, which are designed to solve various specific problems. Only two solvers are selected to solve the problem regarding the experiment in this thesis. These are:

pisoFoam – which is a transient solver for incompressible flow. It was deployed to simulate the flow field of seawater and turbulences over the considered area.

scalarTranportFoam – which is used to solve a transport equation for a passive scalar. It was used to simulate the dispersion of methane from different leakage locations within subsea conditions.

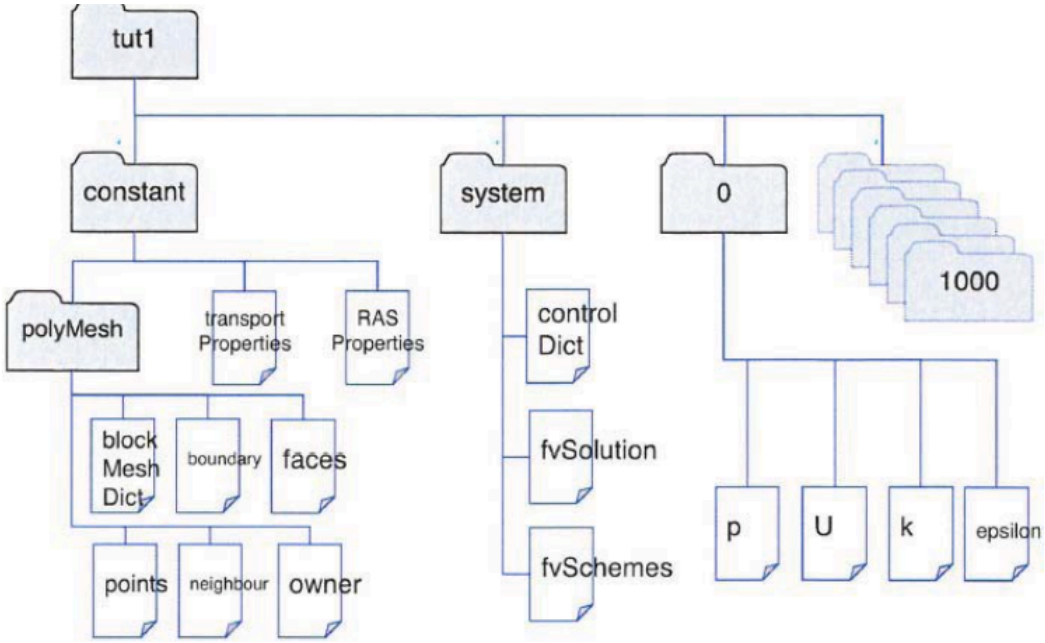

Figure 18 OpenFoam case structure (Hjertager, 2009)

2.4.3. PisoFoam

The pisoFoam is a transient solver for incompressible fluid flows for both laminar and turbulent conditions. The pisoFoam is based on the PISO algorithm, which stands for Pressure Implicit with Splitting of Operators. This algorithm was introduced by Issa in 1995. The pisoFoam is a pressure-velocity calculation process, which basically involves one predictor step and two corrector steps. The predictor step uses the momentum equations in order to solve an intermediate pressure field (It is the concept of the SIMPLE algorithm.). While two corrector steps are used to correct and ensure values of relevant parameters in a way to satisfy the momentum and continuity equations (Versteeg and Malalasekera, 2007). The pisoFoam can therefore be described as an extension of the SIMPLE algorithm by adding corrector steps in order to ensure the calculated values.

The PISO algorithm (or PISO loops) consists of the following steps (Versteeg and Malalasekera, 2007):

Predictor step

1.) Initiate guess p^*
2.) Solve discretize momentum equations:

$$a_{i,J}u^*_{i,J} = \sum a_{nb}u^*_{nb} + \left(p^*_{I-1,J} - p^*_{I,J}\right)A_{i,J} + b_{i,J}$$

$$a_{I,j}v^*_{I,j} = \sum a_{nb}v^*_{nb} + \left(p^*_{I-1,J} - p^*_{I,J}\right)A_{I,J} + b_{I,J}$$

The results obtained from this equations are $u^*$and v^*.

3.) Solve pressure correction equation below to obtain value of p'

$$a_{I,J}p'_{I,J} = a_{I+1,J}p'_{I+1,J} + a_{I-1,J}p'_{I-1,J} + a_{I,J+1}p'_{I,J+1} + a_{I,J-1}p'_{I,J-1} + b'_{I,J}$$

Where; $a_{I,J} = a_{I+1,J} + a_{I-1,J} + a_{I,J+1} + a_{I,J-1}$

And the coefficients are given below:

$a_{I+1,J} = (\rho dA)_{i+1,J}$
$a_{I-1,J} = (\rho dA)_{i,J}$
$a_{I,J+1} = (\rho dA)_{I,j+1}$
$a_{I,J-1} = (\rho dA)_{I,j}$
$b'_{I,J} = (\rho u^* A)_{I,J} - (\rho u^* A)_{i+1,J} + (\rho v^* A)_{I,j} - (\rho v^* A)_{I,j+1}$

The value p' $u^*$and $v^*$are used to calculate with following equations to obtain the new values (p^*, u^* and v^*). These values are more accurate compared to the initial values.

$$p^*_{I,J(new)} = p^*_{1,J} + p'_{1,J}$$

$$u^*_{I,J(new)} = u^*_{I,J} + d_{I,J}\left(p'_{I-I,J} - p'_{I,J}\right)$$

$$v^*_{I,J(new)} = v^*_{I,J} + d_{I,J}\left(p'_{I-I,J} - p'_{I,J}\right)$$

Corrector Step 1

4.) Repeating step 3 again with the new values obtained by the previous steps. Therefore the new values (p'', p^{**} $u^{**}$and v^{**}) are usually closer to the real value.

Corrector step 2

5.) Correcting the values of pressure and velocities by calculating the following equations:

$$p_{1,J}^{***} = p_{1,J}^{**} + p'_{1,J} + p''_{1,J}$$

$$u_{I,J}^{***} = u_{I,J}^{*} + d_{I,J}\left(p'_{I-I,J} - p'_{I,J}\right) + \frac{\sum a_{nb}(u_{nb}^{**} - u_{nb}^{*})}{a_{I,J}} + d_{i,J}\left(p''_{I-I,J} - p''_{I,J}\right)$$

$$v_{I,J}^{***} = v_{I,J}^{*} + d_{I,J}\left(p'_{I-I,J} - p'_{I,J}\right) + \frac{\sum a_{nb}(v_{nb}^{**} - v_{nb}^{*})}{a_{I,J}} + d_{i,J}\left(p''_{I-I,J} - p''_{I,J}\right)$$

The result obtained from this PISO algorithm represents fluid flow at only one time step. Therefore this PISO loop will repeat again for the next time step until it reaches the termination time (stopping criteria). The full explanations of the PISO algorithm can be referenced in the book *An Introduction to Computational Fluid Dynamics: The finite volume method* written by Versteeg and Malalasekera, (2007).

The PISO algorithm solves the pressure correction equation twice to be able to represent the flow field as close to as it occurs in a real environment. In order to achieve this however, the PISO algorithm requires additional storage and computational effort. However, Issa (1986) research found this PISO algorithm is more efficient and faster compared to the standard SIMPLE algorithm.

2.4.4. ScalarTransportFoam

In this study ScalarTransportFoam was used to simulate the dispersion of methane based on the flow field obtained by PisoFoam.

The general transport equation starts from substitution both convective and diffusive effects (Equation 2-3)

$$f(x,t) = v(x,t)u - \mathcal{D}(x,t)\rho\nabla c \qquad \text{Equation 2-3}$$

Into the partial differential equation of mass conversation (Equation 2-4)

$$\frac{\partial u(x,t)}{\partial t} + \nabla \cdot f(x,t) = s(x,t) \qquad \text{Equation 2-4}$$

Then generic transport equation is:

$$\frac{\partial \rho c}{\partial t} + \nabla \cdot (v \rho c) - \nabla \cdot (\mathcal{D} \rho \nabla c) = s$$

If the density ρ is constant and the velocity is redefined as **v:** = **v** + (D ∇ρ)/ρ, then the generic transport equation is a convection-diffusion reaction (CDR) equation for the mass variable u ($u = \rho c$) is:

$$\frac{\partial u}{\partial t} + \nabla \cdot (v u) - \nabla \cdot (\mathcal{D} \nabla \mathrm{u}) = s \qquad \text{Equation 2-5}$$

In the case of the velocity field **v** is incompressible, that is, ∇ · **v** = 0, then the vector identity is

$$\nabla \cdot (v u) = v \cdot \nabla u + (\nabla \cdot v) u \qquad \text{Equation 2-6}$$

Assembling the vector identity (Equation 2-6) with CDR equation (Equation 2-5), therefore the generic transport equation in the standard form is:

$$\frac{\partial u}{\partial t} + v \cdot \nabla u - \nabla \cdot (\mathcal{D} \nabla \mathrm{u}) = s$$

The terms that appear in this equation are explained as the following physical interpretation (Kuzmin, 2010) and (Jasak, 2007):

- The rate of change term $\frac{\partial u}{\partial t}$ or sometimes called temporal derivative. It demonstrates the net gain or loss of mass per unit volume and time. This term represents inertia of the system
- The convection term $v \cdot \nabla u$ represents the convective transport of u by the prescribed velocity field (the existence of the velocity field).
- The diffusive term $\nabla \cdot (\mathcal{D} \nabla \mathrm{u})$ accounts for the transport of u due to its gradients.
- The source or sink term s is the non-transport factor, which contains all effects that can both create or destroy u. In addition, any extra terms that cannot be incorporated into the convection or diffusion terms are considered as a source or sink term.

Once this equation is applied to all control volumes in given meshes and has solved the equation, the result will demonstrate the transport and the revolution of pollutants in the specific condition (i.e. air, water, or soil), as well as its temperature fluctuations.

CHAPTER 3 EXPERIMENTAL DESIGNS

3. Experimental Designs

3.1. Problem definitions

In order to improve the performance of the leakage detection system, Stinger Technology would like to enable the use of the point sensors (methane sniffers) to identify the leakage source, since they are more sensitive than traditional sonars; performing well even with low concentrations of methane dissolved in the water and able to detect even very small leakages and provide early warning to the operator.

Unfortunately, the measurement of methane sniffers is limited to the very local area in which they are mounted. Hence, this study will focus on concepts that aim to extend the ability of the methane sniffers (in the use of leakage source identifications) to be able to pinpoint such leakages. Based on the literature review ANNs and CFD have been selected for the experiments, as the ANN provides fast calculation times and performs well in complex systems and CFD offers an alternative way to simulate each incident without the need to set up real-environment experiments.

Therefore, this chapter is of key importance to examine the hypotheses that the selected techniques combined with point sensors have potential in identifying the leakage sources in oil platform structures in subsea conditions. It summarises in detail the approaches taken, experiments conducted and assumptions applied, as well as the design of experiments. However, the evaluation of results and a discussion will be provided in later chapters.

3.2. Experimental guideline

Experiments have been broadly separated into two different groups with two different modelling approaches:

1) Computational fluid dynamics (CFD) using OpenFOAM – This programme analyses all fluid behaviour in conditions of flow. OpenFOAM has been employed as a support tool to generate essential data sets required by the artificial neural network model. The advantage of using OpenFOAM is the relatively easy generation of data sets and the ability to visualise fluid flows. The use of CFD and all related parameters pertaining to this issue are demonstrated in the ***Experiment 1***.

2) Artificial Neural Network using Matlab - This is the area that this master thesis mainly focuses on. An ANN using Matlab is a system capable of recognising relationships and patterns of input and output data. However, the performance of ANNs depends on its architecture; a well-defined architecture enables ANNs to pro-actively adapt and learn trough the training process. ***Experiments 2 and 3*** show the methodology to define the value of each of the related parameters and to optimize the neural network, used in this instance to predict the source of subsea leakages more accurately.

The Experiment 4 shows the procedure to evaluate the performance of the selected model, in order to prove that ANNs have a potential to help point sensors to identify the leakage locations. Lastly, due to the fact that in a real-world environment there are many factors, which can interfere with a system, ***The experiment 5*** has been added to assess how well this model operates in unusual environmental conditions.

3.3. Experiment 1 Generating input/output patterns to train/test the ANN with CFD

3.3.1. Assumptions

ANNs ideally require large datasets; however these are expensive and time consuming to produce and companies typically face resource and financial constraints. This data would need to be collected through sensors and real-world simulations of leakage incidents. For the purpose and expedience of this study, I have employed computational fluid dynamics or CFD (as forward models) to generate experimental data based on different locations of methane leakages around the considered area that comparatively easily provides a large number of input/output patterns to train and test the ANN.

OpenFOAM, an open source program to model CFD, is employed to simulate the water current and also leakage sources with numerical conditions such as 0.1 m/s for velocity of water, 1200 square metre for the whole area, and other related parameters. The model also represents the real-world case of the leak detection systems, with the same number of methane sniffers relative to the same distance to the considered area and other relevant parameters. However, there are some factors, which have been adjusted to reduce the complexity of the model; the simulation was run in 2D instead of 3D, which preserves computational resources and delivers significant timesaving. In addition, turbulence in this model was increased for evaluations in order to demonstrate the ability of ANN in source identification applications.

3.3.2. Objectives

We are using the forward models (CFD) to simulate 323 different locations of methane leakage, which are deployed for training the neural network, and another 500 different leakage locations for testing the neural network performance. Table 1 provides the details of each set of data generated by CFD in this experiment.

Table 1 The format and details of the data generated by CFD

Total number of simulation	Number of simulation (based on the purpose of the usage)	Types of data	Meaning	Roles and descriptions
828 leakage cases	323 leakage cases for training (each case consists of two different types of data)	Inputs	The methane concentration measured by methane sniffers at 6 different locations after the source start leaking.	All both inputs and outputs are presented to the ANN again and agian, in order to train the network. That can increase the performance of prediction. It means that this 323 input/output patterns are mainly used for training process. Moreover, sometime this 323 inputs and outputs can be used to validate the ANN performance. (feed inputs to ANN and compare the calculated result from ANN with outputs generated by CFD)
		Outputs (targets), or Actual outputs	The leakage sources	
	500 leakage cases for training (each case consists of two different types of data)	Unseen inputs	The methane concentration measured by methane sniffers at 6 different locations after the source start leaking.	The unseen inputs have to be fed to the ANN, in order to generate calculated outputs (predicted leakage source). This unseen inputs are only used for testing process.
		Actual unseen outputs	The leakage sources	The actual outputs are only data which do not related to the ANN. The main role of these actual outputs are only to compare with calculated output obtained by ANN, in order to see an accuracy of the model.

3.3.3. Solution Strategy

The procedure of input/output pattern generation (for both testing and training process) has been split up into a number of steps.

1.) Modelling and Mesh generation

The outputs of CFD models are highly dependent on both the quality of information inputs and the design and structure of the mesh prior to any actual CFD modelling.

We have created a model that represents the area for which leak detection systems were installed. The dimensions and area of the computational domain are 50 metres in length and 24 metres in width (1200 square meter in total for area). This area contains eight square pillars, which are submerged at the bottom of the seabed and obstruct the normal flow of sea currents. They create turbulences or vortices behind the obstructions. However, turbulence or vortex depends on a structure's design and

external environment such as current velocities, density, and kinetic viscosity. The dimensions of the study area are shown in Figure 19.

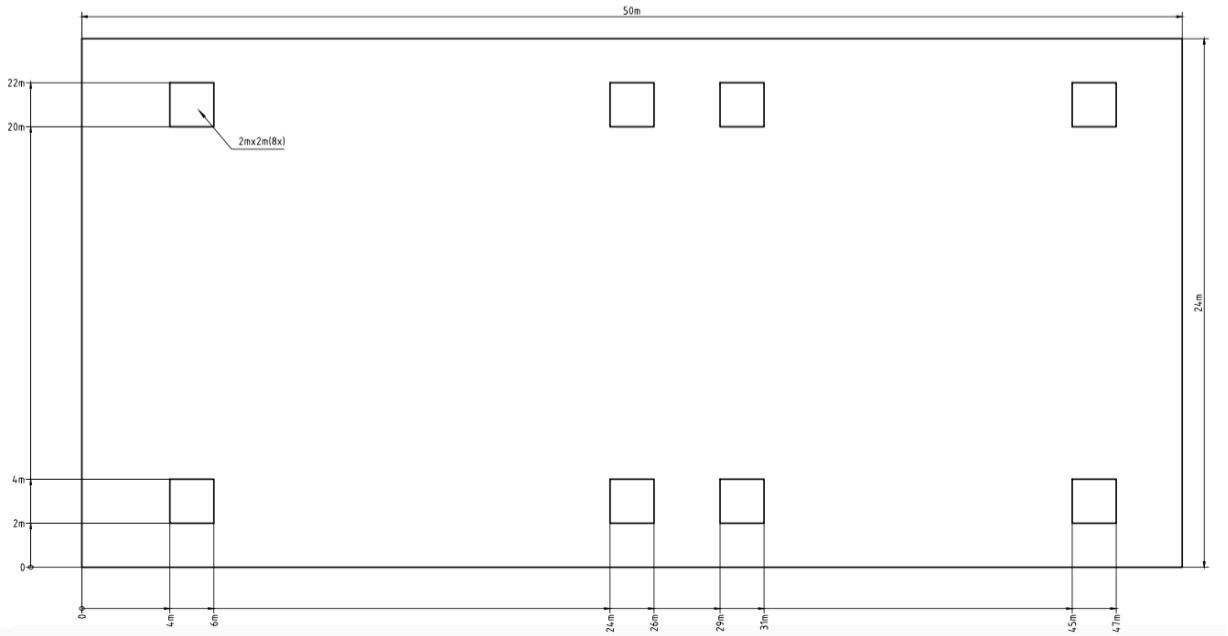

Figure 19 dimensions for study area

The next stage, after the model has been well defined, is the meshing process. This process basically discretises the total area into several small grids that can be used in finite element analysis. According to Versteeg and Malalasekera (2007), the definition of the mesh (including size, shape, connections, etc.) is important to the success of the experiment and the accuracy of the results. Fortunately, there are now many meshing tools that cover a range from basic, moderately complex to a very high complexity of meshing tasks such as the blockMesh function from OpenFOAM, NetGen, and Lagrit software respectively. In this case a simple blockMesh function from OpenFOAM has been deployed, because it is easy to use and simpler to determine the coordinates of each cell, which makes it more compatible with Matlab in the later evaluation.

The computational domain of this area (50m x 24m length/width) is meshed into 29200 square boxes (cells) connected by 59466 nodes. The size of each cell in the computational domain is constant at 0.20 metres for both length and width and the average number of cells in X and Y directions are 260 and 120 respectively. In addition, the discretized grids are also indexed based on coordinates (x, y) starting from (0, 0) up to (50, 24) along x and y directions and the distance between each coordinate is 0.2 increments. The total mesh of this area and mesh coordinates are shown in Figure 20 and Figure 21 respectively.

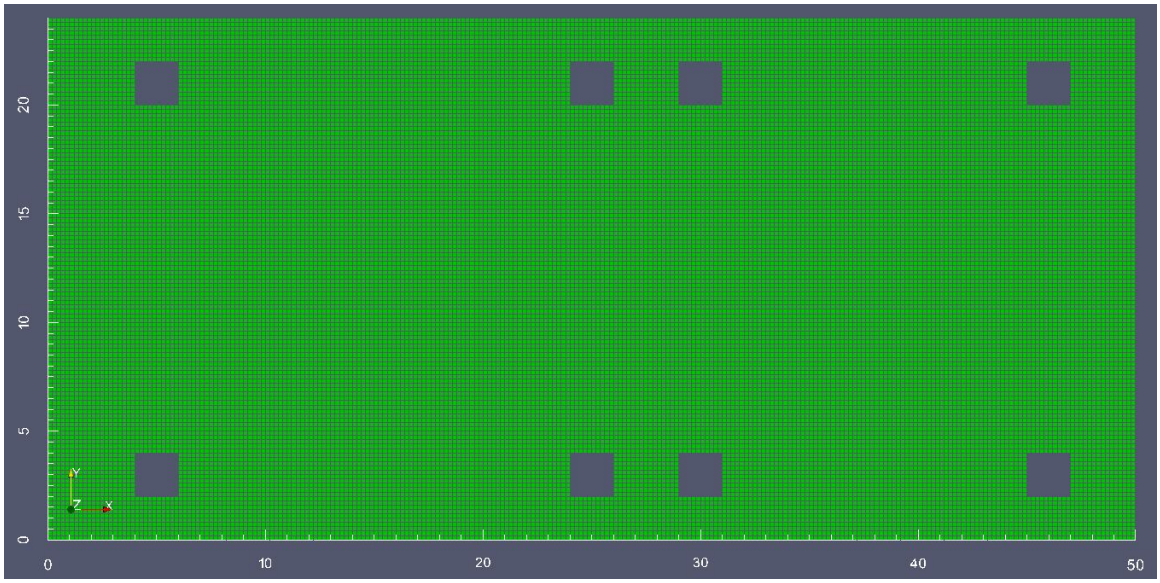

Figure 20 Mesh of the computational domain

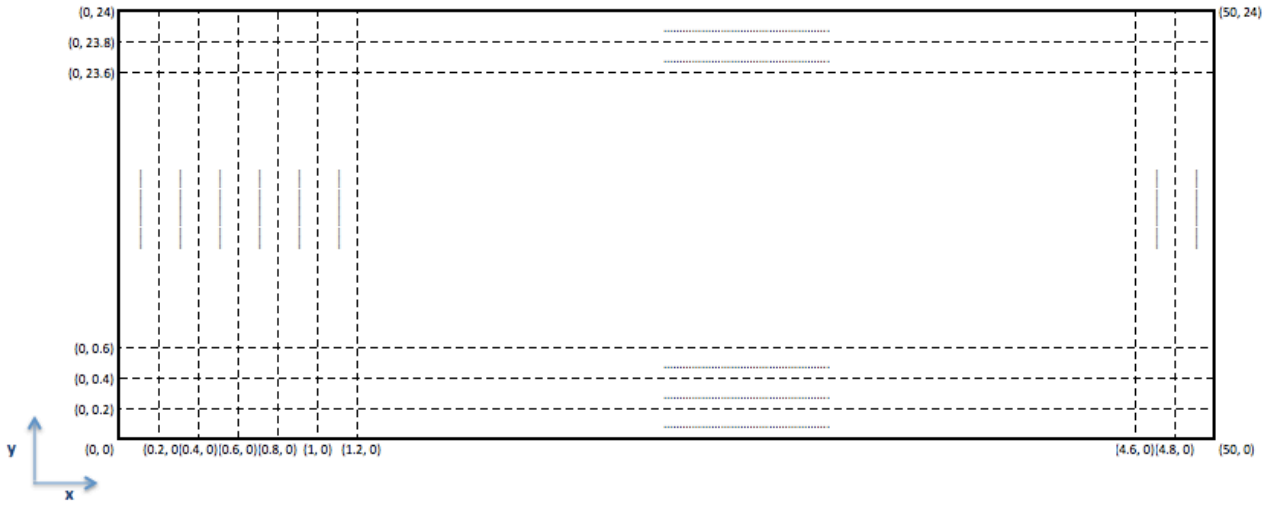

Figure 21 Mesh coordinate

2.) Set up the parameters

In this step, all related parameters are defined and fed into the program in order to create the flow field and turbulence in the considered area. In addition, (Dreyfus, 2010) states that the ANN could be used to solve both linear and nonlinear problems, however, the benefit for linear systems are very limited and insignificant. The implication is that artificial neural networks should be used to solve problems, which are not lending themselves to simple explanations based on equations. Therefore the flow field and turbulence in this evaluation was generated in order to create high complexity of the area rather than using normal conditions. These create greater challenges for the ANN and also allow us to assess the potential of using ANNs in severe conditions. The velocity of current and turbulence in the considered area is shown in Figure 22.

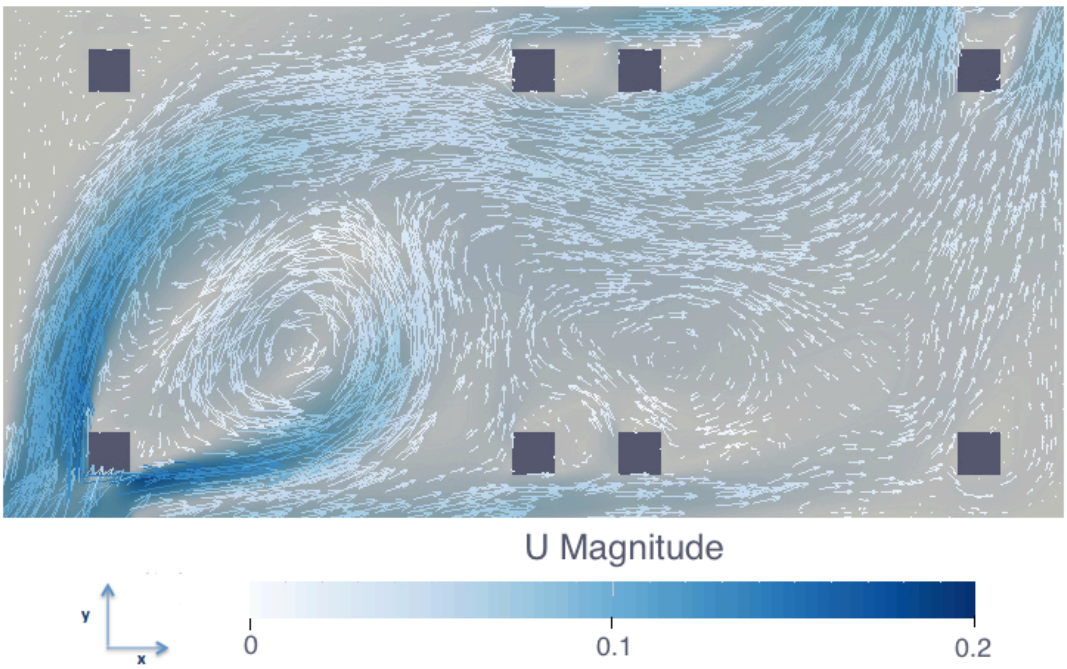

Figure 22 current path and turbulence, which calculated by CFD method around considered area

3.) Solving the case

Once the flow field and turbulence are defined, the parameters and conditions to simulate the leakage of methane are exported to OpenFOAM, which will automatically run to solve the case.

The parameters and conditions to solve the case are listed in Table 2. In addition, even the possible leakage locations used were scaled down to 288 square metres

instead of the whole area (1200 square metres) in order to save simulation time and increase efficiency as shown in Figure 23. This is sufficient to represent the probable leakage sources in order to answer the research's main questions as well as prove or disprove the hypothesis.

Table 2 Parameters and conditions for experiment

Number of leakage locations	
- training	323
- testing	500
Injection rate	2 nanomole/second
Sensor sensitivity	1 nanomole
Number of sensors	6
location of sensors (coordinate x,y)	
- sensor No.1	(8, 6)
- sensor No.2	(8, 18)
- sensor No.3	(25, 6)
- sensor No.4	(25, 18)
- sensor No.5	(41, 6)
- sensor No.6	(41, 8)

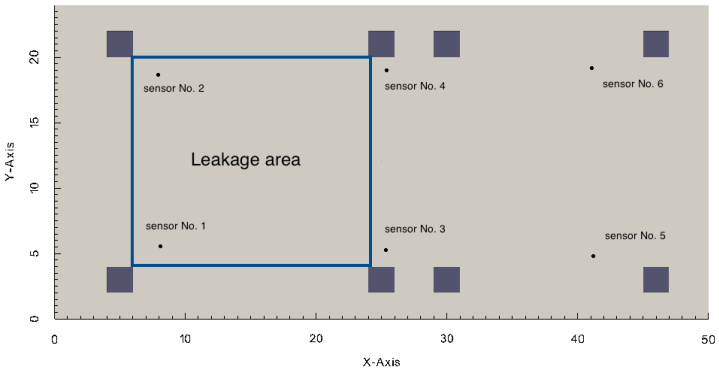

Figure 23 Leakage area for experiment

3.4. Experiment 2 Artificial neural network training function selection.

3.4.1. Assumptions

Within the body of knowledge on artificial neural network, there are many different types of ANN architectures and several methodologies implemented in various applications such as data analysis, pattern recognition, and control functions. Here we have used backpropagation or backprop to solve the specific issues evaluated, as it is the most popular and widely implemented of all neural network paradigms.

The main advantage of artificial neural networks is the ability to solve a non-linear problem. However the limitation of backpropagation is that it only converges to local minima (the smallest error in one specific area), which may or may not be global minima (the smallest error in the total system)(Chester, 1993) (The basic explanation of convergence, local minima, and global minima are shown in Appendix B). Therefore, choosing a suitable architecture and also parameters of the artificial neural network for each application (to ensure it converges to global minima) can be a critical issue. In addition, there are no prescribed solutions or even fully theoretical guidance for backpropagation in order to provide instructions on setting up the ANN model (Zheng and Chen, 2011).

Therefore, the main focus is on customising different parameters and architectures to find out which settings are the most suitable for a certain problem. Moustafa et al., (2011) pointed that the following parameters should be defined carefully to achieve a good ANN model: The input and target sets, the ANN architecture, the number of iterations used to train the network, training rate, number of hidden nodes and the training function. The latter parameter is examined in this experiment.

In simple terms, artificial neural network training functions (training function or training algorithm) are used to update the ANN parameters: weights and bias values, reducing the overall error rate of the total system as quickly as possible. Weights and biases are updated until the ANN converges reaching a steady state of the system. Generally there are two different ways to update weights and biases. Firstly, incremental mode, in which the weights and biases are updated suddenly after each input is applied. Secondly, batch mode which all of the inputs have to be introduced into the network before the weights and biases are updated (Demuth et al., 2008)

Currently, there are many algorithms, which have been created and developed to establish a new training function, that can work well with wider applications and offer fast convergence. A list of the training algorithms available in the Neural Network Toolbox software and comments are shown in Table 3. In addition, the performance of each training algorithm also relies on the complexity of the given problem such as the number of inputs in the training set, number of weights and biases in the network, and whether related parameters are used in the model. The theory behind this approach is that one algorithm might work best in a few specific problems but not in others.

3.4.2. Objectives

The key objective for this experiment is to find the training function is the best suited for deployment in subsea leakage source identification applications.

3.4.3. Solution Strategy

This exercise compares all the various training functions provided by Matlab's Neural Network Toolbox except 'trainbr'. The main reason to exempt it in this benchmarking is that Demuth et al., (2008) stated that ANN's with trainbr are highly recommended for use with 1-20-1 networks (1 input layer- 1 hidden layer (with 20 hidden nodes) - 1 output layer) no matter how large the total number of parameters in the network are, to ensure convergence of the ANN. Moreover, the method to stop a training process of trainbr (used 'regularization' method) is completely different from the other twelve algorithms (which use 'early stopping' method). However, trainbr will be brought

back again in the next experiment to verify a network performance along with one algorithm selected from this experiment.

The comparisons of the various training algorithms (12 cases) can be evaluated by following these steps:

1.) Feedforward backpropagation 1-5-1 network, initial weight and biases value are determined as a constant for all trials for all different twelve training functions. This ensures that those factors will not interfere with the performance of training algorithm,
2.) Supplying 323 input/output patterns generated by CFD to all artificial neural network for training. The set of input/output patterns is constituted by a set of input values together with the corresponding target. The input values are simulating the methane concentration measured by methane sniffers, and the target values represent the methane leakage sources. Therefore, 323 input/output patterns mean that there are 323 consequences of 323 different leakage cases. In addition, before the ANN training process was started, those 323 patterns data were be randomly separated into three sets: training set, validation set, and testing set[2]. By using 70%-15%-15% proportion, out of total, 227 patterns were used for training, 48 patterns were used for validation, and another 48 patterns were used for testing.
3.) In each trial, the network is trained until the network begins to overfit the data (see Appendix C.), when it stops training. Once the training process has finished, 323 inputs (which were used earlier for training) will be fed into the ANN again to generate calculated outputs.
4.) Compare the result between actual outputs (from CFD) and calculated outputs (from ANN)
5.) Select only the most promising methodology amongst the various algorithms.

To verify the performance of each algorithm, it is necessary to define the indicator, which provides a quantitative score describing either the degree of similarity or the level of error between two kind of data; pristine original data and contaminated data by error (Wang and Bovik, 2009). Therefore, the well known statistical tools Mean Square Errors (MSE) is used for this benchmarking.

Mean Square Errors are defined as the sum of the squared difference between the output of the network and the target values over all samples (Anguita et al., 1994).The smaller the value of MSE, the better the performance of training function is, that the total system can obtain.

[2] Training Set is used to adjust the weights and biases of the ANN, Validation Set is only used to minimize overfitting, and Testing Set is used only for testing the final solution ensuring the predictive power of the ANN network

The equation is shown below.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(X_i - Y_i)^2$$

where, N = total number of samples; X_i = actual value; and Y_i = calculated value from model

Table 3 Training algorithms specification (Demuth et al., 2008)(TechSource Systems, 2005)(Torrecilla et al., 2008)

Group	Training Algorithms	MATLAB tools	Acronym	Description
Gradient Descent with Variable Learning Rate	gradient descendent BP	traingd	GD	Basic and original gradient descent but slow response. It used in incremental-mode training.
	gradient descendent with momentum BP	traingdm	GDM	Typically Faster training than traingd and only be used in incremental-mode training.
	gradient descendent with adaptive linear rate BP	traingda	GDA	Adaptive learing rate. Faster than traingd. But only be used in batch-mode training.
	gradient descendent with momentum and adaptive linear BP	traingdx	GDX	
Resilient BP	resilient BP	trainrp	RP	Provide fast convergence and minimal storage requirement. It is a simple batch mode training algorithm.
Conjugated Gradient Descent	Fletcher-Powell conjugate gradient BP	traincgf	CGF	The only conjugate gradient algorithms that has the smallest storage requirement. It also provides fast convergence.
	Polak-Ribiere conjugate gradient BP	traincgp	CGP	Provide faster convergence on some problem but require slightly larger storage than traincgf.
	Powell-Beale conjugate gradient BP	traincgb	CGB	Typically Provide faster convergence on all problems but require slightly larger storage than traincgp.
	scale conjugate gradient BP	trainscg	SCG	It is very good general-purpose training algorithm with fast convegence, it is the only one of the conjugate gradient algorithm does not requires line search.
Quasi-Newton Algorithm	BFGS quasi-Newton BP	trainbfg	BFG	Quasi-Newton method introduce approximate Hessian matrix to solve the problem. It demand a huge storage and more computation in each iteration.
	one-step secant BP	trainoss	OSS	Compromise between quasi-Newton methods and conjugate gradient methods. That results in fast convergence and reduce storage requirement.
Levenberg-Marquardt	Levenberg-Marquard BP	trainlm	LM	It is the fastest training algorithms for moderate size and has memory reduction feature when the training set is large.
Automated Regularization	Bayesian regularization	trainbr	-	Modification of the Levenberg-Marquardt in order to improve generalization capability. That reduces the difficuly of determining the optimum network architecture.

3.5. Experiment 3 Architecture (Hidden layer) optimization and Network performance verification

3.5.1. Assumptions

The information flows from input layer to output layer though the artificial neural network is typically determined by the ANN architecture; meaning that it directly affects the accuracy of output produced by the artificial neural network. Therefore, specifying the appropriate artificial neural network architecture is one of the key factors of applying ANN in tasks. Moreover, lack of suggested solutions for the optimisation process makes ANN design more complex.

To overcome this challenge a good starting point is to train the ANN from several different initial conditions and then select one ANN model, which generates the most useful results for a specific application. This can verify the performance of each ANN among various different ANN network architectures.

For an ANN model, there are many different types of ANN architectures from a very simple network to high-complexity network for use in challenging scenarios as shown in Figure 24 (MathWorks, 2015). For this thesis, a simple backpropagation architecture had been selected to implement.

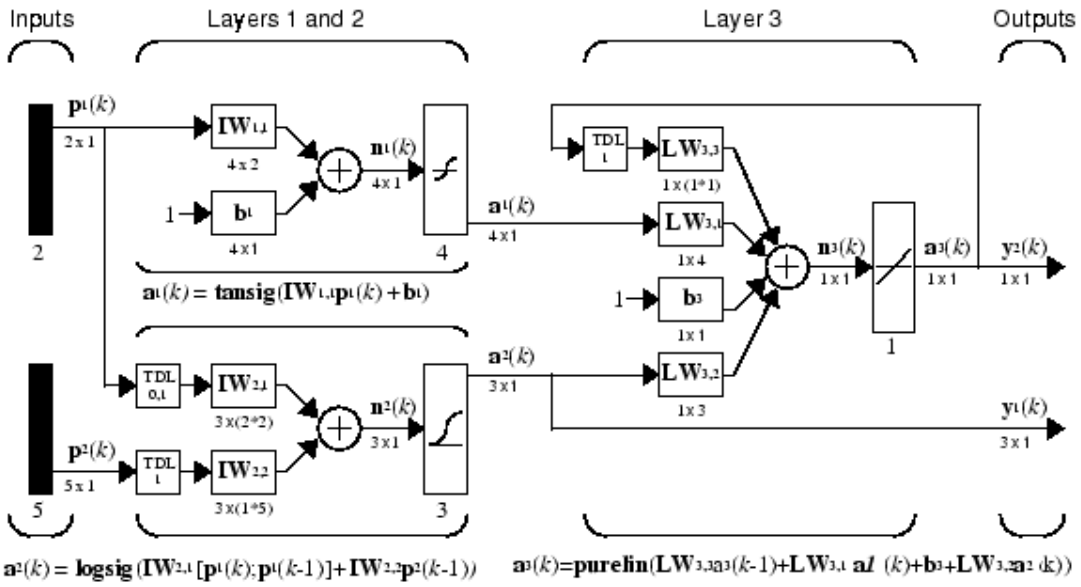

Figure 24 example of custom network for dramatic purpose (MathWorks, 2015)

However, backpropagation itself consists of many properties that could be set in order to specify the behaviour of the network. Based on Figure 25, the process of backpropagation artificial neuron network is defined as follows (MathWorks, 2015)

1.) The process of neurons (hidden nodes) in each hidden layer start when hidden layer receives a number of inputs (either from original data, or other neurons from other layers in the network).
2.) The neurons compose the activation signal of the neuron by forming the weighted sum of the inputs then subtracted with threshold.
3.) Lastly network passes the activation signal through a transfer function to produce the output.

It can be observed that the property and behaviour of the ANN could be affected by many essential properties such as the number of inputs and outputs, number of hidden nodes (typically equal to number of weight and threshold), number of hidden layers, the initial value of each weight and threshold, and transfer function.

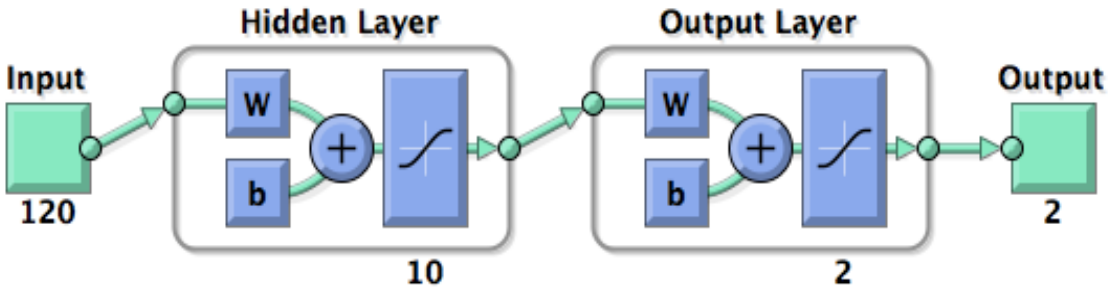

Figure 25 Back propagation algorithm with 120 inputs, 1 hidden layer with 10 neurons, and 2 outputs, where w = weight; b = threshold or bias; and $\int$ = transfer function.

However, as we have already decided to use a backpropagation model to be the main ANN architecture and Levenberg-Marquard BP algorithm to be a training tool in the earlier experiment, the main experiments here are about trialling different options to identify the specific architecture and customisation option that work best for an ANN model with regard to a specific question. Especially the numbers of hidden nodes, which is the main component of hidden layers to define the performance of a total system; too few or too many hidden nodes do not necessarily produce better of worst result of an ANN model.

Therefore, the following sections will explain the process of network architecture determination used in this study.

3.5.2. Objectives

To create several different architectures of ANNs and then train each ANN model with several different initial conditions. Lastly, select one ANN model, which generates the most useful results for a specific application. This can be employed to verify the performance of each ANN amongst various different ANN network architectures.

3.5.3. Solution Strategy

Basically, the value of several parameters used in this experiment has been derived from two different sources: suggestions from related articles, books, and forums, and results from the evaluation process.

1.) *Number of Inputs and Outputs* – 120 inputs and 2 outputs were defined based on a specific scenario. The 120 inputs represent the methane concentration measured by methane sniffers at six different locations every hundred seconds for twenty time steps from gas releases (6 sensors * 20 times = 120 inputs). The 2 outputs represent the location in coordinate systems (x, y), for example the model prediction (3, 5.6) mean that the leakage source located in 3 and 5.6 meter away from reference point (0,0) in X and Y directions respectively.

2.) *Number of hidden layers* – Singh et al., (2004) stated that in tasks with a high complexity and involving nonlinearity, two hidden layers performed better than ANNs with a single hidden layer. However Moustafa et al., (2011)'s research shows that there was no difference in improvement in the accuracy of the solution between using single or two hidden layers for simple tasks, but they require more training time. As a result, one hidden layer was selected for the purpose of this study, to minimise complexity.

3.) *Initial value of weights and biases* – weights and biases initialization is the one of the most effective approaches to speed up the training process of neural networks. There are many new algorithms that have been created such as statically controlled activation weight initialization (SCAWI), optimal initial value setting (OIVS) (Yam and Chow, 2000). In addition, in cases of simple applications, functions such as RAND in Matlab can be implemented as well. However, in this study each layer's weights and biases are initialized by Nguyen-Widrow's randomization method. This method was selected because it is widely used and well known as well as the one of the most effective weight initialization methods available since the 1990s.

4.) *Transfer function* – As mentioned the functionality of transfer functions is to create outputs from each layer of neuron. Generally there are many

differentiable transfer functions to be used in backpropagation algorithm for example log-sigmoid transfer function which generates output between 0 to 1 (before de-normalise the value), or hyperbolic tangent sigmoid transfer function which generates output between -1 to 1 (Demuth et al., 2013b). The latter transfer function was selected to use in this experiment as it is often used for pattern recognition problems, and (Singh and Datta, 2006) also mentions that this transfer function makes backpropagation perform better.

Therefore, in most situations, the experiment has to be created and evaluated to determine the best number of hidden nodes by training the artificial neural networks with a variety of different settings, until the ANN obtains acceptable results. This experiment separates the number of hidden nodes for training and testing processes into five groups: 30, 60, 90, and 120 hidden nodes with Levenberg-Marquardt training algorithm, and 20 hidden nodes with Bayesian regularization training algorithm.

The main reason to use this numbers: 30, 60, 90 and 120 for Levenberg-Marquardt training algorithm is based on suggestions from Blum (1992) stating that the size of hidden layer (number of hidden nodes) should be somewhere between the input layer size and output layer size. The concept behind this is that fewer numbers of hidden units will generate more training errors, as the network does not have enough capacity to manage all data. Contrastingly, for too high a number of nodes, "your network becomes a memory bank that can recall the training set to perfection, but does not perform well on samples that were not part of the training set" (Petersen, 2013). This effect is called overfitting.

In addition, there are many articles quoting the superior performance of artificial neural network model by using Bayesian regularization algorithm (trainbr), such as Singh et al. (2010)'s research comparing trainbfg with trainbr for predicting the value of heat capacity in refrigeration systems. Its conclusion showed that the trainbr training function offers more appropriate results compared to other functions used with the same ANN. Therefore, in this study Bayesian regularization backpropagation with 20 nodes in hidden layers (strongly suggest by Demuth et al., (2008)) will be additionally examined in order to compare with four different settings of artificial neural network architectures evaluated earlier

The results of this experiment provide us with the completed ANN model architecture, which is ready to be implemented for use in real case simulation (*the experiment 4*). The steps to run this evaluation are:

1.) Define all controlled parameters as well as independent parameters (five different numbers of node) as mentioned earlier. The weights and biases of artificial neural network with four different hidden nodes (30, 60, 90 and 120) are updated with Levenberg-Marquardt algorithm (trainlm) that produced a

very good result in the previous experiment. The ANN with 20 hidden nodes is trained with Bayesian regularization algorithm (trainbr).

2.) Supplying 323 input/output patterns generated by CFD to neural network for training process. Similarly as with the second experiment, those 323 patterns of data are randomly separated into three sets 70%-15%-15% for training, validation, and testing respectively.
3.) In each trial, the network is trained until the network begins to overfit the data and then stop training. Once the training process had been finished, 323 inputs (which be used earlier for training) are to be fed into the ANN again to generate calculated outputs.
4.) Compare the result between actual outputs (from CFD) and simulated outputs (from ANN)
5.) Carefully select a good architecture. The criteria used for selecting ANN architecture is based on Mean Square Errors (MSE), which demonstrates the differences between actual and calculated sources locations (in both x and y axis). The lower the value the better the performance. In this experiment, only models which MSE values of less than 0.05 will be verified.

3.6. Experiment 4 Source identification by using the selected model

3.6.1. Assumptions

In the real-world subsea condition, the methane concentration levels are measured by six methane sniffers in different locations after the leak sources start emitting methane. This data will be delivered to the well-optimised ANN network, in order to calculate the output. This output is expected to identify the location of leakage source (in coordinate x - y system).

3.6.2. Objectives

To simulate such a real case scenario, the source identification model (selected network from previous experiment) was run with 500 sets of unseen data generated by CFD (500 different leakage sources). This can verify the performance of selected ANNs to deal with unseen data.

3.6.3. Solution Strategy

This set of inputs represents the methane concentration at six different locations from each of 500 different source locations around the study area. Moreover, these 500

inputs have not been used previously for training purposes, and can therefore be classified as unseen input data.

The procedures to run this evaluation are:

1.) Preparing unseen input data and actual unseen output data (the output data would be used to validate the model performance).
2.) Presenting 500 unseen input data points generated by CFD for the artificial neural network, which are subsequently converted by the ANN into calculated output.
3.) Compare the result between actual unseen outputs (from CFD) and simulated outputs (from ANN). Apart from the previously mentioned Mean Square Errors (MSE), we have added the Normalized Error (NE) here, to measure the misfit of both actual outputs and calculated output in percentage. This makes the result easier and more straightforward to read and compare.

This normalized error criterion is defined as (Mahar and Datta, 2000):

$$NE = \frac{\sum_{i=1}^{N} |Y_i - X_i|}{\sum_{i=1}^{N} X_i} \times 100$$

where, N = total number of samples; X_i = actual value; and Y_i = calculated value

3.7. Experiment 5 Robustness test

3.7.1. Assumptions

Due to the fact that only steady state conditions of water current flows were considered for this thesis, the results could be seen a little bit too optimistic. In a real-world environment there are many factors either controllable or uncontrollable, which can interfere with a system. For example, high water contamination and malfunction of sensors that might result in faulty readings or data loss respectively. This may impact the estimated performance of the leakage source location identification, depending on the models assumptions and complexity (robustness).

3.7.2. Objectives

The robustness test is introduced in this part to assess how this model operates in unusual environmental conditions and in the presence of exceptional inputs.

3.7.3. Solution Strategy

To demonstrate the robustness of this mode, white noise is added to the set of inputs measured by six sensors. Each new input was expressed as the following equations.

$$Input_{new} = Input_{original} + Noise$$

where, the value of the noise was calculated by following equation (Nunnari et al., 2001):

$$Noise = Xran \cdot \frac{\sum_{i=1}^{N} |Value_i|}{N} \cdot \frac{Perc}{100}$$

Where:

Xran = Uniformly distributed random variable created in the interval [-1:1]
Value = Value of the considered input set (methane concentration)
N = Number of elements making up the input set for every leakage point
Perc = Percentage of noise

For this experiment, percentages of noise were set at 1%, 5%, 10%, and 20%. The new inputs with noise added can be seen on Figure 26. The higher the percentage of noise added, the greater the amount of fractured data that is provided to the network.

The procedures to run this robustness test are:

1.) Providing the trained ANN with noise-free input/output patterns (The selected artificial neural network from previous evaluation would be used in this experiment).
2.) Preparing noise added unseen input sets and their actual outputs (for validating the model performance).
3.) Testing the network with noise added inputs starting from noise-free, noise at 1%, noise at 5%, noise at 10%, and noise at 20%.
4.) Collecting all calculated outputs from the artificial neural network from every test groups.
5.) Evaluate the robustness of the network by comparing the result between actual unseen outputs (from CFD) and simulated outputs (from ANN). The criterion used here is normalized error (NE).

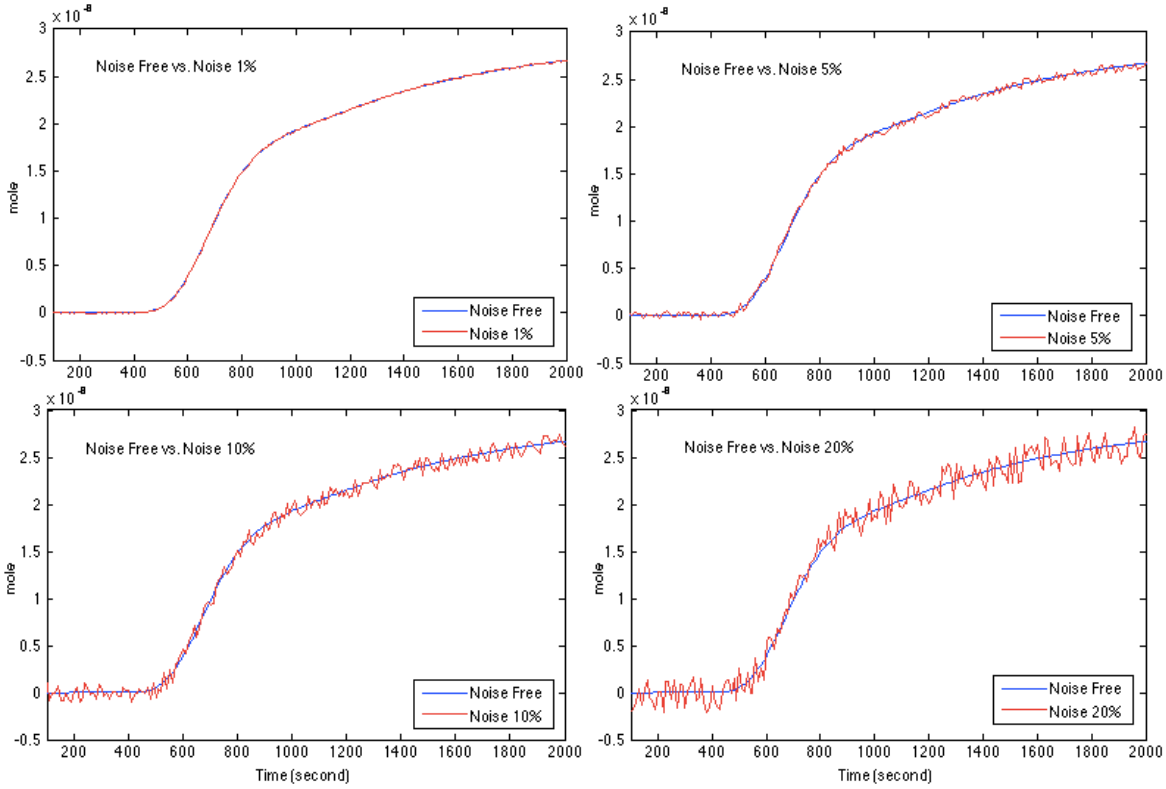

Figure 26 New input with different percentage of noise from t = 100 to 2000 second. (These value read by sensor number six of which methane source was set at position (6, 16.4) from origin point.)

CHAPTER 4 Experimental results and Analysis

4. Experimental results and Analysis

4.1. Experiment 1 Generating input/output patterns to train/test the ANN with CFD

By simulating the leakage source in each different location using CFD, we can visualise and provide an overview of how methane disperses into the water phase, the direction, the methane concentration of each point, and traveling time. The latter two sets of simulated values (methane concentration and traveling time) are measured and collected at six different points around the computational domain area, which represent behaviours of the six different locations of methane sniffers around the seabed in real-world conditions. This kind of data will be manipulated and used as input/output patterns and also unseen inputs/outputs for neural network training and testing processes respectively. The concentration fields at different times (t.) from the different sources are shown in Figure 27 and Figure 28.

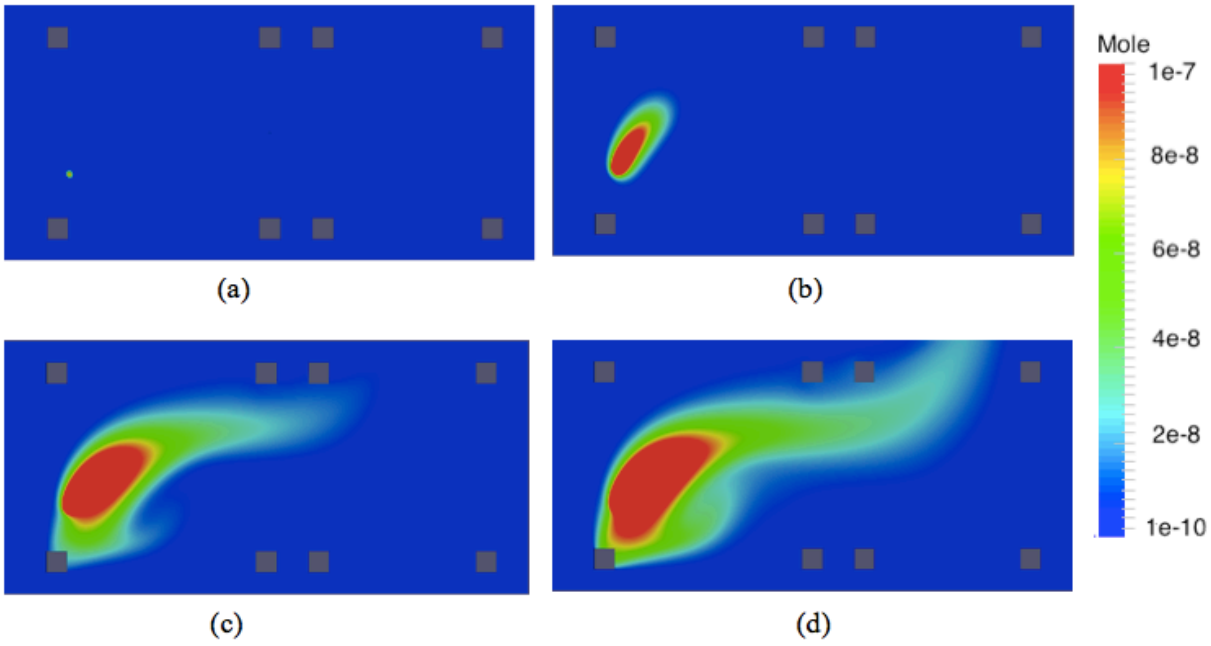

Figure 27 Concentration fields (case of steady state flow) for the source is at location (6, 8), (a) at t = 1 sec. obtained by CFD; (b) at t = 120 sec. obtained by CFD; (c) at t = 600 sec. obtained by CFD; (d) at t = 1200 sec. obtained by CFD.

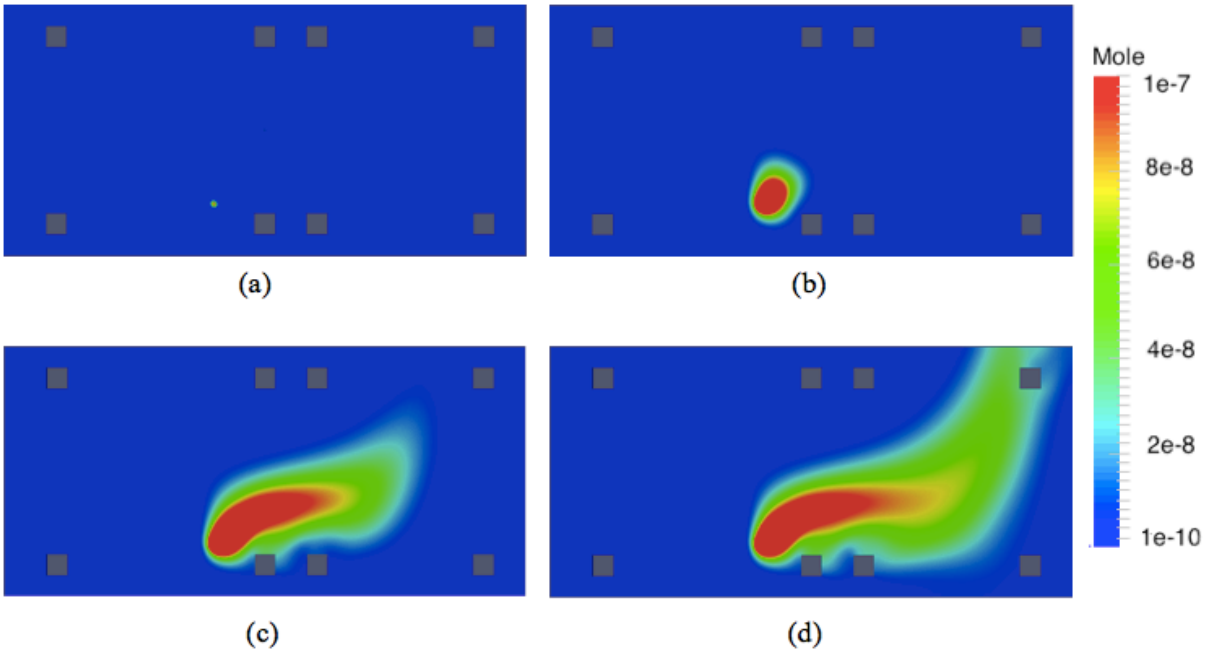

Figure 28 Concentration fields (case of steady state flow) for the source is at location (20, 50), (a) at t = 1 sec. obtained by CFD; (b) at t = 120 sec. obtained by CFD; (c) at t = 600 sec. obtained by CFD; (d) at t = 1200 sec. obtained by CFD.

To decide which kind of data has to be collected and in which form is one of the most important parts that have to be considered carefully. For example, less input/output patterns for the training process would decrease the performance of the artificial neural network in terms of predicting the possible outcome. On the other hand, too many patterns would demand longer processing times and higher memory capacity of computers for training processes, as well as possibly resulting in a dramatic drop in performance if the sets of data/data structures are too complex.

For the specific problem of leakage source identification features, there are three main factors which should be considered; sea current velocity, sea current direction, and methane concentrations. However, as this evaluation only uses a steady state flow field to frame the problem, sea current velocity and directions at the different time (t.) can be eliminated here. Therefore, only the levels of methane concentration at 20 different time steps measured by all six different locations were selected for use in this ANN model.

Basically, there are two different types of data sets required by an artificial neural network model.

1.) *Input/output patterns* – artificial neural network are typically considered a black box process, meaning that the system considers only input and output without accounting for any concept or knowledge of its internal workings.

Therefor only input and output patterns are required for training/learning processes. During the training/learning process the black box (model) adjusts its mechanism (weight and bias values) by itself, to represent the relationship between stimuli inputs and output reactions, which were fed as an input/output pattern. That increases the ANN's ability to predict the possible output from unseen data based on its perception and understanding of the input/output pattern used to train the ANN.

2.) *Unseen (input/output) data* – This type of data set was generated in order to represent the scenarios that the neural network up until that time had not experienced. This type of data is normally used to measure the performance of the ANN model as to how precise it is in terms of prediction, by comparing the outcomes between the calculated output from the ANN model and the unseen output data from the simulation (CFD).

In simple terms, input/output patterns are used to make models smarter and unseen data is used as a measurement tools to identify the performance of a model; both data sets are generated by CFD. The flow chart of the data collection process (including the use of data) is shown in Figure 29.

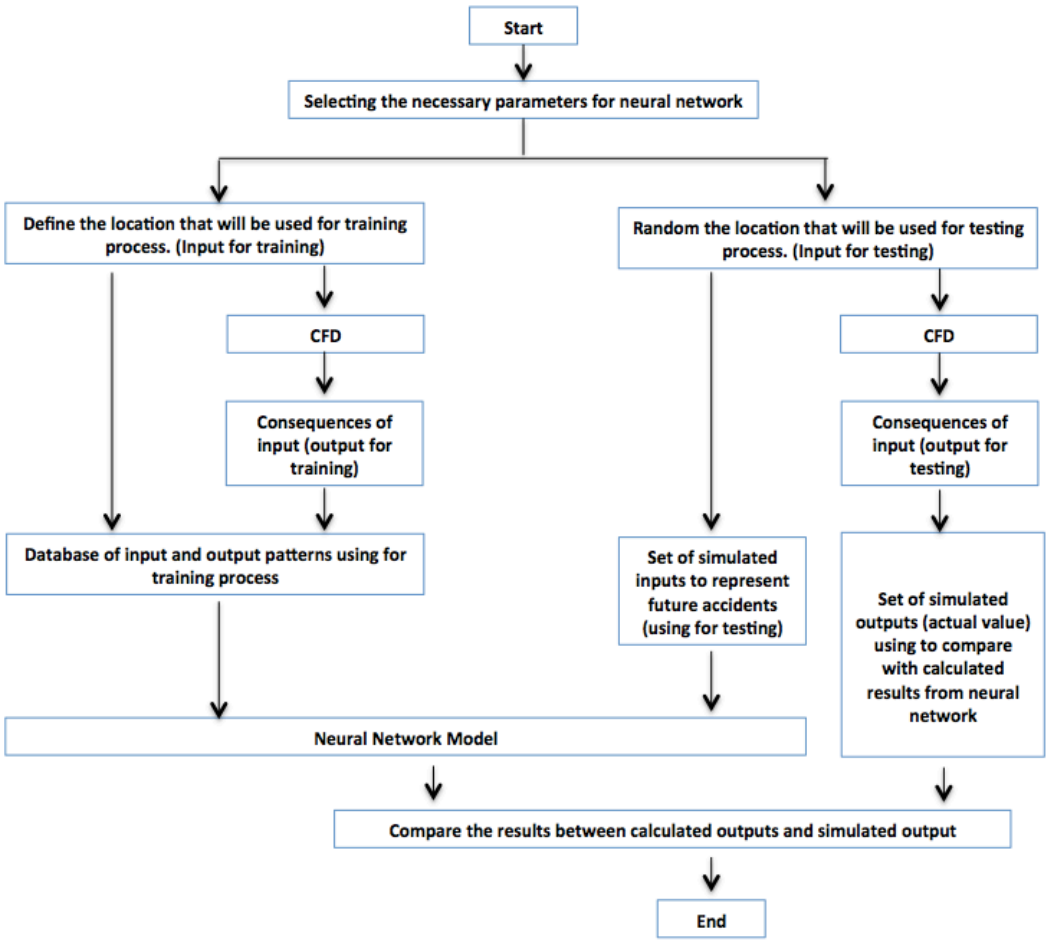

Figure 29 The flow chart of data collection process

4.2. Experiment 2 Artificial neural network training function selection.

The following table summarizes the results of the network using twelve different training algorithms; running10 trials per algorithms.

Table 4 the results of the network using twelve different training algorithms. Each algorithm is tested ten times.

Trial		Algorithms											
		GD	GDM	GDA	GDX	RP	CGF	CGP	CGB	SCG	BFG	OSS	LM
1	Mean Squar Error	59.90	59.36	60.59	62.03	3.38	9.17	28.70	2.28	5.71	14.65	26.63	44.63
2		60.95	59.86	68.87	59.56	20.66	19.83	3.44	3.47	13.41	0.37	42.70	0.25
3		55.25	57.50	61.66	58.61	9.33	53.99	2.68	3.90	16.30	27.30	16.17	45.18
4		58.62	59.17	61.60	56.46	5.37	12.67	56.31	1.81	6.17	0.32	26.64	45.52
5		60.52	57.94	62.25	60.98	4.74	18.28	1.82	1.49	16.24	1.10	14.95	7.73
6		58.28	59.93	65.54	58.90	6.46	56.40	3.88	2.65	1.75	1.87	15.33	51.33
7		58.01	59.90	61.40	59.35	7.38	19.96	3.55	3.93	14.88	0.39	3.84	43.42
8		60.78	57.40	61.20	57.97	17.30	21.54	14.87	1.25	16.80	0.85	15.56	42.97
9		61.23	51.38	64.53	57.52	5.49	21.32	5.57	1.37	4.11	32.89	26.44	1.19
10		59.14	61.57	60.73	59.13	16.17	11.93	6.46	3.79	17.43	0.58	13.47	0.19
Max		**61.23**	**61.57**	**68.87**	**62.03**	**20.66**	**56.40**	**56.31**	**3.93**	**17.43**	**32.89**	**42.70**	**51.33**
Min		**55.25**	**51.38**	**60.59**	**56.46**	**3.38**	**9.17**	**1.82**	**1.25**	**1.75**	**0.32**	**3.84**	**0.19**
Std.		**1.83**	**2.78**	**2.67**	**1.61**	**6.12**	**16.73**	**17.38**	**1.10**	**6.10**	**12.48**	**10.70**	**22.50**

According to Table 4, there are several algorithm characteristics, which can be deduced from the experiments. Powell-Beale Restarts algorithm or traincgb (CGB) is able to provide relatively low mean square errors with the highest precision (std. is about 1.1) compared to any of the other algorithms tested. However, the advantage of high precision training (to provide almost the same results no matter how different the initial value is) is not especially necessary for this specific problem, but the possibility of the lowest mean square errors the ANN model can obtain (the model, which is able to find the optimum point of the system – highest accuracy) is of greater interest. The concept behind this is that once the best preforming network is found (1 out of 120 models in this experiment for instance), all parameter values in this selected artificial neural network would be fixed as a constant and used to simulate an output until the model owner is willing to start a new training process.

Therefore, for this benchmarking, the Levenberg-Marquardt algorithm (trainlm) is the most interesting training function when compared to the others. Firstly, it provides the lowest Mean Square Errors for this experiment. That means trainlm has a higher potential to provide the best solution (less errors) compared to other algorithms. Secondly, the lowest Mean Square Errors of trainlm is over ten times smaller than Resillian backpropogation (trainrp) which generally were recommended by (Demuth et al., 2008), for using in pattern recognition applications.

4.3. Experiment 3 Architecture (Hidden layer) optimization and Network performance verification

The Table 5, Table 6, Table 7 and Table 8 summarize the training results of Levenberg-Marquardt backpropagation network with a variety of different architectures and initial weight and bias values.

Table 5 Performance evaluation of trainlm backpropagation network with 30 hidden nodes

Number of nodes	Model No.	Trial	Mean Square Errors (MSE)		
			X	Y	TOTAL
30	1	1	0.17320	0.13579	0.15450
	2	2	0.95371	0.80997	0.88184
	3	3	0.43031	0.93028	0.68029
	4	4	0.05500	0.04368	0.04934
	5	5	0.07730	0.66400	0.37065
	6	6	111.000	0.19261	55.59630
	7	7	0.09267	0.63897	0.36582
	8	8	0.10663	0.06808	0.08735
	9	9	0.04545	0.03140	0.03842
	10	10	0.10990	0.55135	0.33062

Table 6 Performance evaluation of trainlm backpropagation network with 60 hidden nodes

Number of nodes	Model No.	Trial	Mean Square Errors (MSE)		
			X	Y	TOTAL
60	11	1	0.09632	0.12523	0.11077
	12	2	0.06079	0.05063	0.05571
	13	3	0.11461	0.09126	0.10293
	14	4	0.10367	0.08513	0.09440
	15	5	0.14508	0.10958	0.12733
	16	6	0.05537	0.04171	0.04854
	17	7	0.19354	0.22779	0.21067
	18	8	0.31804	0.48660	0.40232
	19	9	0.14878	0.14248	0.14563
	20	10	0.04166	0.04870	0.04518

Table 7 Performance evaluation of trainlm backpropagation network with 90 hidden nodes

Number of nodes	Model No.	Trial	Mean Square Errors (MSE)		
			X	Y	TOTAL
90	21	1	0.08318	0.63332	0.35825
	22	2	0.08143	0.05043	0.06593
	23	3	110.99985	0.19049	55.59517
	24	4	0.09462	0.06225	0.07843
	25	5	1.07399	1.47528	1.27464
	26	6	0.04179	0.11147	0.07663
	27	7	0.14636	0.10816	0.12726
	28	8	1.32628	1.72884	1.52756
	29	9	0.05776	0.05008	0.05392
	30	10	0.95564	88.00000	44.47782

Table 8 Performance evaluation of trainlm backpropagation network with 120 hidden nodes

Number of nodes	Model No.	Trial	Mean Square Errors (MSE)		
			X	Y	TOTAL
120	31	1*	0.16179	0.28120	0.22149
	32	2	0.26297	0.26263	0.26280
	33	3	0.05970	0.06225	0.06098
	34	4*	0.16179	0.28120	0.22150
	35	5	0.14635	0.14820	0.14728
	36	6*	0.16179	0.28120	0.22150
	37	7	0.07765	0.12779	0.10272
	38	8	0.12598	0.10854	0.11726
	39	9	0.64791	1.06919	0.85855
	40	10	0.06851	0.06259	0.06555

* Network was running into same optimal point.

Table 9 summarizes the results of the Bayesian regularization backpropagation network with 10 different initial weight and bias values (10 trials).

Table 9 Performance evaluation of trainbr backpropagation network with 20 hidden nodes

Number of nodes	Model No.	Trial	Mean Square Errors (MSE)		
			X	Y	TOTAL
20	41	1	0.08243	0.06413	0.07328
	42	2	0.06302	0.05353	0.05828
	43	3	0.09726	0.05378	0.07552
	44	4	0.11591	0.07899	0.09745
	45	5	0.05702	0.04176	0.04939
	46	6	0.03222	0.06268	0.04745
	47	7	0.04700	0.04869	0.04785
	48	8	1.73626	2.09154	1.91390
	49	9	0.06499	0.06022	0.06260
	50	10	0.24103	0.19220	0.21661

After training the artificial neural network with a variety of different architectures and complexities with 323 input/output patterns, only seven models have MSE values less than 0.05 as shown in Table 10 (sorted by min to max of MSE value).

Table 10 Networks with MSE value less than 0.05

Model No.	Training function	Hidden nodes	Mean Square Errrors (MSE)		
			X	Y	Total
9	trainlm	30	0.04545	0.03140	0.03842
20	trainlm	60	0.04166	0.04870	0.04518
46	trainbr	20	0.03222	0.06268	0.04745
47	trainbr	20	0.04700	0.04869	0.04785
16	trainlm	60	0.05537	0.04171	0.04854
4	trainlm	30	0.05500	0.04368	0.04934
45	trainbr	20	0.05702	0.04176	0.04939

However, only one model will be selected to test with unseen data in the next experiment, which is model number 9. The reason for its selection is quite straightforward; firstly, model number 9 provided the lowest MSE values compared to other models. Secondly, trainlm works better and easier to avoid data overfit compared to trainbr in training process. Lastly, as with reference to Petersen's (2013)

recommendation to secure the ability of the network the number of nodes should be kept as low as possible, and Model number 9 consists of only 30 hidden nodes which is the smallest number among trainlm backpropagation network evaluated here.

In addition, the performance of this model in the training process could also be described using the correlation coefficient (R), whose expression is:

$$R = \frac{Cov(X,Y)}{\sqrt{Cov(X)Cov(Y)}}$$

where, X = actual value; and Y = calculated value

This formula always returns a value between -1 and 1, due to the standardizing procedure of dividing by square roots of the two covariances. 1 indicates a strong positive relationship, whereas -1 indicates a strong negative relationship, and 0 indicates no relationship at all (Ross, 2011).

In this study, larger values of R represent better performance.

The correlation coefficient after the training process of model number 9 is given in Figure 30. It consists of four sub-figures which demonstrate the model performance using four different groups of inputs; training set, validation set, test set, and all three sets combined, in order to be used as unseen data.

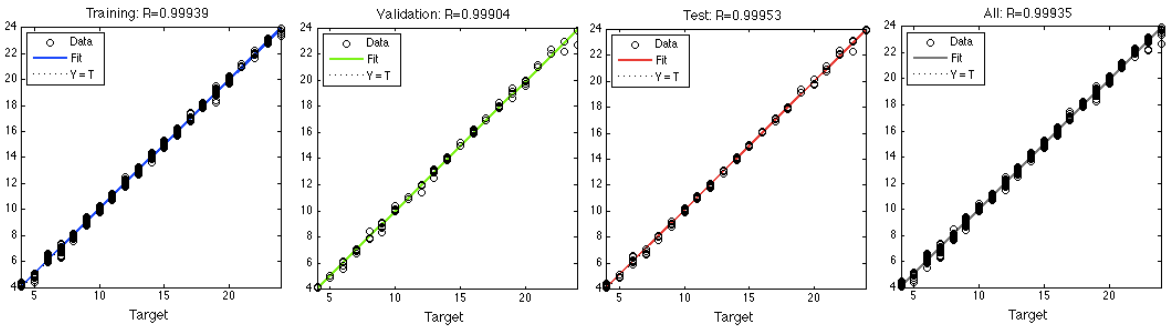

Figure 30 Correlation coefficient of model number 9

4.4. Experiment 4 Source identification by using the selected model

After introducing 500 unseen input data points (500 leakage locations) into the ANN, errors (the difference between calculated outputs and actual outputs) will be calculated and plotted as shown in Figure 31. The results indicate a well performing model; almost all points were plotted very close to the centre of the graph (which indicates a low error rate), and they were all in close proximity to each other. Overall the model showed a high level of accuracy in predicting the location of methane leakage sources, despite the fact that it also generated a number of false outliers, with results with a deviation of about 1 metres from the actual location and one other output that had an error value of more than 1.5 metres. Data visualisation is an excellent method to make data easier and simpler to understand (as shown in Figure 31); however, in order to compare individual data sets and outcome, quantitative data showing the model performance in greater detail should also considered. The distribution plot of the results is shown in Figure 32.

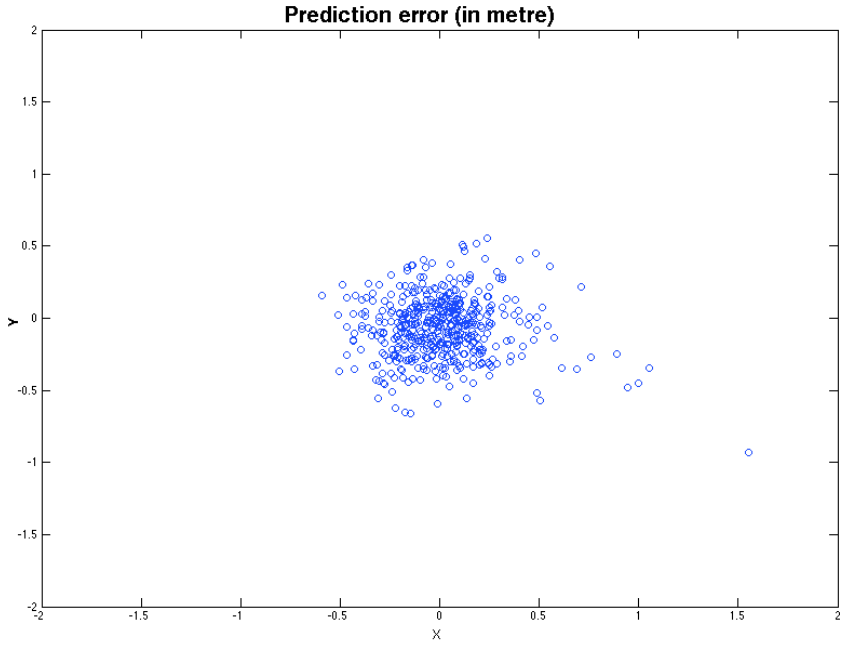

Figure 31 Error between calculated output and actual output after presented 500 unseen input data to the network

For all 500 leakage source scenarios run in the ANN, the average of the MSE (Mean Square Error) value is 0.0534 metres. This value was calculated from the average MSE of all outputs, which represent leakage positions on both x-axis and y-axis; those are 0.0583 and 0.0486 respectively. The correlation coefficient is 0.99783 for total ANN model runs as shown in Figure 33.

The final criterion presented here is Normalize Error (NE). This is very useful by providing error values in percentage terms. This helps the understandability in general, and also can eliminate the different scales from several source parameters. In this study the Normalize Error of this ANN is 1.29%, meaning that on the average calculated outputs are deviating from actual outputs by 1.29%.

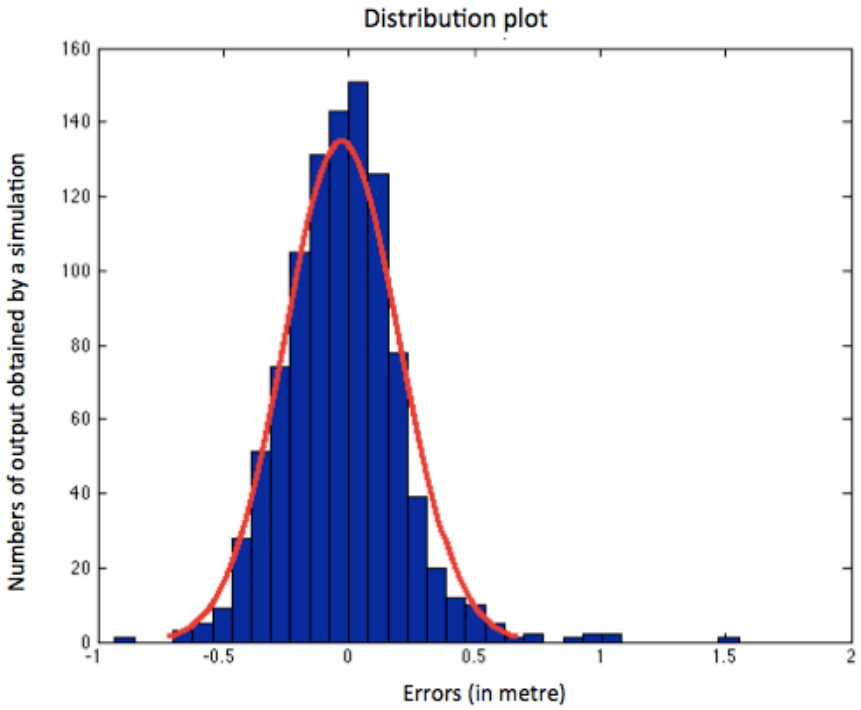

Figure 32 The distribution plot of the error between calculated output and actual output after presented 500 unseen input data to the network

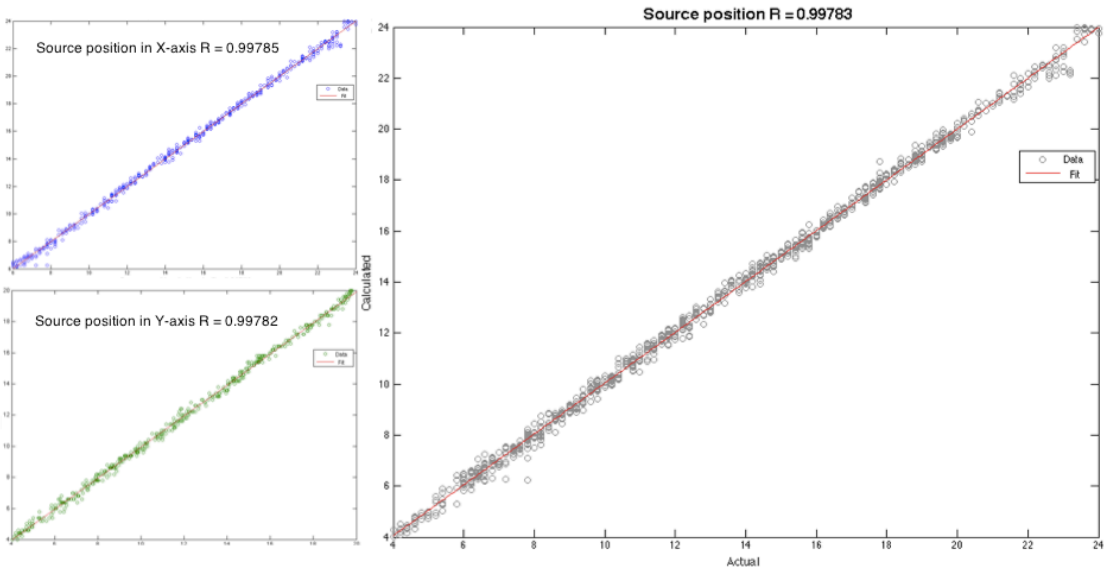

Figure 33 Correlation coefficient of the ANN

4.5. Experiment 5 Robustness test

The robustness of the network (trained with noise-free input/output pattern) was evaluated by testing with different noise levels, based on Perc values from 1% to 20%. The output from the network showing the prediction error is demonstrated in Figure 34. And summaries of the network performance from four different noised levels are shown in Table 11 and Figure 35 respectively.

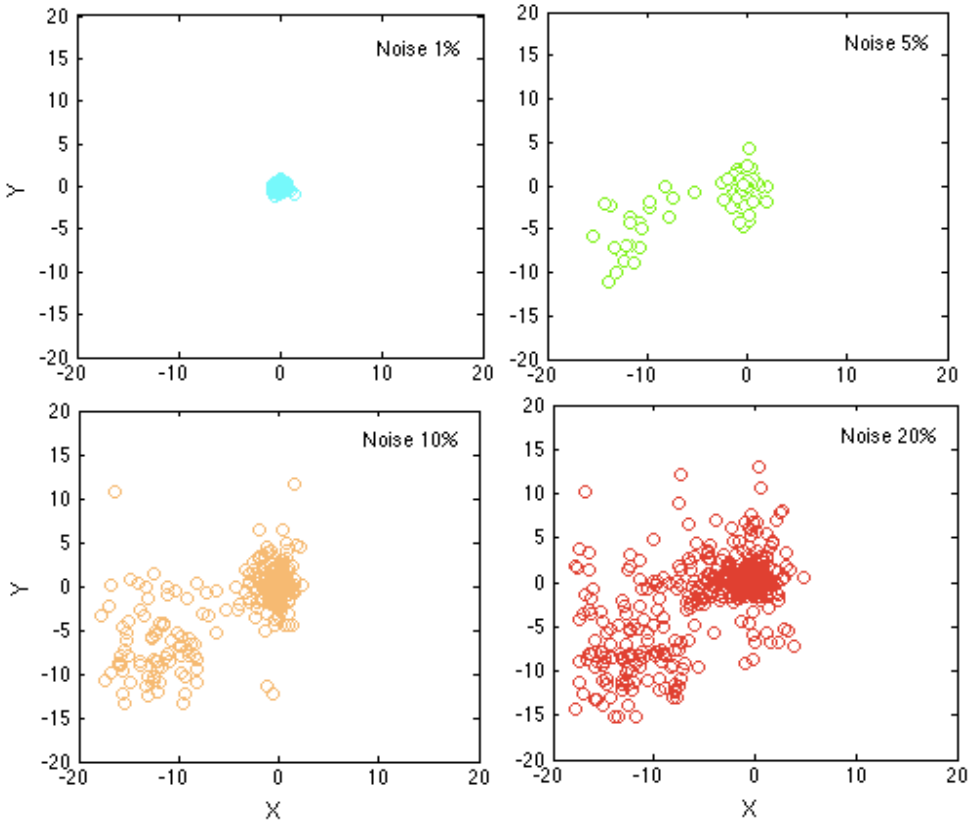

Figure 34 Error (in metre) between calculated output and actual output after presented 500 unseen input data points to the network with different noise levels added.

Table 11 Performance indexes MSE and NE for 5 different noise levels

Noise	MSE [m]			NE [%]		
	x - axis	y - axis	total	x - axis	y - axis	total
Nosie Free	0.0583	0.0486	0.0534	1.16%	1.45%	1.29%
1%	0.0701	0.0617	0.0659	1.28%	1.60%	1.42%
5%	6.0147	2.1880	4.1014	5.37%	5.73%	5.53%
10%	26.1059	11.9478	19.0269	16.57%	15.44%	16.07%
20%	45.8411	22.9109	34.3760	29.05%	25.32%	27.41%

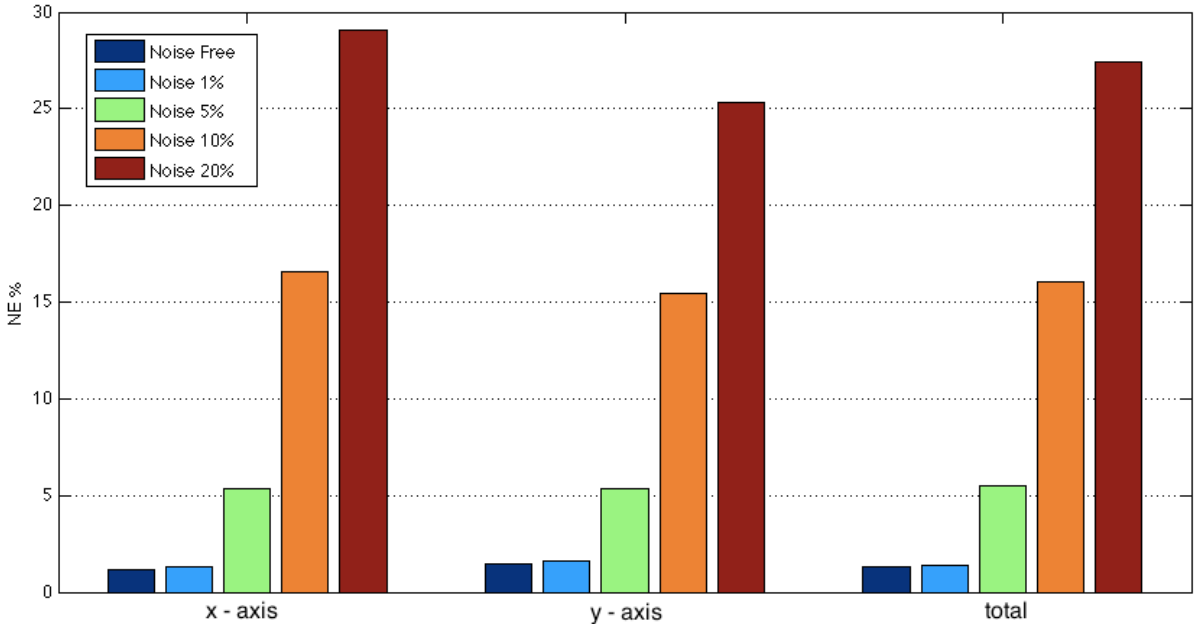

Figure 35 Normalise errors NE% of source position estimation for each axis using different levels of noisy input at the inputs of the ANN model.

From the table and figure above, the normalised error performance index increases with the percentage of noise introduced in unseen inputs. The most significant amplification errors are observed in noise level of 5% and above. There are about twenty outputs that indicate an estimated error of more than 5 metres (in the –x and –y direction) from the target, and this number increases along with noise added in the input. As seen from Figure 34, the errors seem to be separated into two groups (centre at (0,0) and about (-10, -10)) rather than staying only at the centre at point (0,0) with lower accuracy and precision. The reason for this trend is at this point not clear, but may include factors such as data pollution and the lack of data cleaning filters. To identify precisely the real root cause would be beyond the scope of this study.

4.6. Summary and Discussion of the results

Artificial neural networks try to mimic the concept of human cognition; the way humans learn and respond to any situations they encounter. This makes the use of neural networks very attractive for addressing and solving complex and non-linear problems. However, there are many different types of analysis artificial neural network employ, based on their architecture and training processes. Therefore, it is better to look at each individual type of network to evaluate their usefulness for use in leak detection systems. A backpropagation neural network is the principal analysis methodology employed to address the objectives of this thesis. Hence, the artificial

neural networks or ANNs, which are mentioned in this part, are typically referred to as backpropagation neural networks.

The problem solving methodology of the neural network is defined as a soft computing method. This tries to understand the relationship between input and output of training pairs during the ANN's training process, unlike conventional solving methods, which use mathematical equations to create the outputs. Therefore, the accuracy of the output obtained by ANNs is reliant on both the quality and the quantity of the training data. Firstly, high quality here means that the data must fully represent real incidental events as well as eliminating any contaminated data, such as low confidence data and data extracted from poor performing sensors. Secondly, optimal quantity of training data here means that the data has to be sufficiently scattered over a range of possible accident scenarios. That means a comprehensive database (which contains the sets of training data) requires various combinations of modifying parameters and covers as many real incidents as possible. In addition the quantity of training data should be minimised to as little as is required to produce acceptable outputs, as large amounts of training data can require high computational efforts during the training process.

In this study the backpropagation network was employed to estimate the leakage location based on measurement data (the concentration of dissolved methane in the sea water) from six different sensors locations. Setting up the exercise to obtain the training data (leakage location and the methane concentration in six different locations) in real subsea condition is extremely challenging due to constraints such as the difficult environment, costs and time.

A computational fluid dynamic model was deployed to create a comprehensive database by simulating all possible causes and their consequences as it was shown in the *experiment 1*. It was used to generate two required data sets for artificial neural network models. These are:

1.) Input/output patterns – each leakage source and its outcomes in 323 different locations over the considered area were simulated and used in the training process. In addition, to ensure a well running training process and incremental increases to the ANN model performance, all simulated leakage source locations used for input/output patterns must be carefully defined, such that new input/output data should be separately added in the area where there is a high complexity of turbulence or conditions, and possible to reduce in the smooth and calm area. However, in this experiment, the density of generated leakage sources (as an input/output pattern for training) were constant at every one metre along x and y directions, no matter the severity of conditions in each area.

2.) Unseen input/output data – 500 different leakage locations and each of their outcomes were created and used for testing the model performance. This unseen data was not defined by the model owner but created using a randomizing function instead. Uniform distribution with non-repeat values was deployed in this random function. This ensures the leakage location of unseen data is covering all considered areas and providing a new location that had not been used for training before. Therefore the data used in the testing process is new, fresh, and raw in order for the artificial neural network model to perform its ability to predict an output from unseen input data. The unseen output data was used to compare with the calculated output obtained by the model.

The input data represent the methane concentration levels, which are measured by six methane sniffers after the sources start leaking methane, whereas the output data represent the leakage location which led to this outcome.

Typically, the output format of CFD does not match the input format for artificial neural network models and sometimes data from CFD can either be not sufficient or too numerous. A functional data manipulation process has to be applied and deployed throughout the system in order ensure compatibility between CFD and the neural network model. The data manipulation process in this master thesis was created by myself using visual basic. This program selects 120 out of 12000 data points generated by CFD of each case (leakage location), and then arranges this selected data in the format required by ANN. This step is required and helpful to create a valid system with high performance and high reliability, as it can reduce the human error of dealing with too many data points and the configuration can be changed easily to match the different types of required data. Moreover, this is an automatic process, which can run continuously and fast, as all 823 different cases had been manipulated and finished in approximately 5 minutes.

By using CFD, the simulation of all 823 different cases was completed and ready for the ANN model within a few days. In addition, the number of cases or any new scenarios defined by to users needs, can be extracted from the program easily by only changing numerical values. This tool is extremely useful to simulate, demonstrate, visualise, and provide information of fluid dynamics scenarios in conditions where humans find it difficult to be involved. However, to simulate a case approximating real-world conditions as closely as possible, many factors such as mesh size, related equation, related parameters, validation method and interval of calculation have to be carefully considered and optimised. The more accurate the training data can represent the real-world conditions, the more accurate and better performing the model will be and the easier it is to deploy in an operational leak detection system. This can lead to high demands for more resource, time, and money from companies and organisations wanting to undertake these kinds of simulations.

However, even the best set of training data can become useless, if the architecture of the neural networks is not appropriate for the provisioned training data. The neural network and all its relevant parameters have to be well defined in order to fully extract patterns or identify suitable features of the training data. In themselves, the neural networks have no established criteria to determine a value of each parameter in order to extend the fit of the data by the model. This lack of concrete guidelines can be an issue in the implementation of an ANN. Therefore, the alternative solution is to run several experiments to determine the best or optimal network architecture for each specific problem. The parameters that are involved in the experiments carried out for this thesis were the training function, the number of hidden nodes, and the optimum values of both weights and biases. The training function optimization was demonstrated in the *experiment 2*, while the others were shown in the *experiment 3*.

The training function or training algorithm is the mathematical function that is used to update the ANN parameters (weights and bias values), reducing the overall error rate of the total system as quickly as possible. There are several training functions provided by Matlab's Artificial Neural Network Toolbox. All of them have their own specific features as earlier shown in Table 3. Therefore, to find out which training function is the most suitable to deploy in leakage source identification, I have undertaken benchmarking by determining all related parameters as a constant and for all twelve different training functions as shown in *experiment 2*. This ensures that those factors will not interfere with the performance of the training algorithm. In addition, to select the most suitable training function, the key is to find a balance between time and results; slow rates of learning require a large number of iterations to understand the training data, whereas a high rate of learning may result in missing the minimum on the error surface.

For this benchmarking the Levenberg-Marquardt algorithm (trainlm) is the most interesting training function when compared with others. This is due to the fact that one model in trainlm group obtains the lowest Mean Square Errors for this experiment. These results indirectly imply that for using this specific set of training data (simulated leakage point and its consequences) combined with Levenberg-Marquardt algorithm (trainlm) for training the ANN, there is a likelihood to provide the best outcome (global minima), compared to another algorithms. Even though it required longer training times compared to other algorithms, the longer training times need not be taken into the account, as it is pre-processed before the network is implemented within real operational circumstances.

There are no firm rules to create a well performing artificial neural network, but only suggestions based on books, articles, forums, and other relevant sources. These suggestions still cannot provide full explanations for all potential problems, as well as not being able to guarantee success in each application. For example Levenberg-Marquardt is supposed to be the fastest training algorithm and more efficient than other technique when the network contains no more than a few hundred weights

(Torrecilla et al., 2008) (Hagan and Menhaj, 1994). Unfortunately this is not applicable for this specific problem as Levenberg-Marquardt algorithm (trainlm) took longer training times compared to other algorithms.

Even if the training function evaluation process did not create any new insights or may obtain the same results as other sources suggest, it is still advantageous to run this benchmarking, at least to confirm that the selected training functions have a high probability to provide a good result in this specific scenario. From the results of this experiment, Levenberg-Marquardt algorithm (trainlm) was selected, and it was also assumed that its performance would be improved if its number of weights and biases were well optimised for this training data set (a leakage identification scenario).

Therefore, *experiment 3* had to be created and evaluated to determine the best number of hidden nodes by training the artificial neural network with a variety of different settings, until the ANN obtains acceptable results. This experiment separates the number of hidden nodes for training and testing processes into five groups: 30, 60, 90, and 120 hidden nodes with Levenberg-Marquardt and a one extra group: 20 hidden nodes with Bayesian regularization backpropagation. Each group consists of ten models, where each model contains the different initial values of weights and biases.

Theoretically, these initial values are automatically adjusted after feeding the new pair of training data into the training process. When this process is finished, the final values of both weights and biases are fixed as a constant to estimate the output. Therefore, the total performance of the neural network model is determined by the final values of weights and biases of each individual neuron in the system. Unfortunately, the weight and bias values can be another issues with ANNs. This is because of the synaptic weight and bias values of each neuron are only real numbers without physical meaning. This means that to associate the numerical values of the weights and biases with any physical phenomena or knowledge is impossible. Therefore, selecting the optimum values of the weights and biases are based on trial and errors of different initial values of both weights and biases.

In addition, this training process of neural networks can be described as a black box process; we know very little of what the system is actually doing due to the fact that the system is learning by itself and progresses on its own. The users have no other role than to feed the training data and watch it train and await the outputs. This is very simple in term of operation. However, it becomes more complicated for troubleshooting the network problems or understanding the concept behind the network. That is because the final product of the training process is a trained network that provides no mathematical equations or any coefficient numbers. The network itself is the final equation of the relationship.

In order to finalize the best suitable architecture for application in leakage source identification, the 323 input/output patterns were used for training all 50 models of

the artificial neural network. Later, these input/output patterns were used as an unseen data, in order to examine the model performances. All models have their performance evaluated by MSE, which represents the deviation between the actual and the calculated outputs.

The result from these experiments shows that out of the 50 considered model, the ninth was the most promising (we refer to is subsequently as Model 9). It should be considered for testing with unseen data for the following reasons:

1.) The neural network architecture of model 9 consists of 30 hidden neurons, which is the smallest number amongst all models within Levenberg-Marquardt training algorithm. This can help to boost the performance of the network, as the number of hidden nodes should be kept as low as possible, and also be able to reduce the training time.
2.) The model number 9 contains a reasonable and useful number of related parameters, which enables the model to obtain the smallest MSE value (0.03842). This MSE value indicates how well each model performs in this training process, it also indirectly implies the performance of the model in terms of dealing with unseen or unknown data. The lower MSE the values, the higher the potential of success in predictions.
3.) The correlation coefficient values present better performance of this model, since the R value is close to 1. Therefore, apart from having a high accuracy of prediction, this selected model also provides a strong positive relationship between calculated and actual outputs.

To simulate such a real-world scenario the source identification model (the trained model 9) was tested with unseen input data sets (500 different leakage locations) generated by CFD, as was shown in *experiment 4*.

These 500 unseen input data sets were subsequently converted by the ANN (the trained model 9) into 500 calculated outputs and compared with actual unseen outputs. The analysis of the two criteria mean square errors (MSE) and correlation (R value) indicates that this ANN is able to approximate leakage source identification with good accuracy (based on low MSE value). Also, the value between actual and calculated leakage location are well correlated (based on R value close to 1).

In addition, the Normalized Error (NE) is introduced to measure the misfit of both actual outputs and calculated output as a percentage. This makes the result easier and more straightforward to read and compare. The Normalize Error of this ANN model is 1.29%, meaning that on average calculated outputs are deviating from actual outputs by 1.29%. However, in each leakage source scenario, an error can be less or more than 1.29% depending on the quality of inputs. For example, a set of inputs, which contains only methane concentration from two sensors has a higher chance of providing less good predictions than a set of inputs with full data from six sensors.

Therefore the optimal design of the locations for methane measurement on the seabed is important. This can be an opportunity for improving the model.

The experimental error rates of around 1% are unlikely to be replicable in real world conditions. That is because of the subsea conditions employed in this experiment being comparatively simple, as I considered only the steady state of 2-D sea current flow; no changing or disturbance from external factors during the dispersion of released methane. This is one gap that should be addressed in future studies (using ANN in dynamic conditions for subsea source identification). In addition, this model still needs to be validated using realistic field experiments that may incorporate many operational uncertainties. However, the results are nevertheless valuable, important and necessary, as they demonstrate the potential and the advantages of the use of ANN in leakages identification. These are specifically:

1.) Even though the neural networks in this study only used steady state conditions, it provided a powerful modelling tool to identify leakage locations. This becomes a strong indication that with greater amounts of source and research data and more detailed modelling performance improvements, ANNs are highly feasible for use in real world subsea leakage systems.
2.) It has also shown that the neural network can reduce or eliminate the need for specific knowledge of related parameter such as turbulence or vortices occurring in the considered area to be taken into account. That can be an advantage to implement ANNs in any high complexity system. Based on the research in this thesis, the neural network requires only two parameters during the training process. These are the causes (leakage locations) and its consequences (values from methane sniffers).
3.) The neural network can predict the output almost in real time in operational conditions; the model generated the 500 calculated outputs within a few seconds. However, the main time-investment for the ANN approach occurs during the training process, specifically where very large networks and a large amount of data is involved. The model number 9 (the backpropagation neural network with 120 input nodes, 30 hidden nodes, and 2 output nodes) took approximately 30 minutes for training. This is because of the CPU computes the function of each neural node and connection separately. This issue is not really a big problem if the model runs on a parallel computer system.

In a real-world environment there are many factors either controllable or uncontrollable, which can interfere with a system, such as high water contamination and malfunction of sensors that could result in faulty readings or missing data. This may impact on the performance of leakage source location identification. Therefore, the robustness test is introduced in *experiment 5* to assess how this model operates in unusual environmental conditions and in the presence of erroneous inputs. To demonstrate this, white noise is added to the set of inputs to examine the model's

performance. There are four different sets of contaminated unseen inputs, which are separated by the percentage of added noise: 1%, 5%, 10% and 20%.

The result of the robustness test shows that the performance index increases with the percentage of noise introduced in unseen input. Significant amplification errors are observed in noise levels of 5% and above. We can see that the selected ANN can tolerate contaminated data only up to a certain level (approximately 1%– 4%). The reasons for this deteriorating performance may lie with the contaminated input data where a high percentage of noise was located in the area outside the range of variables in the training data set. To find out the real root cause is considered to be outside the scope of this study.

Typically, the robustness testing is part of any quality assurance methodology by developing test cases and test environments, which can be used to assess and verify the overall system performance dealing with unusual events. The more robust the system, the more extreme environmental conditions it can tolerate. In addition, developers can also use this information to find any root causes that degrade a system's robustness, and then explore the solution to overcome these challenges. However, the research and modelling results at this stage can not provide a definitive answer on the final validity of this approach due to the lack of scientific standards around the accuracy of estimation of leakage source position identification (i.e. is a standard above 90% acceptable, or does it require 95% +, etc.) in both presence of noise and without noise.

As the results obtained from the robustness demonstrated that the selected model can tolerate contaminated data only up to approximately 4%, performance degradation of the system could occur, once it deployed in real-world environments, which are highly dynamic and contain many different and various influences. However, there are a variety of approaches and techniques from different disciplines to overcome these issues. The alternative solutions can include:

1.) The training data will have to be carefully selected, especially in the case of the neural network with supervised training method. This is due to the fact that the performance of prediction obtained by these networks are limited by the range of training data. Therefore, this training data should address the exact level of noise; the uncertainty of the system. This applies to all scenarios that possibly occur in the considered area and should be captured and use as training data. This enables the neural network to respond in several environments: different sea current speeds/directions and various levels of disturbed flow fields. However, the minimum quantity of data required should be taken into account, as the model will demand less resource: time for data cleaning, calculating time, and computer and storage requirements.
2.) Introducing filters, either mechanical or electronic. They can help to eliminate any noises in the input data. For example, for unseen data with 20% noise

(which is very vibrant and fluctuating) the use of a Kalman filter can make the input smoother as this filter has an ability to predict an optimal estimate input by operating recursively noisy input data.

3.) Changing the learning method of the artificial neural network from supervised training to unsupervised training. This unsupervised training enables the neural network to adapt and respond to the new inputs on its own; making this system more intelligent to work in a dynamic environment.

No attempt was made to evaluate the above techniques for the purpose of this study.

Lastly, based on all experiments in this thesis we found one additional interesting ability of artificial neural networks, which is their high flexibility. This makes them suitable and adaptable to address a variety of pattern recognition challenges; there is basically no right or wrong solution (model) for each application but only those suitable or unsuitable. Because of this characteristic, ANNs have been widely used to solve non-linear problems that can rarely be explained by equations. Moreover, this high flexibility enables ANNs to easily be combined with other methodologies and technologies. This can result in the improvement of the current ANN performance and also the ability to solve a wide range of specific problems. Some of the techniques suitable for combination have been identified in several articles, amongst them Neuro-Fuzzy, Neuro-PID, and Markov chain-Neural Network for instance. They are integration of the ANN concepts with Fuzzy logic, PID control and Markov chain respectively.

Those are the main reason why I personally believe that application of ANNs (both conventional ANNs or improved ANNs) have significant potential to deal with the dynamic environments that apply to this specific problem too (i.e. subsea oil and gas leakage source identification).

CHAPTER 5 CONCLUSION

5. Conclusion

5.1. Conclusion

As the concerns of severe consequences of the accidents from oil and gas exploration and production activities rise, the safety and protection systems are required to provide higher performance and greater reliability to avoid or at least mitigate a potential disaster. Part of a range of improvements and technological advances that are introduced by the industry and its suppliers are more sophisticated subsea leak detection systems, in particular those using point sensors (methane sniffers), which have a high sensitivity to detect contaminants. These subsea leak detection systems employing methane sniffers have been the basis on which to evaluate the potential for additional neural network-based detection functionality that may be introduced in such systems.

In order to provide extra functionality (identification) for the methane sniffers, there are four main usable approaches, which have been reviewed in this study. Each technique has its own advantages and drawbacks. The analytical approach is simple but not able to simulate the complexity of subsea leak detection scenarios. The probabilistic approach requires large amounts of accurate historical data. This is often not practical in an operational environment. The optimization approach can work in almost real-time but requires significant investment in building a database. Lastly, the direct inverse modelling approach, which can operate with comparatively little information, but required longer calculating times. As a rule of thumb that applies to all approaches, the more accurate the results, the more resources are required.

In case of a serious incident, real-or-near-real-time prediction is one of the most important tools for successful crisis management. I have identified and selected artificial neural networks (classified as an optimization approach) for supporting the methane sniffers used in today's state-of-the-art leak source identification systems. However, as this approach requires a database, I did combine this method with CFD modeling. Therefore, the basic characteristic of this combined approach is to present the methane leakage models obtained by CFD, taking the ability of an ANN to learn several complex non-linear behaviours in these models, in order to estimate the leakage location in subsea oil and gas production environments. This combined approach has been demonstrated to be an appropriate and useful tool to potentially enhance these systems' performances.

Firstly, the ANN approach is very straightforward; it generates the output based on what it has learnt in the training process. That implies that in-depth knowledge of the

physical phenomena (which are often hard to explain and describe) is not required, but only a relevant and high-quality set of training data. Even though an ANN seems a very simple approach in terms of creation and usage, it sometimes can appear complicated with regards to the optimisation required to make the model perform well.

The selected model was used to identify the leakage locations based on the reading from six methane sniffers around the area under consideration. The results revealed that the basic backpropagation neural network (the selected model which is trained using patterns obtained by CFD) could identify the leakage source in calm environmental conditions (steady state) with satisfactory accuracy. Moreover, the neural network offers other benefits to the total system, as was demonstrated in the robustness test. It can operate and provide an acceptable estimation of leakage location in the presence of noise (up to certain levels). The noise represents the contaminated data either from the environment or the system, missing measurement data, or unusual events occurring in the area.

Secondly, CFD demonstrated an alternative solution to obtaining information required by the ANN without incurring the considerable costs of real environmental sampling and testing. It provides a realistic description of the characteristics and behaviours of fluid flow in as many scenarios as the ANN required. However, this comes at the cost of the high computational time usually necessary to run the CFD model, especially several models which are needed to build a sufficiently large database (for training process). Fortunately, the ANN can reduce the size of the database. This is due to the fact that ANNs do not essentially require the high similarity of the matching (between unseen input and database) to calculate the outputs. Therefore, the high computational time requirements are also reduced.

Lastly, the overall results show that the combined approach (CFD and ANN) is a promising tool to support pinpoint sensors (methane sniffers) to be more efficiently used for identifying leakages in calm condition. However, more efforts are required to overcome issues related to the dynamics and uncertainty of several parameters in real world scenarios, such as temperature, sea current directions and velocities, and surge currents. This will makes the subsea leak detection system more attractive to operators, smarter, as it can provide real time monitoring with high sensitivity of detection, as well as more accurately pinpointing the leakage source locations. This in turn enables the operator to respond faster to stop any incidents before they develop into critical events.

5.2. Recommendations to the Industry

Because this thesis is essentially a feasibility study using the combination of CFD and ANN to identify the leakage source, the results obtained principally come from the simulations conducted. That means this study lacks some of the operational data to support its use in industrial environments such as investment costs, rate of return, customer satisfactions, and so on. Therefore, to adapt and develop this study for implementation in an operational environment, I strongly recommend stakeholders to investigate the following topics as a matter of priority:

1.) As the results in this study have been obtained by computer simulation only, it has not been possible to take all operational uncertainties and related parameters into account; some parameters can be identified and detected, whereas others cannot. Therefore, one step required before deploying this approach is to carry out realistic field experiments; testing in real sea conditions or something similar (simulated pools). The data generated by this more realistic performance evaluation of ANN models in real-time and real-world scenarios can be collected and may be used as a reference source.
2.) Optimising the number and location of methane sniffers on the seabed is encouraged. This can ensure that a subsea leak detection system installed would use the minimum number of sensors while remaining accurate and fast at the same time. In addition, the sensor locations as well as the number of sensors installed are very important to ensure reliable and accurate outputs in this combined sensoring/modeling approach. The higher the number of inputs (i.e. the data from all six sensors), the better a prediction we can obtain. Vukovic and Srebric, (2007) mentioned two alternative methods that have been inspired by the artificial neural network theory: “optimal brain damage” or “pruning algorithm”. This starts the optimization with a large number of sensors and then reduces the sensor count until reaching the optimal value. Contrastingly, the “growing algorithm” performs in the inverse direction.
3.) This combined approach requires a large amount of computational time, in terms of training the ANN model and creating the database using CFD in particular. A potential user should balance the time and resources spent on the project with the anticipated obtainable outcome and shift the balance in favour of business goals and requirement. For example, utilising the ANN to identify the leakage zone instead of pinpointing the precise location of the leakage. This can definitely reduce the efforts used in the development phase of the project, as the size of database will be smaller. However, the model can still achieve adequate performance and might provide a more robust solution. Moreover, the ANN based methodology in this thesis is developed and evaluated for identification of unknown leakage sources in terms of locations only. Other features such as magnitude and timing of leakages should be

considered and added into the model, in order to respond to stakeholder requirements.

Lastly, there are undoubtedly other suggestions that are worth considering. A good way to bring out new ideas is to undertake brainstorming sessions with people with different (but relevant) expertise and backgrounds, covering the breadth and depth of experience and need to generate potentially useful ideas as well as feedback and recommendation from different perspectives. This can improve the performance, reliability and relevance as well as reduce the errors of such systems, once this approach is fed to prototyping or the deployment phase.

5.3. Future Scope

Even though this research work is a feasibility study only and the experiment cases are quite simple, the success of the results shown in this thesis support the idea of applying this approach in an operational environment. However, further studies need to be conducted. This may include:

1. Further study, experimentation and design of the ANN model to provide acceptable results identifying the leakage location in more dynamic conditions. For example, taking into account the constant changes in sea current velocities and their direction would enable the model to learn how to deal with greater uncertainty and unexpected situations. This would result in a model much more suitable for real-world application.
2. To make this optimization approach perform with a higher accuracy of prediction, it would be advantageous to capture and collect historical data in areas under consideration and feed that into the current ANN model. In addition related data sources, which affect the characteristic and behaviour of methane dispersion, should be taken into account. The related data sources might be the sea temperature, current direction, current speed, and saltiness for instance. These sources have to be further investigated and applied to the system to significantly increase the performance of identification.
3. There are two training topologies in ANNs: supervised and unsupervised. For the future scope of work, I strongly suggest to look at neural networks with unsupervised training instead. Firstly, the time required to simulate multiple cases for training can be decreased or sometimes eliminated, as only small additional databases are required. Secondly, this approach has been widely used in actual projects in various application fields, for example web searching, which contain very large dimensions of data. That indirectly implies that neural networks with unsupervised training have been proven to deal with such a high complexity system.

4. As ANNs are by their nature problem dependent, this model is only suitable for this specific scenario. Therefore the next steps may include the design of a universally compatible system (based on this approach) that can be deployed on oil platforms in any location (where the subsea leak detection system is installed) and is also suitable for various types of sensor arrays (different number of sensors and different sensors allocations). This should result in the efforts and related costs required for each project decreasing.
5. Whilst it is currently not explicitly required, solutions identifying non-point or multiple leakage sources, may well be of interest in the future, despite clearly posing a significant challenge.

The above are only high-level suggestions to make subsea leak detection systems smarter, more practical, more reliable and more attractive to operators. On their own, the concept of ANN and CFD are probably insufficient to implement all the suggestions and meet all the challenges. Therefore a combination of techniques and methodologies in conjunction with the application of artificial neural networks and their underlying principles may hold the best prospect for delivering a future-facing, operational and cost-effective leak detection system.

REFERENCES

Alavala, C.R., 2008. Fuzzy Logic and Neural Networks: Basic Concepts and Applications. New Age International Pvt Ltd Publishers, New Delhi.

Alifanov, O.M., 1983. Methods of solving ill-posed inverse problems. J. Eng. Phys. 45, 1237–1245. doi:10.1007/BF01254725

Almurib, H.A.F., Mat Isa, A.A., Al-Assadi, ayder M.A.A., 2011. Direct Neural Network Control via Inverse Modelling: Application on Induction Motors, Artificial Neural Networks - Industrial and Control Engineering Applications, Prof. Kenji Suzuki (Ed.). InTech. doi:10.5772/16052

Anguita, D., Parodi, G., Zunino, R., 1994. An efficient implementation of BP on RISC-based workstations. Neurocomputing, Backpropagation, Part III 6, 57–65. doi:10.1016/0925-2312(94)90034-5

Anonymous, 2015. Maxima and minma [WWW Document]. wikipedia. URL http://en.wikipedia.org/wiki/Maxima_and_minima (accessed 5.25.15).

Bady, M., 2013. Fundamentals of Direct Inverse CFD Modeling to Detect Air Pollution Sources in Urban Areas. Comput. Water Energy Environ. Eng. 02, 31–42. doi:10.4236/cweee.2013.22004

Bakker, A., 2002. Lecture 1 - Introduction to CFD. Applied Computational Fluid Dynamics.

Bishop, C., 2007. Pattern Recognition and Machine Learning. Springer, New York.

Blum, A., 1992. Neural Networks in C++: An Object-Oriented Framework for Building Connectionist Systems, 1 edition. ed. Wiley, New York.

Chester, M., 1993. Neural Networks: A Tutorial. Prentice Hall, Englewood Cliffs, N.J.

Cole, K., 2013. Leak Detection Methods for Subsea Pipelines. Marit. Act. Rep. November 06, 2013.

Coley, K., 2013. Leak detection methods for subsea pipelines [WWW Document]. Soc. Marit. Ind. URL (accessed 2.21.15).

Davidon, W., 1991. Variable Metric Method for Minimization. SIAM J. Optim. 1, 1–17. doi:10.1137/0801001

Decomworld, 2014. Industry gets together to plug gaps in offshore leak detection systems [WWW Document]. Decomworld Anal. URL http://analysis.decomworld.com/projects-and-technologies/industry-gets-together-plug-gaps-offshore-leak-detection-systems (accessed 2.20.15).

Demuth, H., Beale, M., Hagan, M., 2013a. Neural Network Toolbox Getting Started Guide 2013a.

Demuth, H., Beale, M., Hagan, M., 2013b. Neural Network Toolbox References 2013a.

Demuth, H., Beale, M., Hagan, M., 2008. Neural Network Toolbox 6 User's Guide.

DNV, 2010. Recommended practice DNV-RP-F302, Selection and use of subsea leak detection systems.

DNV GL, n.d. Offshore leak detection JIP [WWW Document]. DNV GL. URL https://www.dnvgl.com/oilgas/innovation-development/joint-industry-projects/offshore-leak-detection-jip.html (accessed 2.20.15).

Dreyfus, G., 2010. Neural Networks: Methodology and Applications, Softcover reprint of hardcover 1st ed. 2005 edition. ed. Springer, Berlin; New York.

Durrenberger, C., 2014. Guassian Plume Modeling (Chemical Engineering 357). Lecture notes. University of Texas at Austin.

Eberhart, R.C., Shi, Y., 2007. Computational Intelligence: Concepts to Implementations, 1 edition. ed. Morgan Kaufmann, Amsterdam ; Boston.
Elbern, H., Schmidt, H., Talagrand, O., Ebel, A., 2000. 4D-variational data assimilation with an adjoint air quality model for emission analysis. Environ. Model. Softw., Air pollution modelling and simulation 15, 539–548. doi:10.1016/S1364-8152(00)00049-9
Esser, D., 2008. Hydrocarbon sensor systems. Schiff Hafen June 2008, No. 6.
ETH Zurich, 2008. Inverse Modeling [WWW Document]. Res. Tools. URL http://www.up.ethz.ch/Research_Tools/Inverse_Modeling (accessed 3.28.15).
Franatech, n.d. Gas monitoring and leak detection in underwater application for the offshoew oil and gas. a case study. [WWW Document]. Franatech. URL http://www.franatech.com/img/application/pdf/web_franatech_4_seiter_NEU_SF_20130602.pdf (accessed 3.25.15).
Ghahramani, Z., 2011. Bayesian nonparametrics and the probabilistic approach to modelling. Phil Trans R Soc A 1–27. doi:10.1098/rspa.00000000
Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning, 1 edition. ed. Addison-Wesley Professional, Reading, Mass.
Hagan, M.T., Menhaj, M.B., 1994. Training feedforward networks with the Marquardt algorithm. IEEE Trans. Neural Netw. 5, 989–993. doi:10.1109/72.329697
Haupt, S.E., 2005. A demonstration of coupled receptor/dispersion modeling with a genetic algorithm. Atmos. Environ. 39, 7181–7189. doi:10.1016/j.atmosenv.2005.08.027
Hjertager, B.H., 2009. Lecture notes in OpenFOAM - MSK 600. University of Stavanger.
Islam, M.A., 1999. Application of a Gaussian Plume Model to Determine the Location of an Unknown Emission Source. Water. Air. Soil Pollut. 112, 241–245. doi:10.1023/A:1005047321015
Islam, M.A., Roy, G.D., 2002. A Mathematical Model in Locating an Unknown Emission Source. Water. Air. Soil Pollut. 136, 331–345. doi:10.1023/A:1015210916388
Issa, R.I., 1986. Solution of the implicitly discretised fluid flow equations by operator-splitting. J Comput Phys 62, 40–65.
Jasak, H., 2007. Scalar Transport Equation - lectur note - OpenFOAM course.
Jeraj, R., Wu, C., Mackie, T.R., 2003. Optimizer convergence and local minima errors and their clinical importance. Phys. Med. Biol. 48, 2809. doi:10.1088/0031-9155/48/17/306
Kaggle, 2011. Don't Overfit!! [WWW Document]. Home Data Sci. URL https://www.kaggle.com/c/overfitting (accessed 5.26.15).
Kathirgamanathan, P., McKibbib, R., McLachlan, R.I., 2012. Source term estimation of pollution from an instantaneous point source. Res. Lett. Inf. Math. Sci. 3, 59–67.
Kato, S., Kajii, Y., Itokazu, R., Hirokawa, J., Koda, S., Kinjo, Y., 2004. Transport of atmospheric carbon monoxide, ozone, and hydrocarbons from Chinese coast to Okinawa island in the Western Pacific during winter. Atmos. Environ. 38, 2975–2981. doi:10.1016/j.atmosenv.2004.02.049
Keats, A., Yee, E., Lien, F.-S., 2007. Bayesian inference for source determination with applications to a complex urban environment. Atmos. Environ. 41, 465–479. doi:10.1016/j.atmosenv.2006.08.044

Khlaifi, A., Ionescu, A., Candau, Y., 2009. Pollution source identification using a coupled diffusion model with a genetic algorithm. Math. Comput. Simul., The International Conference on Approximation Methods and numerical Modeling in Environment and Natural Resources 79, 3500–3510. doi:10.1016/j.matcom.2009.04.020

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680. doi:10.1126/science.220.4598.671

Krenker, A., Bešter, J., Kos, A., 2011. Introduction to the Artificial Neural Networks, Artificial Neural Networks - Methodological Advances and Biomedical Applications, Prof. Kenji Suzuki (Ed.). InTech. doi:10.5772/15751

Kumar, N., n.d. Direct and indirect search methods, Lecture Note, Advanced Topics in Optimization M8L4. Indian institute of science.

Kuzmin, D., 2010. A guide to numerical methods for transport equations.

Kuzmin, D., n.d. Introduction to Computational Fluid Dynamics - Lecture note. Institute of Applied Mathematics, University of Dortmund.

Li, F., Niu, J., 2005. An inverse approach for estimating the initial distribution of volatile organic compounds in dry building material. Atmos. Environ. 39, 1447–1455. doi:10.1016/j.atmosenv.2004.11.021

Liu, X., Zhai, Z., 2007. Inverse modeling methods for indoor airborne pollutant tracking: literature review and fundamentals. Indoor Air 17, 419–438. doi:10.1111/j.1600-0668.2007.00497.x

Mahar, P.S., Datta, B., 2000. Identification of Pollution Sources in Transient Groundwater Systems. Water Resour. Manag. 14, 209–227. doi:10.1023/A:1026527901213

MathWorks, 2015. Create and Train Custom Neural Network Architectures R2015a [WWW Document]. MathWorks-Doc. URL http://se.mathworks.com/help/nnet/ug/create-and-train-custom-neural-network-architectures.html (accessed 2.23.15).

Moustafa, A.A., Alqadi, Z.A., Shahroury, E.A., 2011. Performance Evaluation of Artificial Neural Networks for Spatial Data Analysis. W Trans Comp 10, 115–124.

Neptune Oceanographics, n.d. Sniffit for detection of subsea hydrocarbons [WWW Document]. Hydrocarb. Leak Detect. URL http://www.neptuneoceanographics.com/documents/sniffit-for-detection-of-subsea-hydrocarbons.pdf (accessed 3.29.15).

Newman, M., Hatfield, K., Hayworth, J., Rao, P.S.C., Stauffer, T., 2005. A hybrid method for inverse characterization of subsurface contaminant flux. J. Contam. Hydrol. 81, 34–62. doi:10.1016/j.jconhyd.2005.07.006

Nunnari, G., Bertucco, L., Ferrucci, F., 2001. A neural approach to the integrated inversion of geophysical data of different types. IEEE Trans. Geosci. Remote Sens. 39, 736–748. doi:10.1109/36.917884

Petersen, S., 2013. How to decide the number of hidden layers and nodes in a hidden layer.

Rajabalinejad, M., 2010. Bayesian Monte Carlo method. Reliab. Eng. Syst. Saf. 95, 1050–1060. doi:10.1016/j.ress.2010.04.014

Rege, M.A., W.Tock, R., 1996. A Simple Neural Network for Estimating Emission Rates of Hydrogen Sulfide and Ammonia from Single Point Sources. J. Air Waste Manag. Assoc. 46, 953–962. doi:10.1080/10473289.1996.10467530

Reich, S.L., Gomez, D.R., Dawidowski, L.E., 1999. Artificial neural network for the identification of unknown air pollution sources. Atmos. Environ. 33, 3045–3052. doi:10.1016/S1352-2310(98)00418-X

Richardson, R.M., Zandt, G., 2009. Inverse problem in geophysics GEOS 567. A Set of Lecture Notes. University of Arizona.

Ross, S.A., 2011. Corporate Finance: Core Principles and Applications, Global ed of 3rd revised ed edition. ed. McGraw-Hill Europe, New York.

Sayma, A., 2009. Computational Fluid Dynamics. Abdulnaser Sayma& Ventus Publishin ApS.

Silva, L.F.L.R., Lage, P.L.C., 2011. Development and implementation of a polydispersed multiphase flow model in OpenFOAM. Comput. Chem. Eng. 35, 2653–2666. doi:10.1016/j.compchemeng.2011.04.011

Singh, D., Maheshwari, G., Mathur, N., Misha, P., Patel, I., 2010. Comparison between training function trainbfg and trainbr in modelling of neural network for predicting the value of specific heat capacity of working fluid LIBR-H20 used in vapour absorption refrigeration system. Int. J. Adv. Res. Enginneering Technol. 1, 118–127.

Singh, R.M., Datta, B., 2006. Artificial neural network modeling for identification of unknown pollution sources in groundwater with partially missing concentration observation data. Water Resour. Manag. 21, 557–572. doi:10.1007/s11269-006-9029-z

Singh, R.M., Datta, B., Jain, A., 2004. Identification of unknown groundwater pollution sources using artificial neural networks. J. Water Resour. Plan. Manag. 130, 506–514. doi:10.1061/(ASCE)0733-9496(2004)130:6(506)

Snieder, R., 1998. The role of nonlinearity in inverse problems. Inverse Probl. 14, 387–404.

Sohn, M.D., Reynolds, P., Singh, N., Gadgil, A.J., 2002. Rapidly locating and characterizing pollutant releases in buildings. J. Air Waste Manag. Assoc. 52, 1422–32.

Sreedharan, P., Sohn, M.D., Gadgil, A., Nazaroff, W., 2006. Systems approach to evaluating sensor characteristics for real-time monitoring of high-risk indoor contaminant releases. Atmos. Environ. 40, 3490–3502. doi:10.1016/j.atmosenv.2006.01.052

The IDEAS Research Institute, n.d. Chapter 3 The Back Propagation Algorithms.

Thomson, L.C., Hirst, B., Gibson, G., Gillespie, S., Jonathan, P., Skeldon, K.D., Padgett, M.J., 2007. An improved algorithm for locating a gas source using inverse methods. Atmos. Environ. 41, 1128–1134. doi:10.1016/j.atmosenv.2006.10.003

Thomson, L.C., Hirst, B., Gibson, G., Gillespie, S., Jonathan, P., Skeldon, K., Padgett, M., 2007. An improved algorithm for locating a gas source using inverse methods. Atmos. Environ. 41, 1128–1134. doi:10.1016/j.atmosenv.2006.10.003

Torrecilla, J., Aragon, J., Palancar, M.C., 2008. Optimization of an artificial neural network by selecting the training function. application to olive oil mills waste. Ind. Eng. Chem. Res. 47, 7072–7080. doi:10.1021/ie8001205

Turner, D.B., 1994. Workbook of Atmospheric Dispersion Estimates: An Introduction to Dispersion Modeling, Second Edition, 2 edition. ed. CRC Press, Boca Raton.

Ushakov, I.A., 2013. Optimal Resource Allocation : With Practical Statistical Applications and Theory. John Wiley & Sons, Somerset, NJ, USA.

Versteeg, H., Malalasekera, W., 2007. An Introduction to Computational Fluid Dynamics: The Finite Volume Method, 2 edition. ed. Prentice Hall, Harlow, England ; New York.
Vukovic, V., Srebric, J., 2007. Application of Neural Networks Trained with Multizone Models for Fast Detection of Contaminant Source Position in Buildings. ASHRAE Trans. 113, 154–162.
Walpole, R.E., Myers, R.H., Myers, S.L., Ye, K.E., 2011. Probability and Statistics for Engineers and Scientists, 9 edition. ed. Pearson, Boston.
Wang, Z., Bovik, A.C., 2009. Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures. IEEE Signal Process. Mag. 26, 98–117. doi:10.1109/MSP.2008.930649
Willmann, M., Kinzelbach, W., Stauffer, F., 2015. Groundwater flow modeling: Inverse problem, lecture note : Groundwater 2 Calibration. Institute of Environmental Engineering, ETH Zurich.
Yam, J.Y.F., Chow, T.W.S., 2000. A weight initialization method for improving training speed in feedforward neural network. Neurocomputing 30, 219–232. doi:10.1016/S0925-2312(99)00127-7
Yumimoto, K., Uno, I., 2006. Adjoint inverse modeling of CO emissions over Eastern Asia using four-dimensional variational data assimilation. Atmos. Environ. 40, 6836–6845. doi:10.1016/j.atmosenv.2006.05.042
Zeng, L., Shi, L., Zhang, D., Wu, L., 2012. A sparse grid based Bayesian method for contaminant source identification. Adv. Water Resour. 37, 1–9. doi:10.1016/j.advwatres.2011.09.011
Zhai, Z. (John), Liu, X., Wang, H., Li, Y., Liu, J., 2011. Experimental verification of tracking algorithm for dynamically-releasing single indoor contaminant. Build. Simul. 5, 5–14. doi:10.1007/s12273-011-0041-8
Zhang, J., Zeng, L., Chen, C., Chen, D., Wu, L., 2015. Efficient Bayesian experimental design for contaminant source identification. Water Resour. Res. 51, 576–598. doi:10.1002/2014WR015740
Zhang, T., 2007. Detection and mitigation of contaminant transport in commercial aircraft cabins (Ph.D.). Purdue University, United States -- Indiana.
Zhang, T.F., Chen, Q., 2007. Identification of contaminant sources in enclosed environments by inverse CFD modeling. Indoor Air 17, 167–177. doi:10.1111/j.1600-0668.2006.00452.x
Zheng, X., Chen, Z., 2011. Inverse calculation approaches for source determination in hazardous chemical releases. J. Loss Prev. Process Ind. 24, 293–301. doi:10.1016/j.jlp.2011.01.002
Zheng, X., Chen, Z., 2010. Back-calculation of the strength and location of hazardous materials releases using the pattern search method. J. Hazard. Mater. 183, 474–481. doi:10.1016/j.jhazmat.2010.07.048

APPENDICES

Appendix A: Franatech METS methane sensors specifications

Table 12 Franatech METS methane sensors specifications (Franatech, n.d.)

Mechanical Specifications		
Model	Standard	Optional
Material	Stainless steel	Titanium
Weight in air	1.3 kg.	0.8 kg.
Weigh in water	1.0 kg.	0.5 kg.
Dept rating	4000 m.	
Electrical Specifications		
Input	9 to 36 VDC, 35 to 100 mA. @ 12VDC	
Output	standard analogue 0.5 V. and digital RS485	
Options	4.2 , RS232, analogue only, digital only, Desktop Converter RS485/RS232	
Calibration ranges		
Temperature	2 - 20 degree C (standard)	
Mehtan	50 nM - 10 uM (standard)	
	1 nM - 500 nM (sensitive)	
	low range 20 nM - 1 uM / high range 1 uM - 40 uM	
Response time	reaction time within few seconds	
	T90 between 1 and 30 min depending on version and deployment conditions	
Special feature	Integrate formula (plug&play)	
	Correction formula for work under variable oxygen levels	

Appendix B: Maxima and minima of a function

The maxima and the minima (the plural form of maximum and minimum respectively) of a function are the largest and smallest value(s) of the function. These values can be either the local or the global values (Anonymous, 2015).

1. A local minima (maxima) is defined as a point from a given domain that all points in a neighbourhood have a value greater (smaller) than or equal to the value at that point, when a specific function is applied.
2. A global minima (maxima) is a value from an entire domain of a function, which offers the lowest value (highest value).

For example, the function, as shown in Figure 36, is $\cos(3\pi x)/x$. If the given domain is the value between 0.5 and 0.8, the local maximum point is approximate x = 0.649 and this is also the global maximum of this specific domain. However, the main goal of the optimization method is to find a global maxima or minima of a specific function. The implication is that the values have to be defined over the whole domain. Therefore, x = 0.649 will become only the local maximum, while the global maximum point is x = 0.1.

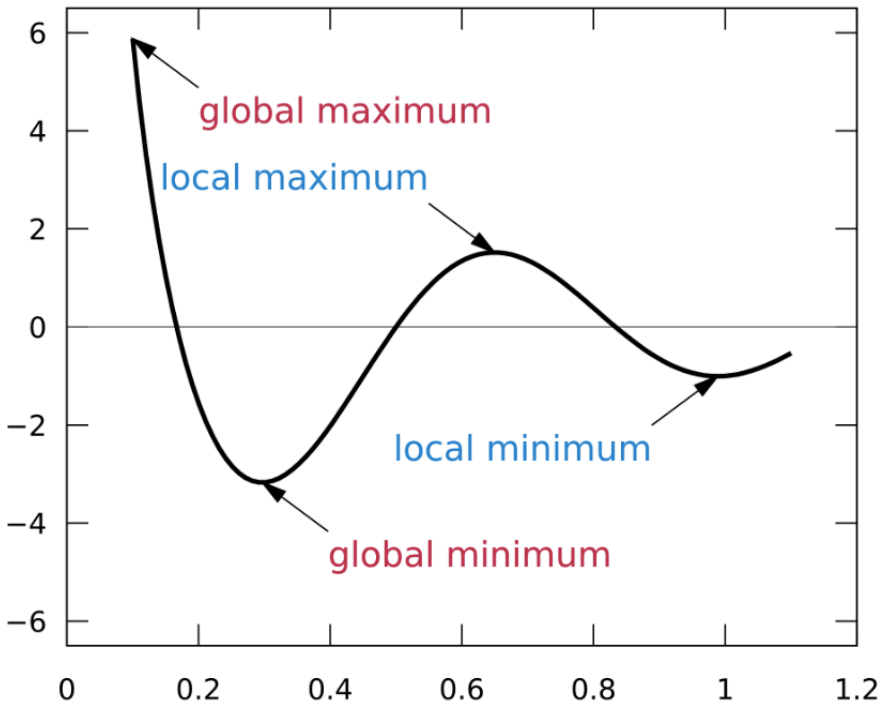

Figure 36 Local and global maximum and minimum (Anonymous, 2015).

To obtain the best outcome from an optimization approach can be difficult, as there are many different sorts of errors embedded in each calculation algorithms. One major error in the training algorithm for the purpose of ANNs is called the local minima error. It can cause the optimal solution to deviate from the true (global) solution (Jeraj et al., 2003).

The main root causes of the local minima error in the ANNs, especially in the backpropagation network, derives from two major sources. Those are an objective function of each specific problem and a characteristic of the learning algorithm in ANNs.

This problem (the local minima error) does not occur when the objective function is convex, such as the function $|x|$ or the parabola's equation. Unfortunately, the selected system (methane leakages in a subsea environment) in this master thesis is a non-linear problem with a high complexity, and also difficult to explain by equations. The implication is that the error surface of this problem has a high number of regions with local minima. Therefore, the optimal solution (values of bias and weight) might get trapped in one of the local minima instead of the global minima.

In addition, most learning methodologies of ANNs in backpropagation belong to the class of algorithms that perform gradient descent; finding a minimum point by taking steps in the negative direction of the function gradient. The implication is that the value may get stuck in one minima, with little or no chance to get out of this point and continue the calculation process. Therefore, the different initial bias and weight values can lead to the different outcome, especially in the presence of high complexity objective functions.

For example, based on Figure 36, three different initial values: x = 0.12, x = 0.8, and x = 1.1, which are provided to the objective function y = $(\cos(3\pi x)/x)$, then apply the gradient descent algorithm to optimise the problem. The optimized outputs (x) are 0.296, 0.988, and 0.988 respectively. The first x provides the global minimum (y = -3.17152), while the latter two provide the local minimum (y = -1.005).

We can see that, in order to avoid the problem of local minima in backpropagation algorithm, the different randomised initial bias and weight values have to be introduced. That can increase the chance to obtain the output (final values of both weights and biases), which is located in the global minima. That results in better performance of the model.

Appendix C: Overfitting of artificial neural networks

The main concept of ANNs is to build a model that is able to generate accurate predictions from unseen data. Apart from a high quality set of training data and the well-defined model architecture, the training process is also a key factor which determines the performance of ANNs. However, to train the model too hard does not guarantee the predictive powers of the model. Therefore, an essential step in modeling the ANN network process is to ensure that there has not been an overfit of the training data, which can lead to sub-optimal predictions from unseen data (Kaggle, 2011). Moreover, there is another problem in the training process, regarding a lack of training data. This problem is named the underfit. However, underfit is more unlikely to occur in ANNs as the model owners always aims to provide as much training data as possible.

The phrase "to overfit" (in machine learning or ANNs area) is synonymous with the expression of reading too much. The implication is that during the ANNs training process the model learns all characteristics of the training data (i.e. errors, noise, etc.) instead of learning the bigger pictures or underlying relationship. Therefore, the overfiting problem happens when a model begins to memorise the training data rather than learning to establish a trend. The concrete example of underfit, right fit and overfit are demonstrated below (Bishop, 2007).

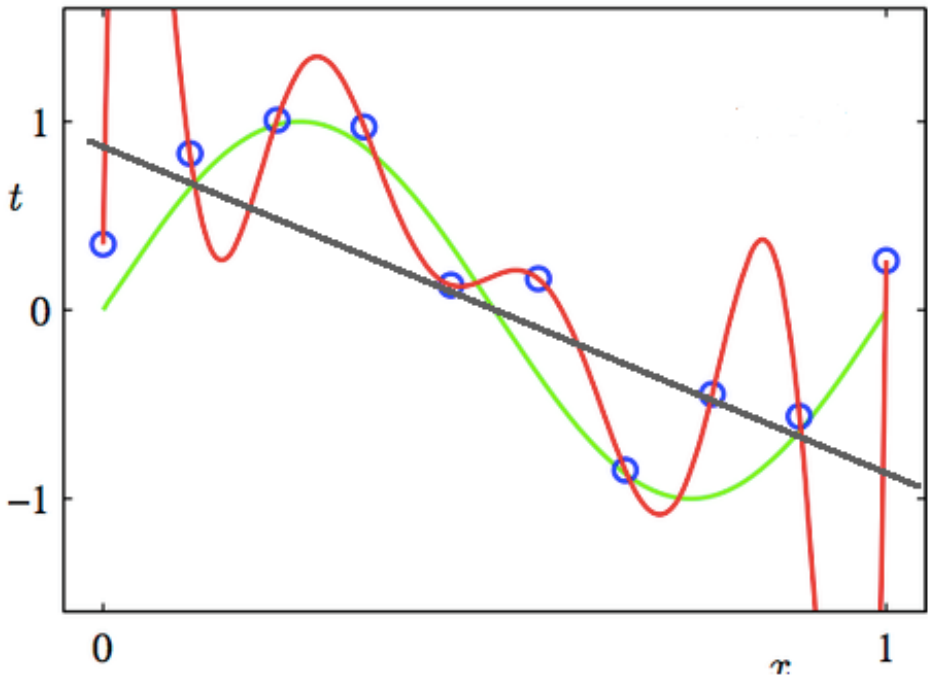

Figure 37 The regression on the data points (adapted from (Bishop, 2007)).

Based on the Figure 37, there are nine data points (blue circles) in the graph. After fitting these nine sample data points with a linear function (linear regression), the result showed that the linear line (gray) does not fit with the data anymore. This is called underfit. Introducing a sine curve (polynomial function of degree 3) instead will match the nine data points (green) reasonably accurately. This is called right fit. In order to match each data point perfectly, the result ended up as shown in the red curve in Figure 37. This curve is a 9-degree polynomial, which can reduce the deviation between nine data points and predictor function close to zero. However, the complexity of 9-degree polynomial makes it likely for the model to result in overfitting.

The root cause of this overfitting is that the model was trained with the same training set too many times, as it show in the Figure 38; the model will become overfitted if it was trained more than 9 times. The consequence of overfitting is that the model only performs well on the training data (decreasing the errors), but deteriorates in the test (unseen) data. That results in poor predictive performance of the model.

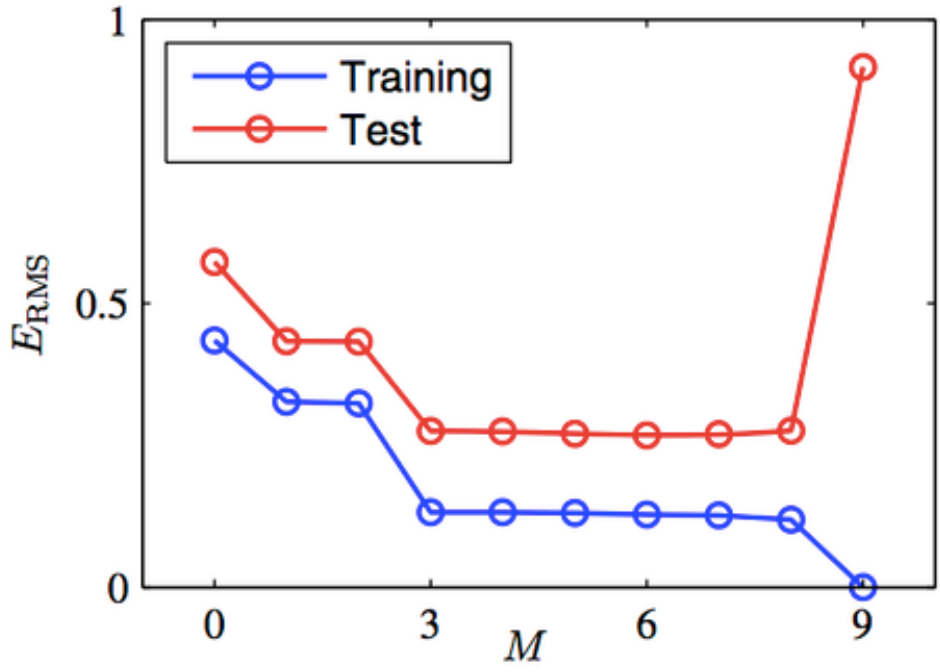

Figure 38 The error obtained by the model, which was trained by the same training data for nine times

Appendix D : blockMeshDict File

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD
Toolbox           |
|  \\    /   O peration     | Version:  2.3.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
      (0 0 0)
      (4 0 0)
      (4 2 0)
      (0 2 0)
      (0 0 0.1)
      (4 0 0.1)
      (4 2 0.1)
      (0 2 0.1)
      (0 4 0)
      (4 4 0)
      (0 4 0.1)
      (4 4 0.1)
      (0 20 0)
      (4 20 0)
      (0 20 0.1)
      (4 20 0.1)
      (0 22 0)
      (4 22 0)
      (0 22 0.1)
      (4 22 0.1)
      (0 24 0)
      (4 24 0)
      (0 24 0.1)
      (4 24 0.1)
      (6 0 0)
      (6 2 0)
      (6 0 0.1)
      (6 2 0.1)
      (6 4 0)
      (6 20 0)
      (6 4 0.1)
```

```
(6 20 0.1)
(6 22 0)
(6 24 0)
(6 22 0.1)
(6 24 0.1)
(24 0 0)
(24 2 0)
(24 0 0.1)
(24 2 0.1)
(24 4 0)
(24 4 0.1)
(24 20 0)
(24 20 0.1)
(24 22 0)
(24 22 0.1)
(24 24 0)
(24 24 0.1)
(26 0 0)
(26 2 0)
(26 0 0.1)
(26 2 0.1)
(26 4 0)
(26 20 0)
(26 4 0.1)
(26 20 0.1)
(26 22 0)
(26 24 0)
(26 22 0.1)
(26 24 0.1)
(29 0 0)
(29 2 0)
(29 0 0.1)
(29 2 0.1)
(29 4 0)
(29 4 0.1)
(29 20 0)
(29 20 0.1)
(29 22 0)
(29 22 0.1)
(29 24 0)
(29 24 0.1)
(31 0 0)
(31 2 0)
(31 0 0.1)
(31 2 0.1)
(31 4 0)
(31 20 0)
(31 4 0.1)
(31 20 0.1)
(31 22 0)
(31 24 0)
(31 22 0.1)
(31 24 0.1)
(45 0 0)
(45 2 0)
(45 0 0.1)
(45 2 0.1)
(45 4 0)
```

```
        (45 4 0.1)
        (45 20 0)
        (45 20 0.1)
        (45 22 0)
        (45 22 0.1)
        (45 24 0)
        (45 24 0.1)
        (47 0 0)
        (47 2 0)
        (47 0 0.1)
        (47 2 0.1)
        (47 4 0)
        (47 20 0)
        (47 4 0.1)
        (47 20 0.1)
        (47 22 0)
        (47 24 0)
        (47 22 0.1)
        (47 24 0.1)
        (50 0 0)
        (50 2 0)
        (50 0 0.1)
        (50 2 0.1)
        (50 4 0)
        (50 4 0.1)
        (50 20 0)
        (50 20 0.1)
        (50 22 0)
        (50 22 0.1)
        (50 24 0)
        (50 24 0.1)
);

blocks
(
hex (0 1 2 3 4 5 6 7)(20 10 1) simpleGrading (1 1 1)
hex (3 2 9 8 7 6 11 10)(20 10 1) simpleGrading (1 1 1)
hex (8 9 13 12 10 11 15 14)(20 80 1) simpleGrading (1 1 1)
hex (12 13 17 16 14 15 19 18)(20 10 1) simpleGrading (1 1 1)
hex (16 17 21 20 18 19 23 22)(20 10 1) simpleGrading (1 1 1)
hex (1 24 25 2 5 26 27 6)(10 10 1) simpleGrading (1 1 1)
hex (9 28 29 13 11 30 31 15)(10 80 1) simpleGrading (1 1 1)
hex (17 32 33 21 19 34 35 23)(10 10 1) simpleGrading (1 1 1)
hex (24 36 37 25 26 38 39 27)(90 10 1) simpleGrading (1 1 1)
hex (25 37 40 28 27 39 41 30)(90 10 1) simpleGrading (1 1 1)
hex (28 40 42 29 30 41 43 31)(90 80 1) simpleGrading (1 1 1)
hex (29 42 44 32 31 43 45 34)(90 10 1) simpleGrading (1 1 1)
hex (32 44 46 33 34 45 47 35)(90 10 1) simpleGrading (1 1 1)
hex (36 48 49 37 38 50 51 39)(10 10 1) simpleGrading (1 1 1)
hex (40 52 53 42 41 54 55 43)(10 80 1) simpleGrading (1 1 1)
hex (44 56 57 46 45 58 59 47)(10 10 1) simpleGrading (1 1 1)
hex (48 60 61 49 50 62 63 51)(15 10 1) simpleGrading (1 1 1)
hex (49 61 64 52 51 63 65 54)(15 10 1) simpleGrading (1 1 1)
hex (52 64 66 53 54 65 67 55)(15 80 1) simpleGrading (1 1 1)
hex (53 66 68 56 55 67 69 58)(15 10 1) simpleGrading (1 1 1)
hex (56 68 70 57 58 69 71 59)(15 10 1) simpleGrading (1 1 1)
hex (60 72 73 61 62 74 75 63)(10 10 1) simpleGrading (1 1 1)
hex (64 76 77 66 65 78 79 67)(10 80 1) simpleGrading (1 1 1)
```

```
hex (68 80 81 70 69 82 83 71)(10 10 1) simpleGrading (1 1 1)
hex (72 84 85 73 74 86 87 75)(70 10 1) simpleGrading (1 1 1)
hex (73 85 88 76 75 87 89 78)(70 10 1) simpleGrading (1 1 1)
hex (76 88 90 77 78 89 91 79)(70 80 1) simpleGrading (1 1 1)
hex (77 90 92 80 79 91 93 82)(70 10 1) simpleGrading (1 1 1)
hex (80 92 94 81 82 93 95 83)(70 10 1) simpleGrading (1 1 1)
hex (84 96 97 85 86 98 99 87)(10 10 1) simpleGrading (1 1 1)
hex (88 100 101 90 89 102 103 91)(10 80 1)simpleGrading(1 1 1)
hex (92 104 105 94 93 106 107 95)(10 10 1) simpleGrading (1 1 1)
hex (96 108 109 97 98 110 111 99)(15 10 1) simpleGrading (1 1 1)
hex (97 109 112 100 99 111 113 102)(15 10 1)simpleGrading(1 1 1)
hex (100 112 114 101 102 113 115 103)(15 80 1)simpleGrading(1 1
1)
hex (101 114 116 104 103 115 117 106)(15 10 1)simpleGrading(1 1
1)
hex (104 116 118 105 106 117 119 107)(15 10 1)simpleGrading(1 1
1)
);

edges
(
);

boundary
(
    inlet
    {
        type patch;
        faces
        (
            (0 4 7 3)
        );
    }
    inlet2
    {
        type patch;
        faces
        (
            (0 1 5 4)
        );
    }
    inlet3
    {
        type patch;
        faces
        (
            (1 24 26 5)
        );
    }
    inlet4
    {
        type patch;
        faces
        (
            (8 10 14 12)
        );
    }
     inlet5
```

```
{
    type patch;
    faces
    (
        (24 36 38 26)
    );
}
outlet
{
    type patch;
    faces
    (
        (116 118 119 117)
    );
}
outlet2
{
    type patch;
    faces
    (
        (81 83 95 94)
    );
}
outlet3
{
    type patch;
    faces
    (
        (114 116 117 115)
    );
}
outlet4
{
    type patch;
    faces
    (
        (105 107 119 118)
    );
}
upperWall
{
    type wall;
    faces
    (
        (20 22 23 21)
        (21 23 35 33)
        (33 35 47 46)
        (46 47 59 57)
        (57 59 71 70)
        (70 71 83 81)
        (94 95 107 105)
        (3 7 10 8)
        (12 14 18 16)
        (16 18 22 20)
     );
}
lowerWall
{
```

```
        type wall;
        faces
        (
            (36 48 50 38)
            (48 60 62 50)
            (60 72 74 62)
            (72 84 86 74)
            (84 96 98 86)
            (96 108 110 98)
            (108 109 111 110)
            (109 112 113 111)
            (112 114 115 113)
        );
    }
    frontAndBack
    {
        type empty;
        faces
        (
            (4 5 6 7)
            (0 3 2 1)
            (7 6 11 10)
            (3 8 9 2)
            (10 11 15 14)
            (8 12 13 9)
            (14 15 19 18)
            (12 16 17 13)
            (18 19 23 22)
            (16 20 21 17)
            (1 2 25 24)
            (5 26 27 6)
            (9 13 29 28)
            (11 30 31 15)
            (17 21 33 32)
            (19 34 35 23)
            (24 25 37 36)
            (26 38 39 27)
            (25 28 40 37)
            (27 39 41 30)
            (28 29 42 40)
            (30 41 43 31)
            (29 32 44 42)
            (31 43 45 34)
            (32 33 46 44)
            (34 45 47 35)
            (36 37 49 48)
            (38 50 51 39)
            (40 42 53 52)
            (41 54 55 43)
            (44 46 57 56)
            (45 47 59 58)
            (48 49 61 60)
            (50 62 63 51)
            (49 52 64 61)
            (51 63 65 54)
            (52 53 66 64)
            (54 65 67 55)
            (53 56 68 66)
```

```
        (55 67 69 58)
        (56 57 70 68)
        (58 69 71 59)
        (60 61 73 72)
        (62 74 75 63)
        (64 66 77 76)
        (65 78 79 67)
        (68 70 81 80)
        (69 82 83 71)
        (72 73 85 84)
        (74 86 87 75)
        (73 76 88 85)
        (75 87 89 78)
        (76 77 90 88)
        (78 89 91 79)
        (77 80 92 90)
        (79 91 93 82)
        (80 81 94 92)
        (82 83 95 93)
        (84 85 97 96)
        (86 98 99 87)
        (88 90 101 100)
        (89 102 103 91)
        (92 94 105 104)
        (93 106 107 95)
        (96 97 109 108)
        (98 110 111 99)
        (97 100 112 109)
        (99 111 113 102)
        (100 101 114 112)
        (102 113 115 103)
        (101 104 116 114)
        (103 115 117 106)
        (104 105 118 116)
        (106 117 119 107)
    );
}

box1
{
  type wall;
  faces
  (
        (2 9 11 6)
        (2 6 27 25)
        (9 28 30 11)
        (25 27 30 28)
  );
}

box2
{
  type wall;
  faces
  (
        (13 15 31 29)
        (13 17 19 15)
        (17 32 34 19)
```

```
            (29 31 34 32)
        );
    }

    box3
    {
        type wall;
        faces
        (
            (37 39 51 49)
            (37 40 41 39)
            (40 52 54 41)
            (49 51 54 52)
        );
    }

    box4
    {
        type wall;
        faces
        (
            (42 43 55 53)
            (42 44 45 43)
            (44 56 58 45)
            (53 55 58 56)
        );
    }

     box5
    {
        type wall;
        faces
        (
            (61 63 75 73)
            (61 64 65 63)
            (64 76 78 65)
            (73 75 78 76)
        );
    }

    box6
    {
        type wall;
        faces
        (
            (66 67 79 77)
            (66 68 69 67)
            (68 80 82 69)
            (77 79 82 80)
        );
    }

    box7
    {
        type wall;
        faces
        (
            (85 87 99 97)
```

```
            (85 88 89 87)
            (88 100 102 89)
            (97 99 102 100)
      );
    }

    box8
    {
      type wall;
      faces
      (
            (90 91 103 101)
            (90 92 93 91)
            (92 104 106 93)
            (101 103 106 104)
      );
    }
);

mergePatchPairs
(
);

// ************************************************************************
//
```

Appendix E: controlDict (pisoFoam)

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |
|
| \\      /  F ield         | foam-extend: Open Source CFD
|
|  \\    /   O peration     | Version:     3.1
|
|   \\  /    A nd           | Web:         http://www.extend-
project.de       |
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     pisoFoam;

startFrom       FirstTime;

startTime       0;

stopAt          endTime;

endTime         1600;

deltaT          0.0005;

writeControl    timeStep;

writeInterval   20000;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;
);

// ************************************************************************* //
```

Appendix F: epsilon File (pisoFoam)

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD
Toolbox           |
|  \\    /   O peration     | Version:  1.6
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 14.855;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 14.855;
    }
    inlet2
    {
        type            fixedValue;
        value           uniform 14.855;
    }
    inlet3
    {
        type            fixedValue;
        value           uniform 14.855;
    }
    inlet4
    {
        type            fixedValue;
        value           uniform 14.855;
    }
    inlet5
    {
        type            fixedValue;
        value           uniform 14.855;
    }
    outlet
    {
```

```
    type            inletOutlet;
    inletValue      uniform 14.855;
    value           uniform 14.855;
}
outlet2
{
    type            inletOutlet;
    inletValue      uniform 14.855;
    value           uniform 14.855;
}
outlet3
{
    type            inletOutlet;
    inletValue      uniform 14.855;
    value           uniform 14.855;
}
outlet4
{
    type            inletOutlet;
    inletValue      uniform 14.855;
    value           uniform 14.855;
}
upperWall
{
    type            epsilonWallFunction;
    Cmu             0.09;
    kappa           0.41;
    E               9.8;
    value           uniform 14.855;
}
lowerWall
{
    type            epsilonWallFunction;
    Cmu             0.09;
    kappa           0.41;
    E               9.8;
    value           uniform 14.855;
}
frontAndBack
{
    type            empty;
}
  box1
        {
              type            epsilonWallFunction;
              Cmu             0.09;
              kappa           0.41;
              E               9.8;
              value           uniform 14.855;
        }
  box2
        {
              type            epsilonWallFunction;
              Cmu             0.09;
                    kappa           0.41;
              E               9.8;
              value           uniform 14.855;
        }
```

```
        box3
        {
                type            epsilonWallFunction;
                Cmu             0.09;
                kappa           0.41;
                E               9.8;
                value           uniform 14.855;
        }
        box4
        {
                type            epsilonWallFunction;
                Cmu             0.09;
                kappa           0.41;
                E               9.8;
                value           uniform 14.855;
        }
        box5
        {
                type            epsilonWallFunction;
                Cmu             0.09;
                kappa           0.41;
                E               9.8;
                value           uniform 14.855;
        }
        box6
        {
                type            epsilonWallFunction;
                Cmu             0.09;
                kappa           0.41;
                E               9.8;
                value           uniform 14.855;
        }
        box7
        {
                type            epsilonWallFunction;
                Cmu             0.09;
                kappa           0.41;
                E               9.8;
                value           uniform 14.855;
        }
        box8
        {
                type            epsilonWallFunction;
                Cmu             0.09;
                kappa           0.41;
                E               9.8;
                value           uniform 14.855;
        }
}

// ************************************************************************* //
```

Appendix G: k File (pisoFoam)

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD
Toolbox           |
|  \\    /   O peration     | Version:  1.6
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0.375;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 0.375;
    }
    inlet2
    {
        type            fixedValue;
        value           uniform 0.375;
    }
    inlet3
    {
        type            fixedValue;
        value           uniform 0.375;
    }
     inlet4
    {
        type            fixedValue;
        value           uniform 0.375;
    }
    inlet5
    {
        type            fixedValue;
        value           uniform 0.375;
    }
    outlet
    {
        type            inletOutlet;
```

```
    inletValue      uniform 0.375;
    value           uniform 0.375;
}
outlet2
{
    type            inletOutlet;
    inletValue      uniform 0.375;
    value           uniform 0.375;
}
outlet3
{
    type            inletOutlet;
    inletValue      uniform 0.375;
    value           uniform 0.375;
}
outlet4
{
    type            inletOutlet;
    inletValue      uniform 0.375;
    value           uniform 0.375;
}
upperWall
{
    type            kqRWallFunction;
    value           uniform 0.375;
}
lowerWall
{
    type            kqRWallFunction;
    value           uniform 0.375;
}
frontAndBack
{
    type            empty;
}
  box1
        {
              type            kqRWallFunction;
              value           uniform 0.375;
        }
  box2
        {
              type            kqRWallFunction;
              value           uniform 0.375;
        }
  box3
        {
              type            kqRWallFunction;
              value           uniform 0.375;
        }
  box4
        {
              type            kqRWallFunction;
              value           uniform 0.375;
        }
  box5
        {
              type            kqRWallFunction;
```

```
                value           uniform 0.375;
            }
        box6
            {
                type            kqRWallFunction;
                value           uniform 0.375;
            }
        box7
            {
                type            kqRWallFunction;
                value           uniform 0.375;
            }
        box8
            {
                type            kqRWallFunction;
                value           uniform 0.375;
            }
}

// ************************************************************************* //
```

Appendix H: nut File (pisoFoam)

```
/*---------------------------*- C++ -*---------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD
Toolbox           |
|  \\    /   O peration     | Version:  1.6
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*-------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            calculated;
        value           uniform 0;
    }
    inlet2
    {
        type            calculated;
        value           uniform 0;
    }
    inlet3
    {
        type            calculated;
        value           uniform 0;
    }
    inlet4
    {
        type            calculated;
        value           uniform 0;
    }
    inlet5
    {
        type            calculated;
        value           uniform 0;
    }
    outlet
    {
        type            calculated;
```

```
    value           uniform 0;
}
outlet2
{
    type            calculated;
    value           uniform 0;
}
outlet3
{
    type            calculated;
    value           uniform 0;
}
outlet4
{
    type            calculated;
    value           uniform 0;
}
upperWall
{
    type            nutkWallFunction;
    Cmu             0.09;
    kappa           0.41;
    E               9.8;
    value           uniform 0;
}
lowerWall
{
    type            nutkWallFunction;
    Cmu             0.09;
    kappa           0.41;
    E               9.8;
    value           uniform 0;
}
frontAndBack
{
    type            empty;
}
  box1
        {
              type            nutkWallFunction;
              Cmu             0.09;
              kappa           0.41;
              E               9.8;
              value           uniform 0;
        }
  box2
        {
              type            nutkWallFunction;
              Cmu             0.09;
              kappa           0.41;
              E               9.8;
              value           uniform 0;
        }
  box3
        {
              type            nutkWallFunction;
              Cmu             0.09;
              kappa           0.41;
```

```
                E               9.8;
                value           uniform 0;
            }
        box4
            {
                type            nutkWallFunction;
                Cmu             0.09;
                kappa           0.41;
                E               9.8;
                value           uniform 0;
            }
        box5
            {
                type            nutkWallFunction;
                Cmu             0.09;
                kappa           0.41;
                E               9.8;
                value           uniform 0;
            }
        box6
            {
                type            nutkWallFunction;
                Cmu             0.09;
                kappa           0.41;
                E               9.8;
                value           uniform 0;
            }
        box7
            {
                type            nutkWallFunction;
                Cmu             0.09;
                kappa           0.41;
                E               9.8;
                value           uniform 0;
            }
        box8
            {
                type            nutkWallFunction;
                Cmu             0.09;
                kappa           0.41;
                E               9.8;
                value           uniform 0;
            }
}

// ************************************************************************* //
```

Appendix I: nuTilda File (pisoFoam)

```
/*--------------------------------*- C++ -*----------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD
Toolbox           |
|  \\    /   O peration     | Version:  2.3.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      nuTilda;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
   inlet
    {
        type            fixedValue;
        value           uniform 14.855;
    }
    inlet2
    {
        type            fixedValue;
        value           uniform 14.855;
    }
    inlet3
    {
        type            fixedValue;
        value           uniform 14.855;
    }
    inlet4
    {
        type            fixedValue;
        value           uniform 14.855;
    }
    inlet5
    {
        type            fixedValue;
        value           uniform 14.855;
    }
    outlet
    {
        type           zeroGradient;
    }
    outlet2
```

```
    {
        type            zeroGradient;
    }
    outlet3
    {
        type            zeroGradient;
    }
    outlet4
    {
        type            zeroGradient;
    }
    upperWall
    {
        type            zeroGradient;
    }
    lowerWall
    {
        type             zeroGradient;
    }
    frontAndBack
    {
        type             empty;
    }
      box1
            {
                  type            zeroGradient;
            }
      box2
            {
                  type            zeroGradient;
            }
      box3
            {
                  type            zeroGradient;
            }
      box4
            {
                  type            zeroGradient;
            }
      box5
            {
                  type            zeroGradient;
            }
      box6
            {
                  type            zeroGradient;
            }
      box7
            {
                  type            zeroGradient;
            }
      box8
            {
                  type            zeroGradient;
            }

}
// ************************************************************* //
```

Appendix J: p File (pisoFoam)

```
/*--------------------------------*- C++ -*----------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD
Toolbox           |
|  \\    /   O peration     | Version:  1.6
|
|   \\  /    A nd           | Web:      http://www.openfoam.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }
    inlet2
    {
        type            zeroGradient;
    }
    inlet3
    {
        type            zeroGradient;
    }
    inlet4
    {
        type            zeroGradient;
    }
    inlet5
    {
        type            zeroGradient;
    }
    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    outlet2
    {
        type            fixedValue;
        value           uniform 0;
```

```
    }
    outlet3
    {
        type            fixedValue;
        value           uniform 0;
    }
    outlet4
    {
        type            fixedValue;
        value           uniform 0;
    }
    upperWall
    {
        type            zeroGradient;
    }
    lowerWall
    {
        type            zeroGradient;
    }
    frontAndBack
    {
        type            empty;
    }
      box1
            {
                  type            zeroGradient;
            }
      box2
            {
                  type            zeroGradient;
            }
      box3
            {
                  type            zeroGradient;
            }
      box4
            {
                  type            zeroGradient;
            }
      box5
            {
                  type            zeroGradient;
            }
      box6
            {
                  type            zeroGradient;
            }
      box7
            {
                  type            zeroGradient;
            }
      box8
            {
                  type            zeroGradient;
            }
}

// ************************************************************* //
```

Appendix K: U File (pisoFoam)

```
/*--------------------------*- C++ -*----------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD
Toolbox           |
|  \\    /   O peration     | Version:  1.6
|
|   \\  /    A nd           | Web:      http://www.openfoam.org
|
|    \\/     M anipulation  |
|
\*-------------------------------------------------------------*/

FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object U;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (0.0642 0.0766 0);
    }
    inlet2
    {
        type            fixedValue;
        value           uniform (0.0642 0.0766 0);
    }
    inlet3
    {
        type            fixedValue;
        value           uniform (0.01736 0.09848 0);
    }
    inlet4
    {
        type            fixedValue;
        value           uniform (0.001 0 0);
    }
    inlet5
    {
        type            fixedValue;
        value           uniform (0.0689 0.0121 0);
    }
    outlet
```

```
{
    type            inletOutlet;
    inletValue      uniform (0 0 0);
    value           uniform (0 0 0);
}
outlet2
{
    type            inletOutlet;
    inletValue      uniform (0 0 0);
    value           uniform (0 0 0);
}
outlet3
{
    type            inletOutlet;
    inletValue      uniform (0 0 0);
    value           uniform (0 0 0);
}
outlet4
{
    type            inletOutlet;
    inletValue      uniform (0 0 0);
    value           uniform (0 0 0);
}
upperWall
{
    type            fixedValue;
    value           uniform (0 0 0);
}
lowerWall
{
    type            fixedValue;
    value           uniform (0 0 0);
}
frontAndBack
{
    type            empty;
}
  box1
        {
              type            fixedValue;
              value           uniform (0 0 0);
        }
  box2
        {
              type            fixedValue;
              value           uniform (0 0 0);
        }
  box3
        {
              type            fixedValue;
              value           uniform (0 0 0);
        }
  box4
        {
              type            fixedValue;
              value           uniform (0 0 0);
        }
  box5
```

```
        {
            type            fixedValue;
            value           uniform (0 0 0);
        }
    box6
        {
            type            fixedValue;
            value           uniform (0 0 0);
        }
    box7
        {
            type            fixedValue;
            value           uniform (0 0 0);
        }
    box8
        {
            type            fixedValue;
            value           uniform (0 0 0);
        }
}

// ************************************************************************* //
```

Appendix L: fvSchemes File (pisoFoam)

```
/*---------------------------------------------------------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD
Toolbox           |
|  \\    /   O peration     | Version:  1.6
|
|   \\  /    A nd           | Web:      http://www.openfoam.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/

FoamFile
{
    version         2.0;
    format          ascii;
    class           dictionary;
    location        "system";
    object          fvSchemes;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default Euler;
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}

divSchemes
{
    default         none;


    div(phi,U)      Gauss upwind;
    div(phi,k)      Gauss upwind;
    div(phi,epsilon) Gauss upwind;
    div(phi,R)      Gauss upwind;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss upwind;

       div((nuEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         none;
    laplacian(nuEff,U) Gauss linear corrected;
    laplacian((1|A(U)),p) Gauss linear corrected;
```

```
    laplacian(DkEff,k) Gauss linear corrected;
    laplacian(DepsilonEff,epsilon) Gauss linear corrected;
    laplacian(DREff,R) Gauss linear corrected;
    laplacian(DnuTildaEff,nuTilda) Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
    interpolate(U)  linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p;
}


// ************************************************************* //
```

Appendix M: fvSolution File (pisoFoam)

```
/*---------------------------------------------------------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD
Toolbox           |
|  \\    /   O peration     | Version:  1.6
|
|   \\  /    A nd           | Web:      http://www.openfoam.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/

FoamFile
{
    version         2.0;
    format          ascii;
    class           dictionary;
    location        "system";
    object          fvSolution;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
      p
    {
        solver           PCG;
        preconditioner   DIC;
        tolerance        1e-06;
        relTol           0.01;
    };

      pFinal
    {
        $p;
        tolerance       1e-06;
        relTol          0;
    }


      U
    {
        solver           PBiCG;
        preconditioner   DILU;
        tolerance        1e-05;
        relTol           0;
    };

      k
    {
        solver           PBiCG;
        preconditioner   DILU;
        tolerance        1e-05;
        relTol           0;
```

```
    };
      epsilon
    {
        solver           PBiCG;
        preconditioner   DILU;
        tolerance        1e-05;
        relTol           0;
    };
}

PISO
{
    nCorrectors 2;
    nNonOrthogonalCorrectors 0;
}

/*
relaxationFactors
{
    p               0.3;
    U               0.7;
    k               0.7;
    epsilon         0.7;
    R               0.7;
    nuTilda         0.7;
}
*/

// ************************************************************* //
```

Appendix N: controlDictFile (scalarTransportFoam)

```
/*--------------------------*- C++ -*----------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD
Toolbox           |
|  \\    /   O peration     | Version:  2.3.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     scalarTransportFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         2000;

deltaT          1;

writeControl    timeStep;

writeInterval   60;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;


functions
(
    probes
```

```
    {
        type            probes;
        functionObjectLibs ("libsampling.so");
        enabled         true;
        outputControl   timeStep;
        outputInterval  10;
        probeLocations
        (


            (8 6 0) //sensor 1
            (8 18 0) //sensor 2
            (25 6 0) //sensor 3
            (25 18 0) //sensor 4
            (41 6 0) //sensor 5
            (41 18 0) //sensor6

        );

        fields
        (
            T U
        );
    }

    fieldAverage1
    {
        type            fieldAverage;
        functionObjectLibs ("libfieldFunctionObjects.so");
        enabled         true;
        outputControl   outputTime;
        fields
        (
            U
            {
                mean        on;
                prime2Mean  on;
                base        time;
            }

            p
            {
                mean        on;
                prime2Mean  on;
                base        time;
            }
        );
    }
);

//************************************************************* //
```

Appendix O: T File (scalarTransportFoam)

```
/*--------------------------*- C++ -*----------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD
Toolbox           |
|  \\    /   O peration     | Version:  2.3.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 1 0 0];

internalField   nonuniform List<scalar>

29200
(
0
0
0
0
0
0
0
0
.
.
.
.
.
.
.
.
.
.
.
.
0
0
0
0
0
0
0
0
0
```

```
0
0

)
;

boundaryField
{
    inlet
    {
       type            zeroGradient;
    }
inlet2
    {
       type            zeroGradient;
    }
inlet3
    {
        type            zeroGradient;
    }
inlet4
    {
        type            zeroGradient;
    }
inlet5
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }
 outlet2
    {
        type            zeroGradient;
    }
 outlet3
    {
        type            zeroGradient;
    }
 outlet4
    {
        type            zeroGradient;
    }

    upperWall
    {
        type            zeroGradient;
    }

    lowerWall
    {
        type            zeroGradient;
    }

    frontAndBack
    {
```

```
        type            empty;
    }

      box1
    {
        type            zeroGradient;
    }
    box2
    {
        type            zeroGradient;
    }
    box3
    {
        type            zeroGradient;
    }
    box4
    {
        type            zeroGradient;
    }
    box5
    {
       type            zeroGradient;
    }
    box6
    {
        type            zeroGradient;
    }
    box7
    {
        type            zeroGradient;
    }
    box8
    {
        type            zeroGradient;
    }
}

// ************************************************************* //
```

Appendix P: U File (scalarTransportFoam)

```
/*---------------------------*- C++ -*---------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD
Toolbox           |
|  \\    /   O peration     | Version:  2.3.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*-------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   nonuniform List<vector>
29200
(
(0.0640453 0.0767247 0)
(0.0636308 0.0767734 0)
(0.0630427 0.0767731 0)
(0.0624811 0.0767581 0)
(0.0619399 0.0767384 0)
(0.0614639 0.0767222 0)
(0.0610252 0.0767085 0)
(0.0606287 0.0767 0)
(0.0602542 0.0766955 0)
.
.
.
.
.
.
.
.
//All data points can be seen in enclosed CD.
.
.
.
.
.
.
.
.
(0.0274134 0.0217885 0)
(0.0227766 0.0279405 0)
(0.0196123 0.0341081 0)
(0.0184627 0.0406624 0)
```

```
(0.0184526 0.0468388 0)
(0.0187847 0.0518495 0)
(0.0193677 0.0557131 0)
(0.0198265 0.0582856 0)
(0.0204271 0.0601698 0)
(0.0206874 0.0609575 0)
(0.0213007 0.0616511 0)


)
;


boundaryField
{
    inlet
    {
       type            zeroGradient;
    }
inlet2
    {
        type            zeroGradient;
    }
inlet3
    {
        type            zeroGradient;
    }
inlet4
    {
       type            zeroGradient;
    }
inlet5
    {
        type            zeroGradient;
    }
outlet
    {
        type            zeroGradient;
    }
outlet2
    {
        type            zeroGradient;
    }
outlet3
    {
        type            zeroGradient;
    }
outlet4
    {
        type            zeroGradient;
    }
upperWall
    {
        type            zeroGradient;
    }
lowerWall
    {
       type            zeroGradient;
```

```
    }
frontAndBack
    {
        type            empty;
    }
box1
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
box2
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
box3
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
box4
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
box5
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
box6
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
box7
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
box8
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
}

// ************************************************************************* //
```

Appendix Q: fvSchemes File (scalarTransportFoam)

```
/*---------------------------*- C++ -*---------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD
Toolbox           |
|  \\    /   O peration     | Version:  2.3.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*-------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}
gradSchemes
{
    default         Gauss linear;
}
divSchemes
{
    default         none;
    div(phi,T)      Gauss linearUpwind grad(T);
}
laplacianSchemes
{
    default         none;
    laplacian(DT,T) Gauss linear corrected;
}
interpolationSchemes
{
    default         linear;
}
snGradSchemes
{
    default         corrected;
}
fluxRequired
{
    default         no;
    T               ;
}


// ************************************************************* //
```

Appendix R: fvSolution File (scalarTransportFoam)

```
/*---------------------------*- C++ -*---------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD
Toolbox           |
|  \\    /   O peration     | Version:  2.3.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*-------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    T
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-06;
        relTol          0;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;
}


// ************************************************************************* //
```

Appendix S: fvOptions File (scalarTransportFoam)

```
/*--------------------------*- C++ -*----------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD
Toolbox           |
|  \\    /   O peration     | Version:  2.3.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvOptions;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

scalarTracer          //this is only the name of the object
{
   type                   scalarSemiImplicitSource;
   active                 true;     //turns on and off the source
   timeStart              0;        //starting time
   duration               12000;        //duration in seconds
   selectionMode          points;   //it could be also a cellZone
   //selectionMode        cellSet

   points              //list of coordinates of the source terms
   (

(20 5 0) //source term at 20 50 0 meters in x y z axis from
reference point (0,0,0)

   );

   scalarSemiImplicitSourceCoeffs
   {
      volumeMode            absolute;
//     - absolute: values are given as \<quantity\>
//     - specific: values are given as \<quantity\>/m3
      injectionRateSuSp
      {
         T         (2e-9 0); // T - scalar variable.
//First value = S_u (explicit source term); Second value = S_p.
      }
    }
}

// ************************************************************* //
```

Appendix T: Content of enclosed CD

The enclosed CD consists of three different folders. These are:

1.) PDF version of the thesis.
2.) CFD case file of both pisoFoam and scalarTransportFoam - This includes subdirectories 0, constant and system.
3.) ANN case file - This consists of all ANN models and data, which are used for training and testing. All models in this folder can be opened and the outputs can be manipulated in Matlab (application).