Integrity Assessment of Dents in Pipelines using Finite Element Analysis and Artificial Neural Networks

by

Janine Woo

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Structural Engineering

Department of Civil and Environmental Engineering

University of Alberta

ABSTRACT

Dents are common occurrences along oil and gas pipelines and can be formed due to pipe contact with external forces such as rocks or construction equipment. There are many factors that contribute to the level of integrity concern of a dent including its shape, size, location on the pipe, and proximity to other features, which has made it difficult for industry agreement on a single, consistent, accurate, and efficient dent assessment method. Current regulations for dents, which are based on depth and interaction with stress risers, have been proven to have limitations by other researchers and have been shown to be too conservative in some cases while failing to identify concern in others. A method is proposed to assess dents herein, which uses the results from finite element analysis (FEA) to train artificial neural networks (ANNs) and the trained ANNs can then be used to assess real-life dent features reported by in-line inspection (ILI) tools.

First, a methodology to use FEA to model dents in pipelines was developed and validated against full-scale tests published by another researcher. The effect of using different element properties was investigated and it was concluded that the use of S4R elements (shell, linear elements with reduced integration) and five integration points in the thickness direction was most suitable for this problem. Automation techniques were developed to make the FEA model generation and extraction of results more efficient and ensure the methodology was performed consistently.

Precision of the dent profiles achieved by FEA compared to the profiles reported by ILI tools was found to be important to attain accurate stress and strain results within a dented region. In order to achieve an accurate profile match, a trial and error method was proposed but was found to be time-consuming. It was determined that a ring torus is an appropriate shape to use as an indenter in the FEA to model smooth, symmetric dent profiles and that two-dimensional profiles aligned with the longitudinal and circumferential axes through the most significant, or deepest, point of the dent, can be used as a simplified representation of the full three-dimensional shape of the dent.

Lastly, ANNs were used to produce results as accurate as could be found using FEA but in significantly less time. Given only the dent profile information and basic pipe properties, the ANNs were able to accurately output stress and strain results. This thesis demonstrates the feasibility of using FEA in conjunction with ANNs to provide an accurate and efficient method for assessing dents.

PREFACE

This thesis is an original work by Janine Woo.

Chapter 5 of this thesis has been previously published as:

1. Woo, J., Kainat, M., and Adeeb, S. (2017). Proceedings of the ASME 2017 Pressure Vessels & Piping Conference: *Development of a Profile Matching Criteria to Model Dents in Pipelines using Finite Element Analysis.* Waikoloa, Hawaii: ASME.

For the above work, I was responsible for the data collection, analysis, and manuscript composition. M. Kainat and S. Adeeb assisted in concept formation, provided technical support, and reviewed the manuscript, while S. Adeeb was also the supervisory author.

The above work was awarded “Outstanding Student Paper” of the 25th Annual Rudy Scavuzzo Student Paper Competition during the 2017 Pressure Vessels & Piping (PVP) Conference.

ACKNOWLEDGEMENTS

I would first like to thank my supervisor, Samer Adeeb, for his continued support and guidance throughout my research.

This research was funded partially by Enbridge Pipelines, Inc. and I'd like to thank Enbridge for the support. I would like to acknowledge Sherif Hassanien, who encouraged me to pursue my Master's degree and provided direction during my research. Muntaseer Kainat and Doug Langer provided invaluable technical assistance and advice that was greatly appreciated.

Lastly, I would like to thank my family, friends, and partner, Sam Rendulich, for their love and encouragement during my years of study. This achievement would not have been possible without them. I would especially like to thank my Mom for always believing in me and supporting me.

TABLE OF CONTENTS

Table of Figures

Table of Tables

List of Abbreviations

ANN	Artificial Neural Network
ASME	American Society of Mechanical Engineers
CSA	Canadian Standards Association
FEA	Finite Element Analysis
ILI	In-Line Inspection
MOP	Maximum Operating Pressure
MSE	Mean Squared Error
MSP	Most Significant Point
NEB	National Energy Board
OD	Outer Diameter
PHMSA	Pipeline and Hazardous Materials Safety Administration
SMYS	Specified Minimum Yield Strength

List of Symbols

$\overline{\varepsilon^p}$	Equivalent plastic strain, measure of total history of plastic deformation
$\Delta\sigma_{axial}$	Change in the axial stress component of the stress tensor due to one pressure cycle
$\Delta\sigma_{hoop}$	Change in the hoop stress component of the stress tensor due to one pressure cycle
$\Delta\sigma_{max}$	Maximum of $\Delta\sigma_{axial}$ and $\Delta\sigma_{hoop}$
R^2	R-Squared, statistical measure to evaluate goodness of fit of a data set

CHAPTER 1: INTRODUCTION

1.1 Background

Pipelines offer an efficient, safe, and environmentally friendly way through which oil and gas can be transported throughout the world. Oil and gas delivered by pipelines better the everyday lives of citizens by providing fuel to heat homes, power vehicles, and provide energy for manufacturing processes. As the majority of pipelines are buried underground, threats to the structural integrity of a pipeline can go undetected. If left unmitigated for long enough, pipeline failure could result, which has the potential to impact public safety as well as the environment. Pipeline integrity management systems exist under pipeline owners and operators to ensure the safety and reliability of pipelines throughout conception, engineering and design, construction, operation, inspection, and repair or replacement, when necessary. Pipeline operators must follow regulations put forth by the regulating entity in their area, such as the Pipeline and Hazardous Materials Safety Administration (PHMSA) in the United States, and the National Energy Board (NEB) in Canada.

Potential threats to the integrity of a pipeline include metal loss, cracking, dents, or the interaction of any of these. Dents are permanent inward plastic deformations localized in the pipe wall. Plain dents are defined as dents that do not contain any features that are stress rising, such as metal loss, cracks, or welds (Dawson, Russell, & Patterson, 2006), while the depth of a dent is defined as the gap between the lowest point of the dent and the original contour of the pipe (CSA, 2015). Formation of a dent can occur due to external impact, such as strike by construction equipment or settlement over rocks (Kiefner & Leewis, 2011). The dent could further cause coating damage and accelerate the potential for or growth of existing corrosion or cause the pipe to become more susceptible to cracking in the deformed area (Hassanien & Langer, 2018).

In-line inspection (ILI) tools can take readings of the inner diameter of the pipe and indicate the location, shape, and size of dents. Caliper ILI tools, which indicate the presence of dents, can indicate interaction with other dents or with welds, while overlaying caliper data with data from other tools can indicate interaction between dents and metal loss. In a study completed by an ILI vendor, dents were found to be common occurrences in pipelines. Out of the pipeline miles that were inspected, at least 80% of the

miles had one or more dents found in them and more than 50% of the pipelines contained 10 or more dents (Dawson et al., 2006).

Current industry regulations require repair of dents primarily based on depth, while there is minimal industry agreement on a suitable fitness-for-purpose approach for dents. The Canadian Standards Association (CSA) Standard Z662 (2015) specifies that plain dents greater than 6% of the nominal pipe diameter should be excavated and repaired. CSA also recommends the repair of dents interacting with welds and dents interacting with stress concentrators (such as corrosion, stress corrosion cracks, other cracks, or gouges), although the standard does not provide guidance as to what is defined as interaction. Dents associated with stress concentrators have a greater potential to form and propagate cracks under cyclic pressurization while a pipeline is in operation, which is why they are of greater concern than plain dents (CSA, 2015).

There is room for improvement of the dent repair regulations as there have been instances in the past where plain dents smaller than 6% of the outer diameter (OD) in depth, thereby acceptable per the CSA criteria, have failed (BMT Fleet Technology, 2012). In 2009, two separate dent failures occurred; one dent was 0.51% of OD and the other was 2.7% of OD (National Energy Board, 2010). After failure investigation, it was determined that each dent had led to the initiation of a crack, which then propagated due to pressure cycle induced fatigue until the crack reached through-wall and the pipe leaked. The failures prompted the NEB to issue an advisory in 2010 pertaining to shallow dents on pipelines. The advisory suggested that operators should improve their dent assessment procedures and consider the safety of all dent features, regardless of depth (National Energy Board, 2010). On the other hand, many dents are excavated and found to be non-injurious, with each excavation requiring a significant amount of resources to complete. Therefore, advanced dent assessment methods should aid in optimizing dig selection to ensure that pipeline safety is maintained while operators' resources are also allocated wisely (Dawson et al., 2006).

Recently, industry focus has moved towards additional criteria for the integrity assessment of dents beyond what is suggested by regulations, including strain-based models (Noronha, Martins, Jacob, & Souza, 2010; Ramezani, 2013) and fatigue-life assessment based on stresses associated with pressurization on the dented region (Rosenfeld, 1999; Dawson et al., 2006; Tiku, Semiga, Dinovitzer, &

Vignal, 2012; Kainat et al., 2019). An option for using strain-based criteria for dent assessment was proposed in ASME B31.8, which provides non-mandatory analytical equations to predict the maximum strain in dents (The American Society of Mechanical Engineers, 2010). More recently, finite element analysis (FEA) has been proposed in literature as an accurate technique with which to evaluate dents (Tiku et al., 2012; Arumugam, Gao, Krishnamurthy, Wang, & Kania, 2016; Hassanien, Kainat, Adeeb, & Langer, 2016; Turnquist & Smith, 2016).

Due to the many different variables that affect the severity of a dent (including size, shape, interacting features, and the pipe's operating conditions), a universal equation or model to assess the integrity of a dent does not exist in industry (Hassanien et al., 2016). Although FEA has proven accurate in providing the stresses and strains within a dented region, FEA is inefficient for analyzing a large number of dents as it is computationally expensive (Hassanien & Langer, 2018).

A method that can handle a complex problem with a large number of inputs, such as this, is an artificial neural network (ANN). ANNs imitate the architecture and concept of neural networks in the human brain by learning from samples of data to detect complex relationships, where a simple formula cannot be used to connect inputs and outputs (Layouni, Hamdi, & Tahar, 2016; Xu, Chun, Joonmo, & Lee, 2017). The database of samples used to train the neural network can be based on numerical analyses.

1.2 Objective of Thesis

The main objective of this thesis is to work towards an improved assessment methodology of the integrity of dented pipe. The research evaluates the use of finite element analysis to model dents reported by in-line inspection tools and assess their severity. In addition, the research considers the use of artificial neural networks (ANN) to predict the stress-strain state of dents using results from FEA and ultimately bypass the need to develop the time-intensive models.

The first objective is to develop a procedure to model indentations in pipelines using FEA, which includes determining specific properties of the model that would lead to obtaining consistent and accurate results in an efficient manner. An appropriate mesh configuration for the dent models is investigated and the general procedure is validated against experimental results.

The second objective is to describe how automation can be used with FEA to create an efficient dent analysis process. Several models are required for the trial and error process of matching the FEA profile to the ILI, and an even larger number is required to create a comprehensive ANN. The generation of these models manually would be highly time-consuming and are susceptible to human error. The automation of the FEA model generation and results extraction steps will save an immense amount of time in this research and allow for consistently developed results.

The third objective is to develop a method to match the dent profiles obtained from FEA to the dent profiles reported by ILI tools. In current uses of FEA to model dents in literature, the dents being modelled are typically hypothetical or experimental. However, dents in operating pipelines may have various sizes and shapes. Although the complex geometries often found in real-life can be modelled using FEA, there is minimal guidance in literature regarding how to ensure the FEA model aligns with what was reported by ILI. Another objective is to investigate a criterion with which to judge if the alignment between the FEA dent profile and ILI dent profile is sufficient for an accurate assessment.

The fourth objective is to use artificial neural networks to estimate dent severity. The ANNs would take the ILI dent profile and pipe properties as inputs and output the maximum stresses and strains within the dent. This would remove the need for modelling each dent with FEA but harness the accurate results that can be obtained from it. The development of the ANNs would utilize the automated FEA dent model generation procedure resulting from the first two objectives as well as the profile matching logic from the third objective. The ANNs will be limited to a small subset of pipe properties and dent shapes, however, this research is intended to prove the capability of the ANNs to go from ILI profile to stresses and strains and further research can expand on the included properties.

1.3 Organization of Thesis

This thesis is organized into five chapters:

Chapter 1 describes the background of pipeline integrity and dents on pipelines.

Chapter 2 is a comprehensive literature review of historical dent failures and existing methods for dent assessment, which includes analytical and numerical methods. Proposed finite element analysis

procedures for modelling dents will also be reviewed, as well as areas requiring additional research from the current methods. Literature describing background of artificial neural networks will also be discussed.

Chapter 3 presents the general FEA methodology that will be used throughout this research, as well as the methodology and results of the mesh configuration studies, including the identification of an optimal mesh size in the indented region, an applicable element type, and the number of thickness integration points that should be used in this research for accuracy and efficiency. In addition, the procedure will be validated against experimental results.

Chapter 4 presents the methodology of automating the generation of dent FEA models and extraction of results. The automation of each step of the process will be described so that human intervention would not be required for the analysis of a large number of dent features.

Chapter 5 presents the methodology and results of modifying the indenter shape and size, and other parameters to produce a FEA dent profile that is in close alignment with a dent profile reported by ILI. Several examples are presented, including the comparison of 2-dimensional profiles and 3-dimensional surfaces to investigate suitable profile matching criteria. Chapter 5 is from a paper that was previously published in the proceedings of the 2017 ASME Pressure Vessels and Piping Conference.

Chapter 6 presents the methodology and results of the use of artificial neural networks to obtain the stresses and strains that would be determined using FEA with only the ILI profile and pipe properties as inputs. Validation of the ANN and comparison to existing dent assessment methods is also presented.

Chapter 7 summarizes the key findings through this research and recommends future work.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction of Dents and Definitions

Dents are permanent plastic deformations that cause a gross disturbance in the curvature of a normally circular cross-section of pipe (Kiefner & Leewis, 2011). Dent depth is treated as the difference between the maximum reduction in diameter and the original diameter of the pipe (Kiefner & Leewis, 2011). This depth includes both local indentation and any change in the overall cross-section (Cosham & Hopkins, 2004).

A dent can either be constrained or unconstrained. A constrained dent, which can be formed during operation or while the pipe is in-service, occurs when the indenter (such as a rock) remains in contact with the pipe and prevents the dented region from further movement due to internal pressure (Cosham & Hopkins, 2004). An unconstrained dent is formed when the object that indented the pipe is removed sometime afterwards, allowing the pipe to flex and push out due to the internal pressure. Unconstrained dents can be formed during contruction or operation, and can be formed by a rock indenter that is subsequently exposed and removed, or a strike by an object such as construction equipment (Cosham & Hopkins, 2004).

Mechanical damage includes both dents and gouges, where dents consist solely of radial displacement of the pipe and gouges consist of mechanical removal or displacement of metal that can form a deformed, work-hardened surface layer (Canadian Standards Association Group, 2015). Metal loss refers to an area of pipe wall that has a measurable reduction in thickness and can be caused by corrosion, gouges, and/or cracks (Gao & Krishnamurthy, 2015). A plain dent is defined as a dent containing no wall thickness reductions (such as corrosion, a gouge or a crack), or other defects or imperfections (such as a girth weld) (Kiefner & Leewis, 2011).

A dent creates a local reduction in the pipe diameter, which also leads to local stress and strain concentraion in the pipe (Cosham & Hopkins, 2004). While the stress and strain distributions vary between dents based on the dent length and width, studies have shown that the maximum stress and strain can be found at the peak of the dent in long dents, and at the flanks of the dent in short dents (Cosham & Hopkins, 2004).

2.2 History of Pipeline Dents

Studies have shown that mechanical damage is one of the major causes of pipeline failures. Between 1985 and 2003 in the United States, mechanical damage was the cause of 21% of the total immediate incidents on liquid petroleum pipelines (Gao & Krishnamurthy, 2015). In the same study, it was found that mechanical damage was the cause of 153 delayed incidents, which was 5% of the total number of delayed incidents on liquid pipelines. Mechanical damage that was not severe enough to cause immediate failure could result in delayed failure if corrosion or cracking develops within the dented material, if internal pressure is increased sufficiently, or if the deformed pipe reaches the end of its fatigue life due to pressure cycling (Gao & Krishnamurthy, 2015). This thesis will focus on the mitigation of dents causing delayed failures. Dents causing immediate failure (such as a third-party strike that causes cracking during contact) would not be reported by ILI tools and would be outside of the intended scope of the methodologies presented in this thesis.

2.3 Existing Dent Assessment Methods

While extensive industry efforts have focused on a suitable integrity assessment method for pipeline dents, a single, agreed-upon analytical or empirical model does not currently exist (Hassanien et al., 2016). This lack of an industry-wide model can be attributed to several factors such as the unique nature of dent sizes and shapes, and the large number of variables associated with dents (i.e. pressure cycling, restraint condition, age, size, shape, etc.) that contribute to accuracy limitations in any one model (Hassanien et al., 2016). Even rigorous methods to assess a single dent (such as FEA) are not suitable for system-wide application due to the large amount of time required for each dent assessment (Hassanien & Langer, 2018).

2.3.1 Regulatory Requirements

The current regulatory requirements for repairing mechanical damage in both liquid and gas pipelines are solely based on two parameters: the nature of the mechanical damage (i.e. plain dents, dents with gouges, cracks or welds, etc.), and the depth of the dent, expressed as a percentage of the outer diameter of the pipe (Gao & Krishnamurthy, 2015).

In the United States, the federal regulations for liquid and gas pipelines state that the following anomalies must be excavated (Gao & Krishnamurthy, 2015):

- A dent that has any indication of metal loss, cracking, or a stress riser
- A dent with depth >6% of the nominal outer diameter (OD) of the pipe
- A dent with depth >3% OD on the upper 1/3 of the pipe
- A dent with depth >2% OD on the upper 2/3 of the pipe
- A dent with depth >2% OD that affects the pipe curvature at a girth weld or longitudinal weld

In Canada, the regulations stated in CSA Z662 are similar in nature to the United States regulations, as the following anomalies must be repaired (Gao & Krishnamurthy, 2015):

- A dent associated with a crack or gouge
- A dent with depth >6% OD
- A dent with depth >2% OD at a weld
- A dent with a corrosion feature that does not meet the criteria described in ASME B31.G (The American Society of Mechanical Engineers, 2012)

2.3.2 Strain-Based Methods

Literature has shown that strain-based models may be more effective than existing depth-based criteria to assess the integrity of a dented pipe (Gao & Krishnamurthy, 2015; Arumugam et al., 2016; Turnquist & Smith, 2016). If the strains in the dented region exceed the strain limit of the pipe, the excessive local deformation can lead to immediate crack formation on the internal surface of the pipe (Gao & Krishnamurthy, 2015). Once a crack has formed within a dent, it will generally grow to critical size quite quickly due to the elevated strain resulting from the deformed dent shape, leading to failure (Turnquist & Smith, 2016). Dents far shallower than the 6% OD limit allowed by regulations have been known to fail in the past, indicating that dent assessment should not be solely based on depth (National Energy Board, 2010). For these reasons, strain-based methods could provide more holistic and accurate assessment of dent severity than depth-based criteria.

In recent years, there have been numerous proposed techniques to predict dent strains. Although not required by standards, equations have been provided in ASME B31.8 to use data provided by ILI tools

to estimate the strains in dents (The American Society of Mechanical Engineers, 2010). The equations consider the general dent profile by taking the radius of curvature, length, and depth as input variables. The complete dent profile is not accounted for, nor are other factors such as the material grade of the pipe, loading history, or pressure cycling history of the dented region.

Limitations of the ASME B31.8 strain equations have been identified by other researchers. The bending strains predicted from the equations are highly sensitive in the region nearest the dent apex, which could result in large errors in strain estimates if a low-resolution caliper tool is used (Noronha et al., 2010). In addition, the estimation of global longitudinal membrane strain is inaccurate, particularly considering that no definition of length was provided by ASME B31.8 and the longitudinal strain is highly dependent on length (Lukasiewicz, Czyz, Sun, & Adeeb, 2006; Noronha et al., 2010). The ASME B31.8 equations also neglect circumferential membrane strains and shear strains, which have been shown to be significant, according to FEA models of real-life dents (Gao & Krishnamurthy, 2015). Additional analytical strain estimation methods have been proposed that intended to improve on the aforementioned shortcomings of the ASME B31.8 equations (Lukasiewicz et al., 2006; Noronha et al., 2010; Gao & Krishnamurthy, 2015; Okoloekwe, 2017). The methods were validated using FEA and were proven to produce similar results to FEA while requiring far less time and resources. Okoloekwe (2017) presented a method of interpolating a dented surface using spline functions to assess the radius of curvature, and subsequently the strain, at any point on a dented section of a pipeline. Compared to FEA, the analytical methodology presented by Okoloekwe (2017) proved accurate, but conservative. Although quicker and easier to perform than FEA, ASME B31.8 equations, and other similar analytical strain equations, cannot be used to predict the strain of dents interacting with stress risers or dents with complex shapes, such as dents with multiple apexes.

A novel approach for dent assessment has recently been proposed by Hassanien and Langer (2018), where the strain demand is calculated using the ASME B31.8 equations and includes the uncertainties based on the ILI tool resolution information, the strain limit is the maximum tensile capacity of the pipe, and qualitative multipliers are incorporated to de-rate the strain capacity. These three components are used to determine the probability of failure or the probability that the demand curve will exceed the capacity curve (Hassanien & Langer, 2018). The incorporation of the multipliers allows for the consideration of additional factors not applicable with the ASME B31.8 equations alone, such as interacting features, complex dent

shapes, and pressure cycling effects. The research shows that the maximum strains found using the ASME B31.8 equations are typically non-conservative compared to results found using FEA (Hassanien & Langer, 2018). In addition, the model in the paper is proposed as a ranking tool. The results found using the semi-quantitative model are always conservative compared to fully quantitative results that utilize FEA and is meant to identify features that would require further analysis using FEA.

Hassanien et al. (2016) has also proposed a quantitative assessment of dents that considers all properties of the dent (true shape and size), pipe properties, and interacting features and uses a combination of FEA and reliability analysis to assess the probability of failure of pipeline dents. The research shows that FEA can provide an accurate assessment of the stresses and strains within the dented region, but it is argued that the fully quantitative assessment using FEA is a time-consuming process and is not feasible for system-wide assessment (Hassanien et al., 2016; Hassanien & Langer, 2018).

2.3.3 Fatigue-Based Methods

The elevated strain in a plain dent can be assessed using the profile of a dent and can be made more severe by stress concentrators, such as metal loss (Hassanien et al., 2016). However, using strain alone for dent assessment may result in non-conservative results, as pressure cycling on a dent can have severe effects on the dent's fatigue life. For example, a dent with a lower peak strain but subjected to very aggressive pressure cycling may have a higher risk of failure than a dent with a higher peak strain but experiencing minimal pressure cycling (Turnquist & Smith, 2016). In a study by Kainat et al. (2019), no definite relationship was observed in dents between the maximum equivalent plastic strain and the stress range caused by pressure cycling, indicating that both strain and fatigue should be considered when assessing dents. Assessment solely based on strains could be sufficient for some natural gas pipelines, where cycling is minimal. However, for liquid pipelines, or other applications where pressure cycling has the potential to increase damage to dented areas of pipe, it is important to consider both the stresses and strains within the dent (Turnquist & Smith, 2016).

From the research presented by Turnquist and Smith (2016), the remaining life of a dent is calculated by considering the stress concentration caused by the dent profile and the effect of pressure cycling at the dent location. Stress concentration factors are calculated based on FEA of the dent geometry built-in

directly to the pipe mesh and the application of internal pressure load. A rainflow counting approach is then used to process previous pressure cycling data for the dent location and the amount of damage accumulated is estimated using S-N curves. The resulting accumulated damage can be converted to a rate by dividing by the number of years that the pressure cycling history spanned. The rate could be used along with knowledge of the dent's assumed existing life (for example the amount of time the pipe has been in-service) to determine the dent's remaining life. This proposed methodology demonstrates that FEA is required to obtain the peak stresses caused by the dent in order to accurately predict the dent's remaining life and prioritize its need for mitigation (Turnquist & Smith, 2016). The FEA would be the time-consuming step that would prevent system-wide application of the proposed methodology by Turnquist and Smith (2016).

A similar fatigue life assessment methodology was proposed by Pinheiro, Pasqualino, and Cunha (2014). FEA models were developed to obtain stress concentration factors using three different indenter shapes: spherical, cylindrical with the curved section along the longitudinal direction of the pipe and cylindrical with the curved section along the circumferential direction of the pipe. Analytical expressions for calculating the stress concentration factors were developed using the FEA results of a parametric study and had the pipe diameter, pipe wall thickness, and dent depth, length, and width as input factors (Pinheiro et al., 2014). The analytical expressions were validated against full-scale fatigue tests available in literature. The research affirms the concept that simpler, analytical equations need to be developed based on various input parameters to obtain results that can be provided by FEA models, without losing significant accuracy or precision.

Kainat et al. (2019) assessed the effect of different load sequences on the remaining fatigue lives of dents. In general, it was noted that the remaining life of an unconstrained dent is significantly shorter, almost half, compared to the remaining life of the same dent with the only difference being that it is constrained. In addition, it was determined that for constrained dents, the load sequence of indentation, then pressurization and de-pressurization had a significantly greater stress range than the load sequence of pressurization, indentation, de-pressurization, then pressurization. The former sequence would represent a dent formed during construction, while the latter sequence would suggest a dent formed while the pipe is in-service.

Therefore, if it is unclear when a dent was formed, it would be more conservative from a fatigue perspective to assume that the dent formed during construction.

2.4 Finite Element Analysis

FEA is a numerical method used to solve complicated problems by discretizing them into smaller elements. Several studies in literature have used FEA for validation of full-scale denting tests, comparison to analytical models, and development of new dent assessment techniques, proving its dependability, accuracy, and ability to account for a variety of material properties, complex loading situations, and interaction with stress concentrators (Arumugam et al., 2016; Ghaednia & Das, 2018; Hassanien et al., 2016; Pinheiro et al., 2014). This section describes the background of several aspects of FEA and how it is used by other researchers.

2.4.1 Material Properties

The uniaxial true stress versus true strain curve for the appropriate steel grade being modelled is sufficient for use in the FEA models of pipe steels, according to Hyde, Luo, and Becker (2011) and Arumugam et al. (2016). Validated models have confirmed that it is sufficient to assume that all materials obey an isotropic hardening rule (Hyde et al., 2011; Kainat et al., 2019).

2.4.2 Indenter Properties

The indenter has been modelled in literature as a rigid surface and indents the pipe by applying a vertical downward displacement and contacting the pipe wall (Hyde, Luo, & Becker, 2001). Surface to surface contact is defined between the indenter and the outer surface of the pipe (Arumugam et al., 2016).

2.4.3 Boundary Conditions

The modelling of only a half, or quarter of the pipe segment and utilizing symmetry boundary conditions has been proposed in literature to reduce computational efforts (Hyde et al., 2011; Tiku et al., 2012; Arumugam et al., 2016). In several procedures, the bottom edge of the pipe (opposite from the indenter) is restrained from movement in the vertical direction (Tiku et al., 2012; Arumugam et al., 2016).

2.4.4 Mesh Configuration

There are various mesh configurations for FEA models used in literature. Many sources use a fine mesh in the indentation region and coarser mesh elsewhere to ensure the most accurate results are obtained in the indentation region but to not sacrifice computational time where accuracy is not needed (away from the indenter) (Arumugam et al., 2016; Hassanien et al., 2016; Hyde et al., 2011). A partition can be created in the indentation region to allow this mesh configuration to be possible (Hyde et al., 2011).

2.4.4.1 Element Type

Shell elements can be used in problems where the structure is much smaller in the thickness dimension than in the other dimensions. The geometry of the structure is defined by a reference surface, while the thickness is defined by the section properties (Dassault Systèmes, 2014). This is opposed to a structure made of solid elements, where all three dimensions of the structure are discretized into three-dimensional elements. A comparison between the two types is shown in Figure 1 below. The shell approximation assumes that straight lines perpendicular to the middle reference surface stay straight after deformation and thus, their rotation becomes a degree of freedom as well.

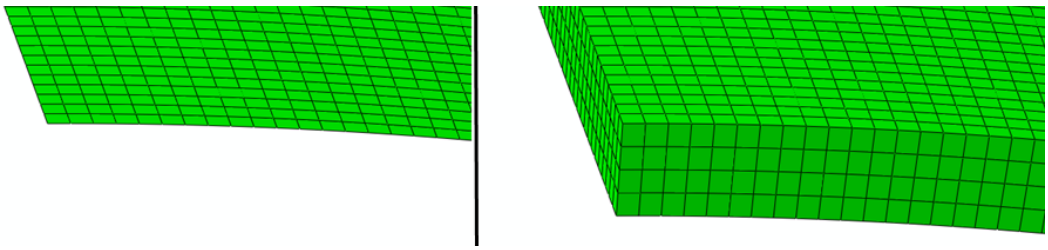

Figure 1: Structure with shell elements (left) and solid elements (right)

Several different types of mesh elements for use in pipeline dent models have been proposed in literature. According to Hyde et al. (2001), similar accuracy for predicting residual stresses in dented pipes is achieved between using shell, 2D and 3D solid elements. The benefit of using shell elements to model plain dents is that the computational time is significantly reduced with minimal loss of accuracy; however, solid, 3D, brick elements must be used to model metal loss or crack features interacting with a dent (Langer, Kainat, & Woo, 2017). This is because the specific geometrical characteristics of the defects, such as rounded corners, cannot be modelled with shell elements as they can with solid elements (Xu et al., 2017). In addition, the effect of internal pressure on the pipe is more accurately represented when there is a defined wall thickness, as opposed to a shell pipe (Xu et al., 2017).

2.4.4.2 Geometric Order

In finite element analysis, linear (first-order) or quadratic (second-order) elements can be used. The order of the elements refers to the order of the interpolation function between the nodes of each element. Quadratic elements provide higher accuracy than linear elements, although analysis of quadratic elements would require greater computation time than equally-sized linear elements (Adeeb, 2018). Quadratic elements are particularly effective over linear elements in modelling curved surfaces and for bending-dominated problems (Dassault Systèmes, 2014).

In literature, linear elements have been recommended over quadratic elements based on sensitivity analysis showing that linear elements are more reliable and require less computational time (Xu et al., 2017).

2.4.4.3 Integration Method

Abaqus uses Gaussian quadrature to assess the material response in each element (Dassault Systèmes, 2014). Compared to full integration, reduced integration uses fewer points to calculate the Gauss integral and to form the element stiffness. The number of computations required for reduced integration is decreased, which results in a drastic reduction in overall running time when performing the analysis in Abaqus. For example, for three-dimensional 20-node brick elements, full integration has 27 integration points, but with reduced integration, only 8 integration points are used (Adeeb, 2018).

The comparative accuracy between the two integration methods is highly dependent on the nature of the problem. Spurious modes can be a problem with reduced integration elements, where the structure is unstable, and the lack of integration points causes zero strains to be predicted. The effect can be minimized by distributing loads and boundary conditions over a greater area of adjacent nodes by creating a finer mesh in areas of loading. On the other hand, in certain problems where the stiffness matrix may be over-predicted in displacement-based problems using full integration, reduced integration may provide a better approximation to the physical problem at hand (Adeeb, 2018).

2.4.4.4 Shell Section Integration

Abaqus can capture linear and nonlinear material behavior through the thickness of a shell cross-section using section points. For shell models, the behavior of the cross-section of the structure is captured directly in terms of section engineering quantities, such as area, moment of inertia, etc., rather than having

to integrate any quantities over the cross-section. Five section points is generally sufficient for nonlinear problems, while more section points may be required for complex problems (Dassault Systèmes, 2014). The integration point in an S4R (reduced integration, linear, shell element) and the corresponding section points through the thickness direction of the shell, are shown in Figure 2.

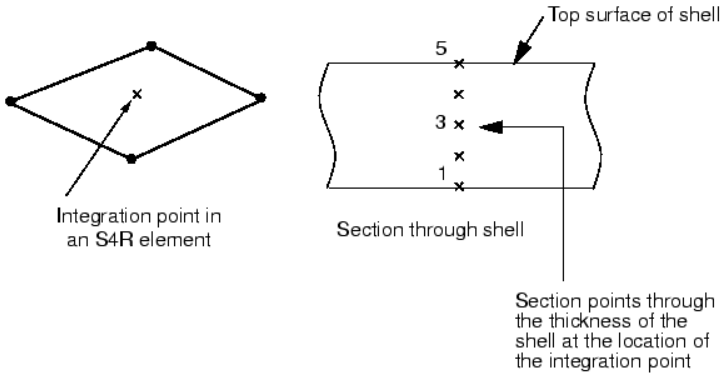

Figure 2: Section points in shell elements (Dassault Systèmes, 2014)

2.4.4.5 Mesh Algorithm

To mesh a region with quadrilateral or quadrilateral-dominated elements, there are two meshing algorithms available in Abaqus: medial axis or advancing front. The medial axis algorithm breaks up the region to be meshed into a group of simpler regions and then uses structured meshing techniques to fill in the smaller regions while minimizing element distortions. On the other hand, the advancing front algorithm generates elements starting at the boundary of the region, according to the specified element sizes at the boundaries, and continues to systematically generate elements as it moves to the interior of the region (Dassault Systèmes, 2014). The difference between using the medial axis and advancing front algorithms on a region is shown in Figure 3. According to Dassault Systèmes (2014), the two algorithms should both be tested to find the optimal mesh for a specific problem.

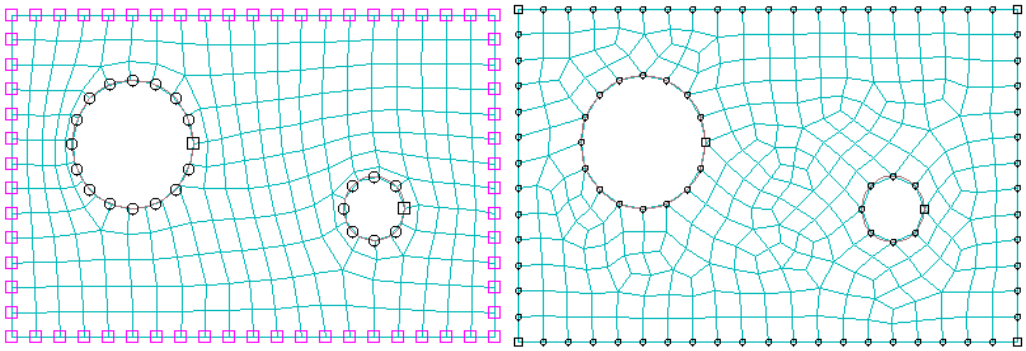

Figure 3: Region meshed using medial axis algorithm (left) and advancing front algorithm (right) (Dassault Systèmes, 2014)

2.5 Artificial Neural Networks

An artificial neural network (ANN) is a powerful computational model that can be used to provide predictive estimates for problems that cannot be expressed by a simple formula and where typical multi-variable, non-linear regression may fail (Xu et al., 2017). The system is capable of learning information from samples and then using this knowledge to perform tasks, similar to the ability and architecture of biological neural networks in the human brain. For problems where complex, non-linear relationships exist between multiple independent and dependent variables, ANNs can process information from samples and use this to infer relationships. A large database of various inputs and outputs is required to train the ANN and can be made up of results from experimental or numerical analyses.

In biological neural networks, most of the neural "computation" occurs within the neuron's nucleus in the cell body (Graupe, 2007). Neural activity travels from one cell to the other across the neuron's axon, which is considered the neuron's connection wire. Electrical signal transmission continues to move down the axon towards synaptic junctions at the end of the axon to dendrites of the next neuron. Any one neuron can receive messages from many other neurons and simultaneously pass on information to many other neurons (Graupe, 2007).

A neuron in an artificial neural network consists of similar parts with similar functions as a biological neuron (Shanmuganathan & Samarasinghe, 2016):

1. Input connections (inputs) – the inputs to the neuron have different weights (or coefficients) associated with them.

2. Input functions (total input) – calculates the aggregated net input signal taking into account the inputs and their weight factors.
3. Activation function (signal) – calculates the activation level of the neuron. This function's primary purpose is to ensure the cell's output is maintained between certain limits.
4. Output function – output signal resulting from the activation function.

ANNs are typically defined by the type of neurons, the architecture of the connections (for example, a network can contain multiple layers with rules over whether the network remembers previous states or not), the learning algorithm, and the recall algorithm (Shanmuganathan & Samarasinghe, 2016). During the training phase, the network is trained with examples to develop rules within its structure and during the recall phase, new data is fed to the trained network to obtain results.

An example of a simple neural network is shown in Figure 4 from Shanmuganathan and Samarasinghe (2016). The figure shows a network with four input nodes, two intermediate nodes, and one output node. The numbers next to each of the lines are the connection weights.

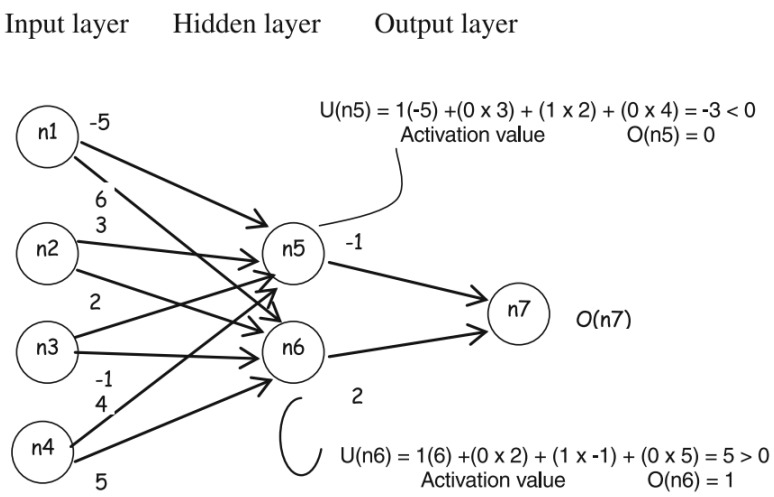

Figure 4: Example of simple artificial neural network (Shanmuganathan & Samarasinghe, 2016)

2.5.1 Training Algorithms

Although there are many different algorithms that can be used to train an artificial neural network, three common algorithms that can be used for training feedforward neural networks are: Levenberg-Marquardt, Bayesian Regularization, and Scaled Conjugate Gradient. The training algorithms work by using different decision functions to update the weights and biases of the network by minimizing the network

errors (Cömert & Kocamaz, 2017). There are benefits and drawbacks to each training algorithm and several algorithms may have to be tested to obtain desired results for a certain application (Cömert & Kocamaz, 2017; The MathWorks, Inc., 2018a).

The Levenberg-Marquardt algorithm is the most efficient method for training feedforward neural networks, but also requires the most memory (Cömert & Kocamaz, 2017; The MathWorks, Inc., 2018a). In terms of ANNs, the Hessian matrix is a matrix of second-order partial derivatives of the performance function at the current values of the weights and biases, when the performance function has the form of a sum of squared errors (Cömert & Kocamaz, 2017; The MathWorks, Inc., 2018a). The Levenberg-Marquardt algorithm works by approximating the Hessian matrix using the Jacobian matrix, where the Jacobian matrix contains first derivatives of the performance function given the current weights and biases (The MathWorks, Inc., 2018a). The Levenberg-Marquardt algorithm uses Equation 1 to update the weights.

$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e \quad (1)$$

where x_{k+1} is the new weight,

x_k is the current weight,

J is the Jacobian matrix,

μ is an adaptive scalar parameter, which is decreased after each successful step (reduction in error) and is increased when a tentative step would increase the error,

I is the identity matrix,

e is a vector of the network errors.

In MATLAB, the Bayesian Regularization training algorithm is used in conjunction with the Levenberg-Marquardt optimization technique described above (The MathWorks, Inc., 2018a). The weights are updated using Equation 1 as in the Levenberg-Marquardt algorithm, and Bayesian techniques are used by assigning a probabilistic nature to the network weights to ensure the network generalizes well and prevents overfitting and overtraining of the network (The MathWorks, Inc., 2018a; Ticknor, 2013).

Conjugate gradient training algorithms typically provide fast convergence, while requiring little memory, however, they can be unstable for large-scale problems (Cömert & Kocamaz, 2017). In addition, they are suitable for problems with a large number of variables, as matrices are not required to be stored for the algorithm (Cömert & Kocamaz, 2017). The Scaled Conjugate Gradient algorithm uses second order

information of the performance function to calculate a step size and search direction with which to update the weight vector (Møller, 1993). It then uses a step size scaling mechanism to reduce the error in the outputs (Møller, 1993).

2.5.2 Use of ANNs in the Pipeline Industry

ANNs have been used in many industries, including robotics, medicine, and financial markets forecasting, although its use in pipelines has been mostly limited. In a detailed study by Xu et al. (2017), an ANN was used to predict the burst pressure of a pipeline with two interacting corrosion defects. The ANN was developed in MATLAB software and was trained using a database made up of a combination of experimental and numerical analyses. The ANN was validated against published burst test data and against the results of existing corrosion assessment analytical equations. It was also concluded from the research that the ANN provided a solution more efficiently, and with more computational time savings compared to FEA (Xu et al., 2017).

2.6 Conclusions

In summary, extensive efforts have been made in research to improve the accuracy and diligence of dent assessment methods beyond the depth- and interaction-based criteria stipulated by current regulations. Dent mitigation at the appropriate time is paramount in the prevention of pipeline failure, which can have adverse effects on the environment, harm people, and/or affect businesses and their reputations. On the other hand, highly conservative dent analyses could result in a waste of resources in repairing dents that did not yet require repair. The balance between these two factors (safety and efficiency) becomes even more challenging for operators with large pipeline systems, where advanced analysis, such as FEA would be very time-consuming (Hassanien & Langer, 2018).

Finite element analysis has proven to be an adequate means through which the stresses and strains within a dented region can be estimated. Although various modifications to FEA settings or properties are included by different researchers, the general procedure of modelling dents using FEA has been well-documented and validated in literature. Artificial neural networks can be based on numerical analyses (such as FEA) and are useful in complex problems where non-linear regression may fail.

CHAPTER 3: FEA MODELLING METHODOLOGY

3.1 Objective

The primary objective of this chapter is to describe a general procedure for using FEA to model real-life indentations in pipelines to be used for the analysis in the remainder of this thesis. There are several different properties of the FEA model, particularly the mesh used, that will impact the final solution and time required for computation, including the size, shape, and type of elements, and the integration method used. As the number of integration points is increased (either by changing the integration method, reducing the elements' sizes, or increasing the number of integration points in the thickness direction), the accuracy of the computations is improved although the speed with which they are determined is reduced (Adeeb, S).

In this chapter, a suitable mesh configuration to accurately and efficiently model dents in pipelines was investigated by comparing the FEA models to the results of full-scale experimental tests performed at the University of Windsor (Ghaednia & Das, 2018). There were three settings pertaining to the mesh that were assessed:

- Element type (shell versus solid)
- Geometric order (linear versus quadratic)
- Integration method (reduced versus full)

The maximum strains using different combinations of the above settings were compared to the maximum strains reported by strain gauges in the experimental tests. As the efficiency of each mesh configuration was also under investigation, the computational time of the different models was recorded.

The effects of reducing the element size and increasing the number of section integration points through the thickness direction for shell elements were also investigated. The results of these analyses led to conclusions about the procedure that will be incorporated in the remainder of this research.

An additional goal of the chapter was to validate the FEA methodology for modelling the indentation of a pipeline through the comparison to full-scale test results.

3.2 General FEA Methodology

The FEA was performed using the software package, Abaqus Version 6.14. The general modelling approach described is the process through which FEA can be used to simulate real dent data obtained from ILI and assess the stresses and strains of the deformed pipe body. Specific areas in the FEA models that deviated from the general approach to meet the objectives of the subsequent sections will be described in detail in those sections.

First, a deformable, solid (in the case of using solid elements) or shell (in the case of using shell elements), 3D part was created in the modelling environment to represent a segment of the pipeline structure. The outer diameter and desired length of the pipe section were specified at this stage. An appropriate length of pipe section had to be selected that fully demonstrates the strains and stresses of the area of interest but was not too long that it was computationally exhaustive. The longer the selected pipe section, the greater number of elements that the model would have and the longer that the FEA solver would take to complete the analysis of the model. For modelling dents according to ILI data, an appropriate indication to select the length of the pipe section is to use the entire length over which the inner diameter readings from the ILI tool seem to deviate from the nominal diameter. For example, a length of 6000 mm could be used to model the hypothetical ILI profile, along the cross-sectional longitudinal profile through the minimum point of the dent, shown in Figure 5 a). The cross-section of the pipe is extruded along the length of the section and the result in Abaqus is shown in Figure 5 b).

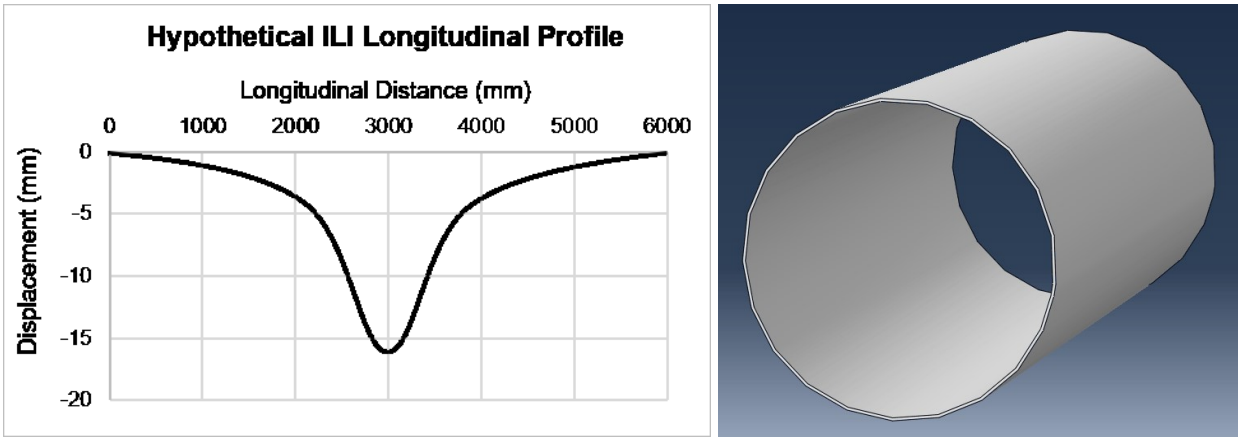

Figure 5: a) Hypothetical ILI dent profile with 6000 mm length; b) Pipe (solid) part in Abaqus

Next, another part was created, separate from the pipe part, to represent the indenter. The indenter part was a 3D, analytical rigid body. The indenter shape used in this study is a torus indenter, which is pictured in Figure 6. The reason for the selection of this indenter shape was for its simplicity, and the ease with which its aspect ratio could be adjusted by changing the major and minor radii. The indenter was then positioned in relation to the pipe so that it was just in contact with the outer surface of the pipe and was centered in the middle of the length of the pipe segment.

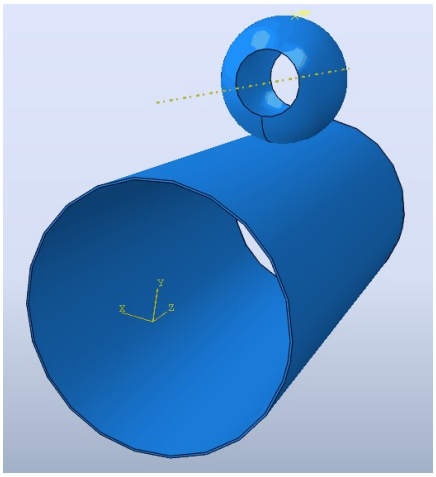

Figure 6: Torus indenter part in Abaqus

The major radius of the torus indenter aligns with the direction of the dent profile that is flatter and thus requires a larger radius than the other side. For example, in Figure 7, the torus indenter is oriented in Abaqus such that a change in the major radius will affect the dent shape in the circumferential direction and a change in the minor radius will affect the dent shape in the longitudinal direction of the pipe. In this case, the profile is flatter in the circumferential direction than the longitudinal direction. For simplicity, the radii of the indenter aligned with each of the directions of the pipe will be referred to as the circumferential and longitudinal radii of the indenter to indicate the respective directions of the dent profile.

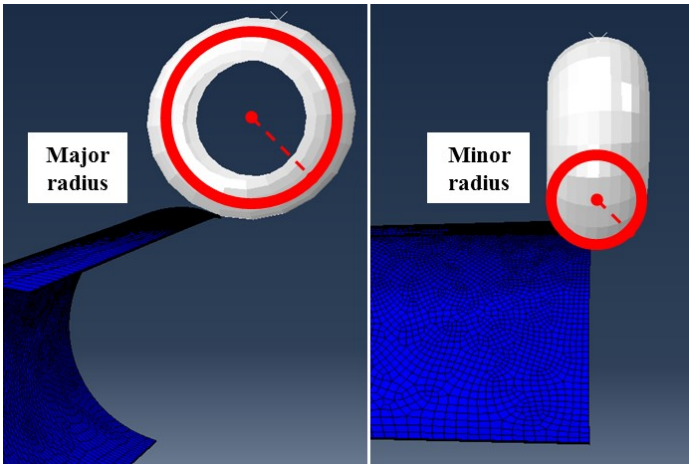

Figure 7: Diagram of the major radius and minor radius of the toroidal indenter

A partition was created in the area closest to the point on the pipe that the indenter was first in contact with. This was to section off a portion of the model and allow the area of the pipe closest to the most significant point (MSP), or minimum point, of the dent to have a fine, structured mesh. The fine seed size could be defined on the partition edges. In the area outside of the partition, the mesh becomes coarser further away from the point of indentation. In dents, the areas further away from the MSP have relatively small deformations that can adequately be captured by coarser mesh. FEA methodology described in literature typically follows this mesh configuration (Hyde et al., 2011; Arumugam et al., 2016).

Once the geometries of the two parts were set up, the elastic and plastic material properties of the pipe were defined. The elastic behavior was assumed to be linear and isotropic, while uniaxial stress strain curves defined the plastic behavior of the material. The plastic hardening was assumed to be isotropic. In this study, the value of yield strength was taken as the specified minimum yield strength (SMYS) of the specific grade of the material and the true stress-true strain curves were determined for the different grades of pipe material based on research published by Lin (2015) and using the Ramberg-Osgood equation (CSA, 2015). In CSA Z662 (2015), the stress-strain relationship can be estimated for steel pipe using Equation 2, shown below.

$$\varepsilon = \frac{\sigma}{E_s} + \left(0.005 - \frac{F_y}{E_s}\right)\left(\frac{\sigma}{F_y}\right)^n \quad (2)$$

where ε is the strain,

σ is the stress,

E_s is the modulus of elasticity of steel pipe,

F_y is the specified minimum yield strength,

n is the strain hardening parameter.

Note that there is also a temperature derating factor that should be included in Equation 2 for pipes operating at temperatures greater than 120°C but it will be assumed for this study that the pipes are operating below this temperature and therefore its effect will be ignored. The resulting values of yield stress versus plastic strain were input in the FEA software as the plastic properties of the material. The specific grades and true stress-true strain curves used in this research are provided in the subsequent chapters.

Surface-to-surface interaction between the rigid body indenter and the pipe was set up in the FEA software. The interaction properties included 'penalty' friction formulation for tangential behavior, with a standard friction coefficient of 0.5, and 'hard contact' for normal behavior.

The initial boundary conditions were set up such that a reference point at the center of each end of the pipe was fixed and a coupling condition was applied to the end of the pipes to restrict the pipe from movement except in the radial direction. This was to simulate real-life pipe conditions: the pipe cannot move vertically or laterally (due to restriction by soil), but can expand (due to internal pressure). In addition, symmetry boundary conditions were used and only a quarter of the pipe was modelled in order to reduce computational efforts. Symmetry boundary conditions for the pipe are only applicable when the dent shape is symmetrical in both the longitudinal and circumferential directions. This is the scenario that will be considered throughout this research. The boundary conditions are shown in Figure 8.

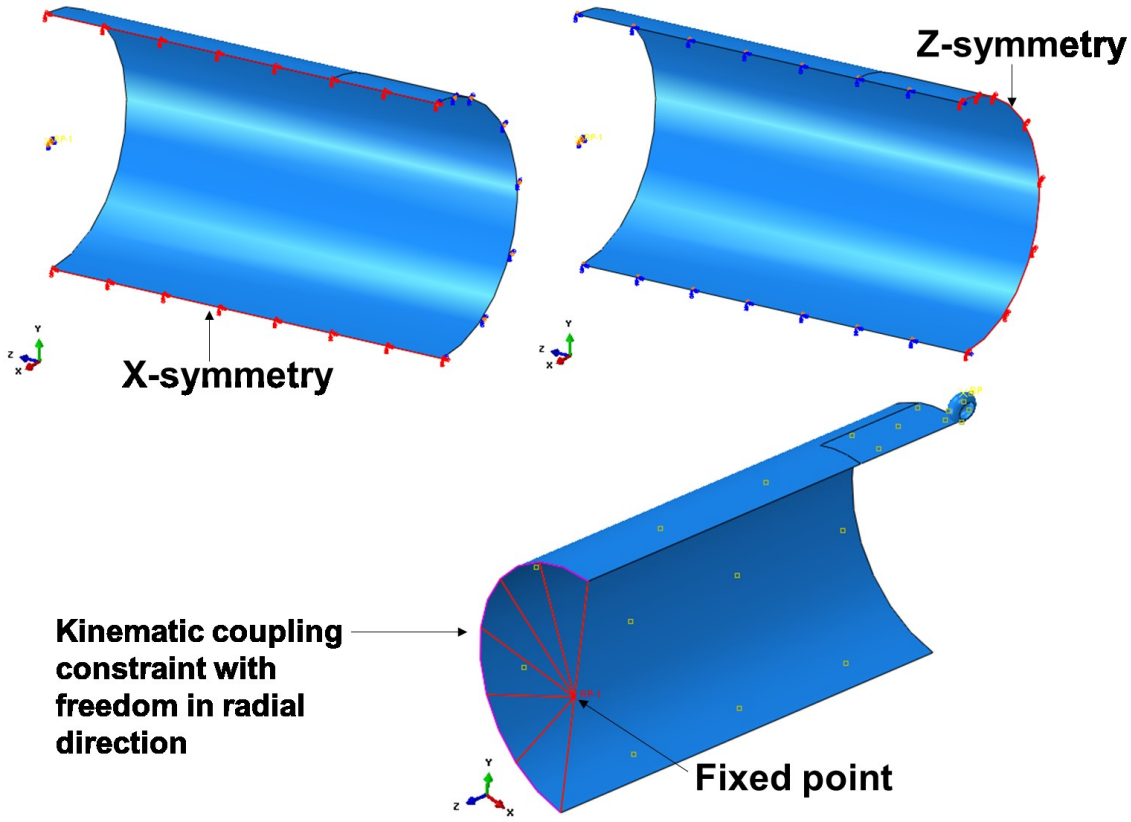

Figure 8: Boundary conditions for FEA pipe model

There are several scenarios that can lead to dent formation that can be reflected in the FEA model by modifying the load sequence. For this research, the consistent loading sequence of indentation, and then a pressure cycle was applied in the FEA models to simulate a constrained dent formed during construction. For unconstrained dent models, the loading sequence applied would be indentation, followed by removal, and then a pressure cycle. The indentation and then pressurization load sequence was chosen since assuming the dent was formed during construction is a conservative estimate from a fatigue life perspective, and from a strain perspective, there is minimal difference in maximum equivalent plastic strain between the loading sequence of indentation and then pressure cycle (construction dent) versus pressure cycle then indentation (operation dent) (Kainat et al., 2019). The indentation step was applied in the FEA model by translating the indenter vertically downward and a pressure cycle was applied by assigning a pressure load on the inside surface of the pipe equal to the maximum operating pressure (MOP), removing the pressure load, then applying the maximum operating pressure load once more.

The maximum operating pressure was calculated for this research using Equation 3 below, from the Canadian standard, CSA Z662 (CSA, 2015). The standard states that pipelines in Canada can operate at no more than 80% of the specified minimum yield strength (SMYS) of the pipe.

$$P = 80\% \times \frac{2St}{D} \tag{3}$$

where P is the maximum operating pressure (in MPa),

S is the specified minimum yield strength of the pipe (in MPa),

t is the wall thickness (in mm),

D is the outer diameter of the pipe (in mm).

For the mesh, four-node, reduced integration, shell elements were used with linear geometric order. The size of the mesh within the partition was 2 mm and the mesh grew coarser outside of the partition, far away from the indentation (the area of interest). Five integration points were used in the thickness direction of the shell elements. The justification for these choices is described in the following section.

3.3 Mesh Configuration Studies

3.3.1 Mesh Configuration Methodology

FEA was performed using Abaqus software and compared to one of the full-scale tests presented by Ghaednia and Das (2018). Only “Specimen 2” described in the paper was used for this research. There were four steps to the analysis performed in this study:

1. The initial model was set-up and run. The force-displacement curve obtained from FEA was compared to the force-displacement curve resulting from the full-scale tests to validate the procedure and ensure the material and geometric properties, boundary conditions, interactions, and loading steps were input similarly to what was performed in real-life.
2. A mesh convergence study was performed with various mesh sizes in the indented region of the pipe. An appropriate mesh size was chosen as a result.
3. The model was re-run with different mesh properties to determine their effects on the final solution.

4. Separate shell models were run to determine the optimal number of thickness integration points to use.

The FEA models used are described in more detail in the sections below. The overall experimental test set-up used in Ghaednia and Das (2018) is shown in Figure 9, where a pipe with end caps was supported at the bottom and a denting load was applied at the top.

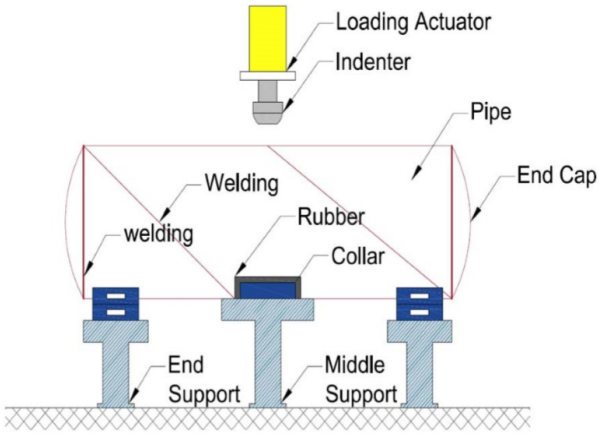

Figure 9: Schematic of experimental test set-up from Ghaednia and Das (2018)

3.3.1.1 Geometric Properties

There were three parts modelled: the pipe, the end caps, and the indenter. The geometric properties of the pipe are shown in Table 1 and follow what was given by Ghaednia and Das (2018).

Table 1: Pipe Geometric Properties for Mesh Configuration Study

Outside diameter (mm)	762
Wall thickness (mm)	8.5
Length (mm)	1250

Although the entire segment of the pipe in the experimental tests was 2500 mm long, only a quarter of the test set-up was modelled in FEA to save computational time and only 1250 mm length was included in the model. Partitions were added at 100 mm from each end of the pipe to allow for boundary conditions to be applied within the sections created by the partitions. A 100 mm (in the longitudinal direction) by 50 mm (in the circumferential direction) partition was also added at the top of the pipe. This is the area of the pipe where the indentation was applied. The partition geometry can be seen in Figure 10.

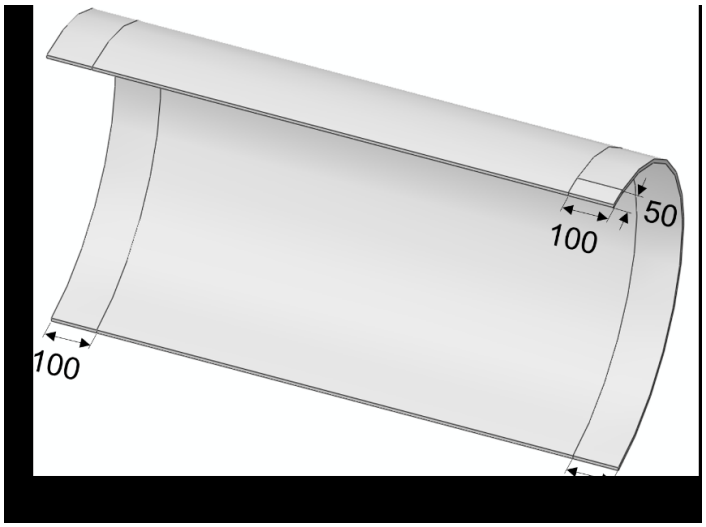

Figure 10: Pipe with partitions dimensioned in mm

The exact geometry of the end caps was not specified in the paper (Ghaednia & Das, 2018). Only one end cap was included due to the symmetry of the model. The end cap was modelled as a 3D deformable part with the section geometry shown in Figure 11 revolved 180° about the vertical Y-axis. As seen in Figure 11, the length of the end cap was modelled as 267 mm as specified in ASME B16.9 (The American Society of Mechanical Engineers, 2001) for 762 mm diameter pipe.

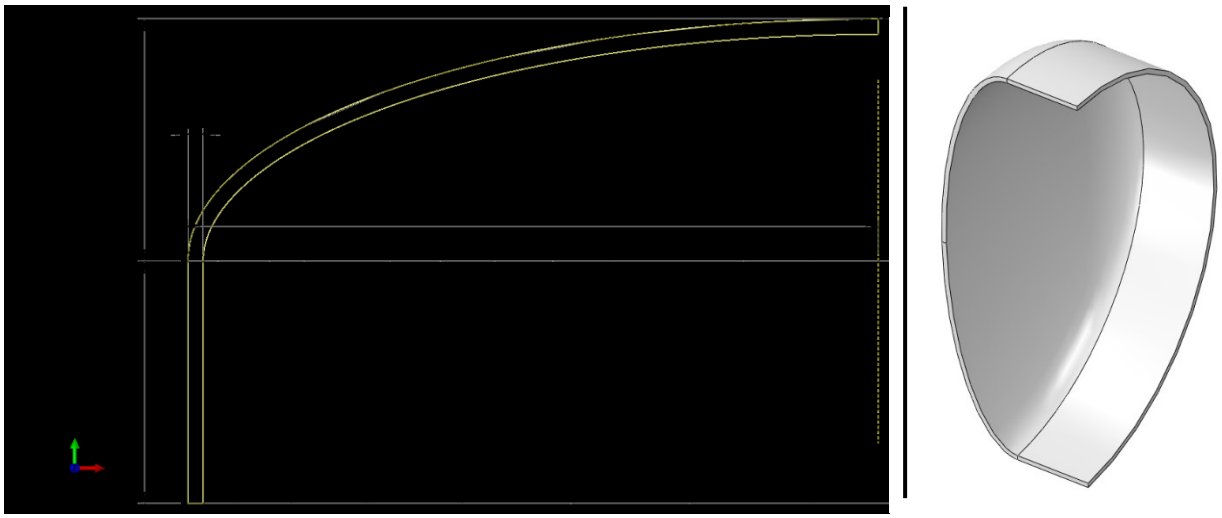

Figure 11: End caps section sketch with dimensions in mm (left) and 3D part image (right)

The indenter was described in the paper to have plan dimensions of 100 mm by 50 mm. In the FEA models, the indenter was modelled as a 3D discrete rigid part. It first was modelled as a rectangular prism that was 100 mm x 50 mm x 5 mm. The exact details of the indenter geometry were not described in the

paper, although the bottom of the indenter was shown to be rounded in the images. As a result, the indenter in the FEA models was rounded at the bottom using fillets of 5 mm radius, as shown in Figure 12.

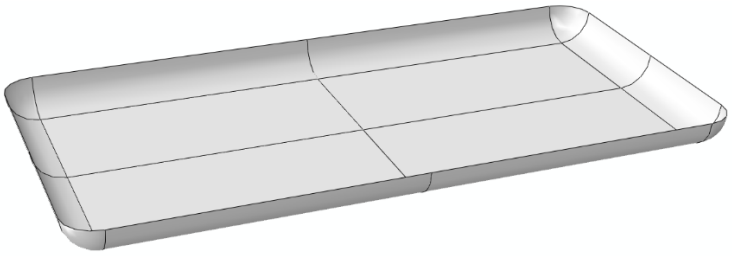

Figure 12: 3D discrete rigid indenter

3.3.1.2 Material Properties

The pipe specimens used in the experiments were made from X70 steel with a modulus of elasticity of 204 GPa (Ghaednia & Das, 2018). The material was modelled as non-linear with isotropic hardening. In the paper (Ghaednia & Das, 2018), the yield strength of the material was specified to be 543 MPa, but the full true stress-true strain curve was not provided in the paper. According to the results of experimental tests published by Lin (2015), the strain hardening exponent (n) for use in the Ramberg-Osgood equation was determined to be approximately 8 for X52 pipe. Using the Ramberg-Osgood equation, presented earlier in Equation 2, with E_s equal to 204 GPa, F_y equal to 543 MPa, and with four different assumptions for the strain hardening exponent (n = 6, 8, 10, and 12) the four different true stress vs. true strain curves are shown in Figure 13. The true stress-true strain curve using a strain hardening exponent of 12 was used for the analysis in this chapter, which is described in more detail in Section 3.3.3.

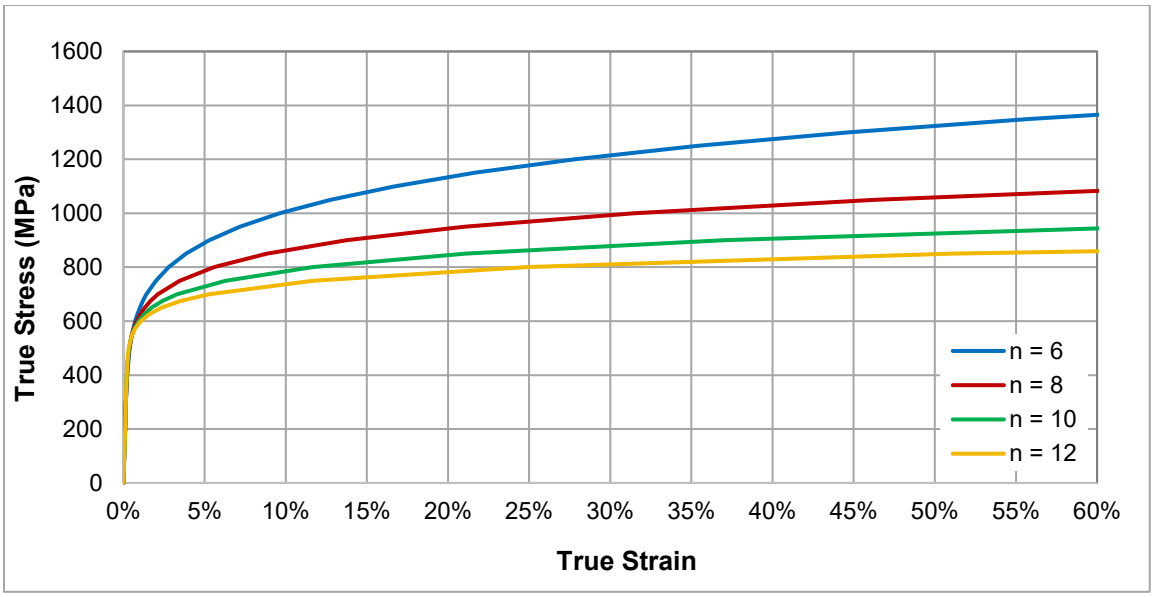

Figure 13: True stress versus true strain material curve

3.3.1.3 Interactions, Boundary Conditions, and Constraints

The interaction between the indenter and the outer surface of the pipe was modelled as surface-to-surface contact, as described in the General FEA Methodology section. “Hard” contact was modelled for the normal behavior property and penalty friction formulation with a friction coefficient of 0.5 was used for the tangential behavior.

The applied boundary conditions are shown in Figure 14. Symmetry in the X and Z directions were applied at the areas shown. The dimensions of the supports below the pipe were not specified by Ghaednia and Das (2018), however, the regions within the partitions (as shown in Figure 14) were pinned so that the pipe was restrained from translation in the X, Y, and Z directions but rotation was free.

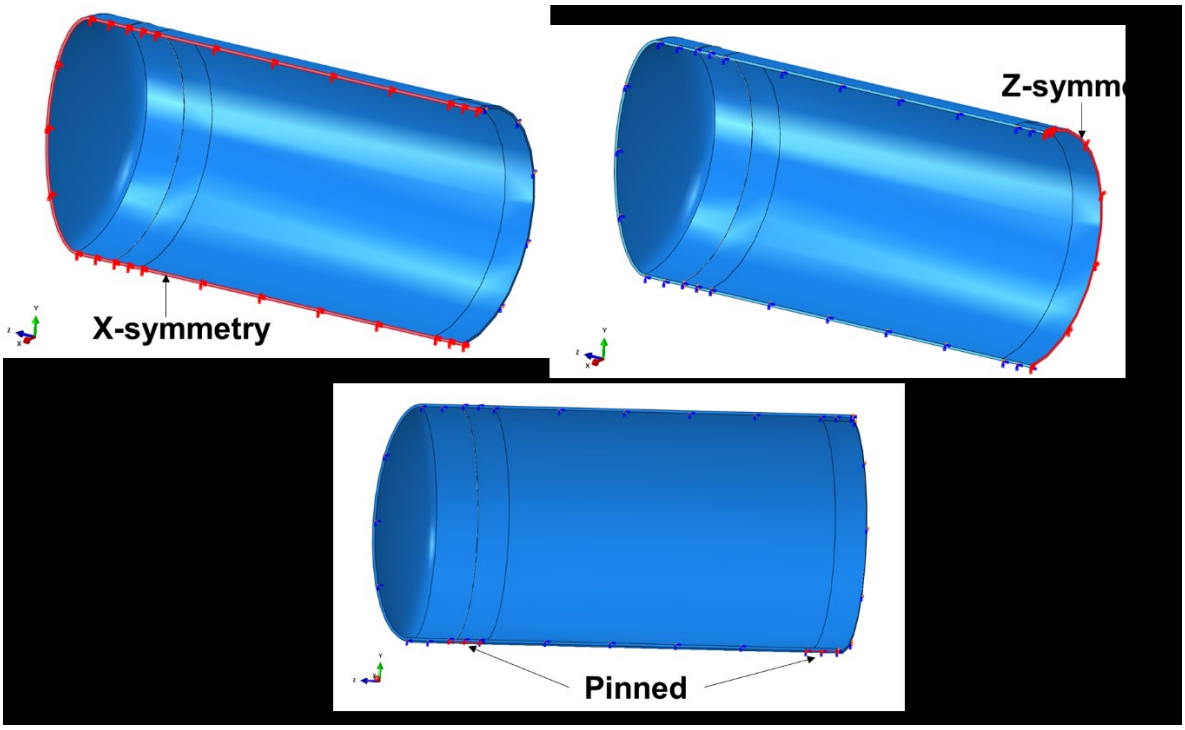

Figure 14: Boundary conditions applied in FEA model

A tie constraint was applied between the surface at the end of the pipe and at the end cap to model how the end cap was welded to the pipe in the experimental set-up. The region of the tie constraint is shown in Figure 15.

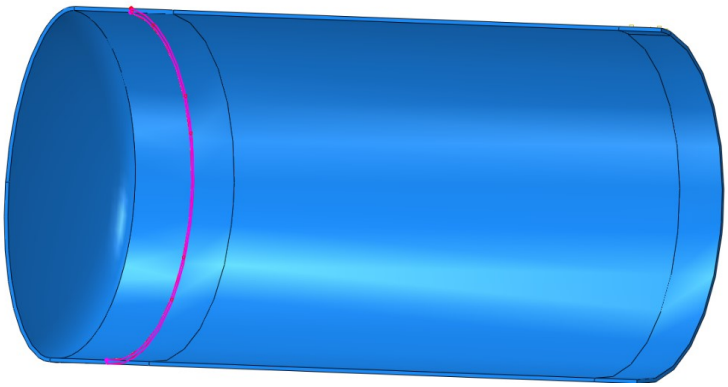

Figure 15: Tie constraint region between end cap and pipe end

3.3.1.4 Loading Steps

As described in the paper, the loading sequence applied was pressurization, indentation, and removal, which simulates an unconstrained dent that forms during operation. During the experimental pressurization step, internal pressure of 3.8 MPa was applied using water, to simulate operating pressure

of oil in a pipeline in real-life. The 3.8 MPa pressure load was applied on the entire internal surface in a single step in the FEA model. Internal pressure of 3.8 MPa was maintained throughout the indentation and removal steps.

The next step in the FEA was the indentation step, where the indenter was translated vertically downward by 64 mm to indent the pipe. Lastly, the indenter was translated vertically upwards so there was no longer contact with the pipe. As there was some rebounding of the pipe due to the elastic properties of the material, the indenter had to be translated a greater amount so that the maximum depth of the permanent, plastic deformation was 30 mm (4% of the outer diameter). The loading steps for the FEA are shown in Figure 16.

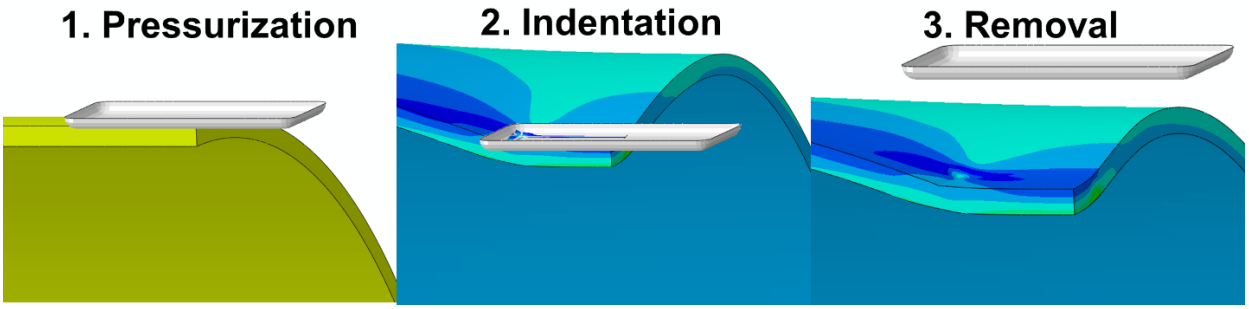

Figure 16: Loading steps in FEA model

3.3.1.5 Mesh Properties

The mesh was generated so that it was fine within the partitioned area and became coarser in the rest of the model. The mesh was finer within the partition as this was where the indentation occurred and was the area of greatest interest. As the indenter was a discrete rigid part, it also required meshing and was meshed with elements of constant size of 2 mm. First, a mesh convergence study was performed with different models using 4 mm, 2 mm, 1 mm and 0.5 mm mesh size within the partition. Following the mesh convergence study, no significant changes were seen with a mesh size smaller than 2 mm within the partition. Thus, the mesh used in the remaining models was 2 mm within the partition, with the longitudinal and circumferential directions of the pipe modelled with a single bias from 2 mm to 30 mm, as shown in Figure 17. The approximate element size within the end cap was 50 mm.

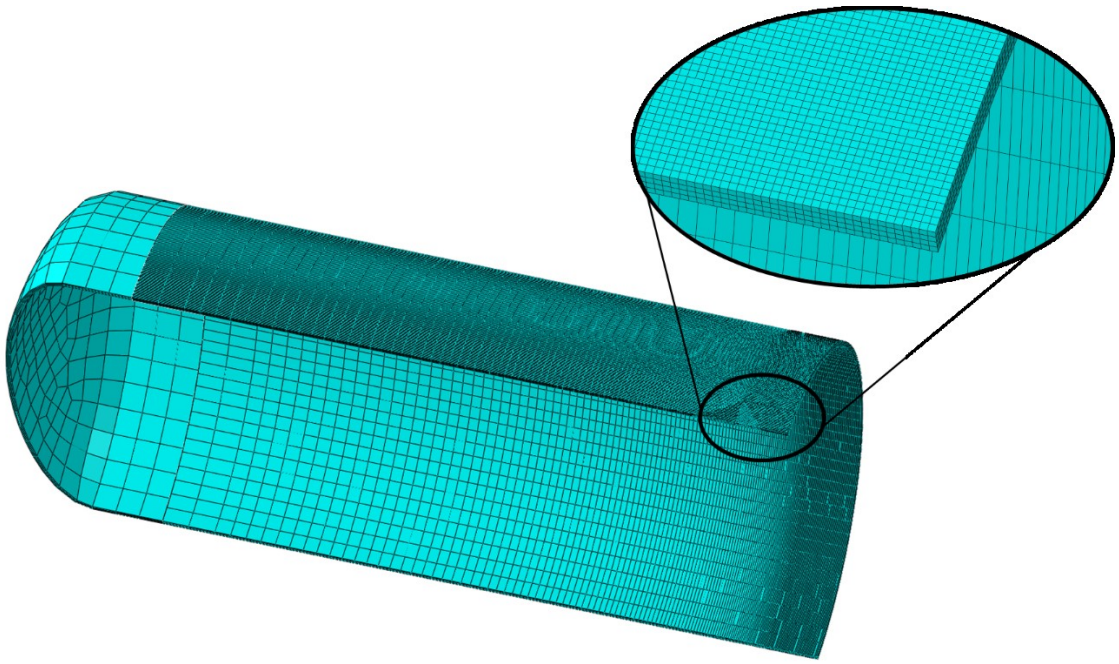

Figure 17: Meshed pipe with finer mesh within partitioned region

For the solid models, there were 4 elements in the through thickness direction. For the shell models, the number of thickness integration points was modified under the section properties, and the effects on the results were observed. These tests were performed using the S4R (shell, linear, reduced integration) elements. The Simpson thickness integration rule was used, and the different integration points tested were 3, 5, 7, 9, and 11, in separate models. In addition to using the indenter used in Ghaednia and Das (2018) described in Section 3.3.1.1, two separate torus indenters (sized 50 mm major radius in the longitudinal direction by 25 mm minor radius in the circumferential direction, and 500 mm major radius in the longitudinal direction by 250 mm minor radius in the circumferential direction), were also tested with different integration points in the pipe section. For the models with each of the indenters, all other properties were held constant while only the number of section points was adjusted. Images of the two ring indenters are shown in Figure 18.

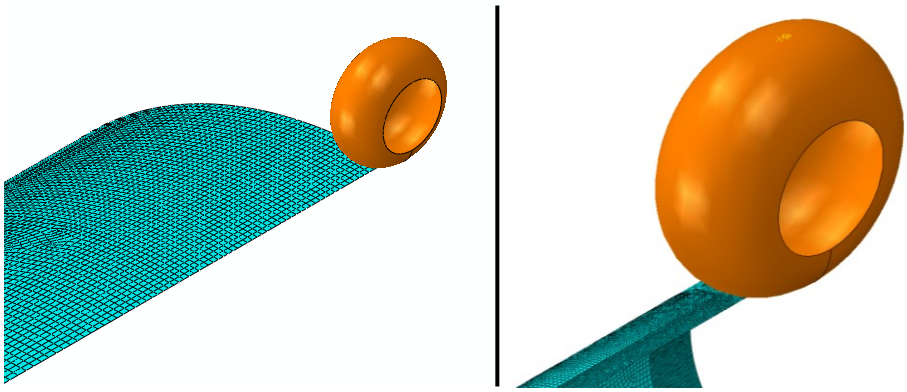

Figure 18: Torus indenters – 50 mm x 25 mm (left) and 500 mm x 250 mm (right)

Under mesh controls, hex-dominated element shapes were used with sweep technique and medial axis algorithm within the partition region. The same controls were used for the rest of the model except with the advancing front algorithm.

Different models were created with the same properties described in this Mesh Configuration Methodology section, but the only difference was the element type. The different element types investigated in this study and their properties are shown in Table 2.

Table 2: Properties of the Element Types Used

Element Type	Solid or Shell	Geometric Order	Integration Method	# of Nodes
S4	Shell	Linear	Full	4
S4R	Shell	Linear	Reduced	4
C3D8	Solid	Linear	Full	8
C3D8R	Solid	Linear	Reduced	8
C3D20	Solid	Quadratic	Full	20
C3D20R	Solid	Quadratic	Reduced	20

3.3.1.6 Strain Outputs

The strain output recorded out of the FEA results was the nominal strain (labelled NE in Abaqus). The principal values used to calculate the nominal strains in Abaqus are the ratios of the change in length to the undeformed length, which provides the closest results to the experimental strain data recorded using strain gauges.

3.3.2 Mesh Configuration Results

There were four different analyses on mesh configuration: validating the load versus displacement curve from FEA to the experimental tests (Ghaednia & Das, 2018), performing a mesh convergence analysis, evaluating the effects of different element types on strain, and determining the optimal number of thickness integration points for the shell section. The results of each analysis are presented in this section.

3.3.2.1 Load Versus Displacement Validation Analysis

The load applied in the vertical direction by the indenter was plotted against the maximum displacement in the pipe. The resulting curves obtained from the different element types using FEA was compared to the curve from the experimental tests (Ghaednia & Das, 2018), shown in Figure 19.

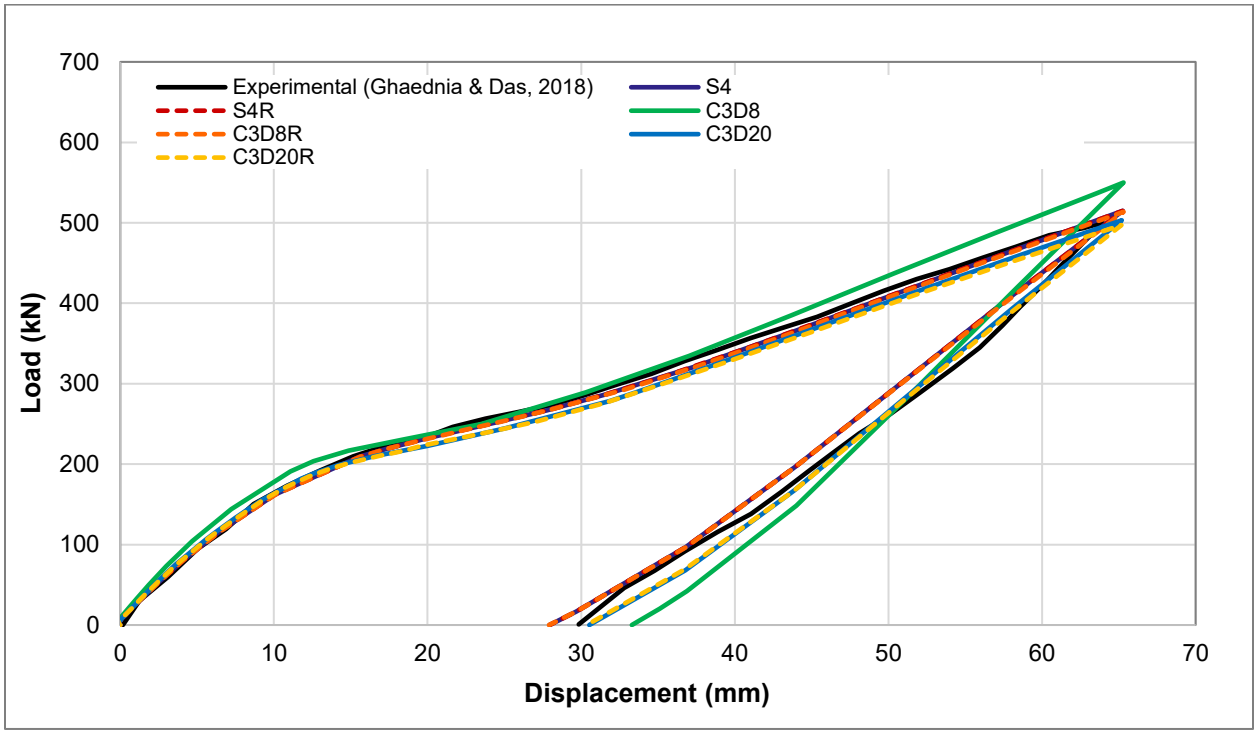

Figure 19: FEA to experimental comparison of load versus displacement curve

The load versus displacement curves from FEA using the S4, S4R, C3D8R, C3D20, and C3D20R were almost identical to each other and aligned closely with the experimental curve. The C3D8 elements (solid, linear elements with full integration) produced a slightly different curve from the experimental one, as a higher load was required to achieve the maximum displacement.

3.3.2.2 Mesh Convergence Analysis

The mesh convergence study involved adjusting the mesh size within the partition to be 4 mm, 2 mm, 1 mm, and 0.5 mm, and the nominal strain distributions were recorded. The elements were kept as squares within the partition (i.e. 4 mm refers to the elements within the partition being 4 mm in the longitudinal direction by 4 mm in the circumferential direction). The mesh sizes in the rest of the model outside of the partition were kept constant between the models. The element type used for the mesh convergence study was S4R as the use of this element required the least computational time. In Figure 20, the longitudinal strain distributions over the axial distance from the dent center are shown, while the circumferential strain distributions over the distance from the dent center in the hoop direction are shown in Figure 21. The strain distributions on the outer surface are shown in the figures.

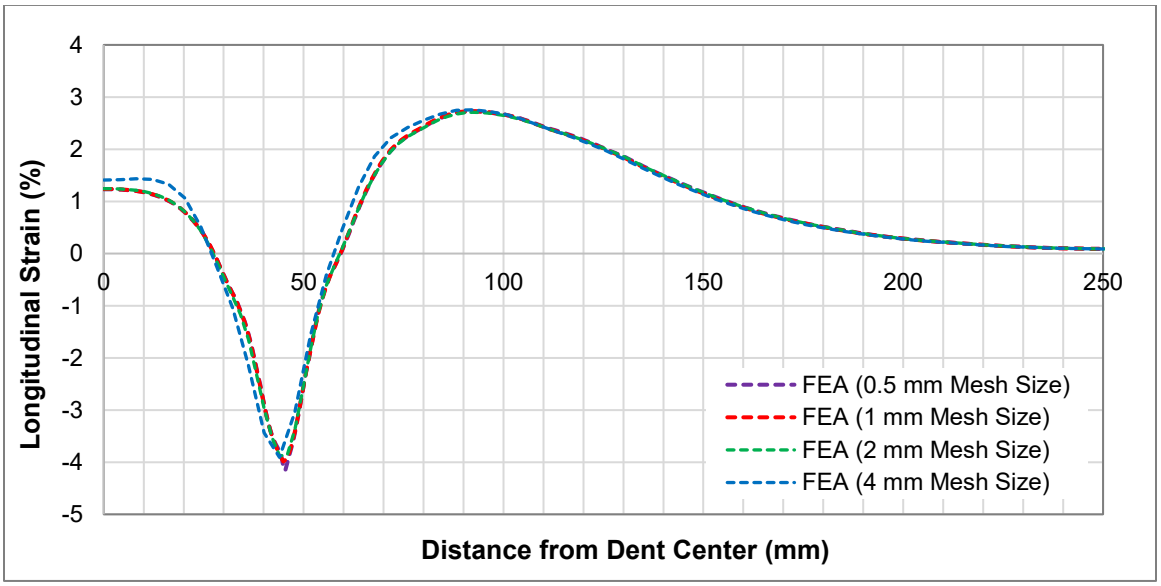

Figure 20: Longitudinal strain comparison for various mesh sizes

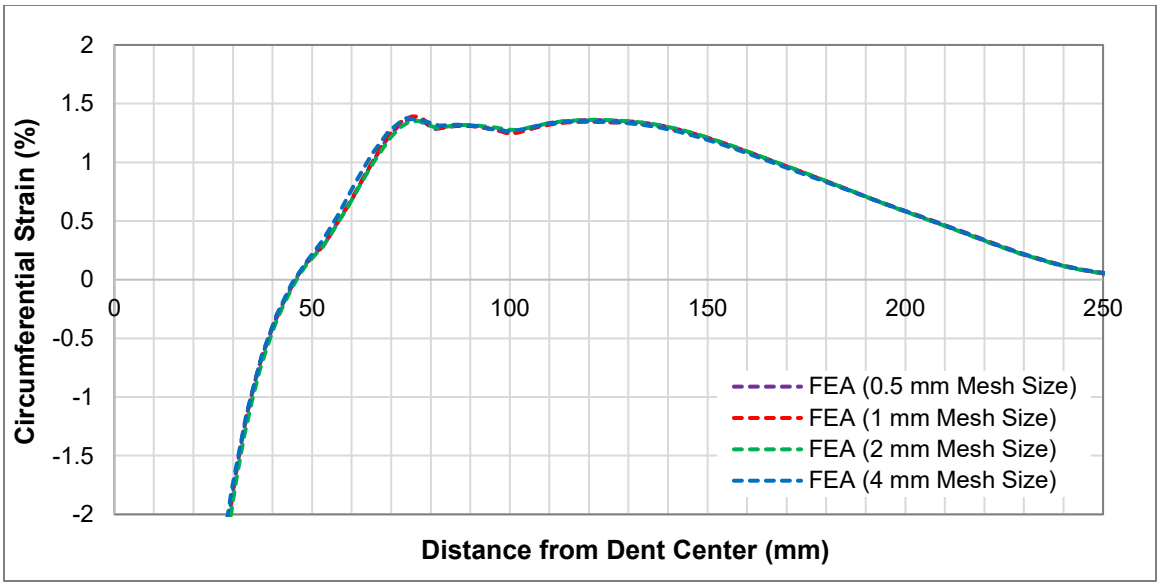

Figure 21: Circumferential strain comparison for various mesh sizes

There is a slight but negligible difference between the strain distributions in each direction from the model with 4 mm mesh size to the other models. There was negligible difference between the strain distributions from the models with 2 mm, 1 mm, and 0.5 mm mesh sizes. This shows that with a mesh finer than 2 mm within the partition, little accuracy would be gained with significant increases in computation time. As a result, 2 mm mesh size within the partition was chosen as the optimal balance of accuracy and efficiency for the rest of the models for the study when considering the strain results.

3.3.2.3 Element Type Analysis

Six different FEA models were created with various element types, as presented earlier in Table 2, and their resulting nominal strain distributions are shown in Figure 22 and Figure 23. The strain distributions were recorded from the top surface of the FEA model, to imitate how the strain gauges were placed on the outer surface of the pipe during the full-scale tests.

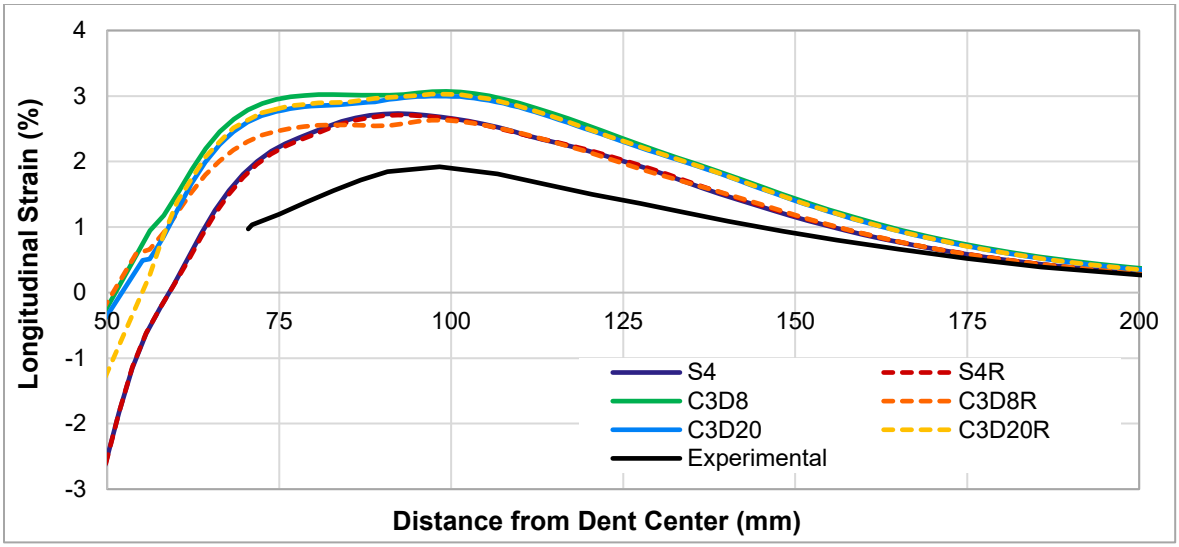

Figure 22: Longitudinal strain distributions for various element types

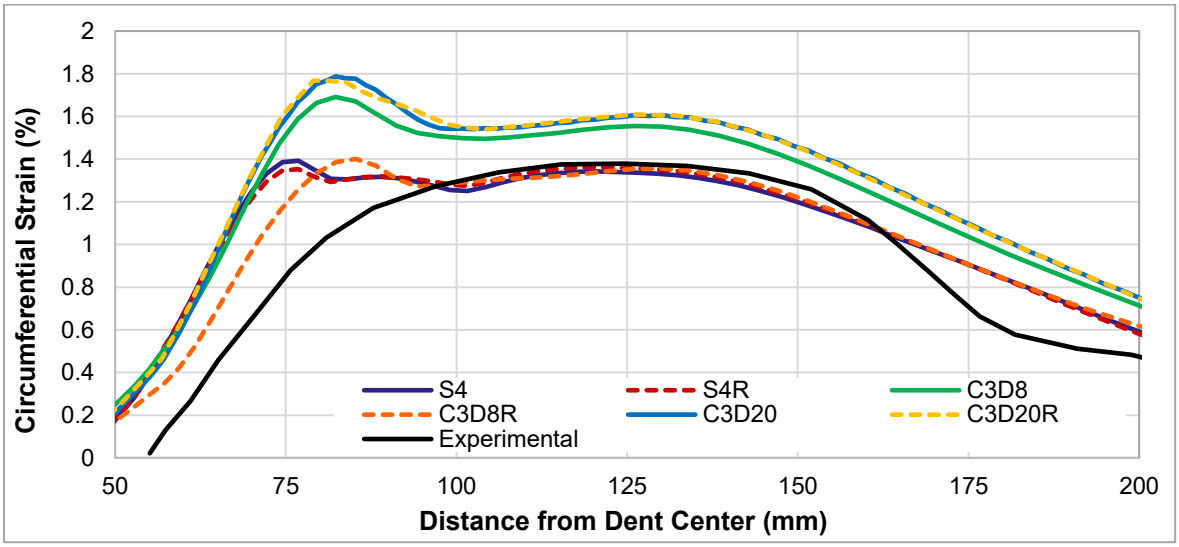

Figure 23: Circumferential strain distributions for various element types

The results show that the nominal strain distributions from FEA are similar to the strains recorded by the gauges during the full-scale tests presented by Ghaednia and Das (2018). The strain distribution curves from the experimental tests were recorded by fitting a smooth curve through the reported measurements from strain gauges that were spaced 25 mm apart along two lines that ran through the dent centerline (one in the circumferential direction and one in the longitudinal direction).

The computational running time for the FEA models was also recorded to evaluate the level of efficiency of each element type. Table 3 shows the computational time required for the models with each of the element types.

Table 3: Computational Time Required for FEA Models with Different Element Types

Element Type	Computational Time (sec)
S4	613
S4R	609
C3D8	2,093
C3D8R	1,882
C3D20	95,207
C3D20R	23,611

Figure 24 shows the maximum longitudinal strains for each element type compared to the maximum longitudinal strain of 1.92% from the experimental tests (Ghaednia & Das, 2018). The computational time for each model is plotted against the secondary axis in the plot in Figure 25.

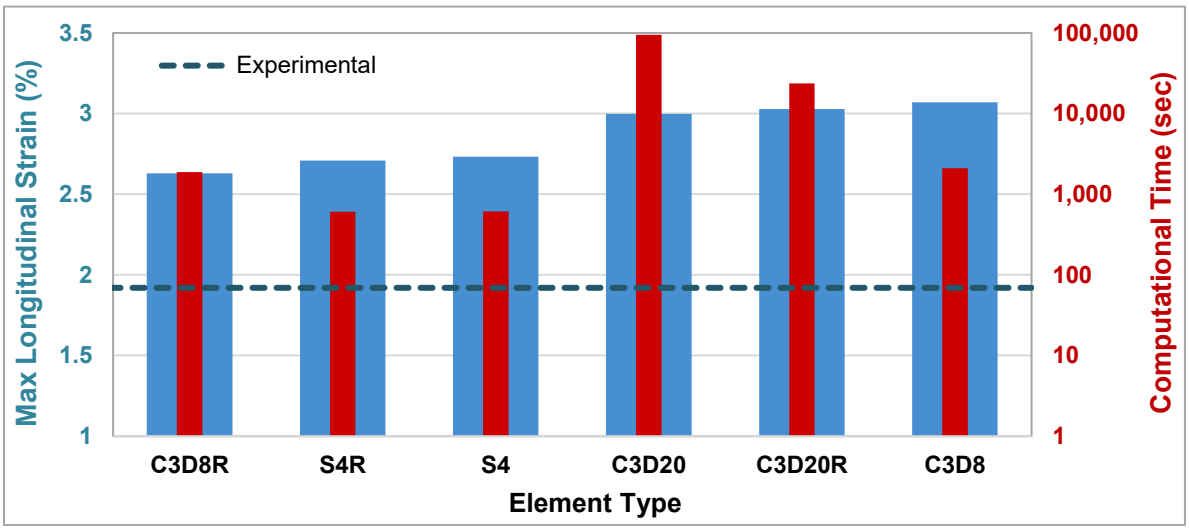

Figure 24: Maximum longitudinal strains and computational times for each element type

The maximum strain in the circumferential direction recorded by the strain gauges during the full-scale tests was 1.37% (Ghaednia & Das, 2018). Figure 25 shows the maximum circumferential strain and computational time from each model.

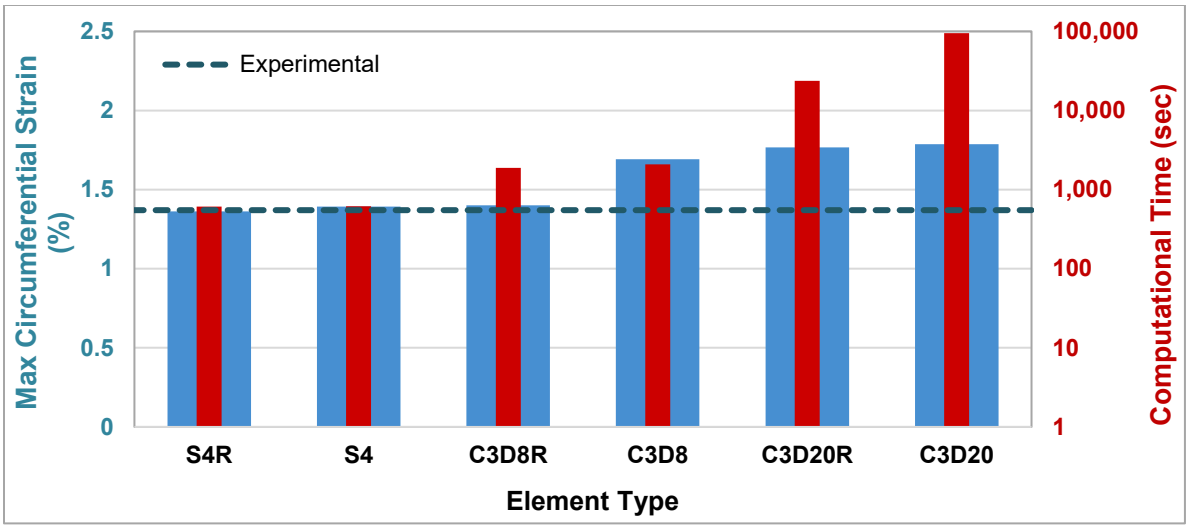

Figure 25: Maximum circumferential strains and computational times for each element type

Figures Figure 22, Figure 23, Figure 24, and Figure 25 all demonstrate that the FEA models that used the element types S4, S4R, and C3D8R resulted in the closest strain results to the experimental results from Ghaednia and Das (2018). The full integration, linear solid elements (C3D8) and two quadratic types (C3D20R and C3D20), produced higher, more conservative strain results in both directions than the S4, S4R and C3D8R elements.

The results also show that the use of quadratic element types in the FEA models requires significantly greater computational time than the linear types. Using full integration requires more time than reduced integration, and solid elements require more time than shell elements. The models that required the least computational time to run overall were the models with shell elements and there was only a negligible difference in computational time and strain outputs between the full integration and reduced integration shell models.

In Figure 26, the maximum displacement and computational time from the different models are shown, where the maximum deformation in the pipe was 30 mm after the indenter was removed during the experimental tests (Ghaednia & Das, 2018). It is apparent that the shell elements caused the pipe to behave in a more flexible manner than with the solid elements, which caused the maximum amount of rebounding of the dent to occur after the indenter was removed. The linear, full integration elements and both quadratic element types resulted in the least amount of permanent, plastic deformation.

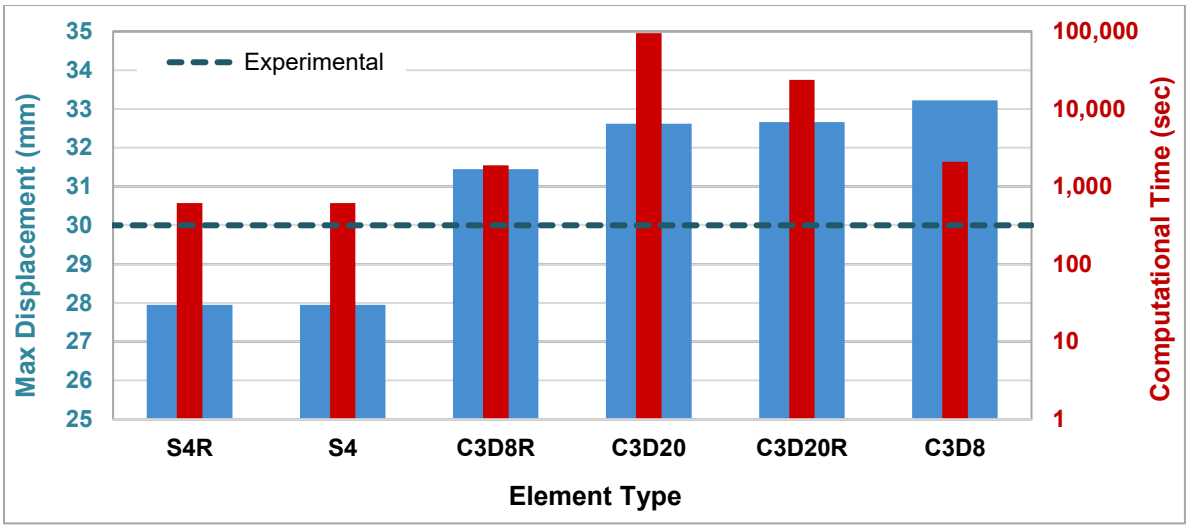

Figure 26: Maximum displacement and computational times for each element type

3.3.2.4 Thickness Integration Points Analysis

The maximum nominal strains and the maximum equivalent plastic strains were recorded in the longitudinal and circumferential directions from the models with varying numbers of integration points in the thickness direction. The strain results with the discrete rigid, rectangular indenter, along with the computational time for each model are presented in Table 4, with the strain results shown graphically in Figure 27. The maximum strains presented by Ghaednia and Das (2018) from the experimental results are also shown in Table 4 for reference.

Table 4: Results by Number of Thickness Integration Points for Rectangular Indenter

Number of Thickness Integration Points	FEA Computational Time (sec)	Maximum Nominal Strain		Maximum Equivalent Plastic Strain	
		Longitudinal Direction	Circumferential Direction	Longitudinal Direction	Circumferential Direction
3	1103	3.21%	1.94%	5.22%	5.11%
5	601	2.71%	1.36%	4.91%	4.40%
7	607	2.70%	1.50%	4.79%	4.43%
9	586	2.73%	1.50%	4.90%	4.43%
11	628	2.73%	1.46%	4.77%	4.42%
Experimental	N/A	1.92%	1.37%	N/A	N/A

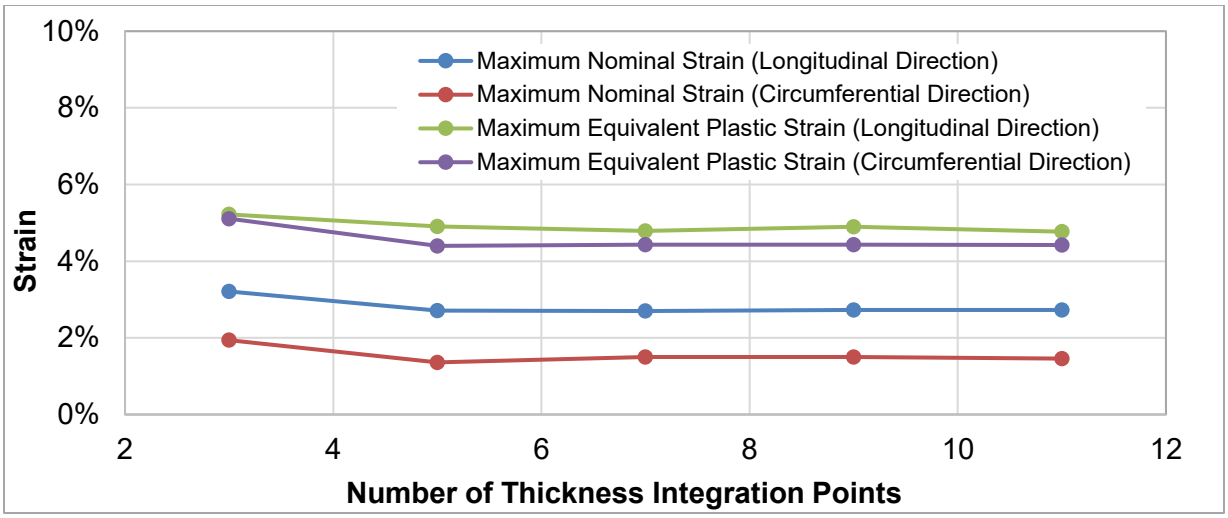

Figure 27: Strain results with increasing number of thickness integration points

In Table 5, the strain results and FEA computational times are shown for each of the models using the torus indenters.

Table 5: Results by Number of Thickness Integration Points for Torus Indenters

Number of Thickness Integration Points	FEA Computational Time (sec)		Maximum Equivalent Plastic Strain	
	50 x 25 mm	500 x 250 mm	50 x 25 mm	500 x 250 mm
3	129	208	23.6%	4.5%
5	269	212	24.4%	4.0%
7	273	196	24.2%	4.3%
9	267	213	24.0%	4.0%
11	262	152	24.0%	4.0%

As shown in Table 4, Table 5, and Figure 27, the models with five integration points or greater all had similar computational times and strain results, regardless of indenter. The FEA strain results with five or more thickness integration points were close to the maximum nominal strains from the experimental results from Ghaednia and Das (2018). The default number of thickness integration points for shell sections in Abaqus is five.

3.3.3 Material Properties Results

The applied load versus maximum displacement in the pipe was also plotted using the four different assumptions for strain hardening exponent in the Ramberg-Osgood model (n=6, 8, 10, and 12) and the results are shown in Figure 28.

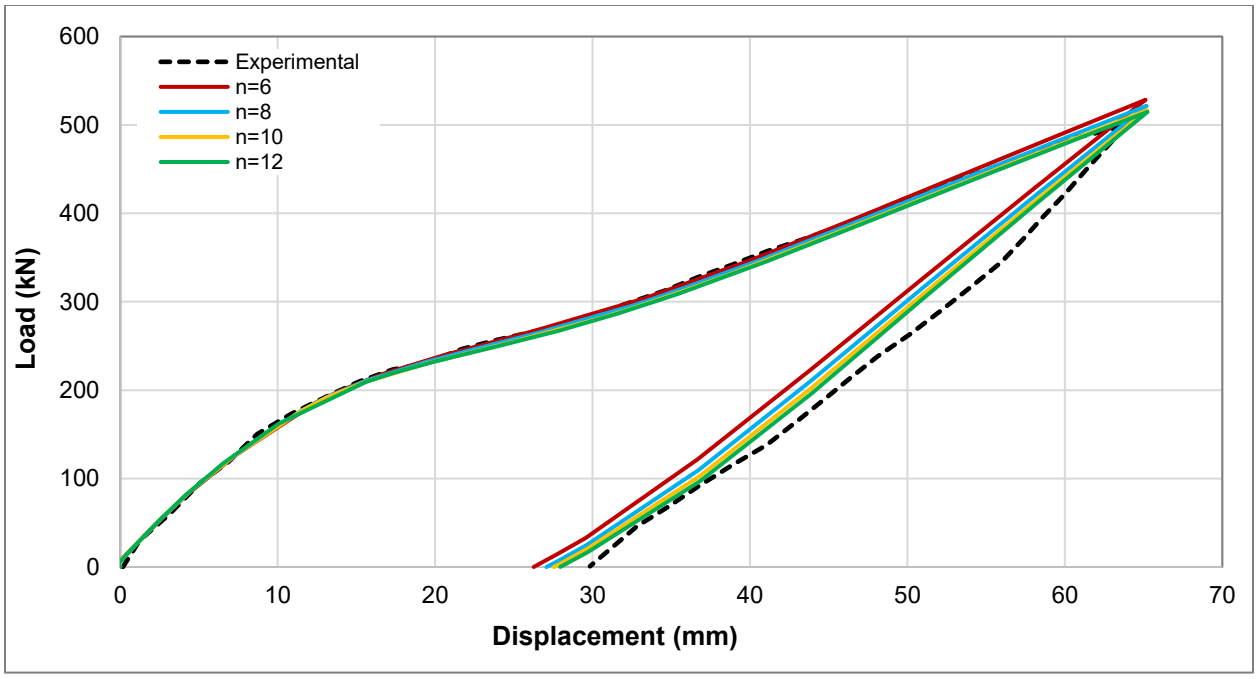

Figure 28: Load versus displacement curve with varied strain hardening exponents

Furthermore, Table 6 shows the maximum nominal strains and maximum equivalent plastic strains in both the longitudinal and circumferential direction with all other variables kept constant except for the strain hardening exponent used to define the plastic material properties in Abaqus. The models to test the effect of the strain hardening exponent on the strain results were run using the S4R elements.

Table 6: Strain Results by Changing Strain Hardening Exponent

Strain Hardening Exponent	Maximum Nominal Strain		Maximum Equivalent Plastic Strain	
	Longitudinal Direction	Circumferential Direction	Longitudinal Direction	Circumferential Direction
6	2.07%	1.19%	4.31%	3.99%
8	2.37%	1.25%	4.64%	4.14%
10	2.57%	1.30%	4.81%	4.23%
12	2.71%	1.36%	4.91%	4.40%
Experimental	1.92%	1.37%	N/A	N/A

3.3.4 Discussion

The two objectives in the study: validating the FEA procedure of indenting a pressurized pipeline with experimental results and investigating a suitable mesh configuration for the FEA models, were demonstrated by the results presented in the last section. The results and their implications are discussed in further detail in this section.

3.3.3.1 Load Versus Displacement Validation

The load versus displacement graph obtained from the FEA model followed very closely to the load versus displacement graph presented by Ghaednia and Das (2018) from the full-scale tests, which indicates that Abaqus is a suitable tool that can be used to perform FEA to model the process of indentation of a pipeline and that the input boundary conditions and parameters were accurate. One of the possible reasons for small discrepancies between the curves from FEA and from the experimental test is the fact that the exact support dimensions were unknown and were assumed from the information available in the paper. The supports were idealized as pinned and modelled as such in the FEA, although in the experimental tests, the supports were made of rubber and might have had some compressibility during the indentation process, which indicates that the FEA boundary conditions created a stiffer structure than the real-life supports. Variations in diameter, wall thickness, material properties and loading between a physical experiment and an idealized model also contribute some error between the FEA and experimental load-displacement curves. When comparing the different element types against each other, the model with full integration, solid, linear elements (C3D8) resulted in the load versus displacement curve that was furthest away from the load versus displacement curve from the experimental tests. Since full integration results in stiffer elements due to the higher number of integration points (Dassault Systèmes, 2014), a greater load was required to achieve the same amount of maximum displacement as with the other element types.

3.3.3.2 Mesh Convergence Analysis

Mesh convergence analysis is important to ensure that the mesh is sufficiently fine to obtain accurate solution from the FEA. In the mesh convergence study used for this thesis, S4R elements were used and the mesh size was only reduced within the partitioned region of the pipe. After reducing the size of the

square elements (4 mm x 4 mm, 2 mm x 2 mm, 1 mm x 1 mm, and 0.5 mm x 0.5 mm), it was determined that the nominal strain results were sufficiently accurate using the 2 mm x 2 mm elements.

3.3.3.3 Element Type Analysis

All of the FEA models, regardless of element type, produced similar longitudinal strain and circumferential strain distributions to those obtained from the experimental results (Ghaednia & Das, 2018). The strain results further support the positive validation of the FEA procedure against the experimental results. One of the reasons for the differences between the FEA and experimental strain distributions is that the exact indenter dimensions from the experimental models were not specified and were thus assumed for the FEA model (i.e. the 5 mm radius fillets used to round the bottom of the indenter). The indenter shape and size have significant effects on the strain distributions, so it is predicted that the strain distributions would be much closer if the correct indenter dimensions from the experiments were modelled. In addition, the strain distribution can be extracted continuously across the dented region from the FEA model, while the experiment utilized strain gauges to measure strains at increments of 25 mm and the strains in between the measurements were interpolated. Furthermore, the FEA model assumes idealized inputs (such as uniform diameter and wall thickness throughout the pipe), whereas a physical experiment has the potential to have inconsistencies in the pipe and material properties.

As expected, the models using shell elements required significantly less (approximately 3 times less) computational time than the models using solid elements. According to the initial literature review, the accuracy level between shell elements and solid elements was largely dependent on the problem at hand. In the problem here, the thickness direction (8.5 mm) that was modelled as a shell was significantly smaller than the pipe in the axial (1250 mm) or circumferential (1197 mm) directions. The dimensions of the pipe make it a suitable structure to be modelled with shell elements and the strain distributions indicated very little difference between the use of the shell elements and the use of the linear, reduced integration solid elements.

The results of this study show that shell elements are suitable for modelling the indentation of a pipeline. However, this would only be the case if a plain dent, or dent not interacting with any other features, needed to be modelled. If the dent is interacting with a corrosion feature, for example, the pipe wall would

be thinner within the corroded region and thus the wall thickness would not be uniform throughout. In this case, solid elements would be required to model the dent with corrosion feature using FEA.

For this problem, the models with quadratic elements required more than 10 times as much computational time as the models with linear elements while their strain outputs were nearly identical to the outputs from the solid, linear, full integration model. Quadratic elements are typically used in FEA problems to better capture the results of highly curved structures. However, the pipe structure in this case and the deformation of it could be sufficiently captured by linear elements. In addition, the mesh within the partitioned region was fine enough that changing the geometric order to quadratic and adding an additional inner node within each of the elements did not add any additional accuracy to the results. Thus, quadratic elements significantly increased the computational time of the models while they did not contribute to any increases in accuracy and are therefore a poor element choice to model deformations in pipelines in this research.

The integration method created insignificant differences in the strain results and the computational times for the models with shell elements. For the linear solid elements, full integration took a greater amount of computational time than reduced integration, although the difference was not significant (full integration took approximately 1.1 times longer than reduced integration). The use of full integration, linear solid elements (C3D8) resulted in greater maximum strains than the reduced integration, linear solid elements (C3D8R), which was further away from the experimental results (Ghaednia & Das, 2018). This indicates that the use of full integration overestimated the stiffness in the structure and that the reduced integration elements more closely modelled what occurred in the experimental tests. The fact that the model with full integration, linear solid elements (C3D8) resulted in the highest final displacement suggests that following indentation and removal of the indenter, the C3D8 elements resulted in the least elastic rebound, which can be attributed to the higher stiffness of the elements (Dassault Systèmes, 2014). This was observed by the fact that the displacements and strains were relatively similar following indentation, but following removal, the displacements and strains were higher for the model with solid, linear, full integration elements versus the same model with reduced integration elements.

The integration method had minimal effect on the strain responses for the models with quadratic elements, although the model with full integration, quadratic elements (C3D20) required 4 times as long to run as the model with reduced integration, quadratic elements (C3D20R).

3.3.3.4 Thickness Integration Points Analysis

The use of three integration points in the thickness direction produced maximum strain results that were furthest away from the experimental results (Ghaednia & Das, 2018). The FEA models with five thickness integration points or more all produced nominal strains that were close to the experimental results from Ghaednia and Das (2018). As there was negligible difference in computational time and strain results between the models with five integration points or more, five thickness integration points will be consistently used throughout the remainder of the research.

3.3.3.5 Material Properties

The results shown in Section 3.3.3 indicate that there was minimal difference in the load versus displacement curve when varying the strain hardening exponent used in the Ramberg-Osgood equation (Equation 2) to determine the plastic material properties. Using a strain hardening exponent of 12 resulted in the closest load versus displacement curve to the experimental test from Ghaednia and Das (2018), while the use of a strain hardening exponent of 6 resulted in the greatest error from experimental.

The strain results in Table 6 show that with the strain hardening exponent of 12, the highest strain values were found, and the nominal strains were closest to the experimental strains in the circumferential direction.

For simplicity throughout the rest of this research, a strain hardening exponent of 12 will be used with the Ramberg-Osgood equation (Equation 2), since it provided the most accurate results compared to the experimental tests from Ghaednia and Das (2018).

3.4 Conclusions

Overall, the methodologies presented in this chapter can be used to model the indentation of pipelines using finite element analysis. The load versus displacement curve and strain distributions found using FEA were positively validated against the results obtained from full-scale tests (Ghaednia & Das, 2018). A mesh convergence analysis was also performed to confirm that a sufficiently fine mesh was used within the area of interest for the purposes of this thesis. There will always be some error in using a model to emulate a real-life problem, as evidenced by the fact that the FEA did not predict the exact same strains as those found experimentally (Ghaednia & Das, 2018). As long as a consistent modelling method, including the mesh configuration and properties, is used and documented, the results obtained from FEA should still be accurate enough to assess the general severity of the dent and aid in making decisions over whether mitigation is required. Potential uncertainties in the material properties of the pipe, geometric properties of the dent, and FEA model can be accounted for in future works using structural reliability analysis.

Different element properties were tested: element type (shell versus solid), geometric order (linear versus quadratic) and integration method (reduced versus full). The results indicated that the elements most suitable for modelling dents in pipelines are S4 (shell, linear, full integration elements) or S4R (shell, linear, reduced integration elements). These elements produced the closest approximation to the results obtained experimentally while requiring the least amount of computational time to run. The integration method did not make a difference for the linear shell elements with this problem. If solid elements must be used for the model (for example, if a non-uniform wall thickness distribution had to be modelled or corrosion or cracks had to be modelled interacting with the dent), then the most suitable element type in that case would be the C3D8R elements, which are solid, reduced integration, linear elements. These elements produced similar results to the shell elements, which were quite close to the results from the experiments (Ghaednia & Das, 2018). The solid, full integration, linear elements (C3D8) and solid, quadratic elements (C3D20 and C3D20R) were found to cause the structure to behave stiffer than with the other three element types (S4, S4R, and C3D8R), which was evidenced by the higher final displacements, indicating less elastic rebound following removal of the indenter. This led to more conservative results while also requiring far greater computational time. As a result, C3D8, C3D20, and C3D20R are not recommended element types

for modelling dented pipelines. The results presented in this chapter support the choice to use S4R elements throughout the models used in the remainder of this thesis.

The optimal number of thickness integration points to be used in the models was also investigated. The models using five integration points in the thickness direction were found to be both accurate and efficient.

Lastly, the true stress-true strain curve was not provided by Ghaednia and Das (2018), so an approximate curve was estimated using the Ramberg-Osgood equation. The effect of differing strain hardening exponents on the load-displacement curve and strain results was observed. A strain hardening exponent of 12 was found to be suitable for this case and will be used throughout the remainder of this thesis.

CHAPTER 4: AUTOMATION OF FEA MODEL GENERATION AND RESULTS EXTRACTION

4.1 Objective

The objective of Chapter 4 is to describe the automation processes of generating a FEA model and extracting the associated results. For this research, the processes were automated using Python script, which is compatible for use with Abaqus.

Automation results in the benefits of increased efficiency and decreased potential for human error or inconsistency in the models. This research requires the generation of a large number of models with only some parameters changing in each model. In the profile matching section (Chapter 5), the indenter size in the FEA model will need to be adjusted in several iterations until the FEA displacement profile closely aligns with the displacement profile reported by an ILI tool. In Chapter 6, the neural network section, hundreds of models with various parameters will be required to train the network. As a result, automation is essential to complete the large number of repetitive tasks required for this research, as long as the varying properties can be adjusted in each model. Outside of this research, automation can be used in industry to efficiently match FEA dent profiles to ILI profiles or model a large number of dents in a system.

4.2 Model Generation

The use of automation during the model generation step reduces the time of human operation required from hours to mere seconds. The bulk of the time required to obtain the FEA results then becomes purely computational time. However, a large batch of models can be generated and programmed to run one after another, so that even if the computational time is lengthy, the models can run continuously without requiring human interference.

The key components in setting up the Python script were identifying which variables would be changing between the different FEA models and then determining how to link each of the model generation steps back to the input variables. The Python script can be thought of as a step-by-step instruction manual for the computer. Each line of code defines a different step in the procedure (described in Chapter 3) that the user would take if operating Abaqus manually. An example of this is shown in Step 2 of Figure 29.

There were two files required for the model generation step to be automated: a text file containing the input variables and a Python script that reads the input variables and describes the model steps. The Python script is then run in Abaqus to generate the model. This process is illustrated in Figure 29.

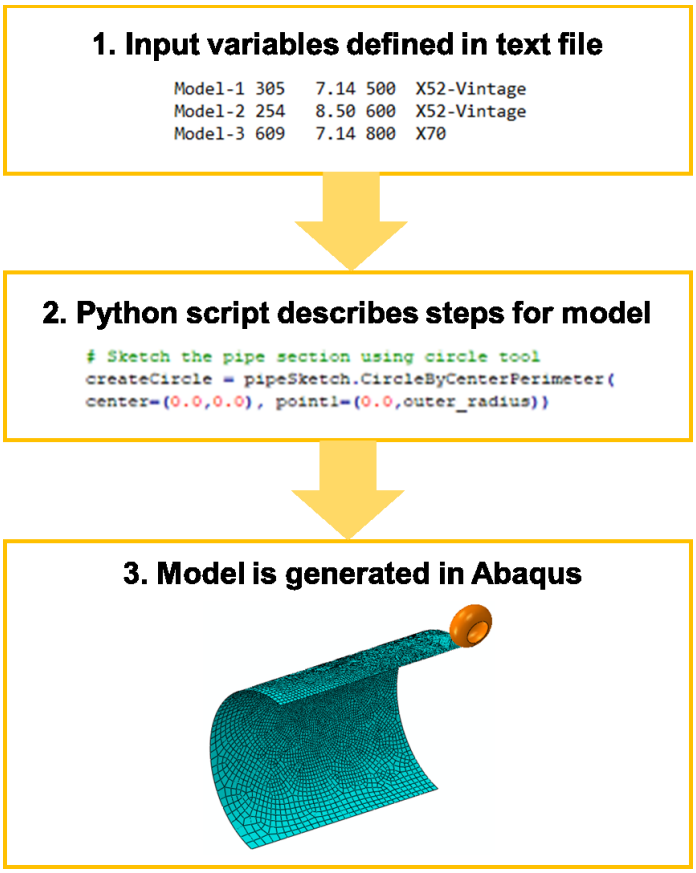

Figure 29: Automation process flow for FEA model generation

4.2.1 Input Variables

The input variables were defined in a separate text file so that the variables could be modified by the user without having to enter them into the Python script each time. In addition, having the variables in a separate file allows for a batch of models to be generated at once (in conjunction with a “for” loop in the Python script). Each column of the file defines a different variable and each row defines the set of variables for a new model. An example set-up of the text file is illustrated in Table 7.

Table 7: Input Variables Example Text File Set-Up

Model Name	Outer Radius	Wall Thickness	Length of Pipe	…
Model #1	254	7.14	500	
Model #2	305	8.5	1000	
Model #3	381	10	2000	
…				

The first portion of the Python script is reading in each input variable from the text file and defining each of them as variables for later use by the script. The input variables needed to define the FEA model are presented in Table 8. All steps in the methodology described in Chapter 3 can be performed automatically with reference to the 15 variables shown.

Table 8: Input Variables Required for Automatic Model Generation

No.	Category	Input Variable	Units
1	Label	Model name	N/A (text)
2	Pipe geometric property	Outer radius	mm
3		Wall thickness	mm
4		Length	mm
5	Pipe material property	Steel grade	N/A (text)
6	Partition geometric property	Length of partition	mm
7		Width of partition	degrees
8	Indenter geometric property	Radius of indenter in longitudinal direction	mm
9		Radius of indenter in circumferential direction	mm
10		Indentation depth	mm
11	Internal pressure	Maximum operating pressure	MPa
12	Mesh seed sizes	Seed size for partition edges	N/A (unitless number)
13		Seed size for opposite end of pipe from indenter – longitudinal edge	N/A (unitless number)
14		Seed size for opposite end of pipe from indenter – circumferential edge	N/A (unitless number)
15		Seed size for opposite corner of pipe from indenter	N/A (unitless number)

The following sections describe the overall aspects of the script required to generate a dent model with various parameters. The exact syntax of the lines of code were determined using the Record Macro

function in Abaqus and modified to incorporate the variables without errors to mimic the button-clicks a user would go through to generate the model manually.

4.2.2 Parts

After reading in the input variables, the Python script creates a model using the name specified by the user. Following that is the creation of three parts: the pipe, partition, and indenter. The most common command used in the script is the “findAt” command. This command uses an arbitrary point found on the object and specified in X, Y, and Z coordinates, to select a vertex, edge, face, or cell. As a result, different selections in the model can be made based on the geometric property input variables. For example, the code used to create the pipe part is shown in Figure 30. As shown, “findAt” is used to select which portion of the circle to trim. The text shown in green following the “#” symbol are comments to describe what the script is doing in each step but are not read in by Abaqus.

```
# Sketch the pipe section using circle tool
createCircle = pipeSketch.CircleByCenterPerimeter(center=(0.0,0.0), point1=(0.0,outer_radius))

# Create vertical construction line
pipeSketch.ConstructionLine(point1=(0.0,0.0), angle=90.0)

# Auto-trim the circle so semi-circle (left-hand side) remains
trimCurve1 = pipeSketchg.findAt((outer_radius,0),)
pipeSketch.autoTrimCurve(curve1=trimCurve1, point1=(outer_radius,0))

# Create a 3D deformable part named "Pipe" by extruding the sketch
pipePart=Model.Part(name='Pipe', dimensionality=THREE_D,type=DEFORMABLE_BODY)
pipePart.BaseShellExtrude(sketch=pipeSketch, depth=length_of_pipe)
```

Figure 30: Example script for creating pipe part in Abaqus

Next, the partition part is created using the length and width of the partition, and outer diameter of the pipe as input variables. The partition is then merged with the pipe part and the excess faces of the partition are removed to create the final pipe part.

The indenter part is created as a three-dimensional analytical rigid body. The size of the torus-shaped indenter is defined by two of the input variables: the radius in the longitudinal direction and the radius in the circumferential direction. The larger value of the two defines the major radius of the torus, while the smaller value defines the minor radius. A spherical indenter results if the two radii are equal.

4.2.3 Materials Library

A library of different materials with unique properties can be defined using the script and only one will be selected for the model (using the input variables) and assigned to a section. In this case, the elastic properties (Young's modulus and Poisson's ratio) and plastic properties (yield stress and plastic strain) are defined for each of the materials and the material itself is given a descriptive name.

4.2.4 Section Assignment, Orientations, and Assembly

Next, the section properties of the pipe are assigned. The section is defined as a homogenous shell section with a certain steel grade defined as the material (using the same name as one of the materials in the material library). At this stage, the wall thickness of the pipe is also defined using the input variable and the number of integration points in the thickness direction is specified to be five.

A cylindrical coordinate system is set-up and applied to the cylindrical pipe part so that the results can be extracted properly (for example the hoop and axial stress components are properly aligned with the hoop and axial directions of the pipe).

In the Assembly, the indenter is positioned in relation to the pipe: at the top and just in contact with the corner of the pipe. The indenter is rotated (if required) so that the specified radii are properly aligned with the longitudinal and circumferential directions.

4.2.5 Mesh

As the mesh is not uniform throughout the pipe and different edges require different seed sizes, the "findAt" function is again used in the script to define the mesh. Each of the edges is selected and assigned the seed size specified in the input variables. The edges that make up the partition are given a constant seed size, while the rest of the edges have a single bias to cause the mesh to become coarser in the areas of the pipe that are further away from the area of indentation. The elements are specified to be type S4R (4-node, linear shell elements with reduced integration) and the mesh controls are set so that the region inside the partition follows the medial axis algorithm and the rest of the pipe follows the advancing front algorithm. The use of the medial axis algorithm is more suitable for simple, structured regions, while advancing front should be chosen for areas with mesh transitions (Dassault Systèmes, 2014). This results in the combination of the medial axis algorithm used in the partition and the advancing front algorithm used

in the rest of the pipe to create an acceptable mesh that does not distort the elements far from their specified seed sizes and does not lead to any mesh errors or warnings due to element distortions. After the seeds, element type, and mesh controls are specified, the mesh is then generated using the script.

4.2.6 Interactions, Boundary Conditions and Steps

The next portion of the script sets up the interaction properties of the model for the surface-to-surface contact between the indenter and pipe. The tangential behavior is defined with “penalty” formulation and normal behavior is defined with “hard” pressure overclosure. The indenter surface is selected to be the master surface, while the outer pipe surface is selected to be the slave surface (using the findAt command for the region selections). The default options for contact controls, initializations, and stabilizations are selected.

Four boundary conditions are set-up in the initial step: X-symmetry, Z-symmetry, the indenter displacement, and the end boundary conditions. In the script, the appropriate regions are selected, and X-symmetry and Z-symmetry are respectively applied. These boundary conditions are propagated throughout the remainder of the steps without modification. The reference point of the indenter is selected, and the displacement boundary condition is applied. This sets up the boundary condition for a displacement to be applied in a future step. Lastly, two reference points are created, one at each end of the pipe in the center of the pipe cross-section. A fixed boundary condition is applied to these reference points and a coupling condition is used to tie the pipe edges to these reference points. The kinematic coupling condition is fixed in all directions except the radial direction. This combination of boundary conditions means that the pipe is fixed from movement in the vertical and axial direction as well as from rotation, but the nodes on the pipe end are free to move radially as a result of the pressurization. These boundary conditions mimic the connection between the modelled segment and the remainder of the pipe network.

The steps are then created in the following order: Indentation, Pressure-MOP, Pressure-Zero, and Pressure-MOP2. In the indentation step, the indenter is displaced downwards into the pipe by the indentation depth specified in the input variables. In the three pressure steps, a pressure cycle is applied. The entire internal surface of the pipe is selected and throughout the three pressure steps the maximum operating pressure is applied, zero pressure is applied, then maximum operating pressure is applied again. The maximum operating pressure was applied twice so that the maximum stress range could be extracted

accurately. According to Kainat et al. (2019), there could be an observed difference in the stress-strain path between the first applied pressure cycle and subsequent pressure cycles; however, after the first pressure cycle, the stress-strain response oscillates along the same path, given that the maximum applied pressure does not exceed the initially applied pressure.

The field output requests are set so that the two variables in Abaqus: “S” (stresses) and “PEEQ” (maximum equivalent plastic strain), are turned on and these outputs will be available for extraction during the post-processing phase.

4.2.7 Jobs

The final portion of the script creates the job. The job name is the same as the model name and the properties are adjusted so that maximum computer memory and processing is allocated to running the job (to minimize run time). At the end of the script, two command lines can be used to submit the job and wait for completion before moving to the next line of the script. If all of the script for model generation and job submission is run under a for loop, a batch of a large number of models can be submitted at once. This way, no human interference is required while the FEA models in the batch are continuously generated and submitted.

4.3 Results Extraction

After the models are created and run automatically in Abaqus using the Python script, the next major step is to efficiently extract the results from the output files. For this study, the displacement profiles (in both the longitudinal and circumferential direction) and the maximum equivalent plastic strain ($\overline{\varepsilon^p}$) must be extracted from each model. Next, the maximum delta sigma in the axial direction ($\Delta\sigma_{axial}$), which is the maximum difference in the axial stress component in the pipe when maximum operating pressure is applied (end of Pressure-MOP2 step) and when zero pressure is applied (end of Pressure-Zero step), is extracted. The maximum delta sigma in the hoop direction ($\Delta\sigma_{hoop}$) is also extracted, where $\Delta\sigma_{hoop}$ has the same definition as $\Delta\sigma_{axial}$, just with the hoop stress component rather than the axial stress component. According to Kainat et al. (2019), the principal stress component that governs a pipeline dent’s fatigue life typically aligns with either the axial or the hoop stress component of the stress tensor. As a

result, the $\overline{\varepsilon^p}$, as well as the $\Delta\sigma_{hoop}$ and $\Delta\sigma_{axial}$ were extracted for this study for practical assessment of the severity of dents from both a strain- and fatigue-based standpoint. The maximum of $\Delta\sigma_{hoop}$ and $\Delta\sigma_{axial}$, or $\Delta\sigma_{max}$, would be the value used for fatigue analysis, which is beyond the scope of this research.

4.3.1 Displacement Profiles

A separate script was written to extract the displacement profiles of the pipe in the longitudinal and circumferential directions once the finite element analysis was complete. The script uses a few of the input variables described in Table 8.

First, the output Abaqus file (ODB file) is opened using the model/job name as the identifier. A longitudinal path is created using the "Point List" path type. With this type of path, a linear path is automatically created between two specified coordinates; in this case, the path starts at (0, outer radius, 0) and ends at (0, outer radius, length of pipe). This way, the script can read from the input variables file and can extract the profile without user interference regardless of variation in pipe properties. A circumferential path is also created using the "Circular" path type and specifying three points on the arc: (0, outer radius, 0), (-outer radius, 0, 0), and (0, -outer radius, 0), which correspond to the top, side, and bottom of the semi-circle of the pipe edge closest to the indenter. The longitudinal and circumferential paths are highlighted in red in Figure 31.

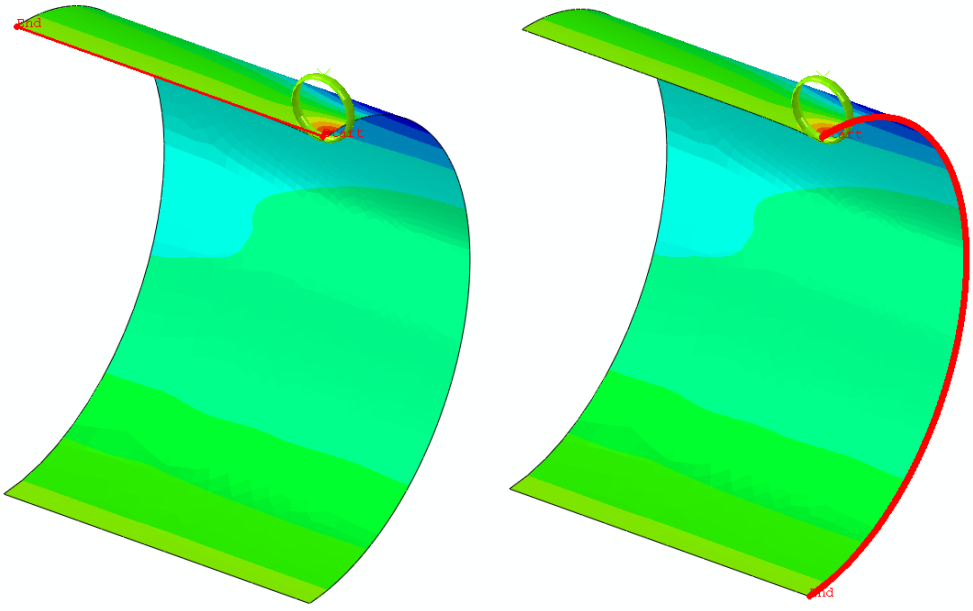

Figure 31: Longitudinal (left) and circumferential (right) paths generated by Python script

After the paths are generated, the reference coordinate system is changed to the cylindrical coordinate system so that the circumferential displacement profile is extracted properly. The XY data, where X is the distance along the path and Y is the displacement (in the radial) direction is extracted from both the longitudinal and circumferential paths. The XY data is then written to file. An example of the obtained data plotted is shown in Figure 32 and Figure 33. As only a quarter of the pipe was modelled with planes of symmetry, the profiles can be considered to be symmetrical about the Y-axis.

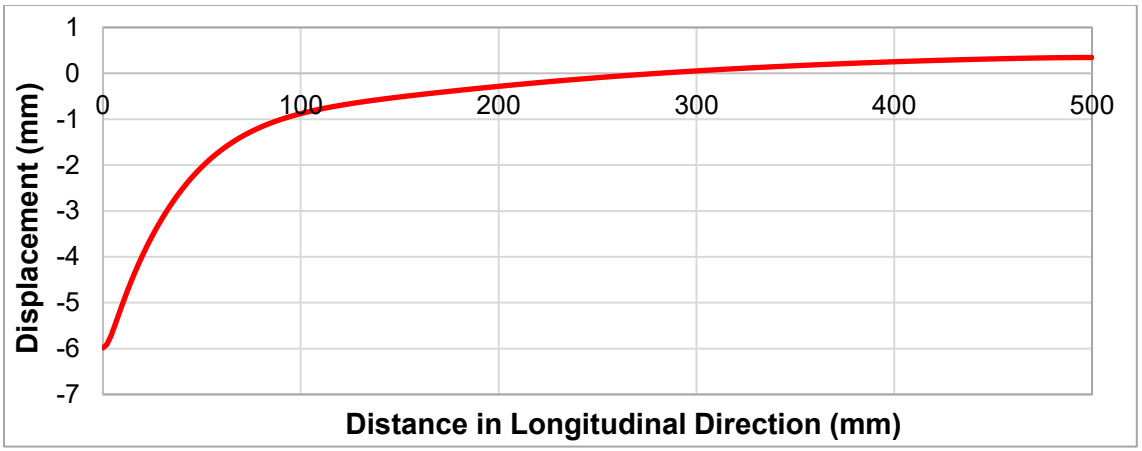

Figure 32: Example longitudinal profile extracted using Python script

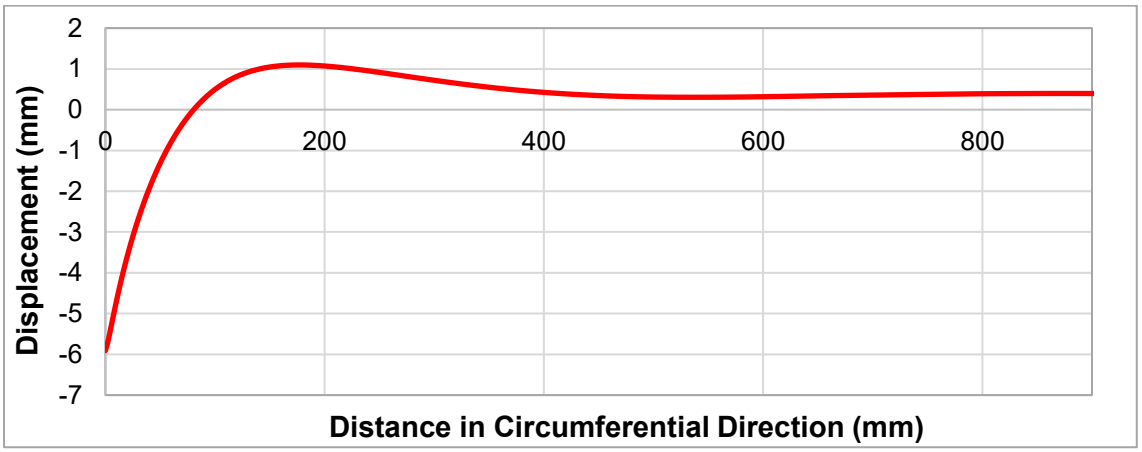

Figure 33: Example circumferential profile extracted using Python script

A “for” loop is utilized so that the script can cycle through a large number of models, extract the profiles from all of them, and save the displacement profiles in separate Comma Separated Values (CSV) files.

4.3.2 Strain and Stress Results

The final aspect of the automation used for this research was extracting the maximum $\overline{\varepsilon^p}$, $\Delta\sigma_{hoop}$ and $\Delta\sigma_{axial}$ results from the output files from the FEA. The maximum $\overline{\varepsilon^p}$ at the end of all steps (after Pressure-MOP2) was desired. The script navigates to the last frame in time within the last step and selects the "PEEQ" field output, which is the label for the maximum equivalent plastic strain in Abaqus. There are two variables created in the script: "maxPEEQ1" and "maxPEEQ2", which represent the top two maximum $\overline{\varepsilon^p}$ values found in the pipe elements during the last frame and step in time. A for loop is created so that the script cycles through all of the elements and checks whether the PEEQ value of the current element is greater than the PEEQ value stored in the maxPEEQ1 variable. If it is, then the new PEEQ value will replace the maxPEEQ1, and if not, then the PEEQ value of the next element will be checked. The PEEQ value of all elements will be checked. Once the for loop has reached the end of all the elements, the value stored in maxPEEQ1 is the maximum PEEQ value in the pipe. The process is repeated but this time with maxPEEQ2 and with the additional stipulation that the current value should be greater than the value of maxPEEQ2 but less than maxPEEQ1 in order to overwrite maxPEEQ2. At the end of both for loops, the top two values of maximum equivalent plastic strain in the pipe will be recorded. The two values are averaged, and this average value of maximum $\overline{\varepsilon^p}$ is written to an output CSV file after the model name is written (for labelling purposes).

For fatigue analysis, $\Delta\sigma_{hoop}$ and $\Delta\sigma_{axial}$ are desired but are not directly output in Abaqus. A temporary field is set up where all the stresses at the end of the step where maximum operating pressure is applied (Pressure-MOP2) are subtracted by the stresses at the end of the step where zero pressure is applied (Pressure-Zero). This creates a session step and a session field output called "Delta-Sigma". The hoop component of stress is first checked. All of the elements are looped through using the script (similar to the procedure described above for $\overline{\varepsilon^p}$) until the top two elements with the maximum $\Delta\sigma_{hoop}$ are identified. The average of the two maximum $\Delta\sigma_{hoop}$ values is calculated and written to the same CSV file in the same row as the maximum $\overline{\varepsilon^p}$ was written to. The process is repeated, but for $\Delta\sigma_{axial}$.

The reason that an average over two elements is taken for maximum $\overline{\varepsilon^p}$, $\Delta\sigma_{hoop}$ and $\Delta\sigma_{axial}$ is to obtain more accurate values in case there are stress concentrations in one element.

At the end of the script, a new line is written to the output file so that all of the results corresponding to one model will be written in the same row with the model name. An example output from five different models is shown in Figure 34.

Model-Name	MAX-PEEQ	MAX-DELTA-SIGMA-HOOP	MAX-DELTA-SIGMA-AXIAL
20OD-12W-500Le-10L-10C-1D	0.2470	513	482
20OD-12W-500Le-50L-10C-1D	0.2188	541	471
20OD-12W-500Le-100L-10C-1D	0.1955	492	447
20OD-12W-500Le-250L-10C-1D	0.1534	467	387
20OD-12W-500Le-500L-10C-1D	0.1216	490	363

Figure 34: Example output of strains and stresses from FEA models

The Python scripts for model generation and results extraction are provided in Appendix A.

4.4 Conclusions

The automation techniques presented in this chapter can be used to efficiently generate a large number of dent FEA models and subsequently extract the displacement, stress, and strain information from them after the analysis in the Abaqus software is complete. The primary benefits of utilizing automation to aid in the FEA analysis are speed, accuracy and consistency from the minimization of human intervention. Even with automation, however, a significant amount of time is still required to perform FEA for many dents due to the extensive computational time required to run the complex models.

In general, the button clicks and keyboard strokes that would be entered by a human to generate the FEA models and extract the data from them are programmed as Python script that can be read in by Abaqus. The aspects of the procedure that would require user input (for example, entering in the pipe's wall thickness) are read from a separate, pre-defined input variables text file.

The automation described in this chapter will be used in the remainder of this thesis to aid in the profile matching case study and in the development of the hundreds of models required for the ANNs.

CHAPTER 5: MATCHING FEA TO ILI DISPLACEMENT PROFILES

5.1 Objective

In this chapter, the effect of changing indenter sizes on the FEA displacement profiles, stresses, and strains is observed. In literature, the exact indenter geometry is known for FEA models used to compare against full-scale dent tests, such as the studies by Tiku et al. (2012), Shuai, Shuai, and Zhang (2018), and Ghaednia and Das (2018). However, for pipelines in operation, the only indication of the dent profile is provided by the ILI tool, but the exact size and shape of the indenter that created the indentation profile seen by ILI is unknown. Researchers such as Gossard, Bratton, Kemp, Finneran, & Polasik (2016) and Turnquist and Smith (2016) have built-in the exact dented geometry from ILI into the modelled pipeline in FEA, however, this does not capture any residual stresses or strains that can result from the indentation process and does not provide flexibility in modelling different loading sequences. In Kainat et al. (2019), it is concluded that the restraint condition (whether the dent is constrained and remains in contact with the pipe, or unconstrained and is removed after indentation), and sequence of whether pressurization occurs before or after indentation, can affect the strain and fatigue behavior of dents.

Operators seeking to analyze in-service dents would have available the ILI profile and would potentially be able to assume the loading sequence. For example, if the dent is located at the bottom of the pipe and has been reported by every ILI tool run on the line since the pipe has been in-service, it can be inferred that the dent is constrained and was formed during construction. Thus, the only unknown piece of information in order to model the dent using FEA would be the size and shape of the indenter.

In this chapter, a case study is presented where the FEA displacement profile is compared to the ILI displacement profile using two-dimensional cross sections and three-dimensional surfaces. The stresses and strains of various dent profiles are presented, as well as a proposed criterion against which the accuracy of a FEA profile can be assessed.

5.2 Methodology

The FEA for this chapter was performed following the General FEA Methodology described in Section 3.2. A model of the pipe was first generated in the Abaqus software based on the input geometric properties of the pipe: the nominal pipe diameter, nominal wall thickness, and length of the pipe section. A plain dent was modelled after a dent reported by an ILI tool on the Enbridge pipeline system that had the pipe properties shown in Table 9.

Table 9: Pipe properties for FEA models in Profile Matching study

Outside diameter (mm)	863.6
Wall thickness (mm)	7.14
Length of section (mm)	3000
Pipe grade	X52 (359 MPa)
Maximum operating pressure (MPa)	4.74

The dent profile from ILI in the longitudinal direction is shown in Figure 35 and in the circumferential direction in Figure 36. As the dent appeared to be roughly symmetrical in the longitudinal direction, the FEA model included only half of the length of the section where the ILI displacement deviated from nominal (3000 mm) and symmetry boundary conditions were used.

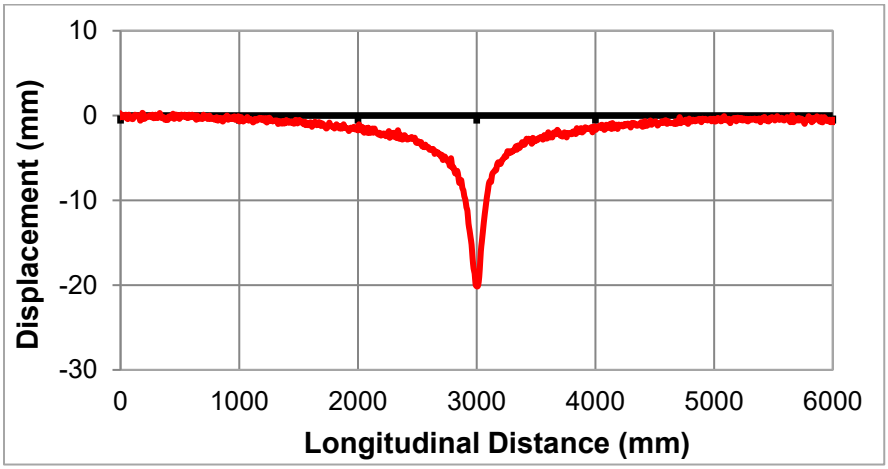

Figure 35: ILI dent profile in longitudinal direction

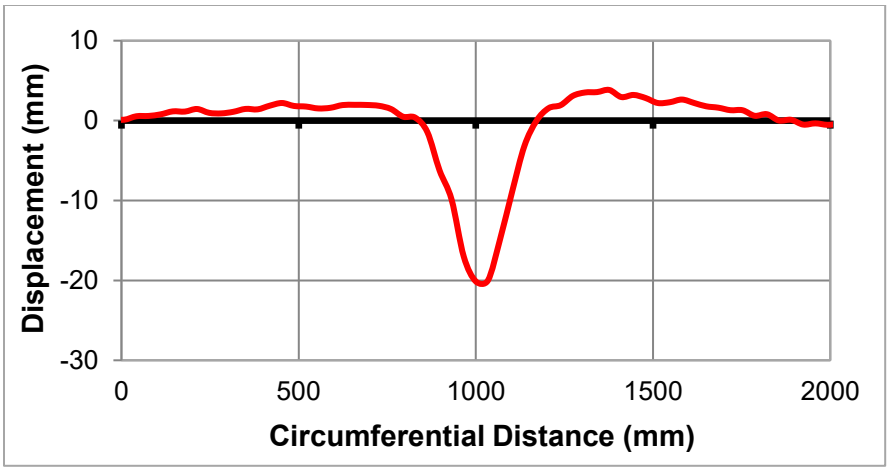

Figure 36: ILI dent profile in circumferential direction

For the models in this section, the value of yield strength was taken as the specified minimum yield strength (SMYS) of API X52 grade material. Figure 37 shows the true stress-true strain curve for the material, which was based on research done by Lin (2015) and approximated using the Ramberg-Osgood model (Equation 2) with E_s assumed to be 210 GPa (CSA, 2015), F_y equal to 359 MPa, and n equal to 12 (assumption determined in Chapter 3). The resulting values of yield stress versus plastic strain were input in the FEA software as the plastic properties of the material.

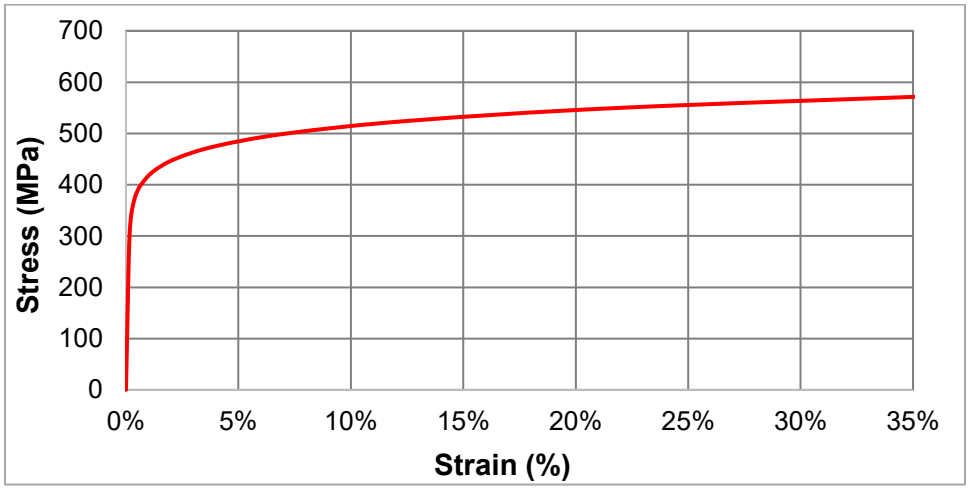

Figure 37: True Stress-True Strain Curve Approximation for X52 Grade Steel

As described in Chapter 3, the indenter shape used in this research is a torus. In the context of this research, the term “longitudinal radius” refers to the radius of the torus that aligns with the longitudinal direction of the pipe and “circumferential radius” refers to the radius of the torus that aligns with the circumferential direction of the pipe.

To get a close profile match between the ILI and FEA profiles, four different parameters can be optimized: longitudinal radius of the indenter, circumferential radius of the indenter, indentation depth, and length of the pipe section. In the case study presented in this chapter, only the radii of the indenter in both directions were modified, while the indentation depth and length of the pipe section were fixed. The depth of the modelled dent was 20 mm (2% of the OD) and the maximum operating pressure was 4.74 MPa (equal to 80% of SMYS from Equation 3). The modeled dent was constrained and was assumed to have formed during construction. As a result, the loading sequence applied was indentation, pressure to MOP, depressurization, and then pressure to MOP again.

The radius of the indenter was modified eleven times in the longitudinal direction and eleven times in the circumferential direction to investigate how the stress and maximum equivalent plastic strain ($\overline{\varepsilon^p}$) results change with indenter size. The indenter radius in one direction was fixed at the optimal value while the indenter radius in the other direction was modified.

The maximum equivalent plastic strain ($\overline{\varepsilon^p}$), and maximum stress range corresponding to one pressure cycle in the hoop direction ($\Delta\sigma_{hoop}$) and axial direction ($\Delta\sigma_{axial}$), were recorded for each indenter size. The maximum between $\Delta\sigma_{hoop}$ and $\Delta\sigma_{axial}$, which is written as the variable $\Delta\sigma_{max}$, was also recorded.

The generation of the FEA models and extraction of the results was performed using the automation described in Chapter 4.

5.3 Results

5.3.1 Two-Dimensional Profiles

For simplicity, the deformations found using FEA can be compared to two cross-sections of the ILI profile: the longitudinal and circumferential cross-sections that run through the most significant point (MSP), or minimum point, of the dent. By trial and error, it was determined that the indenter size that best matched the ILI profile had a radius of 150 mm in the longitudinal direction and a radius of 350 mm in the circumferential direction. The variation of profile shape with the size of indenter radius is shown in Figure 38 and Figure 39. Although 11 different indenter radii were tested in each direction, the profile shapes of only four of the radii are shown in each of Figure 38 and Figure 39. In the longitudinal and circumferential

directions, the profiles of the small, medium, and large indenters are shown in addition to the FEA profile of the closest profile match. This was done so that the differences in the profiles could clearly be discerned.

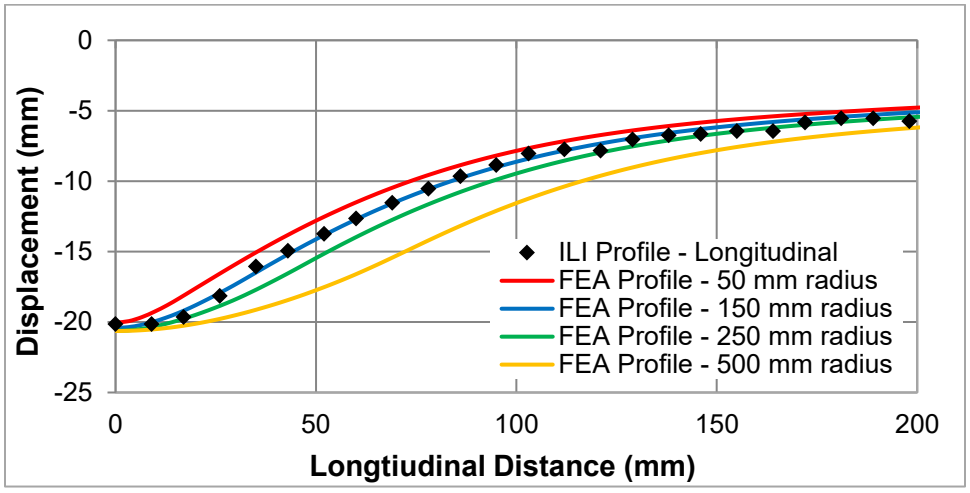

Figure 38: Variation of longitudinal profile with different indenter sizes

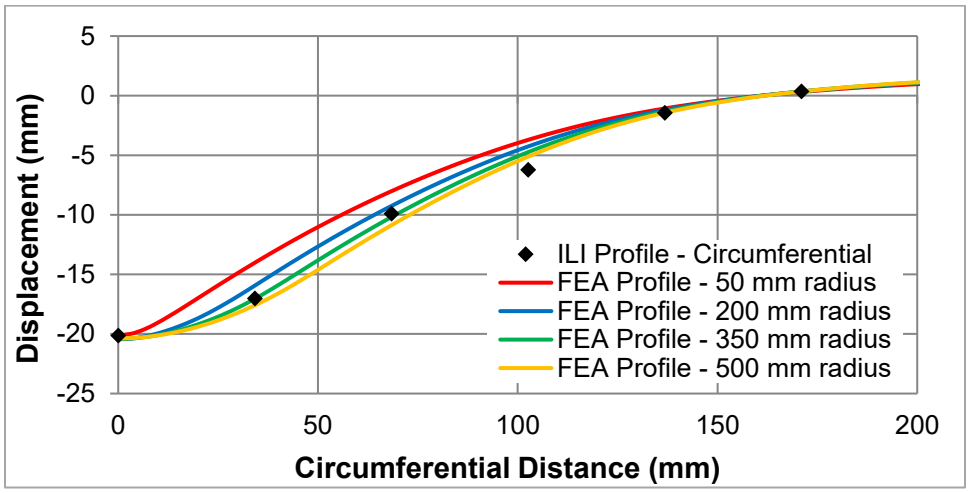

Figure 39: Variation of circumferential profile with different indenter sizes

The $\overline{\varepsilon^p}$ and $\Delta\sigma_{max}$ for each indenter size were also compared to the statistical measure, R-Squared (R^2). R-Squared is a statistical measure to evaluate the goodness of fit of a data set and in this case, indicates how closely the FEA profile aligns with the ILI profile. The R^2 values can only be between 0 and 1, with a number closer to 1 indicating a closer fit of the FEA data to the ILI data. In this case, R^2 is calculated as shown in Equation 4.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_{ILI} - y_{FEA})^2}{\sum_{i=1}^{n}(y_{ILI} - \bar{y}_{ILI})^2} \quad (4)$$

where y_{ILI} is the displacement of a coordinate as reported by the ILI tool

y_{FEA} is the displacement of the same coordinate as reported by the FEA model

$\bar{y}_{ILI}$ is the mean value of the set of ILI displacements

n is the total number of coordinates from which the FEA and ILI displacements were measured.

The $\overline{\varepsilon^p}$ versus R^2 by comparing both the longitudinal and circumferential profiles are shown in Figure 40 and Figure 41.

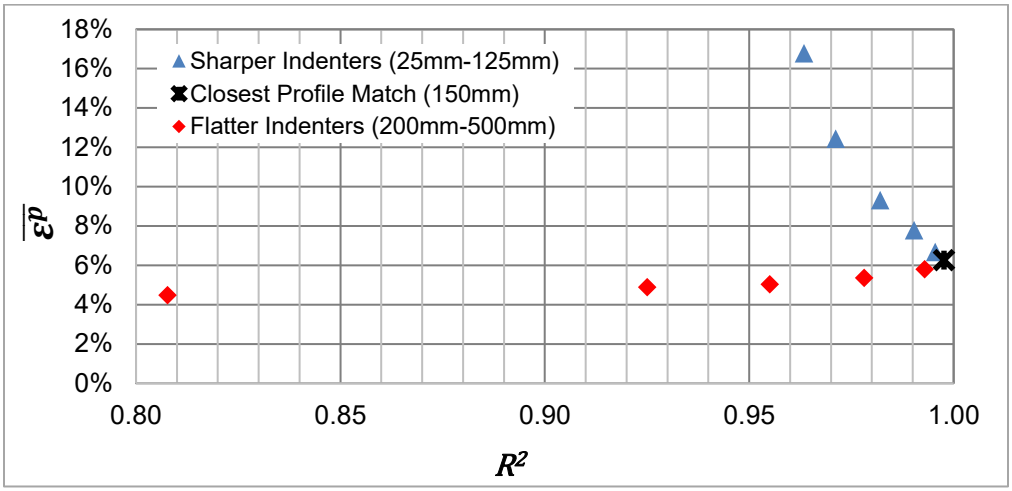

Figure 40: Maximum equivalent plastic strain versus R^2 in longitudinal direction

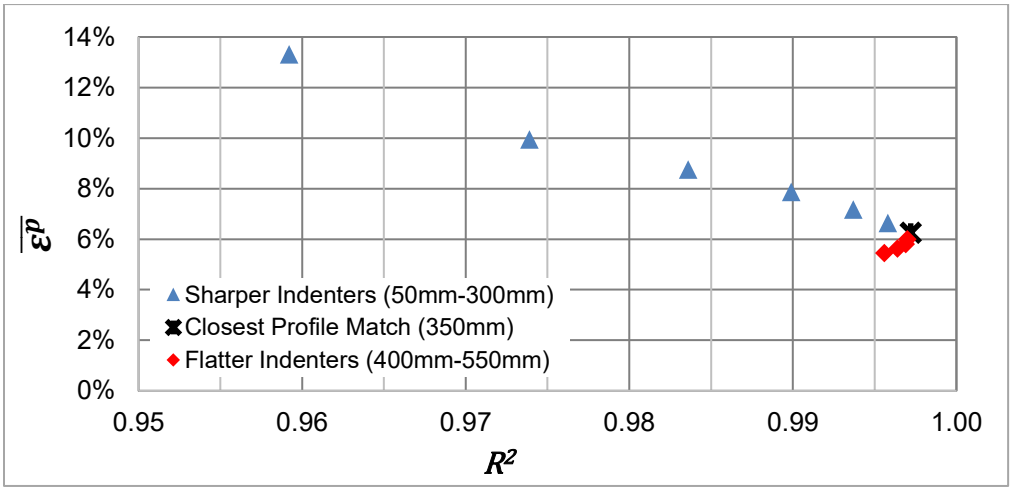

Figure 41: Maximum equivalent plastic strain versus R^2 in circumferential direction

The effect of different indenter radius sizes on the strain components was also observed. In Figure 42 and Figure 43, the change in total strain in the hoop and axial directions is shown, as well as the equivalent plastic strain as the indenter radius changed.

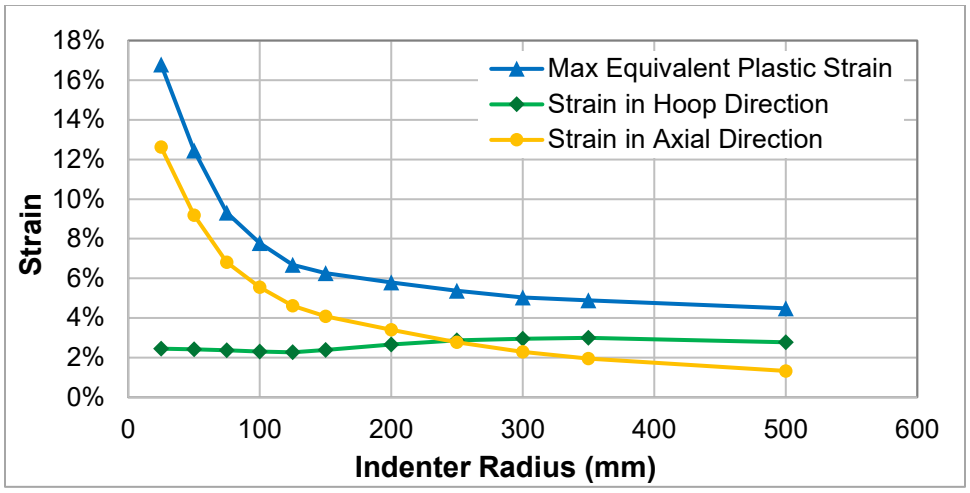

Figure 42: Maximum strain components with differences in longitudinal indenter radius

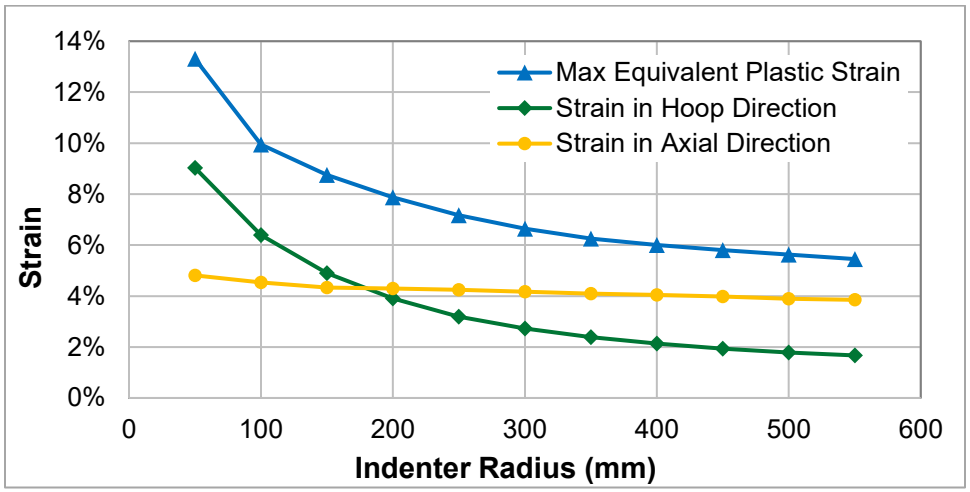

Figure 43: Maximum strain components with differences in circumferential indenter radius

The changes in $\Delta\sigma_{max}$ with changes in R^2 are demonstrated in Figure 44 for the longitudinal profiles and Figure 45 for the circumferential profiles.

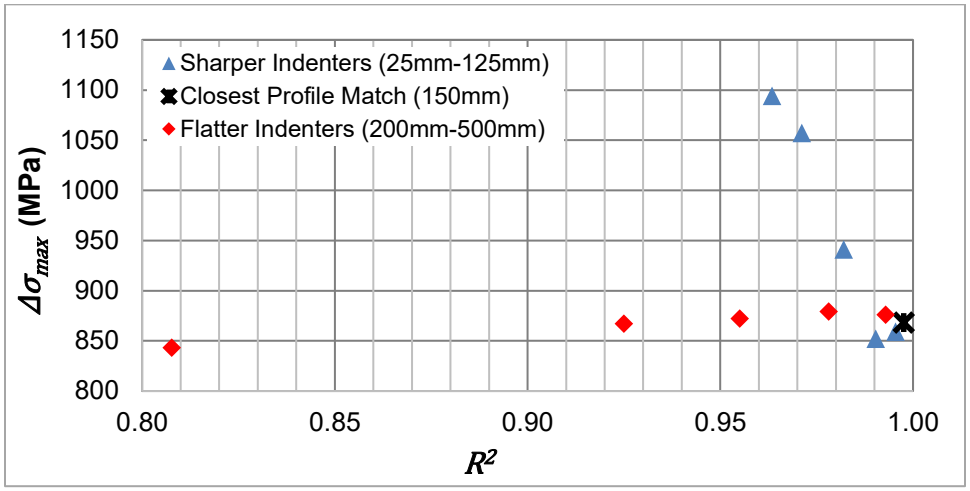

Figure 44: Variation of $\Delta\sigma_{max}$ with differences in R^2 in longitudinal direction

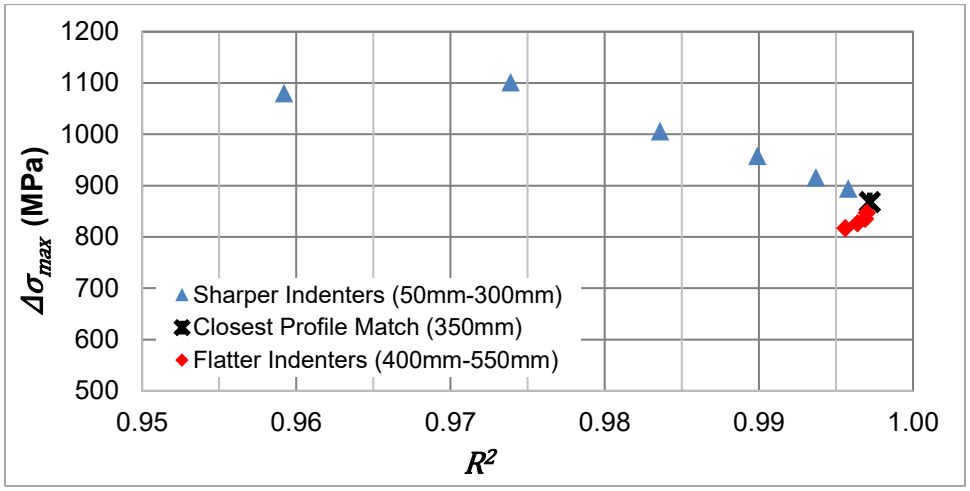

Figure 45: Variation of $\Delta\sigma_{max}$ with differences in R^2 in circumferential direction

The full results showing the effect of the indenter size on $\overline{\varepsilon^p}$, $\Delta\sigma_{max}$, and R^2 are shown in Table 10.

Table 10: Effect of Indenter Size on $\overline{\varepsilon^p}$, $\Delta\sigma_{max}$, and R^2

Longitudinal Radius (mm)	Circumferential Radius (mm)	$\overline{\varepsilon^p}$	$\Delta\sigma_{max}$ (MPa)	R^2
150	50	13.31%	1080	0.9592
150	100	9.95%	1101	0.9739
150	150	8.75%	1006	0.9836
150	200	7.87%	958	0.9899
150	250	7.17%	916	0.9937
150	300	6.64%	894	0.9958
150	**350**	**6.26%**	**868**	**0.9972**
150	400	6.00%	847	0.9970
150	450	5.80%	835	0.9969
150	500	5.63%	826	0.9964
150	550	5.45%	817	0.9956
25	350	16.79%	1094	0.9634
50	350	12.44%	1057	0.9711
75	350	9.32%	941	0.9820
100	350	7.78%	852	0.9903
125	350	6.68%	859	0.9955
150	**350**	**6.26%**	**868**	**0.9976**
200	350	5.80%	876	0.9929
250	350	5.37%	879	0.9781
300	350	5.04%	872	0.9550
350	350	4.89%	867	0.9250
500	350	4.49%	843	0.8077

5.3.2 Three-Dimensional Surfaces

Although an accurate profile match was found using the two-dimensional cross sections of the dent, the correlation of the three-dimensional surfaces was also investigated. For the example in this study, a surface was fit to the X, Y, and Z coordinates of the dent reported by the ILI tool and compared to the surface fit of the displacement coordinates from FEA. This was accomplished using the numerical computing software, MATLAB Version R2018a. The 3D surface fit of the ILI points is shown in Figure 46. As only a quarter of the pipe was modeled in this study, the ILI and FEA points of that quarter were simply mirrored for visualization purposes.

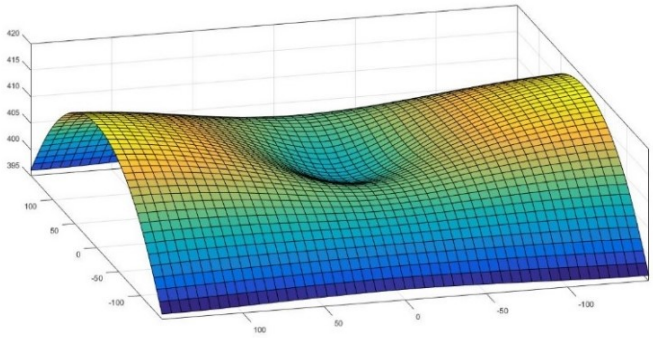

Figure 46: 3D surface fit to ILI coordinates

The FEA model with the closest profile match, as determined by the study, was used for the 3D comparison: 150 mm longitudinal radius and 350 mm circumferential radius. These radii were found using trial and error and comparing the two-dimensional cross-sections of the FEA to the ILI profiles.

The displacement coordinates of the ILI surface were subtracted from those of the FEA surface when both were applied to a common grid of X and Y points. Here, X and Y correspond to the longitudinal and circumferential directions, respectively. The error in the FEA surface based on this exercise is shown in the contour plot in Figure 47. In this figure, the (0, 0) coordinate corresponds to the MSP of the dent.

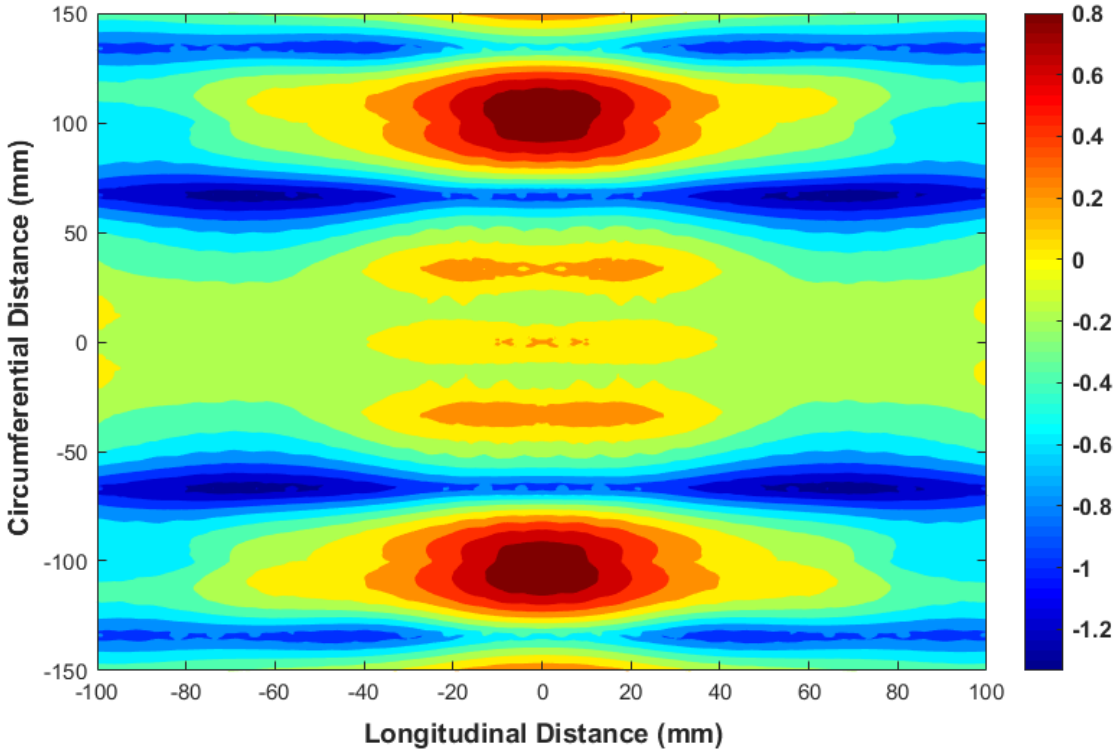

Figure 47: Contour plot of error between FEA and ILI surfaces

As seen in the plot, the absolute maximum error between the FEA and ILI points was 1.3 mm. The average error amongst all points was -0.5 mm, which indicates that the points in the FEA model were on average 0.5 mm shallower than what was reported by the ILI. High-resolution ILI tools reportedly have the capability to measure dent depth with a tolerance of 0.1 inch, or 2.54 mm (Michael Baker Jr., Inc., 2004). As the error between the FEA and ILI surfaces was much lower than the typical ILI depth tolerance for dents, it is confirmed that the indenter size used in FEA was appropriate to model the dent reported by the ILI tool.

In addition, Figure 47 shows that the smallest errors occurred in the region closest to the MSP (in the center of the plot). This is favorable as the location of maximum $\overline{\varepsilon^p}$ and $\Delta\sigma_{max}$ was found to be near the MSP. As the maximum values are of greater interest for assessment, it is desired to have a more accurate profile match in the area nearest the MSP.

5.4 Discussion

From the results, it is seen that the indenter radius can be adjusted in both directions by trial and error until the FEA displacement profiles align closely with the ILI displacement profiles by comparing cross-sections through the MSP of the dent.

5.4.1 Profile Shapes

In Figure 38 and Figure 40 above, it is seen that small differences in indenter size create large differences in profile shape in the longitudinal direction. For example, the flattest indenter modelled (500 mm radius), had a value of R^2 much lower than 1, indicating that the profile shape was very different from the reference ILI shape. Thus, it would be easy to match any different sharpness of ILI profile in the longitudinal direction using the described FEA method, for smooth, round dent shapes.

In comparison, Figure 39 and Figure 41 suggest that at a certain point, increases in indenter size have minimal effect on profile shape in the circumferential direction. This seems to occur once the indenter radius is equivalent to or greater than the pipe radius. Once the indenter radius was greater than the pipe radius (i.e. the indenter radius was 500 mm, which was greater than the pipe radius of 431.8 mm), then the increase in radius of the torus indenter in the circumferential direction (compared to the 350 mm indenter

radius) did not cause the dent profile to be any flatter. In Figure 48, the circumferential profiles resulting from the 350 mm indenter radius and 500 mm indenter radius were converted to cylindrical, then rectangular coordinates so that the circular cross-section of the indented pipe could be viewed. As seen in the figure, there is negligible difference between the deformed shape caused by the 350 mm indenter radius and the 500 mm indenter radius across the entire cross-section of the pipe. This indicates that if the reference ILI profile in the circumferential direction is flatter than the profile that would be produced with a very large torus indenter, then the profile can no longer be modelled in FEA using the torus indenter and a different indenter shape must be chosen to adequately match the ILI profile.

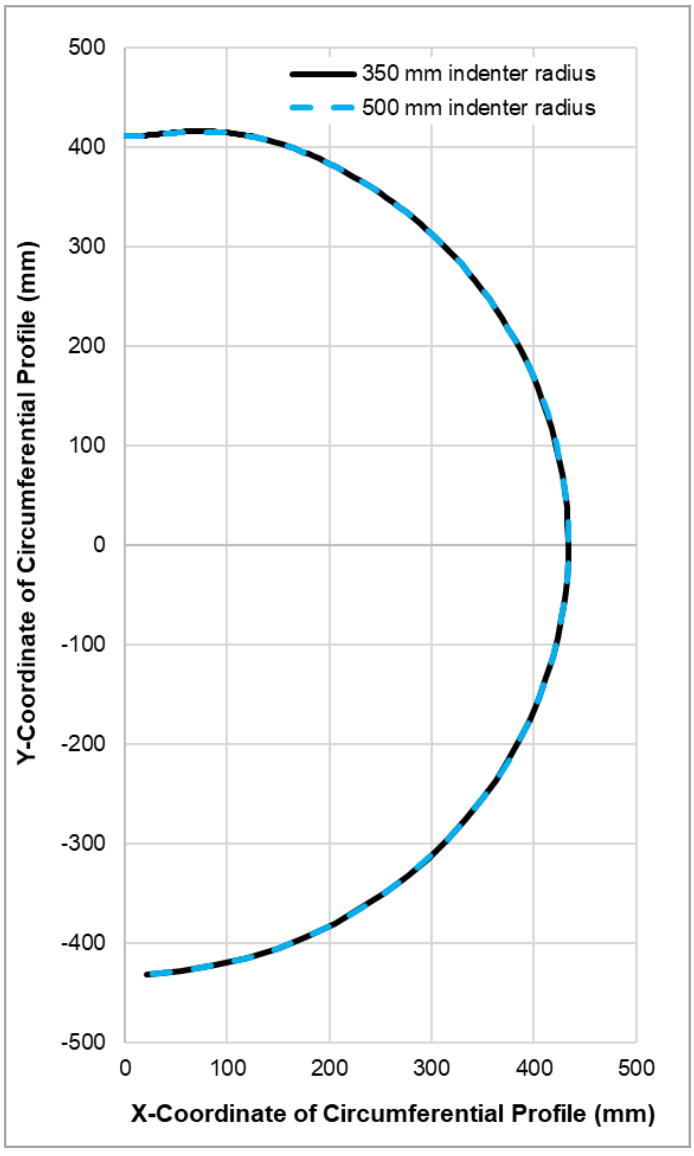

Figure 48: Comparison of circumferential profiles for 350 mm and 500 mm indenter radius

5.4.2 Indenter Impact on Strains and Stresses

The results for this case study demonstrate that the strains and stresses of a dented region are very sensitive to the shape and size of the indenter, particularly for sharp dents. It is seen from Figure 40, Figure 41, Figure 44, and Figure 45 that the sensitivity of both $\overline{\varepsilon^p}$ and $\Delta\sigma_{max}$ with small changes in the profile shape are much higher for the sharper indenters. The $\overline{\varepsilon^p}$ and $\Delta\sigma_{max}$ both increase significantly with each indenter that is subsequently smaller than the indenter for the closest profile match. On the other hand, the $\overline{\varepsilon^p}$ and $\Delta\sigma_{max}$ values stay fairly constant as the indenters become flatter, with radii larger than the indenter used for the closest profile match.

A hypothetical, round and smooth indenter shape was used here that could be modelled using FEA software to determine the strains and stresses. In real-life, however, dents are formed by rocks, ground movement, or construction equipment, which can lead to the dents having complex shapes, such as containing multiple apexes. The study here demonstrated the need for rigorous profile matching when using FEA to determine the stresses and strains within a dented region. A “close” profile match where the FEA profile is slightly flatter than the true profile can lead to non-conservative estimates of strains and stresses and can lead to overlooking defects for excavation and repair, whereas, modelling the dent as sharper than it really is could result in a conclusion that is too conservative and lead to an inefficient use of resources.

From Figure 42 and Figure 43, it is observed that adjustments in the radius size in each direction have a primary effect on the strain in that same direction. In Figure 42, the change in strain in the axial direction governs the change in the equivalent plastic strain while the hoop strains stay mostly constant across the different indenter sizes. This behavior indicates the importance of matching the dent profile in both the longitudinal and circumferential cross-sections to ensure an accurate strain assumption is being estimated.

5.4.3 Profile Matching Criteria

The statistical measure, R^2, was used in this study to provide a quantitative measure of goodness of fit of the FEA profiles to the ILI profiles. However, as only one example ILI profile was presented here, additional study is recommended to set an “R-Squared threshold”. The R^2 between the FEA and ILI profiles would have to be greater than this threshold to accept the profile match and use the FEA results for integrity

assessment. In this study, the highest R^2 for the different circumferential indenters was 0.9972 for the 350-mm indenter and was 0.9976 for the longitudinal 150-mm indenter. As these R^2 values were the highest out of the indenters modelled, it was concluded that the corresponding indenter produced the profiles that fit closest to the ILI profiles. As the sensitivity of stress and strain is higher for sharper dents, it is also recommended that the R^2 threshold be a function of dent sharpness. The R^2 threshold should be higher for sharper dents than for flatter dents.

In this study, the ILI profile was taken to be the "true" profile. However, there are usually uncertainties associated with the tool measurements, suggesting that a reported ILI profile could potentially vary from the real dent profile. For example, an indenter with a longitudinal radius of 25 mm caused the pipe to experience a $\overline{\varepsilon^p}$ of 17%, whereas, the $\overline{\varepsilon^p}$ with a 75-mm indenter was only 9%. If the true profile should have been modelled using the 25-mm indenter, but the ILI reported profile shape was more similar to the FEA profile produced with a 75-mm indenter, the predicted $\overline{\varepsilon^p}$ would be non-conservative using a solely deterministic analysis. This study supports the need for reliability science to be used in conjunction with FEA modelling, as proposed by Hassanien et al. (2016). A probabilistic reliability assessment considers the many different uncertainties input into the FEA model, including the measurement uncertainty in the ILI profile, which can lead to a more comprehensive and accurate assessment of the integrity of the dented region.

The above results and analysis pertain only to the specific combination of pipe properties used in this example. Results may vary with different pipe properties although the methodology with which to model the dents using FEA and compare the profiles to ILI would be the same as described here. Further investigation is required on a larger sample size of dents with varying pipe properties to draw more robust conclusions.

5.4.4 Two-Dimensional Profiles versus Three-Dimensional Surfaces

The example here indicates that determining the indenter size to use in FEA by comparing only the two-dimensional cross-sections, through the minimum point of the dent, is adequate to achieve a close profile match with ILI. For plain dents with simple profiles, 2D profile matching is a sufficient and efficient method to develop an accurate FEA model based on ILI. Although more time- and resource-intensive, the 3D profile matching method discussed in this section can be leveraged for complex deformations, such as

asymmetric, multi-apex, or off-axis dents or dents associated with ovalities. For these cases, the two-dimensional cross-sections may not be enough to capture the full shape of the dent and the entire 3D surface of both the FEA and ILI coordinates may have to be evaluated using the contour plot approach.

5.5 Conclusions

In this chapter, a method to match the dent profiles resulting from FEA to the information given by ILI tools is presented. An indenter in the shape of a ring torus is recommended for FEA modelling; however, very flat dents in the circumferential direction cannot be modelled with this shape.

The case study presented in this chapter demonstrates that the sensitivity of the stresses and strains in the indented region is much higher for sharper dents than flatter dents. This indicates that an error threshold should be developed in future work to ensure a sufficiently close fit between the FEA and ILI dent profiles. The measure used to assess the goodness of fit of the FEA to the ILI data in this case study was R-Squared *(R^2).* It is proposed that an R^2 threshold could be used as a profile matching criterion; the R^2 of the FEA values compared to the ILI values must be greater than the R^2 criteria before the profile match is deemed acceptable for further analysis of the dent. The results of this study indicate that the R^2 threshold value for a specific dent must be based on the sharpness of the dent. Future research should determine if the criteria could also be affected by other factors, such as the restraint condition of the dent, the maximum operating pressure, or the pipe properties.

The results from FEA can be used for strain or fatigue-based integrity assessment methods. Due to the potential error in the FEA results from uncertainties in the input parameters, particularly the measured ILI dent profile described here, the use of reliability techniques in combination with the FEA results is recommended.

The scope of this study was limited to smooth and symmetric, plain dents. In this study, it was determined that matching the dent profile from FEA to the ILI profile can be successfully performed using the 2D cross-sections in the longitudinal and circumferential directions that pass through the minimum point of the dent. As 3D profile matching is more time and resource consuming, the 2D simplification can be used for smooth and symmetric plain dents. Further investigation can include the use of the 3D profile matching

method for complex dent shapes such as asymmetric dents, dents with multiple apexes, and dents with angular orientations.

As the approach presented in this chapter to match the FEA profile to the ILI profile is a trial and error process, this can be quite time-consuming, particularly for complicated shapes. In Chapter 6, an ANN will be used to directly predict the indenter size required from the ILI profile and other known parameters, such as the outer diameter. The 21 unique models in the case study presented in this chapter provide validation points in Chapter 6.

CHAPTER 6: DEVELOPMENT OF ARTIFICIAL NEURAL NETWORKS

6.1 Objective

As demonstrated in Chapter 5, the strain and stress behavior of dented pipe can vary significantly due to changes in dent shape. However, the process of matching the displacement profile obtained from an FEA model to that reported by an ILI tool can be a time-consuming, iterative process. This coupled with the fact that a single FEA model for dent assessment can be computationally expensive due to the complex nature of the analysis makes FEA impractical from a time and resource perspective for system-wide dent analysis.

The objective of Chapter 6 is to demonstrate the feasibility of using artificial neural networks to efficiently and accurately estimate the strains and stresses of a dented region given only the input pipe properties and the ILI profile of the dent. In this chapter, two case studies are described. In the first example, 63 FEA models are automatically generated where all properties are kept constant between the models except for the indenter radius in both the longitudinal and circumferential direction. The results are extracted from the FEA models and this information is used to train the ANNs. The 21 models presented in Chapter 5, a dataset independent of that used to train the ANNs, are used as a validation set to evaluate the accuracy of the ANNs. The second case study uses a different set of pipe properties than used in the first example. The indenter radii and dent depth are adjusted in 216 FEA models and the results are used to train the ANNs. The results obtained from the ANNs are compared to what was obtained using FEA for a validation set of 100 models in terms of both accuracy and speed.

6.2 Case Study 1: Comparison to Results from Chapter 5

6.2.1 Methodology

6.2.1.1 Building Database

The first step required to develop the artificial neural networks was to collect the data required for training. A total of 63 FEA models were run, where the pipe properties were held constant at the values provided in Table 8 (in Section 5.2). The indenter radius was adjusted 7 times in the longitudinal direction and 9 times in the circumferential direction; Table 11 shows the different values used.

Table 11: List of Different Indenter Radii used in FEA Models

Indenter Radius in the Longitudinal Direction (mm)	10
	50
	100
	250
	500
	1000
	2000
Indenter Radius in the Circumferential Direction (mm)	10
	25
	50
	75
	100
	250
	350
	500
	1000

After the FEA models were run, the 2-dimensional displacement profiles were extracted, through the MSP of the dent in the longitudinal and circumferential directions. As the profiles would need to be used in a consistent format to train the ANNs, the horizontal distances (in either the longitudinal or circumferential direction) from the MSP to the profile at 11 different depth increments were recorded. For example, the extracted points are shown as the red diamonds in Figure 49 and bolded in Table 12. The percentage of depth increments were held constant across the different FEA models.

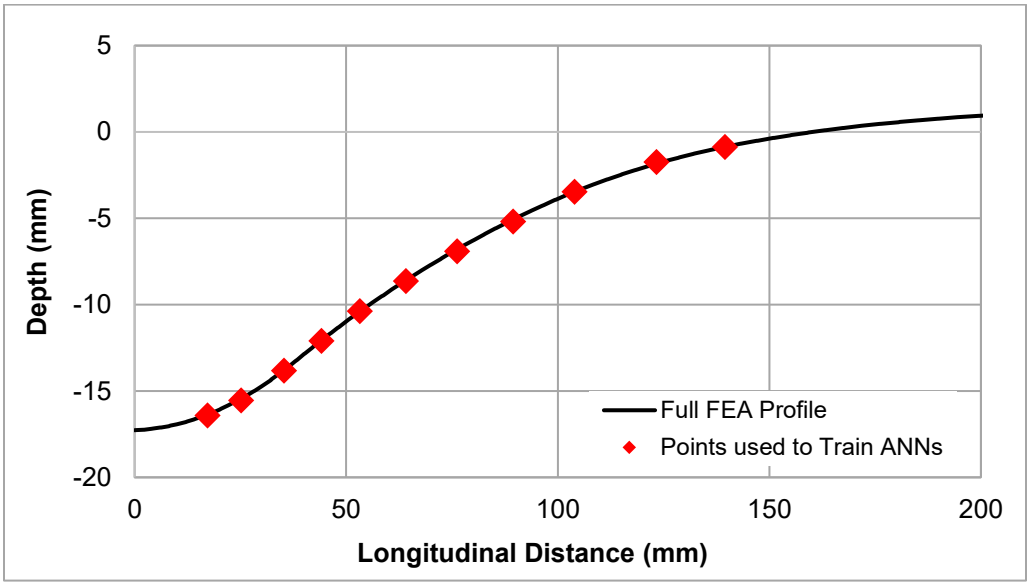

Figure 49: Example of points on FEA profile used to train ANNs

Table 12: Percentage of Depth Increments and X Values used to Train ANNs (Example)

% of Depth	Depth (mm)	X value (used to train ANNs) (mm)
95	-32.8	**13**
90	-31.1	**19**
80	-27.6	**29**
75	-24.2	**40**
60	-20.7	**51**
50	-17.3	**65**
40	-13.8	**80**
30	-10.4	**98**
20	-6.91	**117**
10	-3.45	**141**
5	-1.73	**157**

Therefore, the inputs to training the neural networks to find the indenter radius in each direction were the 11 consistent points along the displacement profiles (i.e. the red diamonds shown in Figure 49). The maximum $\overline{\varepsilon^p}$, $\Delta\sigma_{hoop}$, and $\Delta\sigma_{axial}$, were also recorded from each FEA model.

6.2.1.2 Creating and Training the ANNs

Five separate neural networks were trained using MATLAB Version R2018a, with the desired outputs of: the indenter radius in the longitudinal direction, the indenter radius in the circumferential direction, the maximum $\overline{\varepsilon^p}$, the maximum $\Delta\sigma_{hoop}$, and the maximum $\Delta\sigma_{axial}$. The Neural Net Fitting application within MATLAB was used to create and train the ANNs as well as evaluate their performance.

The type of ANN created using the software was a two-layer feed-forward network designed to fit multi-dimensional mapping problems (The MathWorks, Inc., 2018a). The tool randomly divides the input samples into three groups: training, validation, and testing. The training group has been set to use 80% of the total number of samples, where the network's weights and biases are adjusted based on this group. The validation group uses 15% of the samples and the error on this group is monitored during training. In the initial phase of training, the error on the training and validation sets typically decreases until a certain point when the error on the validation set starts to increase. This behavior during neural network training is called overfitting. Significant overfitting can be avoided by having a large enough sample size to train the network, although the definition of an adequate size varies for different applications and is difficult to predict

before training of the network begins (The MathWorks, Inc., 2018a). Training will cease when the error begins increasing; however, the network weights and biases from the point of minimum error will be saved in the network. Only 5% of the sample was set aside for testing. This is because independent tests will be conducted outside of the MATLAB toolbox (comparison to the independent results from Chapter 5 for this section) and the testing set has no effect on network training.

The next step in setting up the network architecture is to set the number of neurons in the network's hidden layer. The number of hidden neurons affects the performance of the network, but the optimal value varies for different applications (The MathWorks, Inc., 2018a). For this study, the optimal number of hidden neurons was determined by trial and error, where the number of hidden neurons that resulted in the lowest error was chosen for the network (as indicated by the lowest Mean Squared Error (MSE) value and closest R^2 value to 1, where the MSE represents the average squared difference between the outputs and targets and R^2 measures the correlation between the outputs and targets, as described previously in Chapter 5). An example of this process for the longitudinal indenter radius network is summarized in Table 13, where 3 hidden neurons were chosen for the network.

Table 13: Variation of Error for Differing Numbers of Hidden Neurons

Number of Hidden Neurons	MSE	R^2
1	326.4	0.9965
2	79.0	0.9991
3	**63.6**	**0.9993**
4	73.8	0.9991
5	73.5	0.9992
6	75.2	0.9992
7	74.4	0.9991
8	69.4	0.9993

There are three options provided in the software for the training algorithm: Levenberg-Marquardt, which is recommended for most problems, Bayesian Regularization, which is recommended for noisy or small problems, and Scaled Conjugate Gradient, which is recommended for large problems. The three algorithm options, which are described in more detail in Chapter 2, were tested and the algorithm that resulted in the smallest errors was chosen for the network. For the longitudinal indenter radius network, a

comparison between the three training algorithms is shown in Table 14; the Bayesian Regularization algorithm was found to be most suitable for this network.

Table 14: Variation of Error for Different Training Algorithms

Training Algorithm	MSE	R
Levenberg-Marquardt	150.3	0.9984
Bayesian Regularization	**63.6**	**0.9993**
Scaled Conjugate Gradient	388.1	0.9996

A diagram of the neural network architecture for the longitudinal indenter radius network is shown in Figure 50 which shows that the network is a feed-forward network with 11 input variables (the 11 displacement profile points), 3 hidden neurons in the 1 hidden layer, 1 output layer, and 1 output variable (the indenter radius in the longitudinal direction). The default transfer functions built into the MATLAB toolbox were used, which were a sigmoid transfer function in the hidden layer and a linear transfer function in the output layer (The MathWorks, Inc., 2018a). The number of inputs, hidden neurons, and outputs for the 5 different ANNs is shown in Table 15. The Bayesian Regularization training algorithm was found to result in the least error for all 5 ANNs.

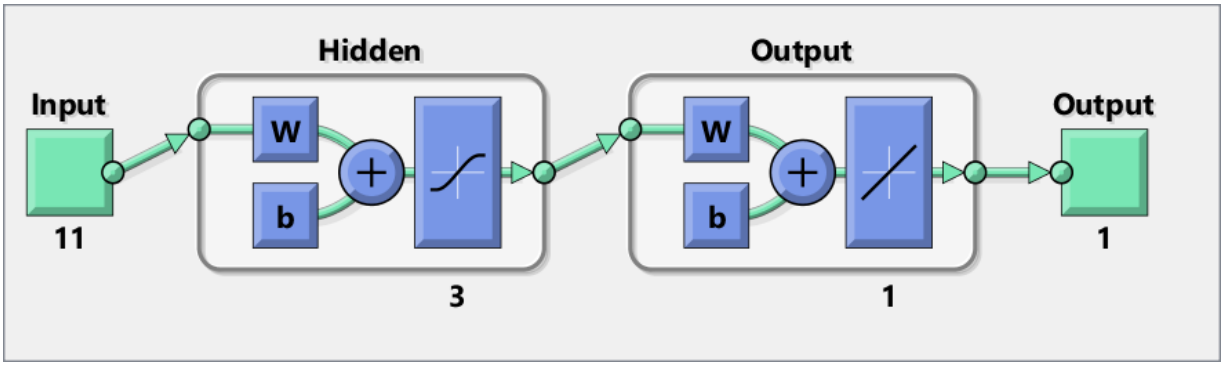

Figure 50: Architecture for longitudinal indenter radius network (The MathWorks, Inc., 2018b)

Table 15: Inputs, Hidden Neurons, and Outputs for the ANNs

Number of Inputs	Inputs	Number of Hidden Neurons	Number of Outputs	Output
11	Points along displacement profile in longitudinal direction	3	1	Indenter radius in longitudinal direction
11	Points along displacement profile in circumferential direction	8	1	Indenter radius in circumferential direction
2	Indenter radius in both directions	8	1	Maximum $\overline{\varepsilon^p}$
2	Indenter radius in both directions	5	1	Maximum $\Delta\sigma_{axial}$
2	Indenter radius in both directions	6	1	Maximum $\Delta\sigma_{hoop}$

After being given a certain input, an ANN is trained by comparing the output to the given target. If the network output does not match the target, the weights, or connections, between the neurons of the network are adjusted. This process is repeated until the error on the network outputs is minimal (i.e. the error fails to decrease over several subsequent iterations). The MATLAB Neural Net Fitting toolbox has been programmed to automatically iterate through this process, which is demonstrated by Figure 51 (The MathWorks, Inc., 2018a).

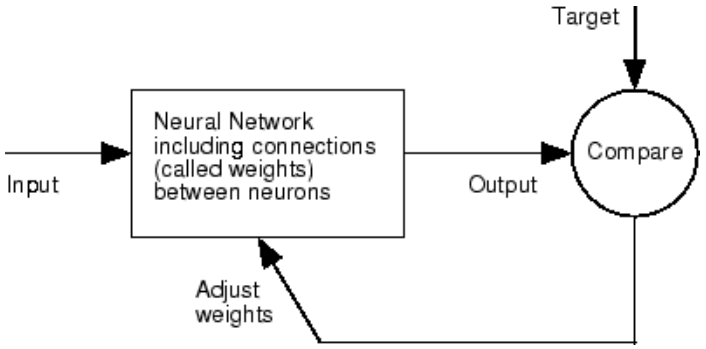

Figure 51: Process of training an artificial neural network (The MathWorks, Inc., 2018a)

After the networks were trained, independent testing was performed by feeding the inputs from the 21 FEA models described in Chapter 5 into the networks and comparing the outputs from the networks to the indenter radii, $\overline{\varepsilon^p}$, and stresses presented in Table 10, from Section 5.3. The 21 models from Chapter 5 had different indenter radii values than were used to train the ANNs, although the values were within the bounds of the training range (i.e. the validation set had indenter radii ranging from 25 mm to 550 mm, while

the training set had indenter radii ranging from 10 mm to 2000 mm). The separate validation data set was used to test if the ANNs could predict accurate results when presented with new data. It was attempted to directly predict the $\overline{\varepsilon^p}$ and stresses from the displacement profiles (without creating separate ANNs to find the indenter radii), but this resulted in a poor comparison between the ANN-predicted results and the FEA-produced results. This was likely due to the high number of inputs and outputs that the neural network would have had to fit between, and this would have required a much larger training sample size for accurate results.

The overall process described in this section (including building the database, training the ANNs, and validating the ANNs), is summarized in Table 16. In the table, symbols are used to show where the same values are used. For example, the dark red dash shows that points from the displacement profile obtained from the FEA models are used as input data to train the ANN where the desired network output is the indenter radius. On the contrary, the yellow triangle pointed to the right shows that the known indenter radius values used in the FEA models were used as the Targets to train the ANN where the desired output was the longitudinal radius.

Table 16: Summary of Inputs and Outputs for Each Step of ANN Development Process

Step	Analysis Mechanism	Inputs	Outputs
Building Database (using Abaqus)	• FEA	▶ Indenter radius in longitudinal direction ▼ Indenter radius in circumferential direction Δ Indenter depth	▬ Displacement profile O Maximum $\overline{\varepsilon^p}$ ▲ Maximum $\Delta\sigma_{hoop}$ and $\Delta\sigma_{axial}$
Creating and Training the ANNs (using MATLAB)	• Training algorithm	**Input Data (to present to network)** ▬ 11 consistent points along displacement profile in longitudinal direction **Targets (desired network output)** ▶ Indenter radius in longitudinal direction	♦ 5 separate, trained neural networks
	• Training algorithm	**Input Data (to present to network)** ▬ 11 consistent points along displacement profile in circumferential direction **Targets (desired network output)** ▼ Indenter radius in circumferential direction	
	• Training algorithm	**Input Data (to present to network)** ▶ Indenter radius in longitudinal direction ▼ Indenter radius in circumferential direction Δ Indenter depth **Targets (desired network output)** O Maximum $\overline{\varepsilon^p}$	
	• Training algorithm	**Input Data (to present to network)** ▶ Indenter radius in longitudinal direction ▼ Indenter radius in circumferential direction Δ Indenter depth **Targets (desired network output)** ▲ Maximum $\Delta\sigma_{hoop}$ and $\Delta\sigma_{axial}$	
Validating the ANNs (comparison to FEA outputs using unity plots)	♦ 5 separate, trained neural networks	• 11 consistent points along "ILI" profile in longitudinal and circumferential direction (case study) • "ILI"-reported dent depth	• Indenter radius (longitudinal direction) • Indenter radius (circumferential direction) • Maximum $\overline{\varepsilon^p}$ • Maximum $\Delta\sigma_{hoop}$ and $\Delta\sigma_{axial}$

6.2.2 Results

The points from the displacement profiles of the 21 models presented in Chapter 5 were given as inputs to the ANNs where the desired outputs were the indenter radii in the longitudinal and circumferential directions. This process is used to imitate the real-life scenario where the ILI-reported dent profile is known, but the indenter radius required to model the feature in FEA is unknown. The outputs from the ANNs were then compared against the true indenter radii used to achieve the displacement profiles in FEA and the results are shown in the plots in Figure 52 and Figure 53.

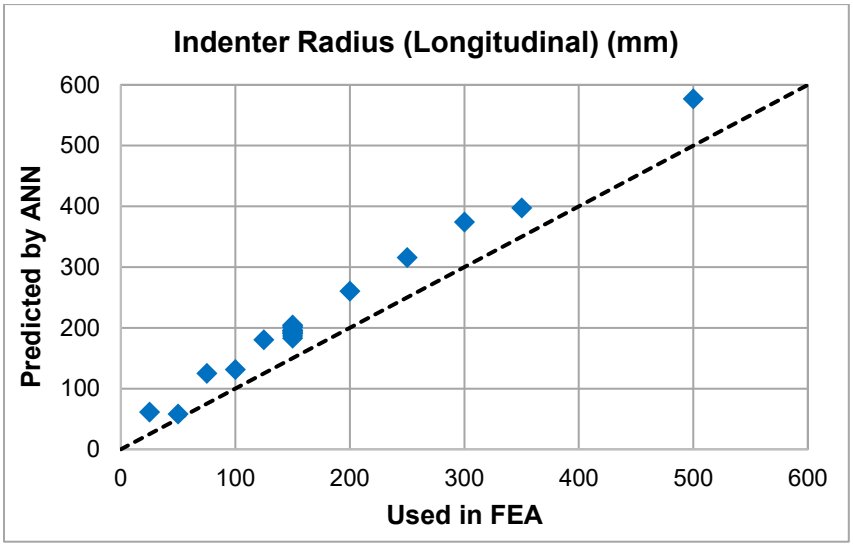

Figure 52: Longitudinal indenter radii predicted by ANN versus radii used in FEA

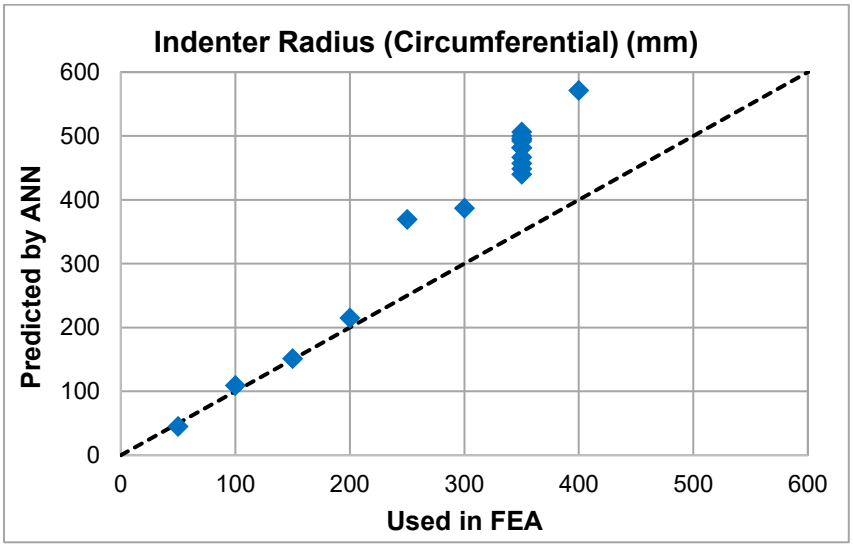

Figure 53: Circumferential indenter radii predicted by ANN versus radii used in FEA

Once the indenter radii were predicted using the ANNs, the radii were then used as inputs to 3 other ANNs to predict the maximum $\overline{\varepsilon^p}$, the maximum $\Delta\sigma_{hoop}$, and the maximum $\Delta\sigma_{axial}$.

For comparison, the analytical equations recommended in ASME B31.8 Appendix R were used with the FEA profiles and pipe properties of the 21 models created in this study. Figure 54 shows the maximum $\overline{\varepsilon^p}$ predicted from the ANN and the $\overline{\varepsilon^p}$ predicted using the ASME B31.8 equations plotted against the $\overline{\varepsilon^p}$ found using FEA. In Figure 54, the $\overline{\varepsilon^p}$ from the alternative method is plotted on the y-axis, while the $\overline{\varepsilon^p}$ from FEA is plotted on the x-axis, with the dashed black line through the center of the plot indicating that the points on the x- and y-axis are equal. As the blue diamonds, which represent the comparison between the $\overline{\varepsilon^p}$ found from the ANNs against the $\overline{\varepsilon^p}$ from FEA, are much closer to the dashed black line than the red diamonds (comparison between ASME B31.8 and FEA), this indicates that the ANNs were more accurate than the ASME B31.8 equations at predicting $\overline{\varepsilon^p}$, with the FEA $\overline{\varepsilon^p}$ used as the benchmark for accuracy.

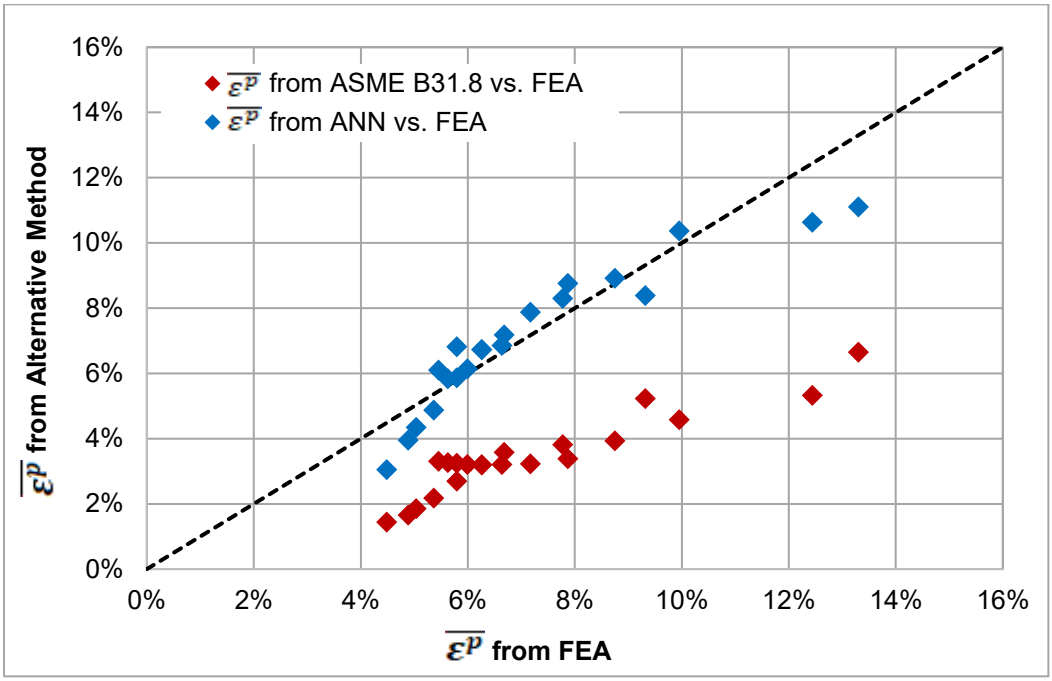

Figure 54: Comparison of $\overline{\varepsilon^p}$ from ASME B31.8 and from ANN versus $\overline{\varepsilon^p}$ from FEA

In Figure 55 and Figure 56, the maximum $\Delta\sigma_{axial}$ and maximum $\Delta\sigma_{hoop}$, respectively, are shown, where the results from ANN are compared against the results from FEA. As the points are all close to the dashed unity line down the center of the plot, this indicates that the results predicted by the ANNs were close to the results found from FEA.

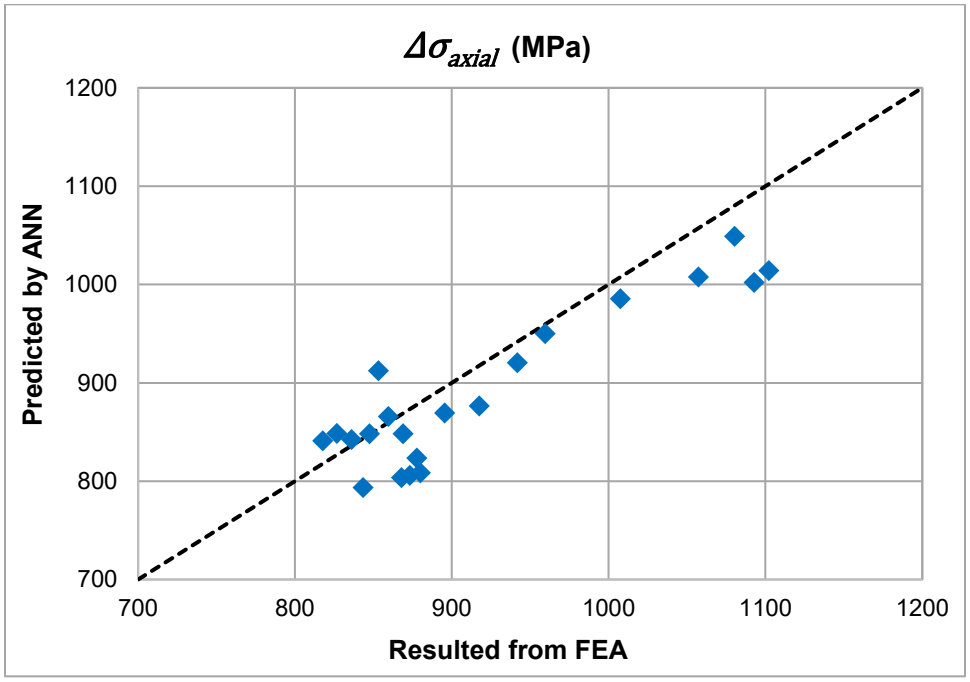

Figure 55: $\Delta\sigma_{axial}$ results from ANN versus results from FEA

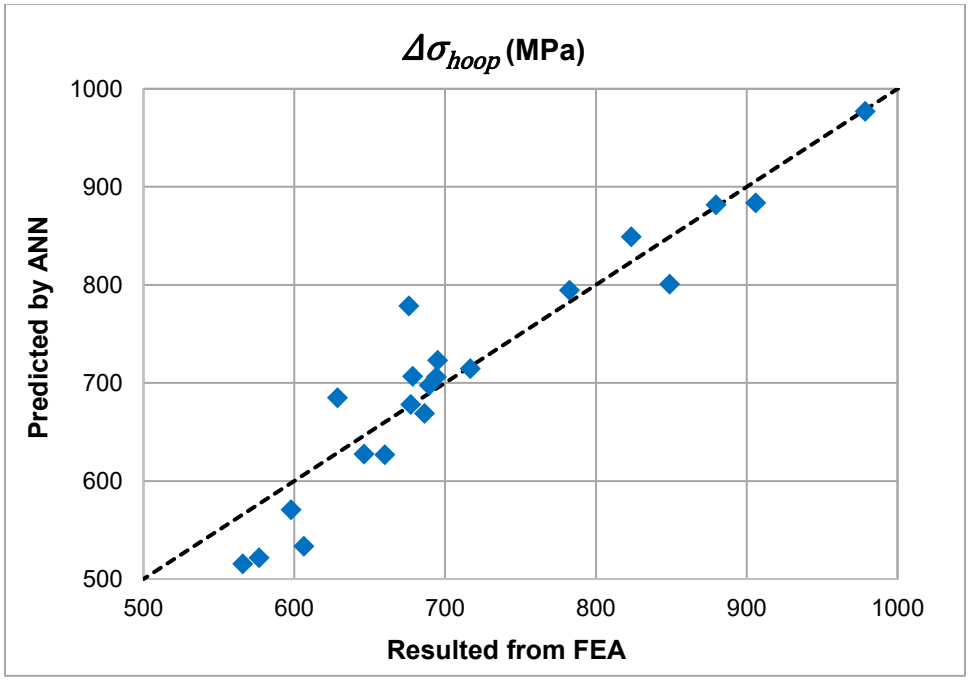

Figure 56: $\Delta\sigma_{hoop}$ results from ANN versus results from FEA

6.3 Case Study 2: Application to Larger Dataset

6.3.1 Methodology

The process described in Section 6.2.1 was repeated in this section but with a larger training dataset where an additional variable, the indentation depth, was also modified. The properties that were held constant across the models is shown in Table 17, while the properties that were adjusted is shown in Table 18. The true stress-true strain curve approximation for X52 steel was shown in Figure 35 in Section 5.2 and the maximum operating pressure shown in Table 17 is equal to 80% of SMYS calculated using Equation 3. As the indenter radius was modified 6 times in both the longitudinal and circumferential directions and the indentation depth was modified 6 times, a total of 216 models were used to train the ANNs.

Table 17: Constant Pipe Properties for FEA models to Train ANNs

Outside diameter (mm)	812.8
Wall thickness (mm)	7.14
Length of section (mm)	1000
Pipe grade	X52 (359 MPa)
Maximum operating pressure (MPa)	5.04

Table 18: List of Different Indenter Properties used in FEA Models to Train ANNs

Indenter Radius in the Longitudinal Direction (mm)	50
	75
	100
	250
	350
	500
Indenter Radius in the Circumferential Direction (mm)	50
	75
	100
	250
	350
	500
Indentation Depth (%OD)	1
	2
	3
	4
	5
	6

Using trial and error again, the Bayesian Regularization training algorithm was found to result in the least error for all 5 ANNs. A summary of the number of inputs, hidden neurons, and outputs for each of the trained ANNs for Case Study 2 is shown in Table 19.

Table 19: Inputs, Hidden Neurons, and Outputs for the ANNs in Case Study 2

Number of Inputs	Inputs	Number of Hidden Neurons	Number of Outputs	Output
12	Points along displacement profile in longitudinal direction and indentation depth	10	1	Indenter radius in longitudinal direction
12	Points along displacement profile in circumferential direction and indentation depth	20	1	Indenter radius in circumferential direction
3	Indenter radius in both directions and indentation depth	20	1	Maximum $\overline{\varepsilon^p}$
3	Indenter radius in both directions and indentation depth	35	1	Maximum $\Delta\sigma_{axial}$
3	Indenter radius in both directions and indentation depth	35	1	Maximum $\Delta\sigma_{hoop}$

For validation, 100 models were generated where the indenter radii in both directions and the indentation depth were randomly selected between the bounds of the training set (i.e. the indenter radius in each direction was randomized between 50 mm and 500 mm and the indentation depth was randomized between 1% OD and 6% OD). The pipe properties of the validation set were the same as the training set, as presented in Table 17.

6.3.2 Results

The results predicted from the ANNs were plotted against the results obtained from FEA for the randomized validation set of 100 models. The comparisons for the longitudinal and circumferential radii are shown in Figure 57 and Figure 58, respectively. In the figures, the results predicted by the ANN are plotted on the y-axis, while the results from FEA are plotted on the x-axis; data points along the diagonal line indicate equal results from the two methods.

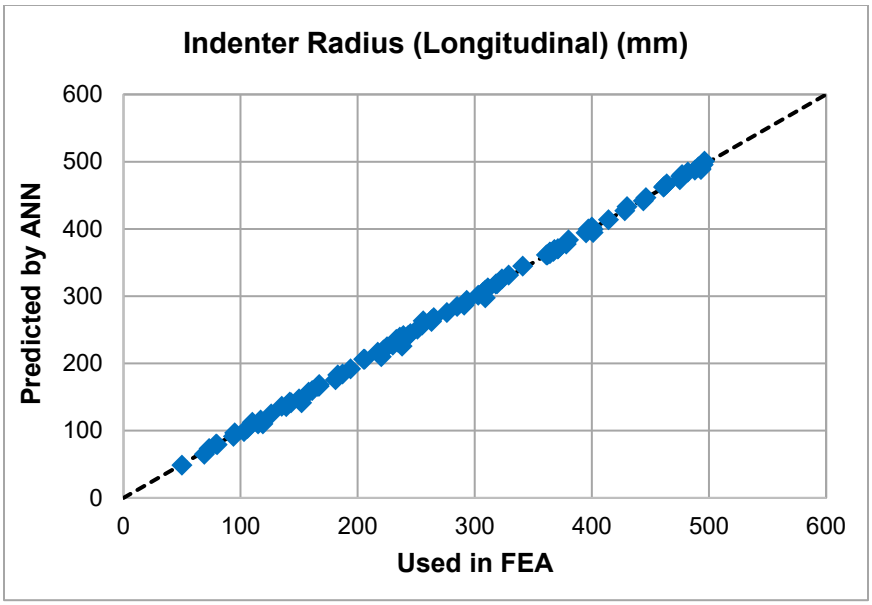

Figure 57: Longitudinal indenter radii from ANN versus radii used in FEA for Case Study 2

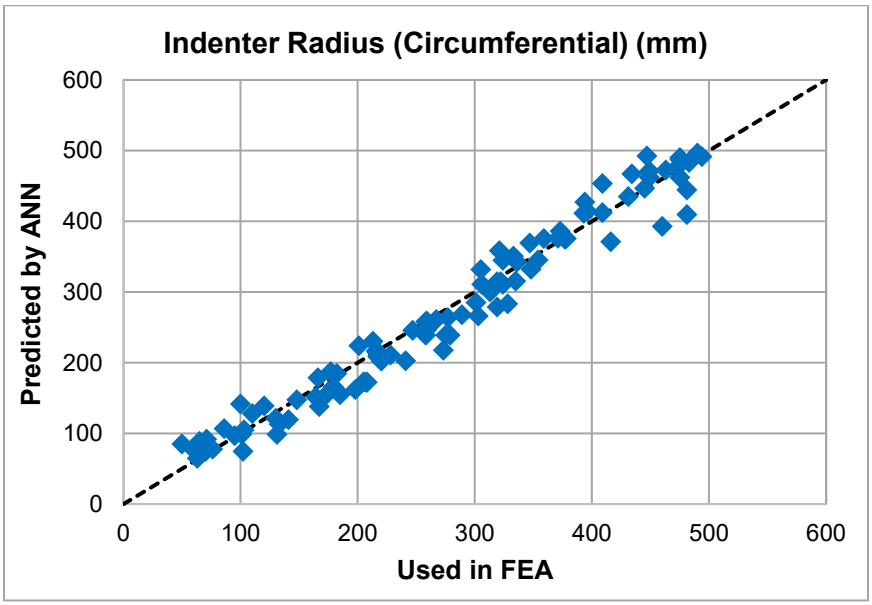

Figure 58: Circumferential indenter radii from ANN versus radii used in FEA for Case Study 2

In Figure 59, the $\overline{\varepsilon^p}$ results estimated by the ANN are plotted against the $\overline{\varepsilon^p}$ values extracted from FEA. For the same dataset of 100 models, the $\overline{\varepsilon^p}$ was also determined using the analytical equations in ASME B31.8, as well as using the analytical methodology described by Okoloekwe (2017) and are plotted against the $\overline{\varepsilon^p}$ from FEA in Figure 59. The maximum $\Delta\sigma_{axial}$ and maximum $\Delta\sigma_{hoop}$ results are shown, respectively, in Figure 60 and Figure 61.

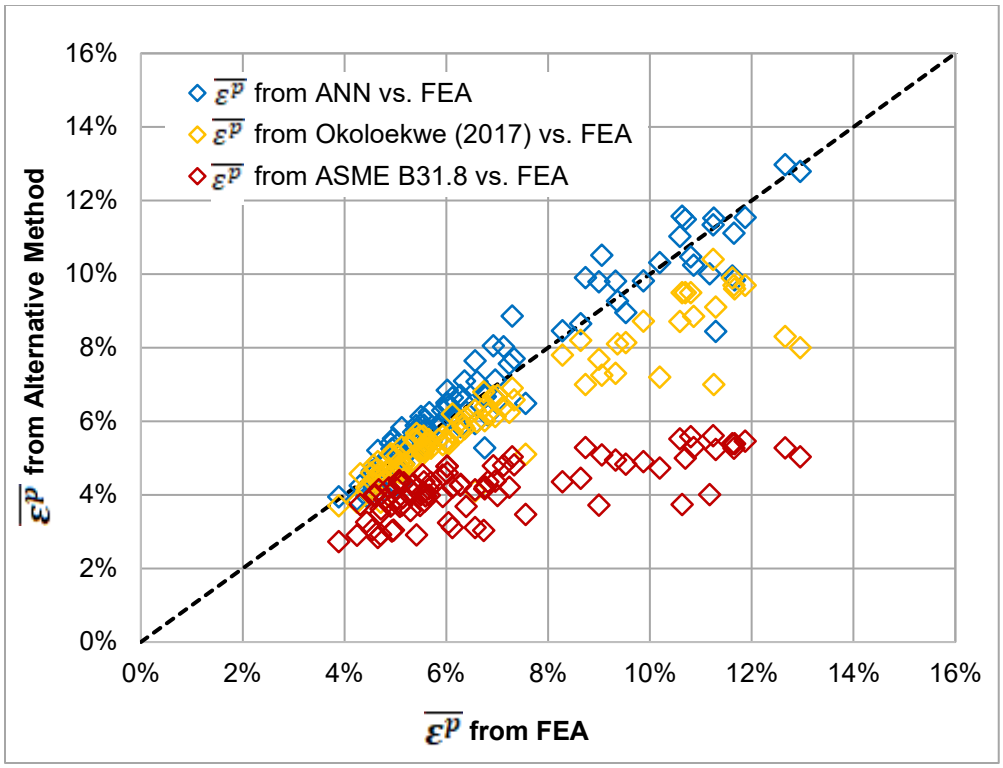

Figure 59: Comparison of $\overline{\varepsilon^p}$ from ASME B31.8 and ANN versus FEA for Case Study 2

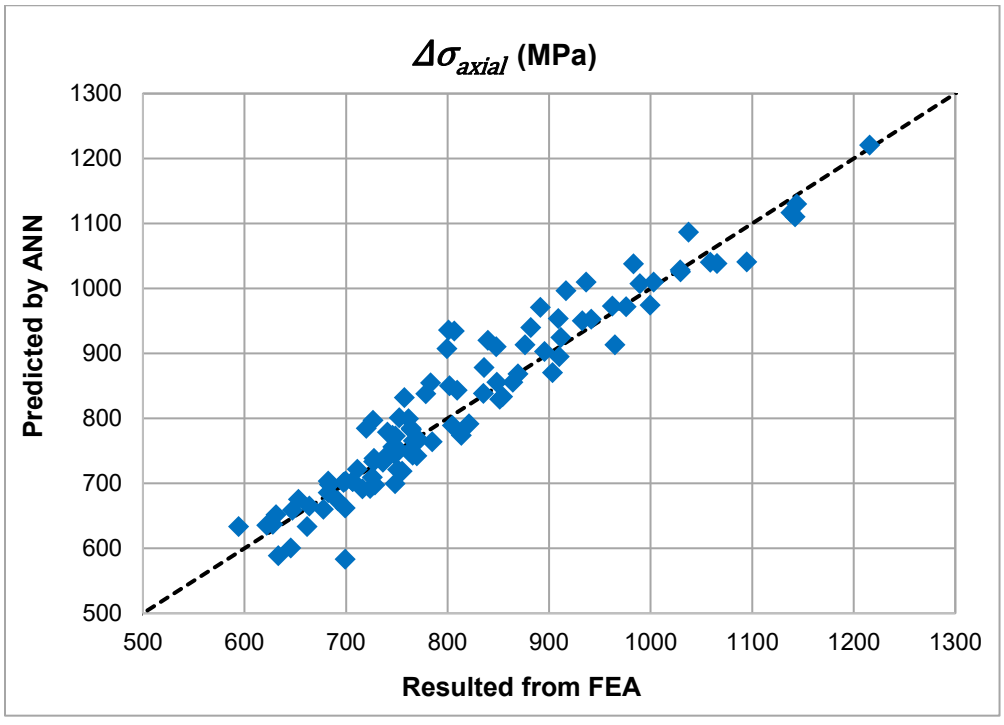

Figure 60: $\Delta\sigma_{axial}$ from ANN versus FEA for Case Study 2

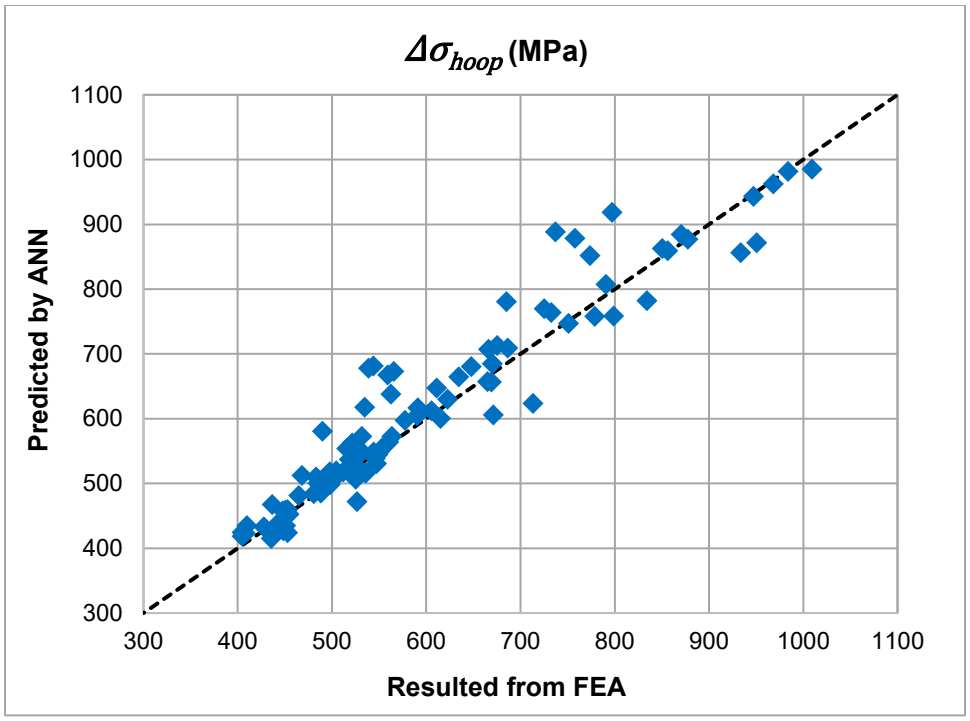

Figure 61: $\Delta\sigma_{hoop}$ from ANN versus FEA for Case Study 2

In terms of time required for this process, the FEA models used to train the ANNs in this chapter took, on average, 5 minutes of computational time to complete. For Case Study 2, the computational time required to run all 216 training models was roughly 18 hours. Once the 100 validation models needed to be analyzed, however, the ANNs provided results in less than 1 minute. On the other hand, using FEA alone requires the time-consuming profile matching process described in Chapter 5. The amount of time required for this process is difficult to quantify accurately as it is subjective, however, an accurate profile match for one dent took approximately 30 minutes to achieve (including both computational time of running the models and user time to assess that the match was not close enough and another trial was required). Applying this to the validation dataset, the amount of time required to analyze the 100 features using FEA would be approximately 50 hours. These results are summarized in Table 20.

Table 20: Comparison of Time between using FEA and ANN versus FEA Only

Task	FEA and ANN	FEA Only
Training the Database	18 hours	N/A
Obtaining Results	1 minute	50 hours
Total Time	**18 hours**	**50 hours**

6.4 Discussion

The results from both case studies demonstrate that the use of ANNs is feasible to obtain accurate stresses and strains for dented regions of pipelines. Figure 52 and Figure 53 show that the ANNs predicted similar radii as what was found using trial and error (as demonstrated in Chapter 5), however, they were predicted in far less time. For Case Study 2, the longitudinal indenter radius predicted by the ANN was the exact same as the indenter radius used in the FEA model to generate the displacement profile for almost all 100 validation cases, as shown in Figure 57. The prediction of the circumferential indenter radius by the ANN produced slightly greater error compared to the prediction of the longitudinal indenter radius when both were plotted against the FEA radius used in their respective directions. This supports the findings from Chapter 5 that changes in the indenter radius have less of an impact on the displacement profile in the circumferential direction than in the longitudinal direction due to the pipe having a circular cross section. However, the error for the circumferential indenter radius predicted by the ANN was minimal and did not significantly impact the final stress and strain results.

Figure 54 and Figure 59 show that the ASME B31.8 equations produce non-conservative results for $\overline{\varepsilon^p}$ when compared to FEA, which is in agreement with what other researchers have found (Noronha et al., 2010; Okoloekwe, 2017). Okoloekwe (2017) attributed the fact that the strain predictions using the ASME B31.8 equations may be underestimated to the fact that the equations ignore the circumferential membrane strains and radial strains, which can contribute to the global strain state within the dented region. When comparing the results from the analytical methodology proposed by Okoloekwe (2017) to FEA, the results were more accurate than using the ASME B31.8 equations but were slightly non-conservative compared to FEA. When observing the results from both case studies presented in this chapter, the $\overline{\varepsilon^p}$ estimated by the ANN align closely with the $\overline{\varepsilon^p}$ obtained from FEA with the added benefit of the estimation from the ANN occurring as quickly as the analytical equations are computed.

There is no method currently published in a standard or recommended practice to compute the $\Delta\sigma_{max}$ in a dent using analytical equations. The $\Delta\sigma_{max}$ is required to estimate the remaining fatigue life of a dent and can be useful for ranking the dent's relative severity against others (i.e. the dent with the highest $\Delta\sigma_{max}$ out of a sample with constant material and operating properties would have the shortest fatigue life and would be considered more injurious than the other dents in the sample). The ANNs were trained to

estimate the $\Delta\sigma_{axial}$ and $\Delta\sigma_{hoop}$ (the maximum of the two is the $\Delta\sigma_{max}$) with the indenter radii in both directions and the indentation depth as inputs. In Figure 55, Figure 56, Figure 60, and Figure 61, the ANNs proved to produce accurate results for $\Delta\sigma_{axial}$ and $\Delta\sigma_{hoop}$ (as would be determined from FEA).

Once the ANNs are trained, then the amount of time required to run new inputs through the network and obtain results is mere seconds. The time-consuming part of the process is running the large number of FEA models needed to train the ANNs. However, if the automation techniques for model generation and results extraction are used as described in Chapter 4, then the models would require minimal user intervention and would simply require extensive computational time. In addition, the ANNs would only have to be trained once to cover all pipe properties and potential dent properties that could be seen on a pipeline system. Any new features that are reported could then be analyzed using the already trained ANNs. Thus, the time investment required to train the ANNs, while initially significant, would pay off over time as future analyses could be performed instantly. The time comparison between the two processes summarized in Table 20 shows that the use of FEA and ANNs is far more efficient than using FEA alone. The effect would be even more dramatic if applied to a larger dataset of features requiring analysis, for example, all dent features across an entire pipeline system.

Although the ANN results from Case Study 1 proved to be fairly accurate, the ANN results in Case Study 2 were even closer to their FEA counterparts and almost equivalent in most cases. This demonstrates that a large training set that adequately covers the full range of input variables is required for accurate results. In addition, the indentation depth was varied for Case Study 2 but kept constant for Case Study 1. This demonstrates that the modification of additional variables that were held constant in the case studies, such as pipe diameter and wall thickness, should not affect the accuracy of the results as long as a sufficient number of training models is used.

6.5 Conclusions

In this chapter, artificial neural networks were trained to predict the indenter radius in both the longitudinal and circumferential direction, the maximum equivalent plastic strain ($\overline{\varepsilon^p}$), the maximum $\Delta\sigma_{axial}$, and the maximum $\Delta\sigma_{hoop}$. These ANNs were trained using FEA models to harness the accuracy that can be obtained using FEA but can produce results in less time. In the first case study in this chapter, the

indenter radii were modified while all other properties were held constant and the results from Chapter 5 were used to validate the indenter radii, $\overline{\varepsilon^p}$, $\Delta\sigma_{axial}$, and $\Delta\sigma_{hoop}$. In Case Study 2, the indenter radii and indentation depth were the only properties modified, and the ANN results were compared against 100 FEA models with randomly generated input variables. After assessing both case studies, the results from the ANNs were close to their associated FEA results demonstrating that this process can be used to efficiently obtain accurate results.

CHAPTER 7: CONCLUSIONS AND FUTURE RESEARCH

7.1 Summary and Conclusions

While dents are a prevalent concern across liquid pipeline systems, current regulations are simply based on depth and interaction with other features. As there are many factors that can contribute to a dent's severity including the dent's shape, size, location on the pipe, interaction with other features, and the pipe's material and operating properties, there has been extensive research in industry to take into account all or some of these factors and propose an accurate and efficient method to assess dents. Accurate and efficient methodologies for assessing other pipeline integrity threats, such as cracks and corrosion, currently exist while the same is not widely agreed upon in industry for dents. The benefit of having an effective integrity assessment method for dents would allow the prevention of pipeline failure, which can negatively affect the environment and the health and safety of people, while ensuring that business resources are optimized (i.e. resources are not spent repairing features that do not pose any threat). While some researchers have proposed the use of finite element analysis to assess dents, this method requires significant computational time and is impractical for system-wide application.

In this thesis, a method to model dents in pipelines using FEA was proposed and validated against full-scale tests conducted by Ghaednia and Das (2018). The effect of changing different element properties including the element type (shell versus solid), geometric order (linear versus quadratic), integration method (reduced versus full) and number of integration points in the thickness direction was investigated. It was concluded that the use of S4R elements (shell, linear elements with reduced integration) with five integration points in the thickness direction produced optimal results for this research in terms of accuracy and speed.

After an effective method for modelling dents with FEA was determined, the process of generating FEA models and extracting results from them was automated using Python scripts. The developed automation techniques allow for the user definition of input variables (such as the pipe diameter and wall thickness, indenter size and depth, and internal pipe pressure), to create several different models at once using a consistent methodology with minimal human intervention required. After the models are run using the FEA solver, the required results, including the displacement profiles, the maximum $\overline{\varepsilon^p}$, the maximum $\Delta\sigma_{axial}$, and the maximum $\Delta\sigma_{hoop}$ could then be extracted using automation.

Next, a trial and error method was proposed to match the dent profiles from FEA to the dent profiles that would be reported by an in-line inspection tool. Using this method, it was determined that a ring torus is an appropriate indenter shape that can be used for FEA modelling to achieve a smooth dent profile with a single apex. However, the limitation of using the ring torus is that profiles that are flat in the circumferential direction cannot be achieved using this indenter shape in the FEA model. It was also observed in this research that stresses and strains can change dramatically based on the dent shape, particularly for sharp dents. This conclusion emphasizes the need to achieve an accurate profile match between the FEA and ILI displacement profiles in order to accurately assess a dent's severity. The proposed profile matching process is time-consuming as it is iterative and requires human subjectivity that can lead to inconsistent results.

Lastly, artificial neural networks were trained based on the results from FEA. It was proven that given a sufficient database size for training the ANNs, the indenter radius in both the longitudinal and circumferential direction could be predicted if the displacement profile in both directions and the indentation depth were given as inputs. Furthermore, using the indenter radius in both directions and the indentation depth as inputs, the maximum of each of $\overline{\varepsilon^p}$, $\Delta\sigma_{axial}$, and $\Delta\sigma_{hoop}$ could be predicted using ANNs. These results show that ANNs can be used to predict the maximum strains and stresses in dented regions with similar accuracy to what is achieved using FEA, which has already been proven to be an accurate representation of real-life by other researchers. The use of ANNs creates the added benefit of achieving results in far less time than using FEA alone and could be feasible for system-wide application from a time and resource perspective.

7.2 Future Research

While this research demonstrated that it is feasible to assess dents with a combination of FEA and ANNs, the research did not investigate all the different pipe properties and dent types that can be found on a pipeline system. For training and validating the ANNs, many parameters were held constant in this study, including the pipe properties and the internal pressure applied. Future work could investigate the adjustment of several different parameters at once (for example, the ANN would cover a range of pipe diameters, wall thicknesses, and grades in addition to various indenter sizes), and evaluate if the ANN could still produce

accurate results. This would allow for practical application to pipeline systems that have many lines with various properties.

Furthermore, this research only explored constrained, plain, symmetric dents with single apexes. Unconstrained dents would create further complications for profile matching as well as prediction of the indenter radius with an ANN and these complications would need to be investigated and resolved. Dent interaction with other features such as other dents, metal loss, and/or cracks has the potential to increase a dent's severity than if the dent was not interacting with any other features. In addition, dents can have complex shapes, for example, they can have multiple apexes, or their peaks may not align with the longitudinal axis of the pipe. The incorporation of these features in the FEA models as well as with the ANNs needs to be investigated further. For dents with complex shapes, the full three-dimensional profile of the dent may need to be considered, rather than just the two profiles along the longitudinal and circumferential axes that were used in this research.

In this research, a strain hardening exponent of 12 was assumed in the Ramberg-Osgood equation to approximate a stress-strain curve for all FEA models. The effect of this assumption could be assessed against the use of stress-strain curves from experimental coupon tests.

Future research could further improve the accuracy and efficiency of the proposed methods. The ANNs were trained using the pre-programmed techniques in the commercially available software, MATLAB. More accurate results could potentially be achieved by using more advanced training algorithms to train the ANNs or a different method to determine the optimal number of hidden neurons required than the trial and error method used in this thesis. In order to improve efficiency further, cloud computing could be implemented with the proposed automation techniques to allow the FEA models to run simultaneously and build the ANN training database faster.

As the strains and stresses are sensitive to changes in the dent profile, as demonstrated in Chapter 5, measurement error by the ILI tool could lead to potentially inaccurate results. While this research was focused on finding the deterministic maximum values for strains and stresses, the research could be expanded to utilize reliability techniques and account for uncertainties in the measurement of the profile as well as other properties such as the pipe's material properties. Model error using FEA and ANN could also be quantified and considered with the results of the proposed methods.

REFERENCES

1. Adeeb, S. (2018). *Introduction to Solid Mechanics & Finite Element Analysis.* Retrieved from https://sameradeeb-new.srv.ualberta.ca/
2. The American Society of Mechanical Engineers. (2001). *Factory-Made Wrought Buttwelding Fittings* (ASME B16.9-2001). New York, NY: The American Society of Mechanical Engineers.
3. The American Society of Mechanical Engineers (2010). *Gas Transmission and Distribution Piping Systems.* (ASME B31.8-2010). New York, NY: The American Society of Mechanical Engineers.
4. The American Society of Mechanical Engineers (2012). *Manual for Determining the Remaining Strength of Corroded Pipelines.* (ASME B31G-2012). New York, NY: The American Society of Mechanical Engineers.
5. Arumugam, U., Gao, M., Krishnamurthy, R., Wang, R., & Kania, R. (2016). Proceedings of IPC2016 11th International Pipeline Conference: *Study of a Plastic Strain Limit Damage Criterion for Pipeline Mechanical Damage using FEA and Full-Scale Denting Tests.* Calgary, AB: ASME.
6. BMT Fleet Technology. (2012). *Dent Fatigue Life Assessment (DOT# 432 Closeout Report).* Washington, DC: U.S. Department of Transportation.
7. Canadian Standards Association (CSA) Group. (2015). *Oil and gas pipeline systems*. (CSA Z662-15). Toronto, ON: Canadian Standards Association.
8. Cömert, Z. & Kocamaz, A. (2017). A study of artificial neural network training algorithms for classification of cardiotocography signals. *Journal of Science and Technology, 7*(2), 93-103.
9. Cosham, A. & Hopkins, P. (2004). The effect of dents in pipelines – guidance in the pipeline defect assessment manual. *International Journal of Pressure Vessels and Piping, 81*(2), 127-139.
10. Dassault Systèmes. (2014). *Abaqus Analysis User's Guide* [Computer software]. Providence, RI: Dassault Systèmes Simulia Corp.
11. Dawson, S.J., Russell, A., & Patterson, A. (2006). Proceedings of IPC2006 6th International Pipeline Conference: *Emerging Techniques for Enhanced Assessment and Analysis of Dents*. Calgary, AB: ASME.

12. Gao, M., & Krishnamurthy, R. (2015). Mechanical Damage in Pipelines: A Review of the Methods and Improvements in Characterization, Evaluation, and Mitigation. *Oil and Gas Pipelines: Integrity and Safety Handbook, First Edition.* Chapter 22.

13. Ghaednia, H., & Das, S. (2018). Structural performance of oil and gas pipe with dent defect. *Journal of Pipeline Systems Engineering and Practice, 9*(1), 1-9.

14. Gossard, J., Bratton, J., Kemp, D., Finneran, S., & Polasik, S. J. (2016). Proceedings of IPC2016 11th International Pipeline Conference: *Evaluating Dents with Metal Loss using Finite Element Analysis.* Calgary, AB: ASME.

15. Graupe, D. (2007). *Principles Of Artificial Neural Networks (2nd Edition).* New Jersey: World Scientific.

16. Hassanien, S., Kainat, M., Adeeb, S., & Langer, D. (2016). Proceedings of IPC2016 11th International Pipeline Conference: *On the Use of Surrogate Models in Reliability-Based Analysis of Dented Pipes*. Calgary, AB: ASME.

17. Hassanien, S., & Langer, D. (2018). Proceedings of NACE CORROSION 2018 Annual Conference: *A Semi-Quantitative Analysis of Dents Associated with Corrosion: A Reliability-Based Approach.* Phoenix, AZ: NACE International.

18. Hyde, T.H., Luo, R. & Becker, A.A. (2001). Proceedings of International Gas Research Conference: *Finite element analysis of indented pipes using three-dimensional solid and shell elements*. Amsterdam, the Netherlands.

19. Hyde, T. H., Luo, R. & Becker, A. A. (2011). Predictions of three-dimensional stress variations in indented pipes due to internal pressure fluctuations. *Journal of Strain Analysis for Engineering Design, 46*(6), 510-522.

20. Kainat, M., Woo, J., Langer, D., Krausert, T., Cheng, J. J., Hassanien, S., & Adeeb, S. (2019). Effects of loading sequences on the remaining life of plain dents. *Journal of Pipeline Systems Engineering and Practice, 10(2),* 1-20.

21. Kiefner, J., & Leewis, K. (2011). *Pipeline Defect Assessment – A Review & Comparison of Commonly Used Methods.* PR-218-05405. Worthington, Ohio: Kiefner and Associates, Inc.

22. Langer, D., Kainat, M., & Woo, J. (2017). Proceedings of NACE CORROSION 2017 Annual Conference: *Reliability Assessment of Corrosion Features Interacting with Pipeline Dents.* New Orleans, LA: NACE International.

23. Layouni, M., Hamdi, M., Tahar, S. (2016). Detection and sizing of metal-loss defects in oil and gas pipelines using pattern-adapted wavelets and machine learning. *Applied Soft Computing, 52*(2017), 247-261.

24. Lin, M. (2015). *Characterization of Tensile and Fracture Properties of X52 Steel Pipes and Their Girth Welds* (MSc Thesis). Structural Engineering, University of Alberta, Edmonton, AB, Canada.

25. Lukasiewicz, S. A., Czyz, J. A., Sun, C., & Adeeb, S. (2006). Proceedings of IPC2006 6th International Pipeline Conference: *Calculation of Strains in Dents Based on High Resolution In-Line Caliper Survey.* Calgary, AB: ASME.

26. Michael Baker Jr., Inc. (2004). *Inspection Guidelines for Timely Response to Geometry Defects: Final Report (TTO Number 7).* Retrieved from https://primis.phmsa.dot.gov/gasimp/docs/TTO07_ InspGuidelinesGeometryDefects_FinalReport_July2004.pdf

27. Møller, M. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks, 6*(4), 525-533.

28. National Energy Board. (2010). *Fatigue Crack Failure Associated with Shallow Dents on Pipelines.* (NEB SA 2010-01). Calgary, AB: Transportation Safety Board of Canada.

29. Noronha, D., Martins, R., Jacob, B., & Souza, E. (2010). Procedures for the strain based assessment of pipeline dents, *International Journal of Pressure Vessels and Piping, 87*, 254-265.

30. Okoloekwe, C. (2017). *A Novel Approach to the Strain Based Analysis of Dented Pipelines* (MSc Thesis). Structural Engineering, University of Alberta, Edmonton, AB, Canada.

31. Pinheiro, B., Pasqualino, I., & Cunha, S. (2014). Fatigue life assessment of damaged pipelines under cyclic internal pressure: Pipelines with longitudinal and transverse plain dents, *International Journal of Fatigue, 68*(2018), 38-47.

32. Rosenfeld, M. J. (1999). *Guidelines for the Assessment of Dents on Welds.* PR-218-9822. Houston, TX: Technical Toolboxes, Inc.

33. Shanmuganathan, S., & Samarasinghe, S. (2016). *Artificial Neural Network Modelling (Vol. 628).* Cham, Switzerland: Springer International Publishing.

34. Shuai, Y., Shuai, J., & Zhang, X. (2018). Experimental and numerical investigation of the strain response of a dented API 5L X52 pipeline subjected to continuously increasing internal pressure. *Journal of Natural Gas Science and Engineering, 56*(2018), 81-92.

35. The MathWorks, Inc. (2018a). *Documentation: Deep Learning Toolbox (R2018a).* Retrieved from http://www.mathworks.com/help/deeplearning/index.html.

36. The MathWorks, Inc. (2018b). *MATLAB and Neural Net Fitting Toolbox Release 2018b.* Natick, Massachusetts, United States.

37. Ticknor, J. (2013). A Bayesian regularized artificial neural network for stock market forecasting. *Expert Systems with Applications, 40*(14), 5501-5506.

38. Tiku, S., Semiga, V., Dinovitzer, A., & Vignal, G. (2012). Proceedings of IPC2012 9th International Pipeline Conference: *Full Scale Cyclic Fatigue Testing of Dented Pipelines and Development of a Validated Dented Pipe Finite Element Model.* Calgary, AB: ASME.

39. Turnquist, M. & Smith, I. (2016). Proceedings of IPC2016 11th International Pipeline Conference: *A Life-Cycle Approach to the Assessment of Pipeline Dents.* Calgary, AB: ASME.

40. Xu, W., Chun, B., Joonmo, C., & Lee, J. (2017). Corroded pipeline failure analysis using artificial neural network scheme. *Advances in Engineering Software, 112* (2017), 255-266.

APPENDIX A: Python Scripts for Model Generation and Results Extraction

In this appendix, the Python codes are provided that can be run in Abaqus to automatically perform: FEA model generation, profile extraction from the FEA output file, and extraction of the maximum $\overline{\varepsilon^p}$, $\Delta\sigma_{axial}$, and $\Delta\sigma_{hoop}$ from the FEA output file. The lines starting with the "#" symbol represent comments to explain what each section of the code is doing and are not part of the main code.

FEA Model Generation Script

```
from abaqus import *
from abaqusConstants import *
import __main__

import regionToolset
import section
import regionToolset
import displayGroupMdbToolset as dgm
import part
import material
import assembly
import step
import interaction
import load
import mesh
import optimization
import job
import sketch
import visualization
import xyPlot
import displayGroupOdbToolset as dgo
import connectorBehavior
import sketch
import sys, math
from numpy import *

# File location of Input Variables text file
file_path = 'C:/Users/Janine Woo/Documents/Thesis/Shell Scripts/Ring/'
input_file = open(file_path + 'Input_Variables.txt')

for line in input_file:

# Read input variables from file - each row of the input file is a new model
       extracted_line = line
       extracted_list = extracted_line.split()
       modelname = '' + str(extracted_list[0])
       outer_radius = float(extracted_list[1])
       wall_thickness = float(extracted_list[2])
       length_of_pipe = float(extracted_list[3])
       steel_grade = '' + str(extracted_list[4])
       length_of_partition = float(extracted_list[5])
       rotation_of_partition = float(extracted_list[6])
       radius_of_indenter_l = float(extracted_list[7])
       radius_of_indenter_c = float(extracted_list[8])
```

```
indentation_depth = float(extracted_list[9])
max_op_pressure = float(extracted_list[10])
mesh_partition = float(extracted_list[11])
mesh_longitudinal = float(extracted_list[12])
mesh_circ = float(extracted_list[13])
mesh_away = float(extracted_list[14])


# ------------------------------------------------

# MODEL

# Create the model

Model = mdb.Model(name=modelname)

# ------------------------------------------------

# PARTS

# Pipe

# Set-up pipe sketch
pipeSketch =
Model.ConstrainedSketch(name='PipeSketch',sheetSize=outer_radius*2)
pipeSketchg = pipeSketch.geometry

# Sketch the pipe section using circle tool
createCircle = pipeSketch.CircleByCenterPerimeter(center=(0.0,0.0),
point1=(0.0,outer_radius))

# Create vertical construction line
pipeSketch.ConstructionLine(point1=(0.0,0.0), angle=90.0)

# Auto-trim the circle so semi-circle (left-hand side) remains
trimCurve1 = pipeSketchg.findAt((outer_radius,0),)
pipeSketch.autoTrimCurve(curve1=trimCurve1, point1=(outer_radius,0))

# Create a 3D deformable part named "Pipe" by extruding the sketch
pipePart=Model.Part(name='Pipe',dimensionality=THREE_D,type=DEFORMABLE_BODY)
pipePart.BaseShellExtrude(sketch=pipeSketch, depth=length_of_pipe)

# -------

# Partition

# Set-up the partition sketch
partitionSketch = Model.ConstrainedSketch(name='Partition Sketch-
1',sheetSize=outer_radius)
partitiong = partitionSketch.geometry

# Create a circle to form the outer edge of the partition
radiusPartition = outer_radius+50
partitionSketch.CircleByCenterPerimeter(center=(0.0, 0.0),
point1=(radiusPartition, 0.0))

# Create construction lines to form sides of partition
partitionSketch.ConstructionLine(point1=(0.0, 0.0), angle=90-
(rotation_of_partition))
partitionSketch.ConstructionLine(point1=(0.0, 0.0),
angle=90+(rotation_of_partition))

# Trim excess portions of circle outside of partition area
```

```
partitionAngle=(rotation_of_partition)*pi/180

trimCurve1 = partitiong.findAt((-radiusPartition,0),)
partitionSketch.autoTrimCurve(curve1=trimCurve1, point1=(-radiusPartition, 0))

trimCurve2 = partitiong.findAt((0,-radiusPartition),)
partitionSketch.autoTrimCurve(curve1=trimCurve2, point1=(0, -radiusPartition))

trimCurve3 = partitiong.findAt((radiusPartition*math.sin(partitionAngle*2), -
radiusPartition*math.cos(partitionAngle*2)),)
partitionSketch.autoTrimCurve(curve1=trimCurve3, point1=(radiusPartition, 0))

trimCurve4 = partitiong.findAt((radiusPartition, 0),)
partitionSketch.autoTrimCurve(curve1=trimCurve4, point1=(radiusPartition, 0))

# Create connecting lines to close shape
partitionSketch.Line(point1=(0.0, 0.0), point2=(-
(outer_radius+50)*math.sin(partitionAngle),
(outer_radius+50)*math.cos(partitionAngle)))
partitionSketch.Line(point1=(0.0, 0.0),
point2=((outer_radius+50)*math.sin(partitionAngle),
(outer_radius+50)*math.cos(partitionAngle)))

# Create 3d deformable part named "Partition" by extruding the sketch
partitionPart = Model.Part(name='Partition', dimensionality=THREE_D,
       type=DEFORMABLE_BODY)
partitionPart.BaseSolidExtrude(sketch=partitionSketch,
depth=length_of_partition)

# Remove the excess face at the top of the part
topFace =
partitionPart.faces.findAt(((0,radiusPartition,length_of_partition/4),))
partitionPart.RemoveFaces(faceList = topFace, deleteCells=False)

# Go to the assembly and create Pipe and Partition instances
pipeAssembly = Model.rootAssembly
pipeAssembly.Instance(name='Pipe-1', part=pipePart, dependent=OFF)
pipeAssembly.Instance(name='Partition-1', part=partitionPart, dependent=OFF)

# Merge Partition and Pipe together to form new part
pipeAssembly.InstanceFromBooleanMerge(name='Pipe-Partitioned', instances=(
pipeAssembly.instances['Partition-1'], pipeAssembly.instances['Pipe-1'], ),
keepIntersections=ON, originalInstances=SUPPRESS, domain=GEOMETRY)
pipeAssembly.makeIndependent(instances=(pipeAssembly.instances['Pipe-
Partitioned-1'], ))

# Define variables for pipe properties for later use
finalPipePart = Model.parts['Pipe-Partitioned']
finalPipePartCells = finalPipePart.cells
finalPipeEdges = pipeAssembly.instances['Pipe-Partitioned-1'].edges
finalPipeCells = pipeAssembly.instances['Pipe-Partitioned-1'].cells
finalPipeFaces = pipeAssembly.instances['Pipe-Partitioned-1'].faces

# Remove excess faces of Pipe-Partitioned part
       # Find 5 faces (inside and outside of the pipe)

       # Faces inside the pipe
removeFace1 = finalPipePart.faces.findAt(((0,outer_radius/2,0),))
       removeFace2 =
finalPipePart.faces.findAt(((0,outer_radius/2,length_of_partition),))
removeFace3 =
finalPipePart.faces.findAt((((outer_radius/2)*math.sin(partitionAngle),(outer_r
adius/2)*math.cos(partitionAngle),length_of_partition/2),))
```

```
removeFace4 = finalPipePart.faces.findAt(((-
(outer_radius/2)*math.sin(partitionAngle),(outer_radius/2)*math.cos(partitionAn
gle),length_of_partition/2),))

       # Faces outside of the pipe
removeFace5 = finalPipePart.faces.findAt(((-
radiusPartition*math.sin(partitionAngle),radiusPartition*math.cos(partitionAngl
e),length_of_partition/2),))

       # Remove the faces from face list defined above
finalPipePart.RemoveFaces(faceList =
removeFace1+removeFace2+removeFace3+removeFace4+removeFace5, deleteCells=False)

# -----

# Ring Indenter

# Re-define input variables ("longitudinal" radius variable must always be
larger than "circumferential" radius variable for geometry to work)

if radius_of_indenter_l < radius_of_indenter_c:
       radius_of_indenter_l = float(extracted_list[8])
       radius_of_indenter_c = float(extracted_list[7])

else:
       radius_of_indenter_l = float(extracted_list[7])
       radius_of_indenter_c = float(extracted_list[8])

# Set-up the indenter sketch
indenterSketch = Model.ConstrainedSketch(name='Indenter Sketch-1', sheetSize =
radius_of_indenter_l*2)
indenterSketchg = indenterSketch.geometry

# Define geometry of indenter
indenterSketch.ConstructionLine(point1=(0.0, -radius_of_indenter_l),
point2=(0.0, radius_of_indenter_l))
indenterSketch.FixedConstraint(entity=indenterSketchg.findAt((0,radius_of_inden
ter_l),))

# Sketch circle based on radius of indenter in longitudinal and circumferential
direction
indenterSketch.CircleByCenterPerimeter(center=(radius_of_indenter_l -
radius_of_indenter_c,0.0), point1=(radius_of_indenter_l -
radius_of_indenter_c,radius_of_indenter_c))

# Set up center point of circle
centerX = radius_of_indenter_l - radius_of_indenter_c

# Use if statement because different curves will have to be trimmed based on
size of indenter

if centerX >= radius_of_indenter_c:

       # If the difference between the two radii is greater than the radius of
the shorter side
       # Use construction line and auto-trim so only the right side of the
circle and less than half remains
       indenterSketch.ConstructionLine(point1=(centerX + 5,0.0), angle=90.0)

       # Trim excess parts of circle
       trimLeft = indenterSketchg.findAt((centerX-radius_of_indenter_c,0),)
       indenterSketch.autoTrimCurve(curve1=trimLeft, point1=(centerX-
       radius_of_indenter_c, 0))
```

```
            trimTop =
            indenterSketchg.findAt((centerX+radius_of_indenter_c*math.sin(0.00001),ra
            dius_of_indenter_c*math.cos(0.00001)),)
            indenterSketch.autoTrimCurve(curve1=trimTop,
            point1=(centerX+radius_of_indenter_c*math.sin(0.00001),radius_of_indenter
            _c*math.cos(0.00001)))

      elif centerX == 0:

            # If longitudinal radius = circumferential radius:
            indenterSketch.ConstructionLine(point1=(radius_of_indenter_l/2,0.0),
angle=90.0)

            # Trim excess parts of circle
            trimLeft = indenterSketchg.findAt((-radius_of_indenter_l,0),)
            indenterSketch.autoTrimCurve(curve1=trimLeft, point1=(-
radius_of_indenter_l, 0))

            trimTop =
indenterSketchg.findAt((centerX+radius_of_indenter_c*math.sin(0.00001),radius_of_inden
ter_c*math.cos(0.00001)),)
            indenterSketch.autoTrimCurve(curve1=trimTop,
point1=(centerX+radius_of_indenter_c*math.sin(0.00001),radius_of_indenter_c*math.cos(0
.00001)))

            trimBottom =
indenterSketchg.findAt((centerX+radius_of_indenter_c*math.sin(0.00001),-
radius_of_indenter_c*math.cos(0.00001)),)
            indenterSketch.autoTrimCurve(curve1=trimBottom,
point1=(centerX+radius_of_indenter_c*math.sin(0.00001),-
radius_of_indenter_c*math.cos(0.00001)))

      else:

            # If the difference between the two radii is less than the radius of the
shorter side
            # Create construction line 5 mm to the right of centerX
            indenterSketch.ConstructionLine(point1=(centerX + 5,0.0), angle=90.0)

            trimLeft = indenterSketchg.findAt((centerX-radius_of_indenter_c,0),)
            indenterSketch.autoTrimCurve(curve1=trimLeft, point1=(centerX-
radius_of_indenter_c, 0))

            trimTop = indenterSketchg.findAt((centerX+radius_of_indenter_c*math.sin(-
0.00001),radius_of_indenter_c*math.cos(-0.00001)),)
            indenterSketch.autoTrimCurve(curve1=trimTop,
point1=(centerX+radius_of_indenter_c*math.sin(-
0.00001),radius_of_indenter_c*math.cos(-0.00001)))

            trimTop2 =
indenterSketchg.findAt((centerX+radius_of_indenter_c*math.sin(0.00001),radius_of_inden
ter_c*math.cos(0.00001)),)
            indenterSketch.autoTrimCurve(curve1=trimTop2,
point1=(centerX+radius_of_indenter_c*math.sin(0.00001),radius_of_indenter_c*math.cos(0
.00001)))

            trimBottom = indenterSketchg.findAt((centerX,-radius_of_indenter_c),)
            indenterSketch.autoTrimCurve(curve1=trimBottom, point1=(centerX,-
radius_of_indenter_c))

      # Create a 3D analytical rigid part named "Indenter" by revolving the sketch
```

```
	indenterPart=Model.Part(name='Indenter', dimensionality=THREE_D,
type=ANALYTIC_RIGID_SURFACE)
	indenterPart.AnalyticRigidSurfRevolve(sketch=indenterSketch)

	# Insert reference point on ring indenter
	RP1 = indenterPart.ReferencePoint(point=(-radius_of_indenter_1,0,0))
	RP1ID = RP1.id

	# ----------------------------------------------

	# MATERIALS

	# Create X52 (Vintage) Mean material
	X52VintageMean = Model.Material(name='X52-Vintage-Mean')
	X52VintageMean.Elastic(table=((210000,0.3), ))
	X52VintageMean.Plastic(table=((300, 0), (325, 0.000243343470806226), (350,
0.000808401541595963), (375, 0.00204445005495397), (386.55, 0.00300842511484258),
(400, 0.00461170168451115), (425, 0.00970711940734048), (450, 0.0194227497102881),
(475, 0.0372984724789057), (500, 0.069153542026477), (525, 0.1243098892775), (550,
0.217356043233442), (575, 0.370645026859678), (600, 0.617774154719637)))

	# ----------------------------------------------------

	# SECTIONS

	# Create a section to assign to pipe
	pipeSection = Model.HomogeneousShellSection(name='Pipe-Section',
	 preIntegrate=OFF, material=steel_grade, thicknessType=UNIFORM,
	 thickness=wall_thickness, thicknessField='', idealization=NO_IDEALIZATION,
	 poissonDefinition=DEFAULT, thicknessModulus=None, temperature=GRADIENT,
	 useDensity=OFF, integrationRule=SIMPSON, numIntPts=5)

	# Select regions that will be assigned a section
	finalPipePartFaces = Model.parts['Pipe-Partitioned'].faces
	section1_1 =
finalPipePartFaces.findAt(((0,outer_radius,length_of_partition/4),))
	section1_2 = finalPipePartFaces.findAt(((0,outer_radius,length_of_pipe),))
	sectionRegion1 = regionToolset.Region(faces=section1_1+section1_2)

	# Assign sections to pipe
	finalPipePart.SectionAssignment(region=sectionRegion1, sectionName='Pipe-
Section', offset=0.0,
		offsetType=TOP_SURFACE, offsetField='',
		thicknessAssignment=FROM_SECTION)

	# ----------------------------------------------------

	# ORIENTATIONS

	# Create cylindrical coordinate system and assign to part

	datum1 = finalPipePart.DatumCsysByThreePoints(name='Cylindrical',
coordSysType=CYLINDRICAL,
		origin=(0.0, 0.0, 0.0), line1=(1.0, 0.0, 0.0), line2=(0.0, 1.0,
0.0))
	datum1ID = datum1.id
	orientation = finalPipePart.datums[datum1ID]
	orientationRegion = regionToolset.Region(faces=section1_1+section1_2)
	finalPipePart.MaterialOrientation(
		region=orientationRegion, orientationType=SYSTEM, axis=AXIS_1,
		localCsys=orientation, fieldName='',
		additionalRotationType=ROTATION_NONE, angle=0.0,
		additionalRotationField='')
```

```
	# ----------------------------------------------------

	# ASSEMBLY

	# Set up assembly - position indenter in relation to pipe
	pipeAssembly.Instance(name='Indenter-1', part=indenterPart, dependent=OFF)
	pipeAssembly.rotate(instanceList=('Indenter-1', ), axisPoint=(0.0, 0.0, 0.0),
			axisDirection=(0.0, 0.0, 1.0), angle=-90.0)

	# If circumferential radius is larger than longituindal radius, the indenter
will have to rotated 90 degrees around the y-axis
	radius_of_indenter_l = float(extracted_list[7])
	radius_of_indenter_c = float(extracted_list[8])
	if radius_of_indenter_c >= radius_of_indenter_l:
		pipeAssembly.rotate(instanceList=('Indenter-1', ), axisPoint=(0.0, 0.0,
0.0),
			axisDirection=(0.0, 1.0, 0.0), angle=90.0)
		pipeAssembly.translate(instanceList=('Indenter-1', ), vector=(0.0,
radius_of_indenter_c + outer_radius+2, 0))

	else:
		# Translate the first indenter to its correct relative position along the
pipe segment
		pipeAssembly.translate(instanceList=('Indenter-1', ), vector=(0.0,
radius_of_indenter_l + outer_radius+2,0))

	# ----------------------------------------------------------------

	# MESH

	# Set up mesh for Pipe-Partitioned instance

	# Define seeds for all edges:

	# Within partition - around all partition edges except through thickness
		# Define XY coordinates for 4 outside edges
	middlePartition = -(rotation_of_partition/2)*math.pi/180
	edgePartition = -rotation_of_partition*math.pi/180
	partitionEdge1 =
finalPipeEdges.findAt(((outer_radius*math.sin(middlePartition),outer_radius*math.cos(m
iddlePartition),0),))
	partitionEdge2 =
finalPipeEdges.findAt(((outer_radius*math.sin(middlePartition),outer_radius*math.cos(m
iddlePartition),length_of_partition),))
	partitionEdge3 =
finalPipeEdges.findAt(((outer_radius*math.sin(edgePartition),outer_radius*math.cos(edg
ePartition),length_of_partition/2),))
	partitionEdge4 =
finalPipeEdges.findAt(((0,outer_radius,length_of_partition/2),))

	# Seed edges within partition around all edges except through thickness - no
bias
	pipeAssembly.seedEdgeBySize(edges=partitionEdge1+partitionEdge2+partitionEdge3+
partitionEdge4, size=mesh_partition, deviationFactor=0.1,
	  constraint=FINER)

	# -----

	# Longitudinal bias, from partition to away from partition
	longbiasEdge1 =
finalPipeEdges.findAt(((0,outer_radius,length_of_partition+0.00001),))
```

```
	pipeAssembly.seedEdgeByBias(biasMethod=SINGLE, end1Edges=longbiasEdge1,
minSize=mesh_partition, maxSize=mesh_longitudinal, constraint=FINER)

	# -----

	# Circ bias, from partition to away from partition
	circbiasEdge1 = finalPipeEdges.findAt(((outer_radius*math.sin(edgePartition-
0.00001),outer_radius*math.cos(edgePartition-0.00001),0),))

	pipeAssembly.seedEdgeByBias(biasMethod=SINGLE, end1Edges=circbiasEdge1,
minSize=mesh_partition, maxSize=mesh_circ, constraint=FINER)

	# -----

	# Longitudinal bias, longitudinal edges opposite of indenter
	longbiasEdge3 = finalPipeEdges.findAt(((0,-outer_radius,0.00001),))

	pipeAssembly.seedEdgeByBias(biasMethod=SINGLE, end1Edges=longbiasEdge3,
minSize=mesh_circ, maxSize=mesh_away, constraint=FINER)

	# -----

	# Circ bias, circumferential edges opposite of indenter
	circbiasEdge3 = finalPipeEdges.findAt(((-
0.00001,outer_radius,length_of_pipe),))

	pipeAssembly.seedEdgeByBias(biasMethod=SINGLE, end2Edges=circbiasEdge3,
minSize=mesh_longitudinal, maxSize=mesh_away, constraint=FINER)

	# -----

	# Mesh controls

	# Partitions - medial axis
	meshFaces1_1 = finalPipeFaces.findAt(((-0.00001,outer_radius,0),))
	pipeAssembly.setMeshControls(regions=meshFaces1_1, algorithm=MEDIAL_AXIS,
minTransition=ON)

	# Outside of partitions - advancing front
	meshFaces2_1 = finalPipeFaces.findAt(((-0.00001,outer_radius,length_of_pipe-
100),))
	pipeAssembly.setMeshControls(regions=meshFaces2_1, technique=SWEEP,
		algorithm=ADVANCING_FRONT, allowMapped=True)

	# Generate mesh
	pipeAssembly.generateMesh(regions=meshFaces1_1, seedConstraintOverride=ON)
	pipeAssembly.generateMesh(regions=meshFaces2_1, seedConstraintOverride=ON)

	# ----------------

	# INTERACTIONS

	# Interaction #1
	# Set up interaction properties
	Model.ContactProperty('IntProp-1')
	Model.interactionProperties['IntProp-1'].TangentialBehavior(
			formulation=PENALTY, directionality=ISOTROPIC,
slipRateDependency=OFF,
			pressureDependency=OFF, temperatureDependency=OFF, dependencies=0,
			table=((0.5, ), ), shearStressLimit=None,
maximumElasticSlip=FRACTION,
			fraction=0.005, elasticSlipStiffness=None)
	Model.interactionProperties['IntProp-1'].NormalBehavior(
```

```
                    pressureOverclosure=HARD, allowSeparation=ON,
                    constraintEnforcementMethod=DEFAULT)

      # Define master surface - Indenter-1
      indenterFaces = pipeAssembly.instances['Indenter-1'].faces
      indenterSurface = indenterFaces.findAt(((0,outer_radius+2,0),))
      masterSurface = regionToolset.Region(side1Faces=indenterSurface)

      # Define slave surface
      finalPipeFaces1 = finalPipeFaces.findAt(((-0.00001,outer_radius,0.00001),))
      finalPipeFaces2 = finalPipeFaces.findAt(((-0.00001,outer_radius,length_of_pipe-
0.00001),))
      slaveSurface=regionToolset.Region(side1Faces=finalPipeFaces1+finalPipeFaces2)

      # Update interaction properties
      Model.SurfaceToSurfaceContactStd(name='Int-1',
                    createStepName='Initial', master=masterSurface,
slave=slaveSurface,
                    sliding=FINITE, thickness=ON, interactionProperty='IntProp-1',
                    adjustMethod=NONE, initialClearance=OMIT, datumAxis=None,
                    clearanceRegion=None)

      # Introduce contact controls, initializations and stabilizations
      Model.StdContactControl(name='ContCtrl-1',
                    stabilizeChoice=AUTOMATIC)
      Model.StdInitialization(name='CInit-1')
      Model.StdStabilization(name='CStab-1')
      Model.interactions['Int-1'].setValues(
                    initialClearance=OMIT, adjustMethod=NONE, sliding=FINITE,
                    enforcement=SURFACE_TO_SURFACE, thickness=ON,
                    contactTracking=TWO_CONFIG, contactControls='ContCtrl-1',
                    bondingSet=None)

      # ----------------------------------------------------------------

      # BOUNDARY CONDITIONS

      # Coupling boundary condition for ends away from indenter

            # Create reference point in center of pipe (on ends away from indenter)
      RPC1 = pipeAssembly.ReferencePoint(point=(0,0,length_of_pipe))
      RPCid_1 = RPC1.id

            # Create cylindrical coordinate system for boundary condition
      datumCreate = pipeAssembly.DatumCsysByThreePoints(name='Datum csys-2',
coordSysType=CYLINDRICAL,
            origin=(0.0, 0.0, 0.0), line1=(1.0, 0.0, 0.0), line2=(0.0, 1.0, 0.0))
      datumID = datumCreate.id

            # For Z = Length of Pipe
            # Select reference point for coupling condition
      refPoints = pipeAssembly.referencePoints
      refPoints1=(refPoints[RPCid_1], )
      controlPoint=regionToolset.Region(referencePoints=refPoints1)

            # Select surface for coupling condition
      couplingEdges = finalPipeEdges.findAt(((-
0.000001,outer_radius,length_of_pipe),))
      couplingFacesRegion=regionToolset.Region(edges=couplingEdges)
      datumCoupling = pipeAssembly.datums[datumID]

            # Create coupling boundary condition and allow freedom in the U1 (radial
direction)
```

```
	Model.Coupling(name='Coupling-Constraint-1',
		controlPoint=controlPoint, surface=couplingFacesRegion,
influenceRadius=WHOLE_SURFACE,
		couplingType=KINEMATIC, localCsys=datumCoupling, u1=OFF, u2=ON, u3=ON,
ur1=ON,
		ur2=ON, ur3=ON)

	# Set up fixed ends boundary condition (encastre)
	encastreRP = (refPoints[RPCid_1], )
	encastreRegion = regionToolset.Region(referencePoints=encastreRP)
	Model.EncastreBC(name='Fixed Ends',
			createStepName='Initial', region=encastreRegion, localCsys=None)

	# Set up X-Symmetry boundary condition
	xSymmEdges =
finalPipeEdges.findAt(((0,outer_radius,length_of_partition/4),),((0,outer_radius,lengt
h_of_pipe-1),),((0,-outer_radius,length_of_pipe-1),))
	xSymmRegion = regionToolset.Region(edges=xSymmEdges)
	Model.XsymmBC(name='X-Symmetry', createStepName='Initial',
			region=xSymmRegion, localCsys=None)

	# Set up Z-Symmetry boundary condition
	zSymmEdges = finalPipeEdges.findAt(((-0.00001,outer_radius,0),),((-0.00001,-
outer_radius,0),))
	zSymmRegion = regionToolset.Region(edges=zSymmEdges)
	Model.ZsymmBC(name='Z-Symmetry', createStepName='Initial',
			region=zSymmRegion, localCsys=None)

	# Set up Indenter-Translation boundary condition
	indenterRP = pipeAssembly.instances['Indenter-1'].referencePoints
	indenterRefRegion = regionToolset.Region(referencePoints=(indenterRP[RP1ID], ))
	Model.DisplacementBC(name='Indenter-translation-1',
			createStepName='Initial', region=indenterRefRegion, u1=SET,
u2=SET, u3=SET,
			ur1=SET, ur2=SET, ur3=SET, amplitude=UNSET,
distributionType=UNIFORM,
			fieldName='', localCsys=None)

	# ----------------------------------------------------------------

	# STEPS

	# Set up Initial-Pressure step
	Model.StaticStep(name='Initial-Pressure',
			previous='Initial', timePeriod=1, initialInc=1,
			minInc=1.5e-05, maxInc=1, nlgeom=ON)
	pressureFaces = finalPipeFaces.findAt(((-
0.00001,outer_radius,length_of_partition/4),),((-(outer_radius),0,length_of_pipe-
100),))
	pressureRegion = regionToolset.Region(side2Faces=pressureFaces)
	Model.Pressure(name='Pressure',
			createStepName='Initial-Pressure', region=pressureRegion,
			distributionType=UNIFORM, field='', magnitude=0.1,
amplitude=UNSET)

	# Set up field output and history output requests
	Model.fieldOutputRequests['F-Output-1'].setValues(
			variables=('S', 'PEEQ', 'LE', 'EE', 'IE', 'U', 'P', 'CNAREA',
'CSTATUS'), frequency=1)
	Model.historyOutputRequests['H-Output-1'].setValues(
			variables=('MASS', ), frequency=LAST_INCREMENT)

	# Set up Indentation step
```

```
	Model.StaticStep(name='Indentation',
			previous='Initial-Pressure', initialInc=0.01, maxInc=0.1,
nlgeom=ON)
	Model.boundaryConditions['Indenter-translation-1'].setValuesInStep(
			stepName='Indentation', u2=-(2+indentation_depth))

	# Set up Pressure-MOP step
	Model.StaticStep(name='Pressure-MOP',
			previous='Indentation', timePeriod=1, initialInc=0.05,
			minInc=1.5e-05, maxInc=0.1, nlgeom=ON)
	Model.loads['Pressure'].setValuesInStep(
	  stepName='Pressure-MOP', magnitude=max_op_pressure)

	# Set up Pressure-Zero
	Model.StaticStep(name='Pressure-Zero',
			previous='Pressure-MOP', initialInc=0.05, minInc=1.5e-05,
maxInc=0.1, nlgeom=ON)
	Model.loads['Pressure'].setValuesInStep(
	  stepName='Pressure-Zero', magnitude=0)

	# Set up P-MOP2 step
	Model.StaticStep(name='Pressure-MOP2',
			previous='Pressure-Zero', timePeriod=1, initialInc=0.05,
			minInc=1.5e-05, maxInc=0.1, nlgeom=ON)
	Model.loads['Pressure'].setValuesInStep(
	  stepName='Pressure-MOP2', magnitude=max_op_pressure)

	# ----------------------------------------------------------------

	# CREATE AND SUBMIT JOB

	job_ID = modelname
	mdb.Job(name=job_ID, model=modelname,
			description='', type=ANALYSIS, atTime=None, waitMinutes=0,
waitHours=0,
			queue=None, memory=90, memoryUnits=PERCENTAGE,
			getMemoryFromAnalysis=True, explicitPrecision=SINGLE,
			nodalOutputPrecision=SINGLE, echoPrint=OFF, modelPrint=OFF,
			contactPrint=OFF, historyPrint=OFF, userSubroutine='', scratch='',
			resultsFormat=ODB, multiprocessingMode=DEFAULT, numCpus=8,
			numDomains=8, numGPUs=8)


input_file.close()
```

Profile Extraction Script

```
from abaqus import *
from odbAccess import *
from abaqusConstants import *
import __main__

import sys
import os
import odbAccess
import numpy as NP
import math
import section
import regionToolset
import displayGroupMdbToolset as dgm
import part
import material
import assembly
import step
import interaction
import load
import mesh
import optimization
import job
import sketch
import visualization
import xyPlot
import displayGroupOdbToolset as dgo
import connectorBehavior

# File location of Input Variables text file
file_path='C:/Users/Janine Woo/Documents/Thesis/Shell Scripts/Ring/'
input_file= open(file_path+'Input_Variables.txt')
COUNT = 1

for line in input_file:
       # Read input variables from file - each row of the input file is a new model
       extracted_line = line
       extracted_list = extracted_line.split()
       modelname = '' + str(extracted_list[0])
       outer_radius = float(extracted_list[1])
       wall_thickness = float(extracted_list[2])
       length_of_pipe = float(extracted_list[3])
       steel_grade = '' + str(extracted_list[4])
       length_of_partition = float(extracted_list[5])
       rotation_of_partition = float(extracted_list[6])
       radius_of_indenter_l = float(extracted_list[7])
       radius_of_indenter_c = float(extracted_list[8])
       indentation_depth = float(extracted_list[9])
       max_op_pressure = float(extracted_list[10])
       mesh_partition = float(extracted_list[11])
       mesh_longitudinal = int(extracted_list[12])
       mesh_circ = int(extracted_list[13])
       mesh_away = int(extracted_list[14])
       instance_name = 'PIPE-PARTITIONED-1'

       # Create output file for longitudinal
       file_pathL = 'C:/temp/' + 'L-' + modelname + '.csv'
       output_file = open(file_pathL,'w')

       # Open odb file
       odb_file_location = 'C:/temp/'
```

```
        odbPath = odb_file_location + modelname
        odbFile = odbPath + '.odb'

        # Create paths
        # Create longitudinal path
        odb = session.openOdb(name=odbPath + '.odb')
        session.viewports['Viewport: 1'].setValues(displayedObject=odb)
        longitudinalName = 'Longitudinal' + str(COUNT)
        session.Path(name=longitudinalName, type=POINT_LIST,
expression=((0,outer_radius,0),(0,outer_radius,length_of_pipe)))

        # Create circumferential path
        circName = 'Circumferential' + str(COUNT)
        session.Path(name=circName, type=CIRCUMFERENTIAL, expression=((0, outer_radius,
0),
          (-outer_radius, 0, 0), (0, -outer_radius, 0)), circleDefinition=POINT_ARC,
          numSegments=1000, startAngle=0, endAngle=180, radius=CIRCLE_RADIUS)

        # Change datum to Cylindrical coordinates
        dtm = session.odbs[odbPath + '.odb'].rootAssembly.datumCsyses['ASSEMBLY_PIPE-
PARTITIONED-1_ORI-1']
        session.viewports['Viewport: 1'].odbDisplay.basicOptions.setValues(
        transformationType=USER_SPECIFIED, datumCsys=dtm)

        # Set variable to U1 to extract Circ Profile
        session.viewports['Viewport: 1'].odbDisplay.setPrimaryVariable(
        variableLabel='U', outputPosition=NODAL, refinement=(COMPONENT, 'U1'))

        # Create XY Longitudinal data
        pth = session.paths[longitudinalName]
        xy1 = session.XYDataFromPath(name=modelname, path=pth,
includeIntersections=True, projectOntoMesh=False, pathStyle=UNIFORM_SPACING,
numIntervals=1000, projectionTolerance=0, shape=UNDEFORMED, labelType=TRUE_DISTANCE)

        # Write XY Data to file
        x0=session.xyDataObjects[modelname]
        session.writeXYReport(fileName=file_pathL, xyData=(x0, ))

        # Create XY Circumferential data
        pthC = session.paths[circName]
        xy1C = session.XYDataFromPath(name=modelname, path=pthC,
includeIntersections=True, projectOntoMesh=False, pathStyle=PATH_POINTS,
numIntervals=10, projectionTolerance=0, shape=UNDEFORMED, labelType=TRUE_DISTANCE)

        # Create output file for longitudinal
        file_pathC = 'C:/temp/' +'C-' + modelname + '.csv'
        output_file = open(file_pathC,'w')

        # Write XY Data to file
        x0=session.xyDataObjects[modelname]
        session.writeXYReport(fileName=file_pathC, xyData=(x0, ))
        COUNT = COUNT + 1


input_file.close()
output_file.close()
```

Strain and Stress Extraction Script

```
from abaqus import *
from odbAccess import *
from abaqusConstants import *
import __main__

import sys
import os
import odbAccess
import numpy as NP
import math
import section
import regionToolset
import displayGroupMdbToolset as dgm
import part
import material
import assembly
import step
import interaction
import load
import mesh
import optimization
import job
import sketch
import visualization
import xyPlot
import displayGroupOdbToolset as dgo
import connectorBehavior

# File location of Input Variables text file
file_path='C:/Users/Janine Woo/Documents/Thesis/Shell Scripts/Ring/'
input_file= open(file_path+'Input_Variables.txt')

# Create output file and write headers in first row
output_fileResults = open('Stress-Strain Output.csv','w')
headers = 'Model-Name AVG-MAX-PEEQ DEL-SIG-S22-MAX S22-Axial-atMOP S22-Axial-atZero
DEL-SIG-S11-MAX S11-Circ-atMOP S11-Circ-atZero'
output_fileResults.write(headers)
output_fileResults.write("\n")

for line in input_file:
       # Read input variables from file - each row of the input file is a new model
       extracted_line = line
       extracted_list = extracted_line.split()
       modelname = '' + str(extracted_list[0])
       outer_radius = float(extracted_list[1])
       wall_thickness = float(extracted_list[2])
       length_of_pipe = float(extracted_list[3])
       steel_grade = '' + str(extracted_list[4])
       length_of_partition = float(extracted_list[5])
       rotation_of_partition = float(extracted_list[6])
       radius_of_indenter_l = float(extracted_list[7])
       radius_of_indenter_c = float(extracted_list[8])
       indentation_depth = float(extracted_list[9])
       max_op_pressure = float(extracted_list[10])
       mesh_partition = float(extracted_list[11])
       mesh_longitudinal = int(extracted_list[12])
       mesh_circ = int(extracted_list[13])
       mesh_away = int(extracted_list[14])
       instance_name = 'PIPE-PARTITIONED-1'
```

```
	# Open odb file
	odb_file_location = 'C:/temp/'
	odbPath = odb_file_location + modelname
	odbFile = odbPath + '.odb'
	odb = session.openOdb(name=odbPath + '.odb')
	session.viewports['Viewport: 1'].setValues(displayedObject=odb)
	# ---

	# Export maximum PEEQ

		# Go to last step (Pressure-ILI) and last frame of that step
	lastStep = odb.steps[odb.steps.keys()[-1]]
	lastFrame = lastStep.frames[-1]

		# Look at the PEEQ Field Output values
	strain = lastFrame.fieldOutputs['PEEQ']

		# Create a variable for the maximum PEEQ
	maxPEEQ1, maxPEEQ2, maxPEEQ3, maxPEEQ4 = None, None, None, None

		# Go through each element in the last step and frame and find the node
with the maximum PEEQ (repeat for loop 3 more times to find the top 4 maximum PEEQ
values)
	for strainValue in strain.values:
		if (not maxPEEQ1 or strainValue.data > maxPEEQ1.data):
			maxPEEQ1 = strainValue
	for strainValue in strain.values:
		if (not maxPEEQ2 or strainValue.data > maxPEEQ2.data and strainValue.data
< maxPEEQ1.data):
			maxPEEQ2 = strainValue
	for strainValue in strain.values:
		if (not maxPEEQ3 or strainValue.data > maxPEEQ3.data and strainValue.data
< maxPEEQ2.data):
			maxPEEQ3 = strainValue
	for strainValue in strain.values:
		if (not maxPEEQ4 or strainValue.data > maxPEEQ4.data and strainValue.data
< maxPEEQ3.data):
			maxPEEQ4 = strainValue

	# Calculate average of top 4 maximum PEEQ values
	PEEQaverage = (maxPEEQ1.data + maxPEEQ2.data + maxPEEQ3.data + maxPEEQ4.data)/4

	# Write the values of max PEEQ (top 4 elements) to a text file
	PEEQresults = str(modelname) + ' ' + str(PEEQaverage) + ' '
	output_fileResults.write(PEEQresults)

	# ---

	instance = odb.rootAssembly.instances[instance_name]


	# Export maximum Delta Sigma

		# Create a temporary field that is equal to the S values at MOP minus the
S values at Zero Pressure
	s_at_MOP = session.odbs[odbFile].steps['Pressure-
Zero'].frames[0].fieldOutputs['S']
	s_at_0 = session.odbs[odbFile].steps['Pressure-Zero'].frames[-
1].fieldOutputs['S']
	tempField = s_at_MOP-s_at_0

	currentOdb = session.odbs[odbFile]
	scratchOdb = session.ScratchOdb(odb=currentOdb)
```

```
	sessionStep = scratchOdb.Step(name='Session Step',
		description='Step for Viewer non-persistent fields', domain=TIME,
		timePeriod=1.0)
	sessionFrame = sessionStep.Frame(frameId=0, frameValue=0.0,
		description='Session Frame')
	sessionField = sessionFrame.FieldOutput(name='DeltaS',
		description='MOP-Zero', field=tempField)

		# Look at the S Field Output values
	delta_sigmaFrame = sessionFrame.fieldOutputs['DeltaS']

		# Go back to the frames where the pipe is experiencing MOP and 0 pressure
and find the S22 values there
	MOPframe = odb.steps['Pressure-Zero'].frames[0].fieldOutputs['S']
	Zeroframe = odb.steps['Pressure-Zero'].frames[-1].fieldOutputs['S']

	# Get S22 Values
	ElementsGroup = odb.rootAssembly.instances[instance_name].elements
	Elements1 = len(ElementsGroup)

		# Create a variable for the maximum Delta-Sigma-S22
	maxDS1_S22, maxDS2_S22 = None, None

		# Go through each element in Delta Sigma Session Step and find the 2
elements with the maximum S22 values
	for elements in range(Elements1):
		element = ElementsGroup[elements]
		DelSigS22 = delta_sigmaFrame.getSubset(position=INTEGRATION_POINT,
region=element).getScalarField(componentLabel='S22').values[0].data
		if (not maxDS1_S22 or DelSigS22 > maxDS1_S22):
			maxDS1_S22 = DelSigS22
			maxDS1_S22_elementlabel = element.label

	for elements in range(Elements1):
		element = ElementsGroup[elements]
		DelSigS22 = delta_sigmaFrame.getSubset(position=INTEGRATION_POINT,
region=element).getScalarField(componentLabel='S22').values[0].data
		if (not maxDS2_S22 or DelSigS22 > maxDS2_S22 and DelSigS22 < maxDS1_S22):
			maxDS2_S22 = DelSigS22
			maxDS2_S22_elementlabel = element.label

	# Write the average value of Delta Sigma to the text file
	DeltaSigmaAvg = (maxDS1_S22 + maxDS2_S22)/2
	DeltaSigmaResults = str(DeltaSigmaAvg) + ' '
	output_fileResults.write(DeltaSigmaResults)

	for index, element in
enumerate(odb.rootAssembly.instances[instance_name].elements):
		if element.label == maxDS1_S22_elementlabel:
			maxDS1_S22_elementlabel = index

	for index2, element in
enumerate(odb.rootAssembly.instances[instance_name].elements):
		if element.label == maxDS2_S22_elementlabel:
			maxDS2_S22_elementlabel = index2

	# Go back to the frames where the pipe is experiencing MOP and 0 pressure and
find the S22 values there
	MOPframe = odb.steps['Pressure-Zero'].frames[0].fieldOutputs['S']
	Zeroframe = odb.steps['Pressure-Zero'].frames[-1].fieldOutputs['S']

	# Get S22 Values
```

```
	stress1 = MOPframe.getSubset(position=INTEGRATION_POINT,
region=instance.elements[maxDS1_S22_elementlabel])
	S22atMOP1 = stress1.getScalarField(componentLabel='S22').values[0].data

	stress2 = MOPframe.getSubset(position=INTEGRATION_POINT,
region=instance.elements[maxDS2_S22_elementlabel])
	S22atMOP2 = stress2.getScalarField(componentLabel='S22').values[0].data

	stress3 = Zeroframe.getSubset(position=INTEGRATION_POINT,
region=instance.elements[maxDS1_S22_elementlabel])
	S22atZero1 = stress3.getScalarField(componentLabel='S22').values[0].data

	stress4 = Zeroframe.getSubset(position=INTEGRATION_POINT,
region=instance.elements[maxDS2_S22_elementlabel])
	S22atZero2 = stress4.getScalarField(componentLabel='S22').values[0].data

	# Write the maximum average values (from top 2 S22 values at MOP and zero) to
the text file
	S22MOPAvg = (S22atMOP1 + S22atMOP2)/2
	S22Results = str(S22MOPAvg) + ' '
	output_fileResults.write(S22Results)

	S22ZeroAvg = (S22atZero1 + S22atZero2)/2
	S22Results2 = str(S22ZeroAvg) + ' '
	output_fileResults.write(S22Results2)

		# Export maximum Delta Sigma

		# Look at the S Field Output values
	delta_sigmaFrame = sessionFrame.fieldOutputs['DeltaS']

		# Go back to the frames where the pipe is experiencing MOP and 0 pressure
and find the S11 values there
	MOPframe = odb.steps['Pressure-Zero'].frames[0].fieldOutputs['S']
	Zeroframe = odb.steps['Pressure-Zero'].frames[-1].fieldOutputs['S']

	# Get S11 Values
	ElementsGroup = odb.rootAssembly.instances[instance_name].elements
	Elements1 = len(ElementsGroup)

		# Create a variable for the maximum Delta-Sigma-S11
	maxDS1_S11, maxDS2_S11 = None, None

		# Go through each element in Delta Sigma Session Step and find the 2
elements with the maximum S11 values
	for elements in range(Elements1):
		element = ElementsGroup[elements]
		DelSigS11 = delta_sigmaFrame.getSubset(position=INTEGRATION_POINT,
region=element).getScalarField(componentLabel='S11').values[0].data
		if (not maxDS1_S11 or DelSigS11 > maxDS1_S11):
			maxDS1_S11 = DelSigS11
			maxDS1_S11_elementlabel = element.label

	for elements in range(Elements1):
		element = ElementsGroup[elements]
		DelSigS11 = delta_sigmaFrame.getSubset(position=INTEGRATION_POINT,
region=element).getScalarField(componentLabel='S11').values[0].data
		if (not maxDS2_S11 or DelSigS11 > maxDS2_S11 and DelSigS11 < maxDS1_S11):
			maxDS2_S11 = DelSigS11
			maxDS2_S11_elementlabel = element.label

	# Write the average value of Delta Sigma to the text file
	DeltaSigmaAvg = (maxDS1_S11 + maxDS2_S11)/2
```

```
	DeltaSigmaResults = str(DeltaSigmaAvg) + ' '
	output_fileResults.write(DeltaSigmaResults)

	for index, element in
enumerate(odb.rootAssembly.instances[instance_name].elements):
		if element.label == maxDS1_S11_elementlabel:
			maxDS1_S11_elementlabel = index

	for index2, element in
enumerate(odb.rootAssembly.instances[instance_name].elements):
		if element.label == maxDS2_S11_elementlabel:
			maxDS2_S11_elementlabel = index2

	# Go back to the frames where the pipe is experiencing MOP and 0 pressure and
find the S11 values there
	MOPframe = odb.steps['Pressure-Zero'].frames[0].fieldOutputs['S']
	Zeroframe = odb.steps['Pressure-Zero'].frames[-1].fieldOutputs['S']

	# Get S11 Values
	stress1 = MOPframe.getSubset(position=INTEGRATION_POINT,
region=instance.elements[maxDS1_S11_elementlabel])
	S11atMOP1 = stress1.getScalarField(componentLabel='S11').values[0].data

	stress2 = MOPframe.getSubset(position=INTEGRATION_POINT,
region=instance.elements[maxDS2_S11_elementlabel])
	S11atMOP2 = stress2.getScalarField(componentLabel='S11').values[0].data

	stress3 = Zeroframe.getSubset(position=INTEGRATION_POINT,
region=instance.elements[maxDS1_S11_elementlabel])
	S11atZero1 = stress3.getScalarField(componentLabel='S11').values[0].data

	stress4 = Zeroframe.getSubset(position=INTEGRATION_POINT,
region=instance.elements[maxDS2_S11_elementlabel])
	S11atZero2 = stress4.getScalarField(componentLabel='S11').values[0].data

	# Write the maximum average values (from top 2 S11 values at MOP and zero) to
the text file
	S11MOPAvg = (S11atMOP1 + S11atMOP2)/2
	S11Results = str(S11MOPAvg) + ' '
	output_fileResults.write(S11Results)

	S11ZeroAvg = (S11atZero1 + S11atZero2)/2
	S11Results2 = str(S11ZeroAvg) + ' '
	output_fileResults.write(S11Results2)
	output_fileResults.write("\n")


input_file.close()
output_fileResults.close()
```