Scripting for Modeling Local Buckling of a Spiral Welded Pipe

by

©Adhi Susilo

A thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of

Master of Engineering

Faculty of Engineering & Applied Science

Memorial University of Newfoundland

May, 2014

St. John's Newfoundland

"Doxa to Theo panton eneken"

St. John Chrysostom

Abstract

The demand of large diameter pipes is significant. In the offshore industry, seamless and UOE (U-ing, O-ing, and Expanding) pipes are used commonly, but the production costs of these kind of pipes is not low. The spiral welded pipe (SWP) is considered the answer to this challenge. The SWP has a low cost of production and yet it can be said that the SWP is as good as the UOE pipe. However, some opinions give bad impressions about a SWP. Later, studies show that weaknesses found with SWPs are due to poor manufacturing processes.

This work is developing a model to stimulate SWP strength with a finite element analysis tool called ABAQUS. The pipe is loaded by internal pressure and bending moment and there is imperfection on the pipe geometry. Therefore, local buckling is expected. The model is built mainly by a Python script. This work emphasizes the use of scripting for simulating a model, but some people may be not familiar with Python. Therefore, other programs (e.g. FORTRAN and GNUPLOT) are involved.

There are two main models for this work. The first model is a UOE pipe; it is called the Benchmark Pipe (BMP) model, this model of this pipe has been calibrated and verified against large-scale physical tests, and the second model is for the SWP model. Referring to the tested model, confidence in the SWP modelling procedures can be established. Both models have the same dimensions, the same material properties, and are isotopic. In this work, the SWPs with different setting are also examined.

The main conclusions from this study include:

1. The SWP modelling procedures provided consistent results with the BMP model in terms of displacement response, moment-curvature behaviour and local deformation mechanisms.

2. The use of scripting provide an effective tool for the pre- and post-processing of the simulation. The main pre-processing attributes included generation of the pipe diameter, wall thickness, pitch, and initial geometric imperfections. The post-processing script facilitates the analysis by transforming field variables such as displacement, rotation and section forces into section moment, curvature, deflection and axial strain.

3. The use of partitioning was found to have detrimental effects on the pipe mechanical response that resulted in non-intuitive behaviour. This further supports the need for scripting tools in the simulation and analysis.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Shawn Kenny, who has given me an opportunity to work in this subject area and for his guidance throughout finishing my studies. I also would like to thank Dr. Amgad Hussein, as co-supervisor, the professors and the staff of the Faculty of Engineering & Applied Science, at Memorial University, in St John's for helping me develop my knowledge in this area. Also, I thank my colleagues for sharing their experiences and knowledge so I can solve my problems, and I thank Mr A. Fatemi for his generosity that allows me using his work.

Furthermore, I would like to thank the Wood Group Chair in Arctic and Harsh Environments Engineering, the Research and Development Corporation (RDC) of Newfoundland and Labrador, and MITACS for their financial support so I can finish this work.

At last, I would like to thank my father who is in heaven and my mother who is overseas for their support and prayers from a distance. Finally, I would like to thank and praise God for His abundant blessings I have received.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Demand for crude oil and gas is still high. Exploration and production of oil and gas will impact on increasing demand for pipes (seamless, UOE, or spiral welded pipe). As reported by TWI (The Welding Institute), spiral welded pipe consumes less cost than UOE, estimated at 10-15% cheaper [22]. This can boost to increase spiral welded pipe production.

This work is modeling local buckling of a spiral welded pipe. The problem is analyzed by a software which is based on the finite element method. The modeling is done by scripting. The scripting will prepare the input for the software and harvesting the output from the software.

This chapter is divided into two sections. The first section is explaining the components of the model and the second section is describing aim and content of this work.

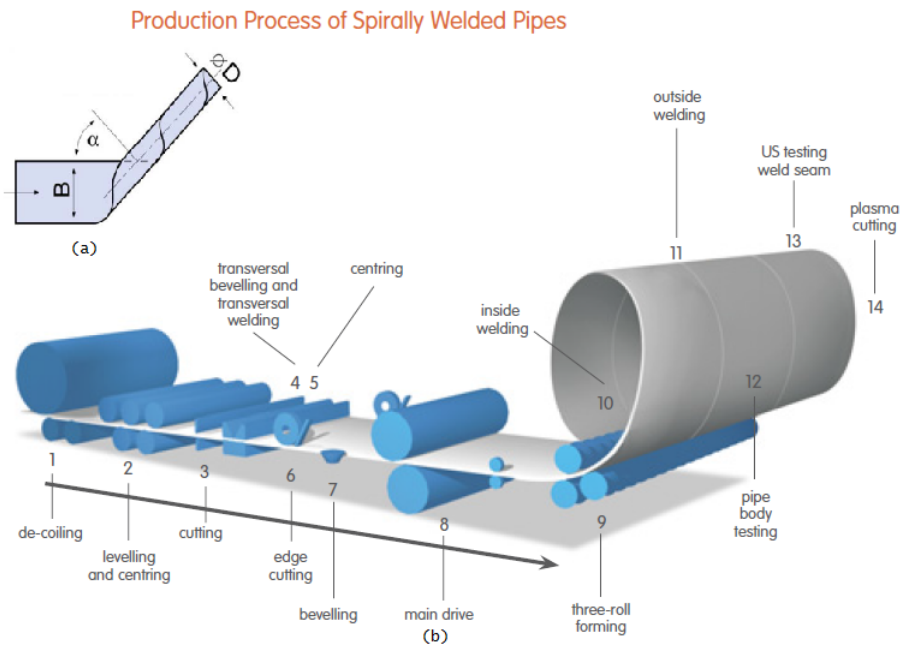


Figure 1.1: Scheme (a) and production process of SWPs (b), from [24] and [1] respectively.

1.1 Background

This section will describe briefly the main factors of this work; the pipe, the software, and the scripting.

1.1.1 The pipe

The transport of oil and gas over long distances is driving the need for producing large diameter pipe that is cost-competitive with mechanical performance equal with other pipe technologies (e.g. UOE). These pipes are manufactured in two ways, one is the UOE process and the other is the spirally welded process. In the UOE process, a plate is bent in U-shape first, then it is formed like O-shape, and finally it is welded longitudinally and expanded to get the correct roundness [9].

The other process is forming the pipe by rolling the plate and welding it spirally, as shown on Figure 1.1. This process gives a lot of advantages. These are the advantages of the SWP as stated by reference [21]; the spiral welded pipes are better weld ability, fewer field welds, smaller tolerances, higher strength steels, and inherent safety margins. However, some opinions describe that the spiral welded pipes are not as strong as UOE pipes. This is not 100% true. The weakness of SWP is not because of the spiral shape, but due to bad manufacturing. Study from [24] concludes that the manufacturing process must be considered to define the axial strain capacity of the spiral welded pipe.

As stated on reference [21], the forming and welding of spiral welded pipes demands good technology of welding, only internal and external submerged arc welding that gives top quality product. Other studies ([5],[11],[28]) also found that the SWP is as good as the UOE. The spiral seam impacts none to the pipe performance. Figure 1.2 depicts an example of the internal and external submerged arc welding. Reference [11] also concludes that spiral welded pipes cost less then UOE pipes.

The manufacturing of spiral welded pipes can be dated back in 1880s. That time, the machine (see Figure 1.3) could produce pipes in sizes 4-30 inches [10]. Today, manufacturers can make the pipes with diameters of up to 60 inches and up to 80 feet in length. Driven by demand, especially in energy sectors, global market for spiral welded pipes and tubes is projected to reach 24.6 million tons by 2018 as predicted by Global Industry Analysts [19].

This work is intended to examine the mechanical performance of spiral welded pipe strength in comparison with UOE pipe. The pipe is internally pressurized and bent

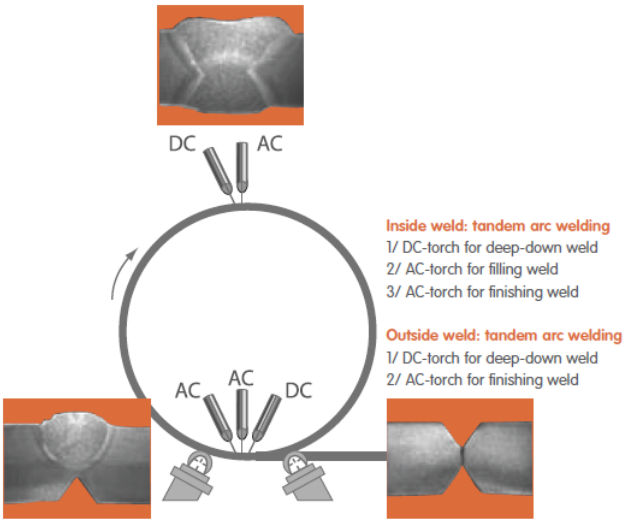


Figure 1.2: The welding process [24].

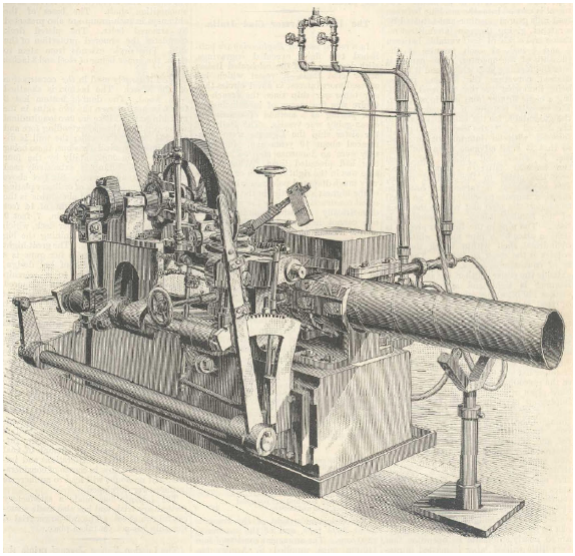


Figure 1.3: Machine for the manufacture of spirally welded tubing [10].

by bending moments on its ends. Bending load causes the pipe cross section to ovalize. Study of [13] shows that rigidity is reduced by ovalization. However, the internal pressure causes the cross section to expand. Therefore the internal pressure reduces the ovalization. Work of [15] shows that the internal pressure makes the pipe 6x stronger than the pipe without internal pressure. This pressure delays the onset of ovalization.

Beside the loads, the strength of the pipe is also influenced by plastic characteristics of the material as shown by [12], [16], and [24]. Further studies also found another factor that influenced the pipe strength which is the imperfection. The imperfections are changes in diameter (ovality) and wall thickness due to the manufacturing process. For the girth weld there are weld process imperfections including radial offset misalignment and differences in pipe diameter of adjoining pipe. The UOE has a long seam weld. The spiral pipe has a continuous spiral weld. Dents and bulges are more damage characteristics. Studies of [12], [16], and [17] show that local imperfections cause the bending strain capacity of the pipe reduce significantly. Recent studies of [6], [7], [8], and [23] also show that the imperfection influences the strain capacity.

For this work, the imperfection will be emphasized. The model, built by scripting and evaluated by Abaqus, will examine local buckling of a pipe due to loads and the imperfection.

1.1.2 The software

Finite element analysis will be used to simulate the model. There are many software base on the finite element method. Abaqus 6.12 is chosen for this work. Abaqus is a sophisticated simulation program. It can be used for variety problems in engineer-

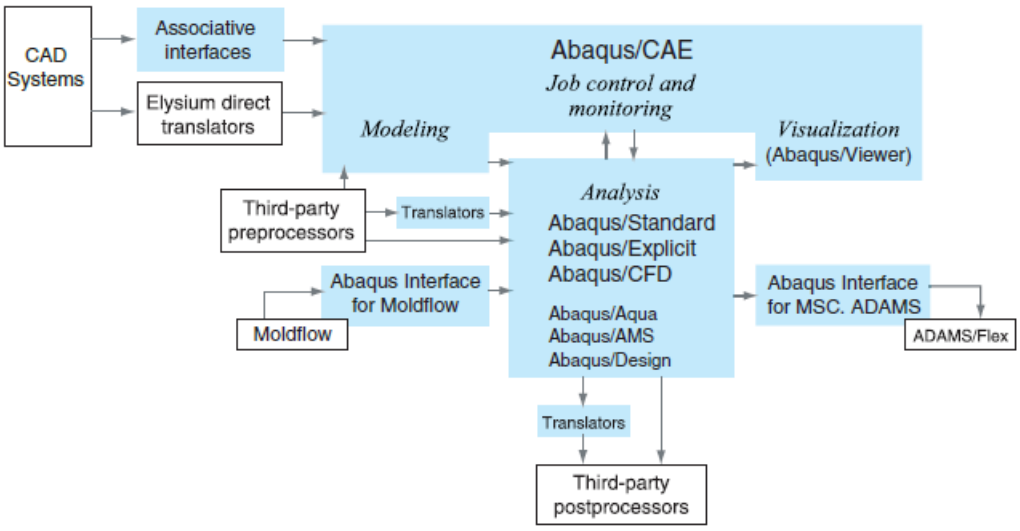


Figure 1.4: Abaqus products [2].

ing; not just for structural problem, but also for heat transfer, mass transfer, fluid dynamics, electrical, magnetism, acoustics, etc. Abaqus can also handle linear and nonlinear problems.

For further detail about Abaqus and its ability can be on reference [2]. Figure 1.4 shows that Abaqus can interact with another software. Input can be created by itself or imported from another software, and then the result also can be exported to another software.

As shown on Figure 1.4 and also stated on reference [26], the Abaqus consists of four core software products; Abaqus/CAE, Abaqus/Standard, Abaqus/Explicit, and Abaqus/CFD. A brief introduction about these products is described bellow:

- Abaqus/CAE

  CAE stands for or '*Complete Abaqus Environment*'. It is like a moderator. The CAE creates the input file or imports it from another software, then it calls the

engine to compute the simulation, finally the CAE will show the result on the visualization module or export it to another software.

- Abaqus/Standard

  A general-purpose finite element analysis tool that uses implicit integration scheme (traditional).

- Abaqus/Explicit

  A special-purpose finite element analysis tool that uses explicit integration scheme to solve highly nonlinear problems.

- Abaqus/CFD

  CFD stands for '*Computational Fluid Dynamics*'. It is a finite element analysis tool that provides advanced computational fluid dynamics capabilities.

Though Abaqus is complex, its work can be divided into three parts; creating input, evaluating the input , and then the last one is harvesting the output. The flow cart is shown on Figure 1.5. Beside knowing the stages of Abaqus process, it will be better if user also knows three programming languages required by Abaqus, which are Fortran, C, and Python. Users can develop their subroutines with Fortran, or write output database with C, or do scripting with Python.

This work will emphasize the ability of Python working on modeling with Abaqus. It will be shown here, knowing Python is enough to do a simulation. Python script is like Abaqus/CAE that moderates the whole process, from preparing the input to harvesting the output.

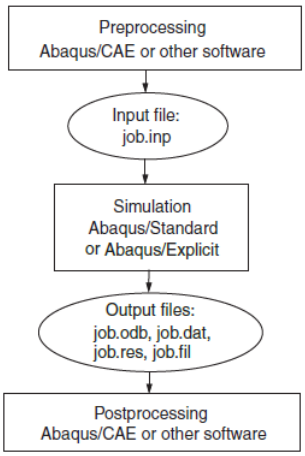


Figure 1.5: Abaqus stages [2].

1.1.3 The scripting

Modern computation is not just computing numbers but also dealing with managing data. Reference [14] defines traditional programming, a program for computing numbers only, as *system programming*. This program usually uses languages such as Fortran, Pascal, C, C++, etc. The other program which is also dealing with managing data and tools is called *scripting*. This program usually is written in Python, Ruby, Perl, etc.

As mentioned in the previous section, Abaqus needs Fortran to do the computation and C to write the output. To manage the data and these tools, Abaqus needs Python. Using the Python scripting language, the input deck, job submission, and analysis of export data can be conducted. This whole process, from beginning to the end, can be done by scripting. The system programming only does the calculation.

Abaqus uses Python as the scripting language. Following is a brief introduction to Python, where more detailed information can be found in online resources or books

(for example reference [18]). Online resources [25] and [27] are good sources to start with. In 1990s, Guido Van Rossum created Python. It is named after a comedy program in television called *Monty Python's Flying Circus*. The program is an open-source programming, structured, and high-level language. It can be used for wide tasks; handling data (numeric or non-numeric), handling tools, and of course the computation. Beside working with different tools, Python can work with different platforms too, such as Microsoft Windows, Macintosh, and Linux. It makes Python accessible from any operating system.

In the following paragraphs a brief introduction to key elements of the Python language is explored. The whitespace or indentation is important in Python, as it defines a block. The indention shows where blocks start and finish. Other program, like C language, uses block delimiters, for example a pair of brackets '{ }'. Next, Python usually handle data in three forms; *list, tuple,* and *dictionary*, as explained below

- List

  It is a list of values, it may be numeric non non-numeric data. A list is mutable, elements of a list can be changed by adding or deleting the elements. A list is defined using square brackets '[ ]'. For example:

  ```
  >>> mylist = ['book', 'pipe', 3.14]
  ```

  This list contains strings 'book' and 'pipe' as well as the number 3.14. Items in a list are accessed by their index, the index starts at zero. See following example:

  ```
  >>> mylist =['book', 'pipe', 3.14]
  >>> mylist[0]
  'book'
  >>> mylist[1]
  'pipe'
  ```

```
>>> mylist[2]
3.14
```

- Tuple

  Tuple is similar to the list, except tuple is **immutable**, the set cannot be changed. You can not add or delete item of a tuple. Once you have set a tuple, there is no way to change it whatsoever: you cannot add, change, or remove elements of a tuple. A tuple is defined using parentheses '( )', but for a simple tuple, it can be written by commas. For example:

  ```
  >>> mytuple = 'force', 180, 'stress'
  ```

  However, it is often necessary to use parentheses to distinguish between different tuples

  ```
  >>> mytuple = ('force', 180, ('stress','weld'))
  >>> mytuple
  ('force', 180, ('stress','weld'))
  >>> mytuple[0]
  'force'
  >>> mytuple[1]
  180
  >>> mytuple[2]
  ('stress','young')
  >>> mytuple[2][0]
  'stress'
  >>> mytuple[2][1]
  'weld'
  ```

- Dictionary

  Dictionary is like a list, it is mutable. However, the elements in a dictionary are not bound to index. The element is accessed using its key. Every element in a dictionary has two parts: a *key*, and a *value*. Calling a key of a dictionary gives its value. Dictionary is stated by curly braces '{ }', each element consists

of three parts; the first part is its key, then a colon, and then the last part is its value. For example:

```
>>> mydictio = {‘material’: ‘steel’, ‘force’:10, ‘type’:‘UOE’}
>>> mydictio
{‘force’:10, ‘material’: ‘steel’,  ‘type’:‘UOE’}
>>> mydictio[‘material’]
‘steel’
>>> mydictio[‘type’]
‘UOE’
>>> mydictio[‘force’]
10
```

Following will be shown an example of an output data from Abaqus, `odb.rootAssembly.elementSets[‘SET-MID’]`, that is arranged in lists, tuples, and dictionaries.

```
 ({’elements’: ([’OdbMeshElement object’, ’OdbMeshElement object’,
’OdbMeshElement object’, ’OdbMeshElement object’,
...
...
’OdbMeshElement object’, ’OdbMeshElement object’],),
’faces’: None, ’instances’: (({’analyticSurface’: None, ’beamOrientations’:
’BeamOrientationArray object’, ’elementSets’: ’Repository object’,
’elements’: ’OdbMeshElementArray object’, ’embeddedSpace’: THREE_D,
’materialOrientations’: ’MaterialOrientationArray object’, ’name’: ’PIPE-1’,
’nodeSets’: ’Repository object’, ’nodes’: ’OdbMeshNodeArray object’,
’rebarOrientations’: ’RebarOrientationArray object’, ’rigidBodies’:
’OdbRigidBodyArray object’, ’sectionAssignments’: ’SectionAssignmentArray
object’, ’surfaces’: ’Repository object’, ’type’: DEFORMABLE_BODY}),),
’isInternal’: False, ’name’: ’SET-MID’, ’nodes’: None})
```

Reference [20] is a good book to start to do scripting for Abaqus. The book contains a lot of examples of scripting.

1.2 Objective and Outline

The objective of this work is creating a model to simulate a spiral welded pipe under pressure and bending moment loads simultaneously. The model is built by scripting and then it is executed in Abaqus. Results of this model are compared to the results from a bench mark model.

A brief outline of the thesis content is presented in the following paragraphs. Chapter 1 contains the introduction of the spiral welded pipe, the software, and the program. Detail in creating the model and its result will be discussed in the next chapters. Chapter 2 explores designing the model. It starts with presenting the element of the model, then it discusses the imperfection, the flow chart, and then it is closed by testing the model.

Further discussion about the scripting is found on Chapter 3. This chapter contains three sections; the first section is about scripting for the bench mark model, the second section is the scripting for the spiral welded pipe, and the last section is discussing scripting for harvesting the output. Results and discussion will be presented on Chapter 4. And finally, conclusions and recommendations are shown on the last chapter, Chapter 5.

Chapter 2

Designing the Model

This chapter discusses the procedure to build the model. The model is a program to simulate a pipe under internal pressure and bending moment load simultaneously. The model is a finite element analysis that runs in Abaqus and it is developed using scripting. Before discussing the program, basic components of the model, which are the element and geometry imperfection, are explained. The programs, the inputs, and the outputs will be shown in the appendices. These appendices are suitable for the models discussed here, another model with different dimensions and grid will have different number of nodes, but the flowchart does not change.

2.1 The Element of the Model

The model is built using elements. The family of element is shell element because it is assumed that the length of the pipe is significantly larger than its thickness and the stress through thickness is negligible. There are two types of shell element, i.e. conventional shell and continuum shell, as depicted on Figure 2.1.

Further detail about this element can be found at Abaqus User's Manual [3]. In this

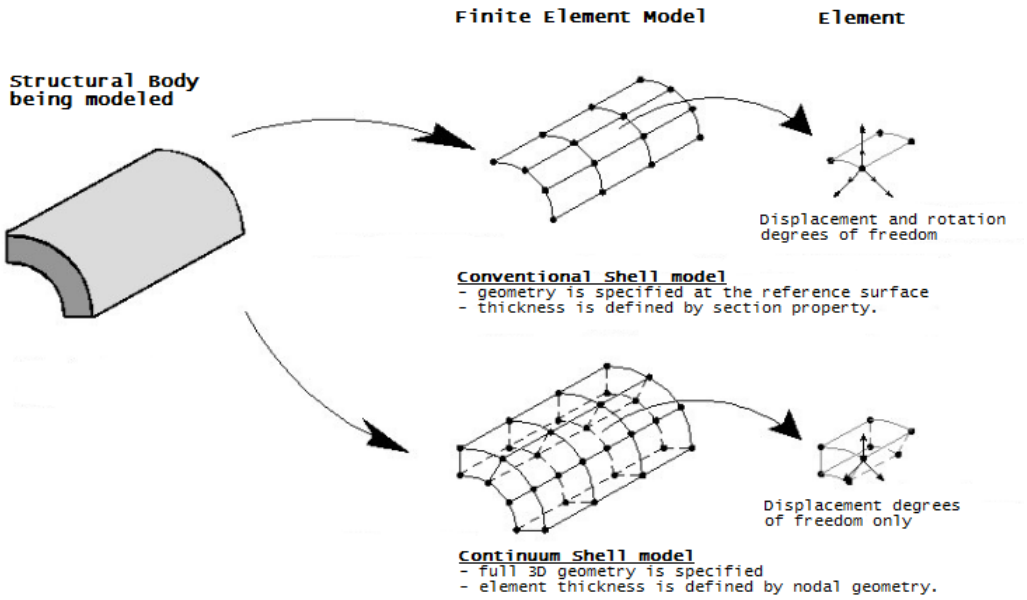


Figure 2.1: Two types of the shell element [3].

work, the model is built using conventional shells. The element can be summarized as following,

- Degrees of freedom (dof)

  Six-dof is used, they are referred to as follows:

  (1) x-displacement

  (2) y-displacement

  (3) z-displacement

  (4) Rotation about the x-axis, in radians

  (5) Rotation about the y-axis, in radians

  (6) Rotation about the z-axis, in radians.

- Number of nodes

  The element is linear element or first-order element, and it is a triangle shape, so number of nodes will be three. It is called S3.

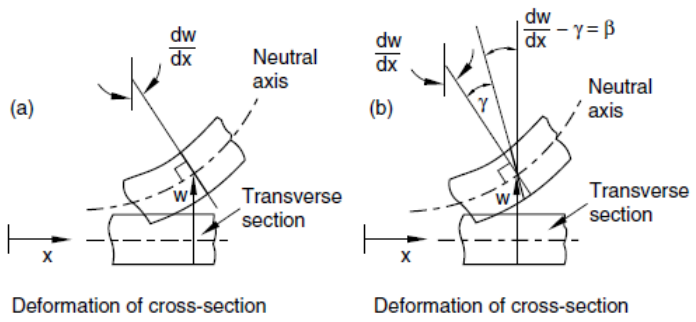


Figure 2.2: Behavior of transverse shell sections in (a) thin shells and (b) thick shells [3].

- Formulation

  Shell element has three classes; general-purpose shell, thin shell, and thick shell. The thin shell analysis neglects the effects of transverse shear deformation, but the thick shell analysis does not neglect those effects. The model developed in this study uses thin shell analysis.

- Integration

  Abaqus uses Gaussian quadrature to evaluates the material response at each integration point in each element. There are two methods of the integration; full and reduced integration [3]. The full integration refers to the number of Gauss points required to integrate the polynomial terms in an element's stiffness matrix exactly, for example a linear quadrilateral element uses two integration points in each direction.

2.2 The Imperfection of the Model

The model simulates a pipe under internal pressure and bending moment. Local buckling is expected to happen. As stated in the Abaqus manual, the local buckling

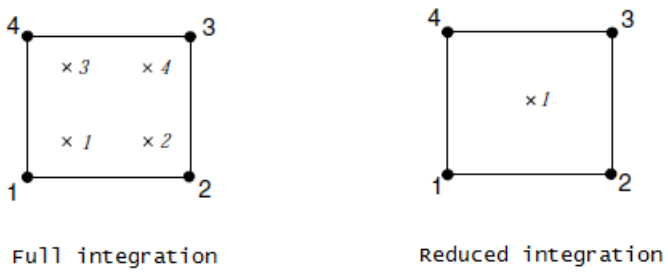


Figure 2.3: Full and reduced integration of a S4 element [3].

creates a discontinuous response (bifurcation) that cannot be analyzed directly. To simulate this problem, we have to change it into a problem with continuous response, which can be done by introducing an imperfection pattern in the geometry of the model so that there is some responses in the buckling mode before critical load is reached [3].

There are three ways defining imperfection in Abaqus:

1. linear superposition of buckling eigenmodes
2. from the displacements of a static analysis
3. by specifying the node number and imperfection values $(\Delta x, \Delta y, \Delta z)$ directly.

The third method is used in this model. For example, there is imperfection on the half length of the pipe with a wavy shape, as depicted on Figure 2.4 (this is only an example of the imperfection, the model uses different shape of imperfection but similar to this wavy shape), then to get the imperfection on the model, the input file of the model is modified by inserting a line code that includes an imperfection file. Detail about implementing the imperfection on the model is discussed in the next section.

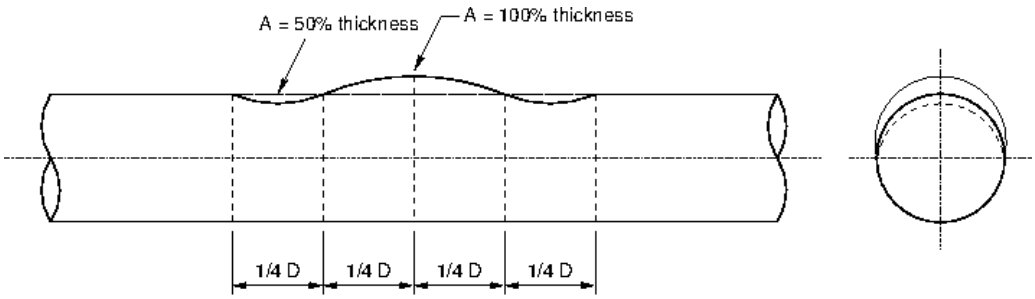


Figure 2.4: The imperfection.

2.3 The Programming of the Model

Simulating model is done in three stages, as shown on Figure 2.5. First is 'pre-processing' which is preparing the input, then is 'processing' which is compiling and executing the program, the last stage is 'post-processing' which is working with the output.

A lot of efforts are done in preparing the input, because the wrong input will give the wrong result and just a waste of effort. Following subsection will discuss each stage further.

Figure 2.5: Flow chart of running the model.

2.3.1 Pre-processing

The pre-processing starts with creating a model; it can be done by Abaqus/CAE or by running a Python script. In this work, a model is created by Python script, it is

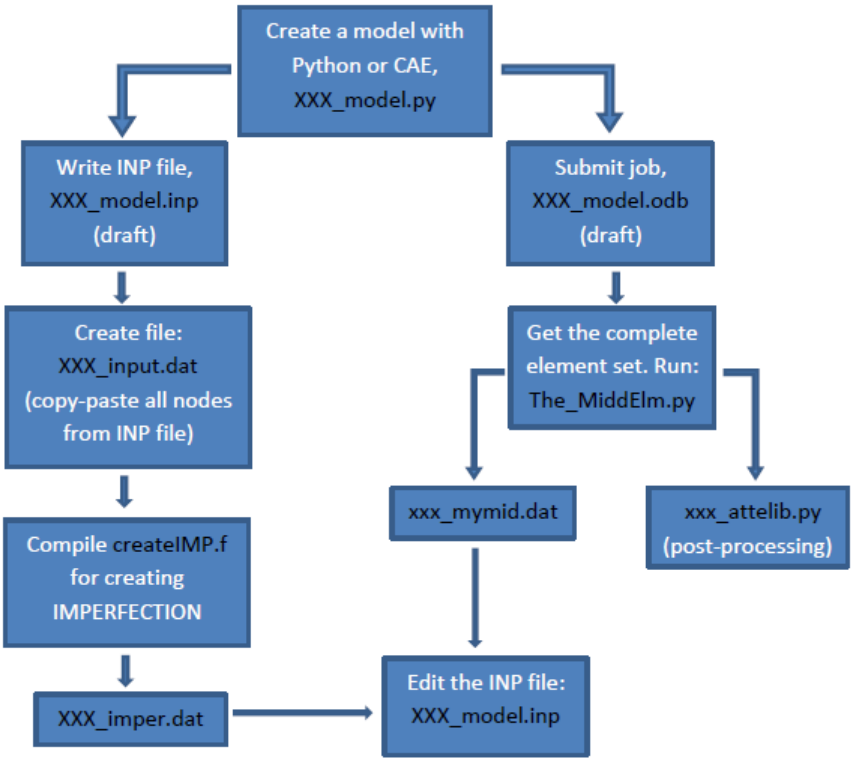


Figure 2.6: Pre-Processing.

named `XXX_model.py`[1] (see Appendix A1 and A2). The script is executed to build the model then the job is submitted. However, this is not the complete model yet. The imperfection needs to be involved in the model. Also, it is found that some elements are missing from the requested set. Most output are property of the element, therefore the lack of element will give error outputs. Figure 2.6 shows the flow chart of the pre-processing. Further explanation will be given in the next section.

There are two things that must be done in the pre-processing stage, i.e.

- inserting the imperfection, and

[1]XXX is either BMP or SWP

- editing the requested element set.

We start with inserting the imperfection. Firstly, we build a model from `XXX_model.py`, then the model produces an input file called `XXX_model.inp`, it is done by clicking 'Write Input' button in the job module of Abaqus/CAE. Then create a file called `XXX_input.dat` using a text editor by copied-pasted the nodes, this file contains all nodes of the model. An example of this file is shown on Appendix A3. These nodes and their coordinates represent 'prefect' geometry of the model.

A FORTRAN program called `createIMP.f` (see Appendix A4) is used to create the imperfection. This program needs the `XXX_input.dat` and will give the `XXX_imper.dat` as the result, as shown on Appendix A5. This file will be called by `XXX_model.inp` to generate the imperfection.

After the imperfection is created, the next step is editing the element set. It is found that some elements on the requested set are missing, especially if triangle elements are used. It might be a bug of the Abaqus v.12. Therefore, another program is developed to fix this program. A Python script called `The_MiddElm.py` (see Appendix A6) will search the missing elements and add them to the requested set. This program reads data from an `odb` file. To get this file, `XXX_model.odb`, we click 'Submit' button in the job manager of Abaqus/CAE. After one step of calculation is done, we kill this job because at this stage we just need the information about nodes and their element set. The complete computation will be conducted after we fix the missing elements using the nodes and the element set that we got from the first run.

The `The_MiddElm.py` will produce complete element set plus additional element set (`xxx_mymid.dat`) and a library of requested node set (`xxx_attelib.py`), as shown

on Appendix A7 and A8 respectively. However, the `xxx_attelib.py` will be needed for post-processing. Running `The_MiddElm.py` takes times when number of node is huge, for example reading 300,000s nodes needs almost 1 hour. Visualizations of this work are depicted on the following figures; Figure 2.7a shows some elements are not included in the requested set, and Figure 2.7b shows edited element set, the complete element set.

Now, with the imperfection file and the new element set, the `XXX_model.inp` is edited. After deleting brackets in file `xxx_mymid.dat`, we copy-paste this data into element set 'SET-MID' therefore the `XXX_model.inp` will have the complete element set of 'SET-MID', see following example:

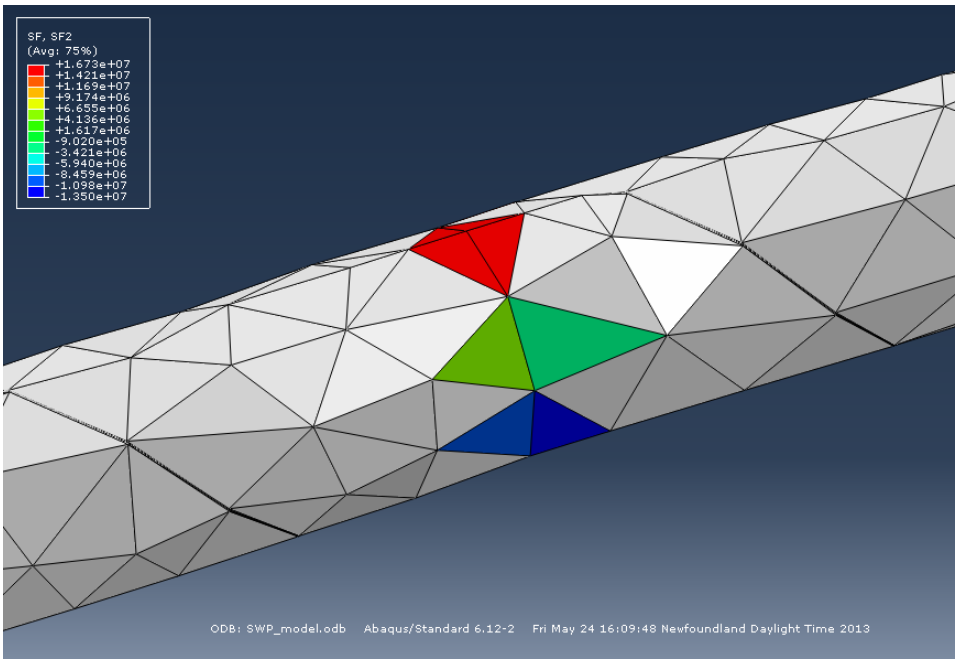
```
*Elset, elset=Set-MID, instance=Pipe-1
<copy-paste new numbers here, old numbers are removed>
*Nset, nset=Set-Tpoint1, instance=Pipe-1
```

To include the imperfection in the `XXX_model.inp`, add this line between 'End Assembly' and 'MATERIALS' as follow:

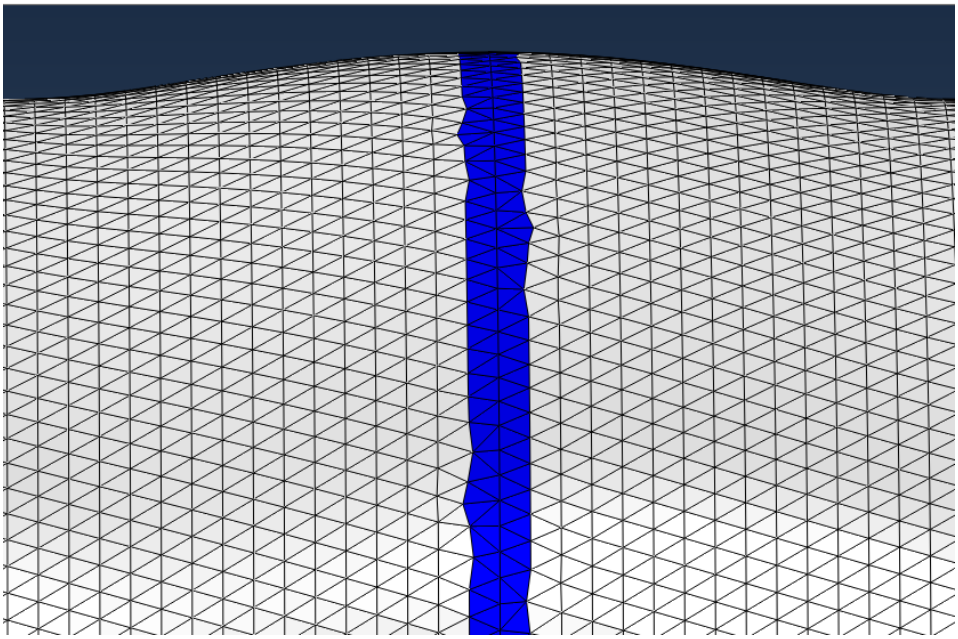
```
*End Assembly
**
**
*IMPERFECTION, INPUT=XXX_imper.dat
**
** MATERIALS
**
*Material, name=Material-Pipe
```

where '*' represents key word and '**' represents comment, 'XXX' is replaced by 'BMP' or 'SWP'.

Now the `XXX_model.inp` is ready for simulation, the element set is correct and the imperfection is included. Appendix A9 through A12 show difference between before



(a) Not correct elemet set



(b) Correct element set

Figure 2.7: The element set.

and after editing of the input file, `XXX_model.inp`.

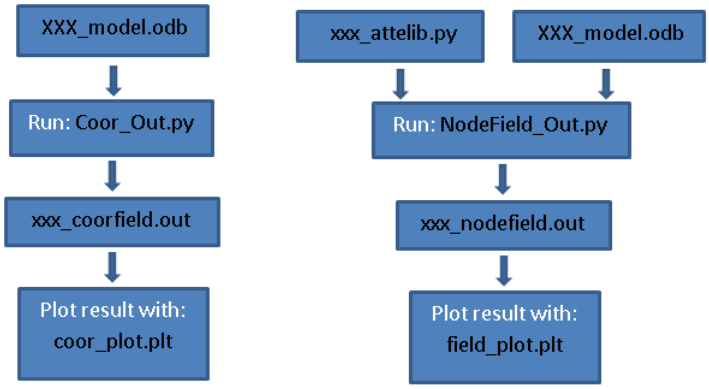
2.3.2 Processing

This part is the easiest task in this simulation. First, we put files `XXX_model.inp` and `XXX_imper.dat` in the same folder, then we run this model in a command prompt, by typingTEMP: '**abaqus job=XXX_model interactive**' or '**abaqus job=XXX_model cpus=2**', where option 'interactive' shows feedback of the process and 'cpus=2' defines number of CPU required. The Abaqus/CAE cannot handle the 'IMPERFECTION', therefore a command prompt is needed.

2.3.3 Post-processing

During computation Abaqus is writing reports in text format and output in binary format. Abaqus/CAE will show the binary format in readable format by human, in form of graphs or reports. It is easy to get 3D graphs, but often 2D graphs are requested. Abaqus/CAE also provide the mechanism to generate 2D plots, but it needs additional effort, and it will be tedious job when dealing with a lot of numbers.

On this step (post-processing), there will be two things to be done: examining the pipe deflection and computing other parameters. Python scripts are created to read the binary output file (`XXX_model.odb`), the readers are called `Coor_Out.py` (see Appendix A13-1) and `NodeField_Out.py` (see Appendix A13-2). The first reader finds the deflection and the second reader finds other parameters; i.e. curvature, strain, and moment. The flowchart of the readers are shown on Figure 2.8. However, only reader two will be discussed further. The Appendix does not include `xxx_coorfield.out` nor `coor_plot.plt` file.



(a) Finding the deflection (b) Finding other parameters

Figure 2.8: Post-Processing.

As shown in Figure 2.8b, the reader needs another file called `xxx_attelib.py` beside the `odb` file. The `xxx_attelib.py` contains list of requested nodes and the elements that attach to those nodes. Most of the variable output belongs to the element, for example the stress (S). Therefore, to find stress on a node, we have to extrapolate stresses on all elements that attach to that node.

The `NodeField_Out.py` takes data from the output file, then does necessary computations, and finally prints the required outputs on ASCII file format, named `xxx_nodefield.out` (see Appendix A14). Now the final output is suitable for plotting 2D, because data are presented in columns. Using GNUPLOT script, `field_plot.plt` (see Appendix A15), we make 2D graphs.

2.4 Testing the Model

There are two tests which will be conducted on this work, i.e. numerical test and parameter test. The numerical test relates to the software and the parameter test

relates to the pipe.

2.4.1 Numerical test

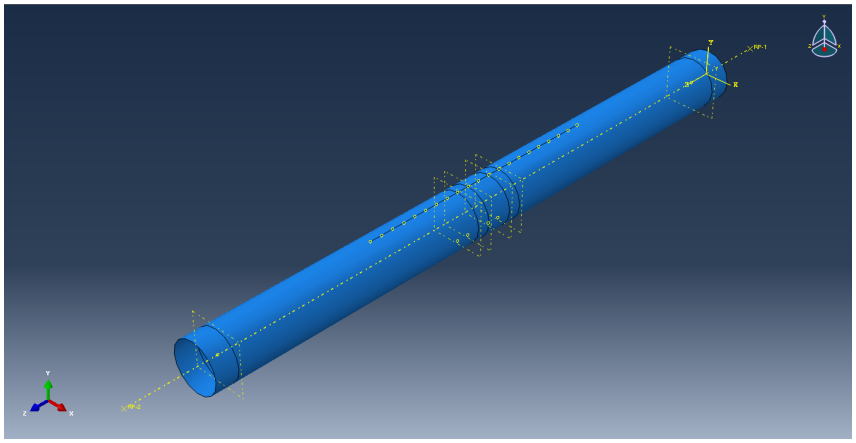
Creating set on the model can be very helpful for harvesting output from the computation. A set on certain location usually is created before meshing. This is done by partition. However, creating partition seems to change the stiffness of the pipe numerically. Therefore, we are going to make a numerical test, a model without partition is compared to a model with partition. To have a set on a model without partition, we are going to use the 'bounding box' module. The module will collect any node inside the box. This module will be discussed further in Chapter 3.

Figure 2.9 shows creating a set without partition with the bounding box module and Figure 2.10 depicts a model with partition. The bounding box makes the model simpler because there are no partitions, but sometimes the bounding box can not capture the wanted location, especially when we have free mesh and it is not a simple geometry.

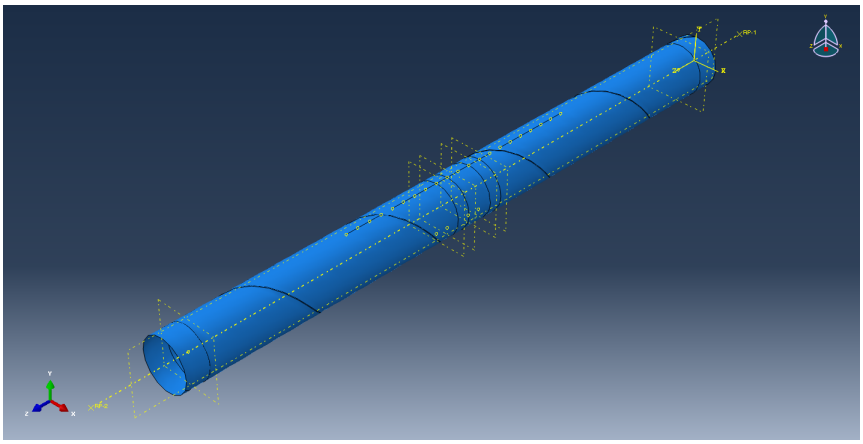
Figure 2.9: Creating a set using a bounding box.

2.4.2 Parameter test

This is not a laboratory test, but still using the software to simulate the pipe under certain conditions. This test relates to the geometry and load applied to the pipe.



(a) Seamless Pipe



(b) Spiral Welded Pipe

Figure 2.10: Creating a set using partitions.

There are four tests will be conducted here, they are called:

- Material Test
- Ovality Test
- Pitch Test
- Internal Load Test

First, we are going the see the material influence on the pipe; we will run a model with uniform material and a model with two kind of material. The material for weld-

ing has yield stress 1.1x than the base material properties of the pipe. Figure 2.11 describes the materials used in this model. Next, we are going the see the influence of the cross section shape of the pipe. There are three shapes; ellipse horizontally, circle, and ellipse vertically, as sketched on Figure 2.12. Figure 2.13 shows 3D of those pipes.

The third test is to see the influence of the pitch angle of the spiral welded pipe. We are going to have 40° and 60° pitch angle respectively, as depicted on Figure 2.14. The last test, we are going to have two internal load conditions; the pipe is pressurized and the pipe with zero pressure.

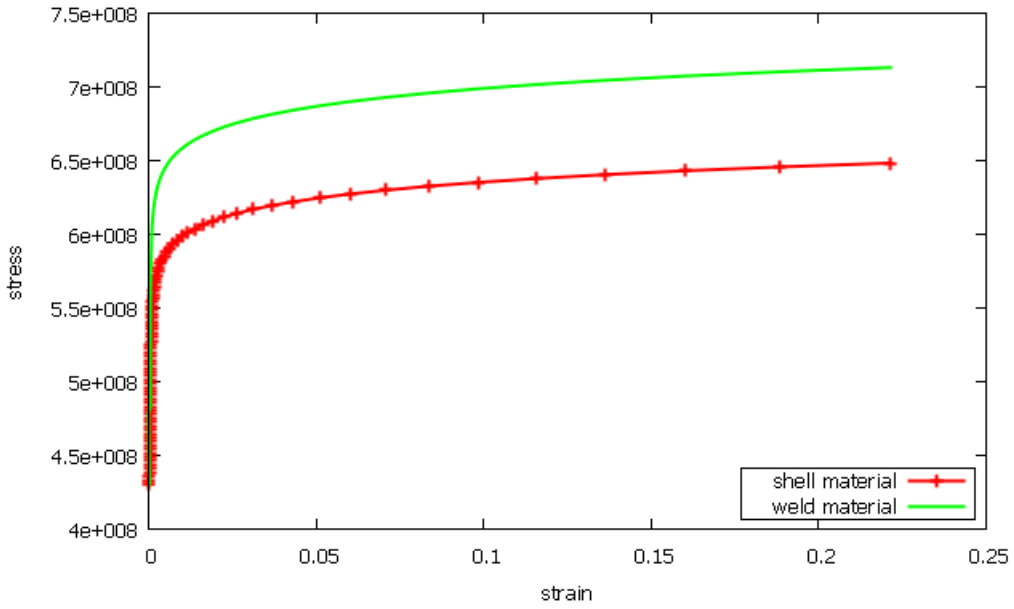


Figure 2.11: Material property of the weld and the shell.

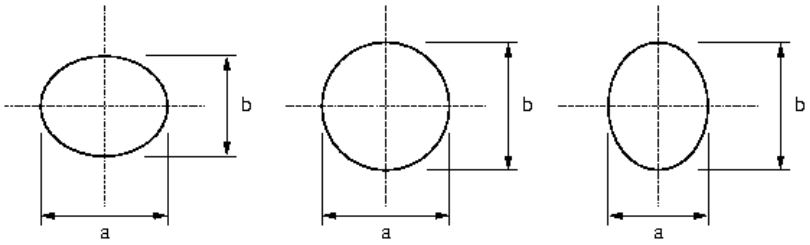
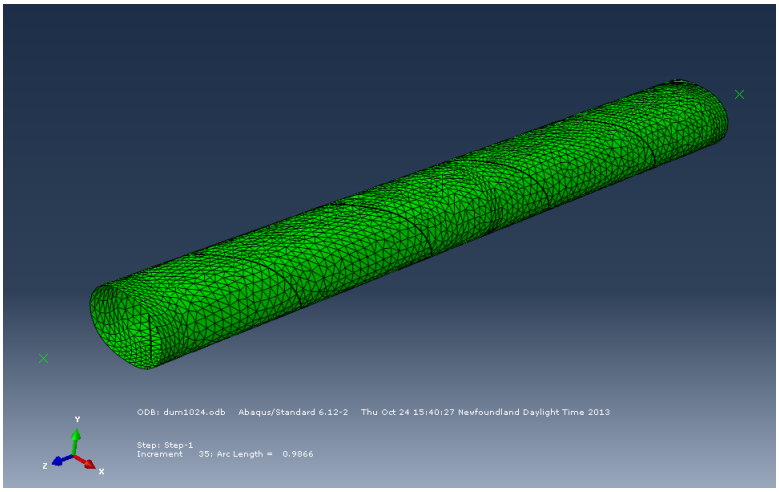
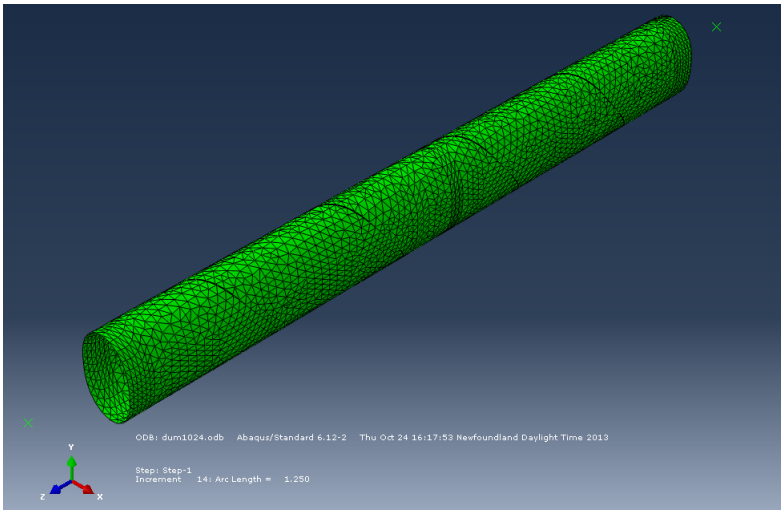


Figure 2.12: Three different shape of the pipe cross section.

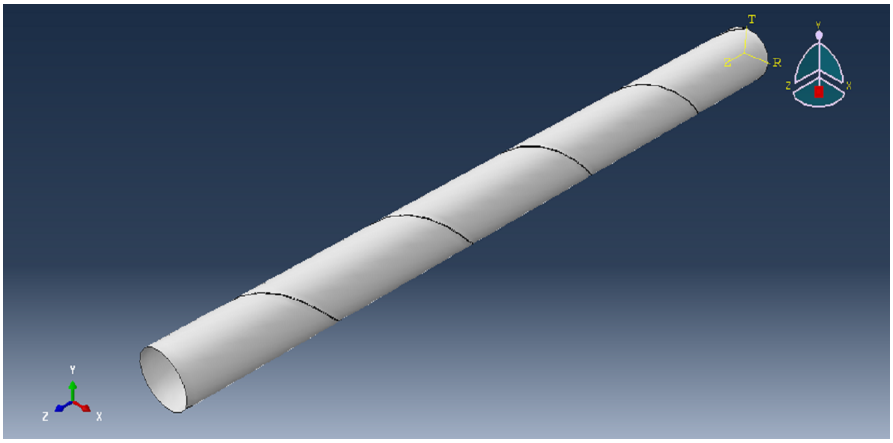


(a) Horizontally ellipse

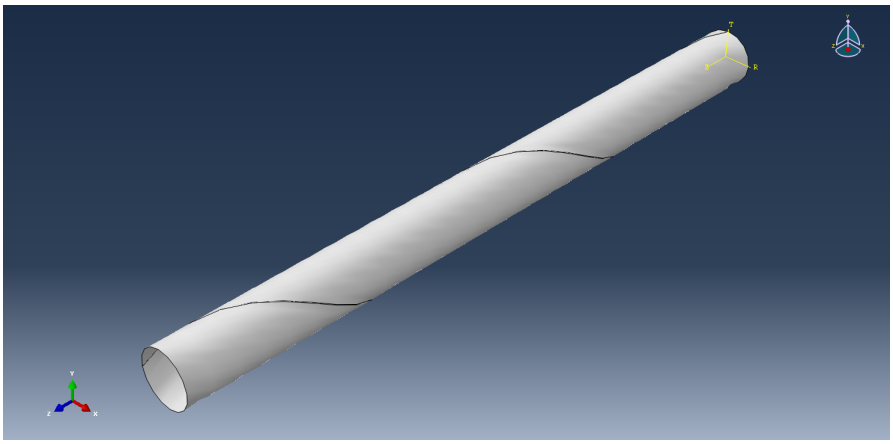


(b) Vertically ellipse

Figure 2.13: Elliptical pipes.



(a) pitch 40°



(b) pitch 60°

Figure 2.14: SWPs with different pitch.

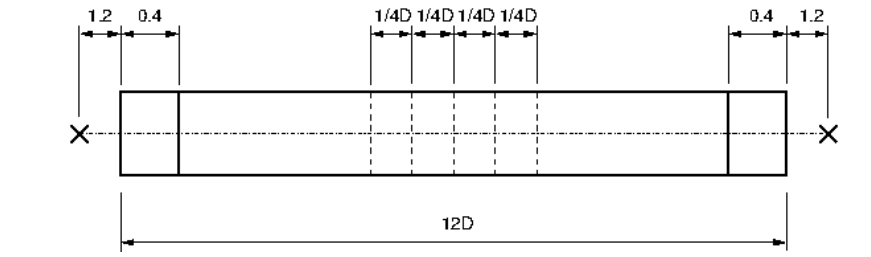
Chapter 3

Detail Model and Python Scripting

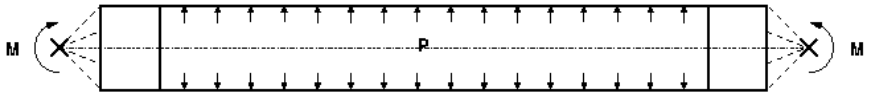
The model and detail Python scripting will be discussed in this chapter. Mainly, there are two models are built: one is the benchmark, a UOE pipe, it is called the BMP (Benchmark Pipe) model, and the other is the spiral welded pipe or the SWP model. The BMP model is built to imitate a model done by A Fatemi and S Kenny [6], let's call this model the 'F&K model'.

In this work, the units are not mentioned, because this work is emphasizing on the programming point of view only. The aim is to work on software with scripting; create a model, run it, and collect the output. The calibration dataset was from a proprietary study that has been reported in the open literature [6] that was calibrated and verified against a physical dataset. This proprietary information was used to calibrate the numerical modelling procedures in the current study in order to provide confidence for extending the algorithm to the local buckling response of spiral pipe.

We will simulate a pipe hold by $0.4m$ collars on both ends. The pipe is pressurized



(a) Dimensions



(b) Loads

Figure 3.1: Dimensions and loads of the pipe.

inside and bent by moments. Detail dimensions and properties[1] can be seen on Figure 3.1 and Table 3.1 respectively. This work is dealing with local buckling which is a nonlinear problem, therefore we also need plastic property of the pipe, as shown on Table 3.2.

Table 3.1: Pipe properties and loads.

Dimensions	
Length	11.0
Radius	0.44577
Thickness	0.01143
Properties	
Young modulus	2.05e11
Poisson ratio	0.3
Loads	
Internal pressure	1.1e7
Bending moment	1.4e7

There are three main scripting will be discussed on the following sections. Two scripting are about creating the pipes and one scripting is about reading the output file.

[1]The internal pressure is 11.3MPa.

Table 3.2: Plastic property.

	Yield Stress	Plastic Strain		Yield Stress	Plastic Strain		Yield Stress	Plastic Strain
1	430500000	0	29	504000000	0.000350725	57	577500000	0.002832422
2	433125000	1.22E-05	30	506625000	0.000365063	58	580125000	0.003272159
3	435750000	2.44E-05	31	509250000	0.000379881	59	582750000	0.003794406
4	438375000	3.66E-05	32	511875000	0.000395283	60	585375000	0.004414603
5	441000000	4.88E-05	33	514500000	0.000411395	61	588000000	0.005150989
6	443625000	6.10E-05	34	517125000	0.000428369	62	590625000	0.006025094
7	446250000	7.32E-05	35	519750000	0.000446389	63	593250000	0.007062316
8	448875000	8.54E-05	36	522375000	0.000465677	64	595875000	0.008292589
9	451500000	9.77E-05	37	525000000	0.000486501	65	598500000	0.009751174
10	454125000	0.000109883	38	527625000	0.000509182	66	601125000	0.011479569
11	456750000	0.000122115	39	530250000	0.000534109	67	603750000	0.013526587
12	459375000	0.000134357	40	532875000	0.000561747	68	606375000	0.015949594
13	462000000	0.000146610	41	535500000	0.000592655	69	609000000	0.018815971
14	464625000	0.000158878	42	538125000	0.000627506	70	611625000	0.022204800
15	467250000	0.000171164	43	540750000	0.000667103	71	614250000	0.026208832
16	469875000	0.000183472	44	543375000	0.000712412	72	616875000	0.030936781
17	472500000	0.000195807	45	546000000	0.000764585	73	619500000	0.036515981
18	475125000	0.000208176	46	548625000	0.000825002	74	622125000	0.043095472
19	477750000	0.000220587	47	551250000	0.000895311	75	624750000	0.050849590
20	480375000	0.000233050	48	553875000	0.000977479	76	627375000	0.059982121
21	483000000	0.000245576	49	556500000	0.001073852	77	630000000	0.070731124
22	485625000	0.000258181	50	559125000	0.001187224	78	632625000	0.083374509
23	488250000	0.000270882	51	561750000	0.001320926	79	635250000	0.098236502
24	490875000	0.000283700	52	564375000	0.001478915	80	637875000	0.115695123
25	493500000	0.000296663	53	567000000	0.001665899	81	640500000	0.136190830
26	496125000	0.000309803	54	569625000	0.001887468	82	643125000	0.160236510
27	498750000	0.000323160	55	572250000	0.002150257	83	645750000	0.188429017
28	501375000	0.000336781	56	574875000	0.002462134	84	648375000	0.221462487

3.1 Scripting for the BMP

The scripting can be found on Appendix A1. This section is discussing this script thoroughly. Abaqus works with modules; in this work, we are using eight modules.

Before entering the module, we do importing objects and defining parameters, as following example

```
from part import *
...

# Dimmesions required
pipeLength = 11.0
...

# Reference points
zp3 = pipeLength/2.
...
```

Now, we are entering to see on each module.

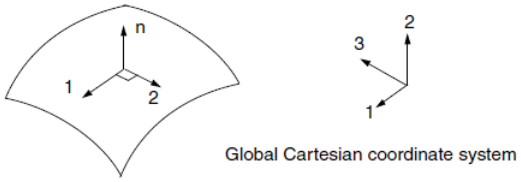
- `MODULE 1: PART`

The name of the model is defined and the pipe is created by extruding a circle a certain length. The circle is made by defining its center and its radius.

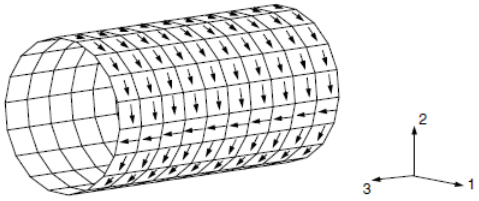
- `MODULE 2: PROPERTY`

Firstly, the material properties (density, Young modulus, Poisson ratio, plasticity) are defined. Then, we create a section and finally assign the property to that section. Before we go to the next section, we have to take care of the orientation.

As explained on [3], default 1-axis of local shell material coincides with X-axis of global axis (see Figure 3.2). This will give incorrect results if there are some elements



(a) Default local shell material directions.



(b) Default local material 1-direction in a cylinder.

Figure 3.2: The direction of stress and the local axis [3].

on the pipe that their plain are perpendicular to global X-axis. Therefore, we have to make sure that there is no element which its normal (n) will be parallel to the global X-axis. It can be done either by rotating the model so its main axis is parallel to global X-axis or by defining orientation on the model. In this work, we define orientation is cylindrical, where Z-axis is main axis of the pipe, X-axis relates to radial direction of the pipe, as shown on Figure 3.3. The different results between wrong and correct orientation of a model are shown on Figure 3.4. This figure shows pipes with different orientation. Both pipes have fixed diameter on the ends. The pipes are pressurized with the same load. However, the result of the hoop stress on the visualization shows different pattern. The wrong orientation gives incorrect contour on the middle, broken ring (see Figure 3.4a) and the correct one gives correct contour, continued ring (Figure 3.4b). The pipe should have uniform hoop stress along its circumference.

- `MODULE 3: ASSEMBLY`

For this model, we assembly nothing. However, on field the pipe is hold by collars on

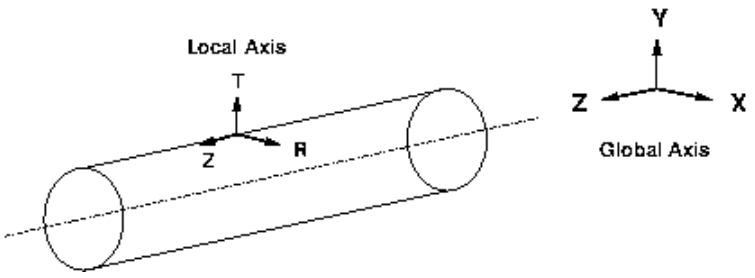


Figure 3.3: Defined local axis.

its ends, therefore we will create collars on the model too. Collar is built by defining position and the length of collar, then the partition is created, so there will be two parts, the pipe and the collar on the same surface.

• `MODULE 4: STEP`

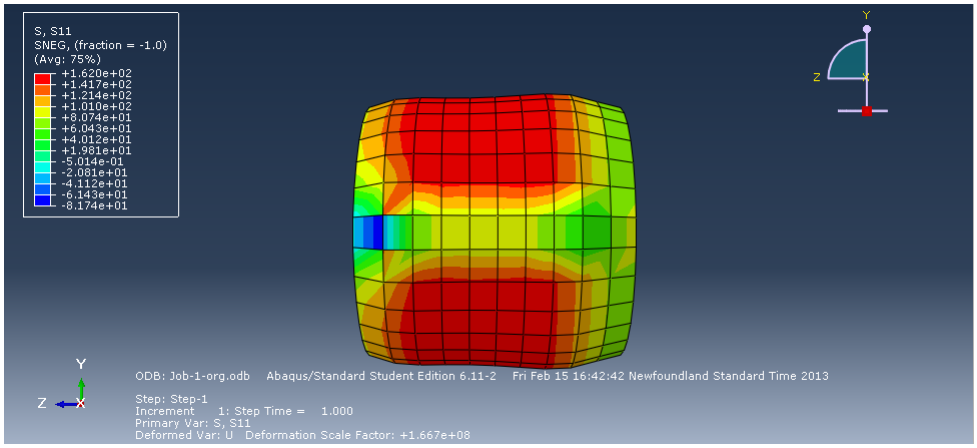
The model is for simulating a nonlinear problem, so we have to use Riks step method and nonlinear geometry function must be turned on. Not all output variables will be written on the output file, some variables must be requested. This module is the place to request the output, Field or History output, but because we request output on certain set, then we will define Field or History output after the set is ready. We will do it on another module.

• `MODULE 5: INTERACTION`

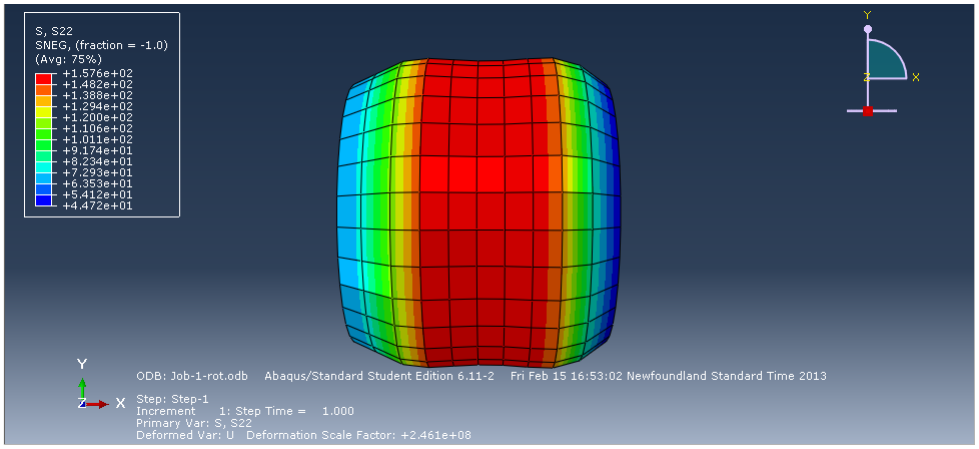
In this module, we are creating a coupling. The collar is stated as a rigid body that has a reference point. The bending moment and boundary conditions are acting on these reference points.

• `MODULE 6: LOAD`

In this module, loads and boundary conditions are defined. Care must be taken for



(a) Z-axis as cylindrical axis



(b) X-axis as cylindrical axis

Figure 3.4: The hoop stress from the wrong orientation (a) and the correct orientation (b).

applying pressure on the surface. The pressure load must hit all surfaces. For bending moment, direction and its axis are the important factors to mimic a real moment load. Together with the boundary conditions, the moments are applied on the reference points. There are two boundary conditions, one is set $u1 = 0.0, u2 = 0.0, u3 = 0.0$, and the other is $u1 = 0.0, u2 = 0.0, u3 \neq 0.0$.

- **MODULE 7: MESH**

There are three parts in this module, i.e. creating set for output, meshing the model, and requesting outputs on the defined set.

Because we want to collect output on a certain location, it is suggested to define that location as a set. To create a set, we have to create partition first. A set contains a group of nodes or may be just a single node. With Abaqus/CAE, a set can be created either before or after meshing. However, in scripting, we have to create a set before meshing. A set is formed by creating a partition, and then name it.

After we have the set, then can proceed meshing. Mesh size is 0.025, mesh shape is TRI (triangle) and not structured. In this step, we have to make sure that all surfaces are covered (remember, we just did partition).

Finally, after we define the location on sets, we can tell Abaqus to write outputs on that locations. This is done by requesting Field or History output on that set. In this work, we will collect the output from the Field.

- `MODULE 8: JOB`

This is the last module. Because we have to create an imperfection, then we can not run the script until getting the final result. The script ends in the job manager. Final result will be done by the input file, `BMP_model.inp`, executed in a command prompt.

3.2 Scripting for the SWP

Two models of SWP will be presented in this section. The first model is a model with uniform material, and the second model has two kinds of material. However, only the

first model will be discussed in detail in this report, and the second model is discussed briefly because this model uses the same principal as the first model, except having two materials.

3.2.1 The SWP model with uniform material

The scripting can be found on Appendix A2. This work emphasizes creating a script to model a SWP, at the moment the SWP has the same properties as the BMP. The model is created with the procedure as the previous model with some exceptional due to nature of making a spiral shape.

- `MODULE 1: PART`

The name of the model is defined and the pipe is created by extruding a circle a certain length. This is not just simple extruding, but also requires a pitch. This process will create a spiral shape.

The circle is made by two arcs. The arc needs three points; a center and two ends of the arc. However, we have to create these arcs **counter clockwise** or we will get wrong normal direction. Figure 3.5 shows the relation between direction of the arc and normal vector.

- `MODULE 2: PROPERTY`

Similar to the BMP, except there are two materials, the shell and the weld. In this work, it is assumed that the weld has the same properties as the shell. Therefore, we copy the property, from the first material to the second material. Of course, this model also has the same orientation as the previous model.

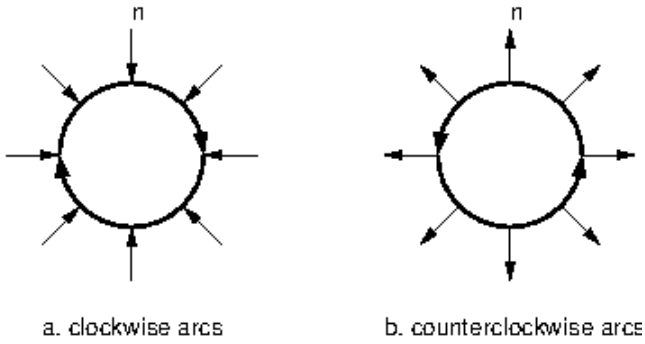


Figure 3.5: Relation between direction of the arc and normal vector.

• `MODULE 3: ASSEMBLY`

No further explanation, it is similar to BMP's assembly module.

• `MODULE 4: STEP`

No further explanation, it is similar to BMP's step module.

• `MODULE 5: INTERACTION`

No further explanation, it is similar to BMP's interaction module.

• `MODULE 6: LOAD`

Loads and boundary conditions module is similar to the BMP model, but extra care must be taken for applying pressure on all materials. The weld width is much smaller than the shell, some part may have no load if we miss to include them in the script. We have to check the `inp` file on the total number of element and number of element under the pressure load. See the following example:

...
...

```
*Element, type=S3
   1,    1,   17,  894
   2,   17, 2634,  894
   3,   17,    2, 2634
...
...
...
158590, 79450, 48465, 48346
*Nset, nset=Set-Shell
...
...
...
*Elset, elset=_Surf-press_SNEG, internal, instance=Pipe-1, generate
      1,  158590,       1
*Surface, type=ELEMENT, name=Surf-press
_Surf-press_SNEG, SNEG
...
...
```

It is found that total of the element 158590, and the element set ('Elset') is generated from 1 to 158590 with step 1, it means total 'Elset' = 158590 or equals to the total of the element. Therefore, it can be concluded that the pressure hits all elements. The same procedure is applied to check the pressure load on the surface of the BMP.

- `MODULE 7: MESH`

No further explanation, it is similar to BMP's mesh module.

- `MODULE 8: JOB`

No further explanation, it is similar to BMP's job module. Final result will be done by the input file, `SWP_model.inp`, executed in a command prompt.

3.2.2 The SWP model with two kinds of material

The second model uses two materials. The material for welding is assumed having yield stress 1.1x than the material of shell. The scripting for this test can be found on Appendix A2. After running the script, then we have to check the input file to make sure that the model contains two kinds of material. Editing this file is necessary, for example including the imperfection file, changing Riks' parameters if required, and checking the model again to make sure we run the model with correct parameters (orientation, material properties, load, boundary condition).

3.3 Scripting for a Model without Partition

As shown on Figure 2.9, we create an imaginary box on the model. This box will capture all points inside the box. Then we define a set of these collected points. Therefore, we have a set without doing partition on the model. However, it may not capture the desired points when we have free mesh or non structure element. The box is created by modul `getByBoundingBox(xmin,ymin,zmin,xmax,ymax,zmax)` in scripting.

The `getByBoundingBox(...)` modul requires 2 points on the corners that separated by space diagonal (remember, the box is 3D object). Following shows a part of the script that explains the bounding box method. As you can see, there are 3 parts; first is defining the points on the corners (`xmin,ymin,zmin,xmax,ymax,zmax`), then invoking the method and name the set (`Set-Top`), and finally using this set to collect the output (`COORD`).

. . .

```
. . .
# Creating sets with bounding box
allNodes = swpModel.rootAssembly.instances['Pipe-1'].nodes
xmin = -0.001
ymin = radius-0.001
zmin = zn1-0.01
xmax =  0.001
ymax = radius+0.001
zmax = zp10 + 0.01

topNodes = allNodes.getByBoundingBox(xmin,ymin,zmin,xmax,ymax,zmax)
swpModel.rootAssembly.Set(name='Set-Top', nodes=topNodes)

swpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-2', rebar=EXCLUDE, region=
    swpModel.rootAssembly.sets['Set-Top'], sectionPoints=
    DEFAULT, variables=('COORD',))
. . .
. . .
```

Before running the program, we must check again the input file, because it is possible we made mistakes in scripting. The software executes the input file only, therefore this file is very important, wrong input will produce wrong output, and of course it is a big waste. Following shows the important parts in the input file that needs to be checked (there are 5 parts);

```
. . .
. . .
*Elset, elset=_Surf-Pressure_SNEG, internal, generate
      1,  154832,       1
*Surface, type=ELEMENT, name=Surf-Pressure
_Surf-Pressure_SNEG, SNEG
. . .
. . .
*Orientation, name=Ori-1, system=CYLINDRICAL
          0.,           0.,           0.,           0.,           0.,
        1.
1, 0.
** Section: Section-Weld
```

```
*Shell Section, elset=Set-2-weld, material=Material-Weld,
orientation=Ori-1, offset=SPOS
0.01143, 9
** Section: Section-Shell
*Shell Section, elset=Set-1-shell, material=Material-Shell,
orientation=Ori-1, offset=SPOS
0.01143, 9
*End Instance
**
. . .
. . .
*End Assembly
**
**
*IMPERFECTION, INPUT=wave2s_imper.dat
**
** MATERIALS
**
. . .
. . .
**
*Step, name=Step-1, nlgeom=YES, inc=100
*Static, riks
 0.05,1.0,0.00000000000000005,0.025,
. . .
. . .
*Dsload
Pipe-1.Surf-Pressure, P, 1.13143e+07
```

The explanation about the important part is given below:

- **Part I**, make sure the surface include all nodes/elements (see keyword 'generate') and define cylindrical axis for the pipe (if the global axis is used, then make the x-axis as the main axis).
- **Part II**, all materials are defined and have correct orientation.
- **Part III**, include the imperfection file.
- **Part IV**, edit Riks parameter if the computation does not converge.

- **Part V**, make sure the load (pressure) hit all nodes/elements.

3.4 Scripting for Working on the Output

The scriptings can be found on Appendix A13.1 and A13.2. The first script (A13.1) is a short script to get y-ordinate of top points set, and the second script (A13.2) will get other parameters. In this work, we want to plot Bending Moment against Curvature and Bending Moment against Strain. Those are 2D plots, so it will be convenient to use a script to manage the output. For that purpose, these variables are required, i.e. `COORD` (coordinates), `UR` (rotation), `SE` (section strain), and `SF` (section force). The `COORD` and the `UR` belong to the node, but the `SE` and the `SF` are property of the element. Therefore, to get strain or force on a node, we have to get it from all elements that attach to that node.

To get those outputs, we have to request them on module `STEP` first, we request Field outputs on the certain sets that have been already defined. Remember, not all variables will be available on output file, by default Abaqus only reserve some. Therefore, requesting variables output before executing the job is very important.

3.4.1 Mathematical equations

Following is explaining to get the final outputs. The final outputs are κ (curvature), ϵ_{top} (strain on the compression side), and M_{mid} (bending moment on the middle of the pipe). We are going to use following equations to compute those variables and Figure 3.6 depicts the variables and the locations that we need.

We are going to compute the curvature. The coordinates, displacements, and rotations

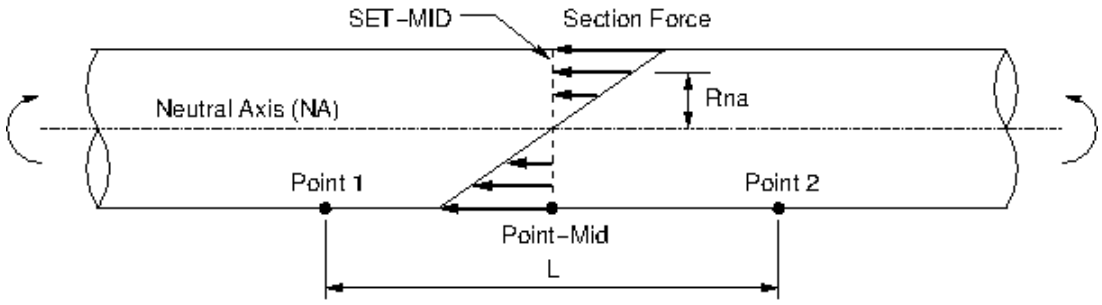


Figure 3.6: For computing the curvature, strain, and moment.

are define in global axis, therefore the variable is computed as

$$\kappa = \frac{\theta_2 - \theta_1}{dx} \tag{3.1}$$

where κ is the curvature, θ_1 is the `UR1` at Point 1, θ_2 is the `UR1` at Point 2. The dx is defined by

$$dx = COOR3_{point2} - COOR3_{point1}. \tag{3.2}$$

The strain on the compression side can be defined as

$$\epsilon_{top} = \kappa D - \overline{\epsilon}_{bot} \tag{3.3}$$

where D is the outside diameter of the pipe and $\overline{\epsilon}_{bot}$ is the average strain on the bottom (tension side) of the pipe. The $\overline{\epsilon}_{bot}$ is defined by

$$\overline{\epsilon}_{bot} = \frac{SE2_{point1} + SE2_{point2}}{2} \tag{3.4}$$

The strain is element property and is governed by local axis. On the orientation we have defined axis-1 is radially, therefore we get axis-2 is axially.

The moment is found by multiplying section force and its arm (distance between the

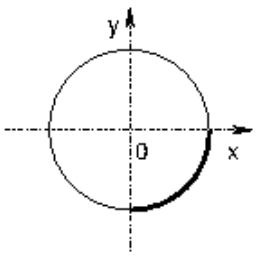


Figure 3.7: Quadrant IV nodes.

force and the neutral axis) and then adding these products to get the total bending moment. The neutral axis is the location where the stress is zero. Before computing the moment, the neutral axis must be found first. See Figure 3.7, the neutral axis will be on $y = 0$ if the load either moment or internal pressure only, but when the both loads are applied, the neutral axis will be on $y < 0$. Therefore, to find the neutral axis faster, only nodes on quadrant IV will be considered.

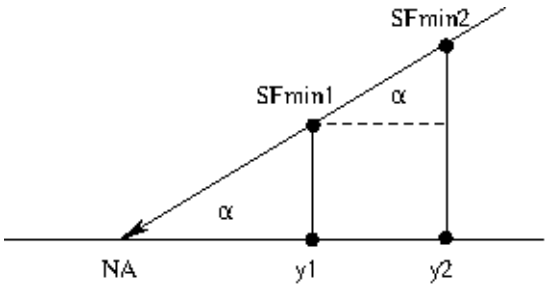


Figure 3.8: Finding the neutral axis.

This model uses two minimum positive section force to search where the location of the zero force is, as shown on Figure 3.8. Therefore the y neutral axis is defined as

$$y_{na} = y_1 - \frac{SF_{min1}}{\left[\frac{SF_{min2} - SF_{min1}}{y_2 - y_1}\right]} \tag{3.5}$$

After the neutral axis has been determined, then we can compute the total bending

moment as

$$M_{mid} = \sum_{i=1}^{i=TotalMidNode} SF_i \times y_{na_i} \tag{3.6}$$

3.4.2 The output database

Abaqus will write an output file in binary format, a file with extension `odb`, stands for *output database*. This file consists of two things; *model data* and *result data*, as shown on Figure 3.9. As explained in [4], the model data contain the parts and part instances that make up the root assembly; i.e. element types, nodal coordinates, set definitions, etc. The results data give the results of the analysis; i.e. displacements, strains, stresses, etc. The results data are requested from the step module, so it can be either field or history output data. In this work, the field output data are requested.

Outputs are kept in object data, to harvest these data we have to know the structure of the file. Figure 3.9 is important, like a map, it locates the desired output; for example you cannot get the 'fieldOutputs' if you open folder 'historyRegions'. Further detailed explanation about the output database can be found in [4].

3.4.3 Scripting algorithm

As mentioned before, the program to read the output is called `NodeField_Out.py`. This program needs the output file, `XXX_model.odb` and a list of requested nodes and their attached elements, this list are kept on the file `xxx_attelib.py`. Put those two files in the same folder as `NodeField_Out.py` first, then we can execute it.

Following is the explanation of the scripting algorithm;

- `OPENING`

This program is a general reader, it can read output from BMP model or SWP model,

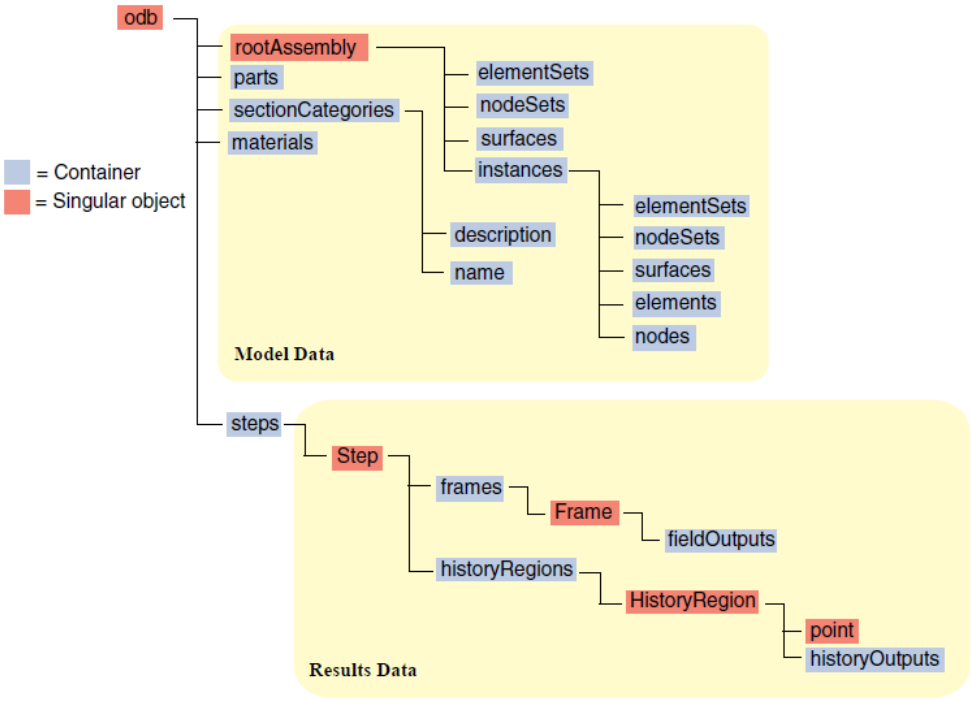


Figure 3.9: The output database [4].

therefore we have to select which model is.

- `SETTING VARIABLES`

In this section we are preparing required sets. One set for nodes on 0.25D from the middle length of the pipe, called 'SET-L25', and set of 0.50D from the middle length of the pipe, called 'SET-L50'.

- `COMPUTATION`

It starts with opening the file for writing the result, then do sorting and computing;

1. Sorting nodes, from top to bottom.

2. Computing the curvatures using Equation 3.1.

3. Reading SE, SF, and S. In this model, S will not be saved, this variable can be removed from the sript.

4. Extrapolating SE, SF, and S for defined nodes.

5. Finding the neutral axis (NA) using Equation 3.5.

6. Calculating the total moment (M_{mid}) using Equation 3.6.

• `PRINT FINAL RESULTS`

The result is kept on file called `xxx_nodefield.out`. It contains: step number, curvature-1, curvature-2, strain on the top, and total bending moment. It is shown on Appendix 14. Then using GNUPLOT program, as seen on `field_plot.plt` (Appendix 15), we plot the results.

Chapter 4

Results and Discussion

This chapter will show results from the test done in this work and result found in Abaqus's visualization and discuss them thoroughly. We start by showing the result from the numerical test first.

4.1 The Numerical Test

This test is to see the influence of partitions on the model. Partitions are imaginary division, it deals with the mesh only, and does not physically separate the model into smaller parts. These partitions are invoked because we need to have a set on a certain location for collecting the output. It is done before meshing. The other way creating a set is by bounding box module, it is done after meshing.

All pipes have the same dimension, material, and geometric imperfection, but different mesh distribution. We are going to see the first result, it is from the BMP model, the BMP model is 'cloning' of the F&K model. As seen on Figure 4.1a, there are a lot of points captured by the bounding box. For different mesh, we may get less or more points. Therefore, number of points captured is function of the mesh configuration

and the bounding box dimension.

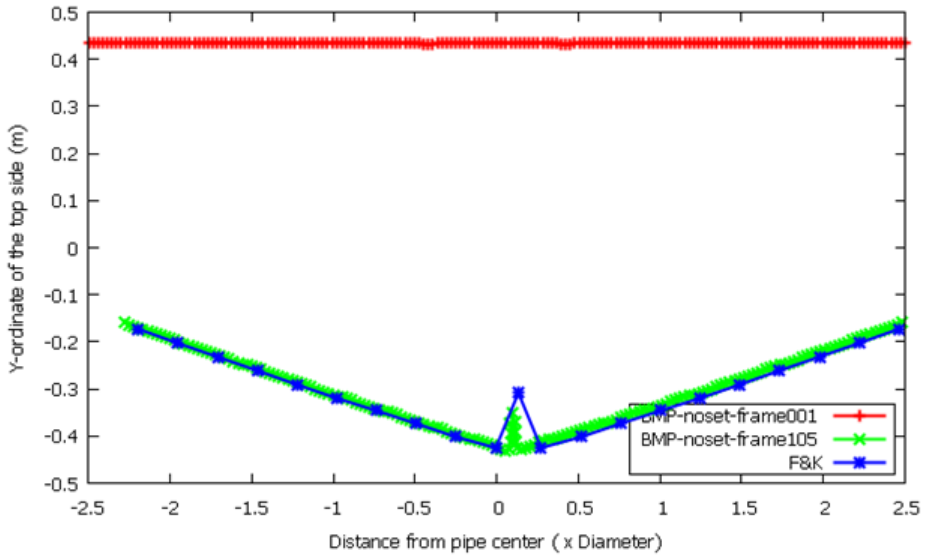
Figure 4.1b is showing a model with a set created by partitions. We get exact number and location of the required points. These points are not function of the mesh, but the mesh is forced to have those points (the mesh is function of the set). In this case we just collect points between 2.5D from the middle length of the pipe.

As shown on Figure 4.1, it is found, beside different number of point of the set, the required step to get the same deflection is also different. The model without partition requires less computation than the model with partition, by around 20% less.

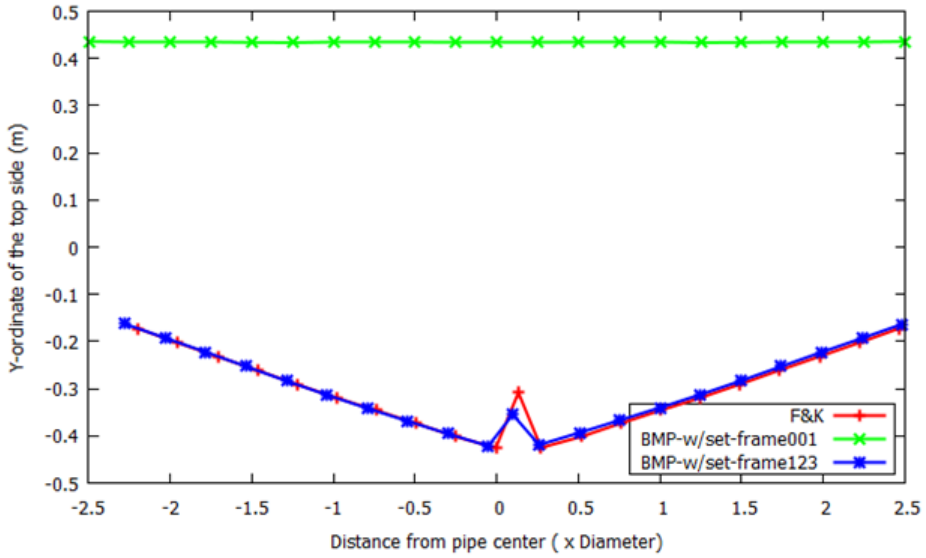
Figure 4.2 shows the 3D image of the both pipes. For the same deflection, we see slightly different shape of bump (see the circle on Figure 4.2a and 4.2b). The maximum von Mises stress is also slightly different, the pipe without partition can reach 6.523E+08, and the pipe with partition reach less (6.522E+08).

As we know, both models have, more or less, the same size of mesh, but the density or distribution is not the same. For the model without partition, we can say that the mesh will be distributed equally or uniform density, but the model with partition will force the model to have mesh on the partition, it will change the density or distribution of the mesh.

Figure 4.3 shows result from the spiral welded pipe. The elements of this pipe are rotated, so when we use a bounding box module, some desired points may not be in the box. As shown on Figure 4.3a, the points are not equally distributed in required domain and unfortunately we can not capture points on the middle of the pipe either.

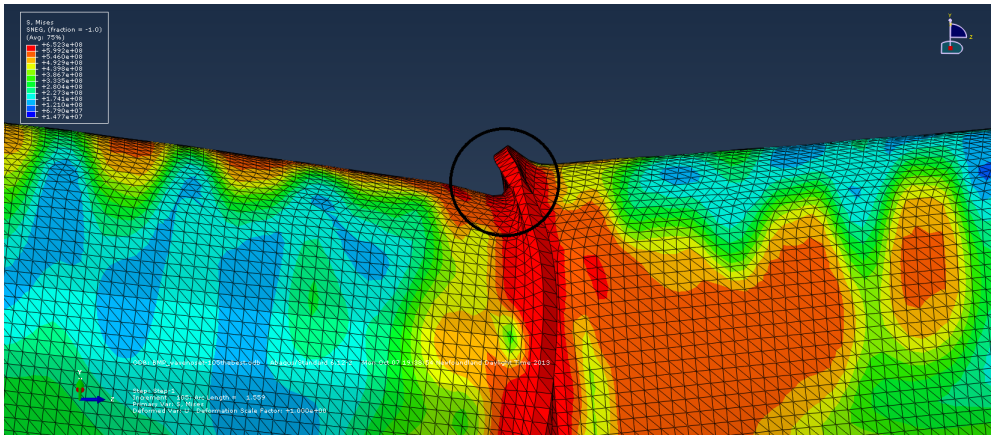


(a) Without partition

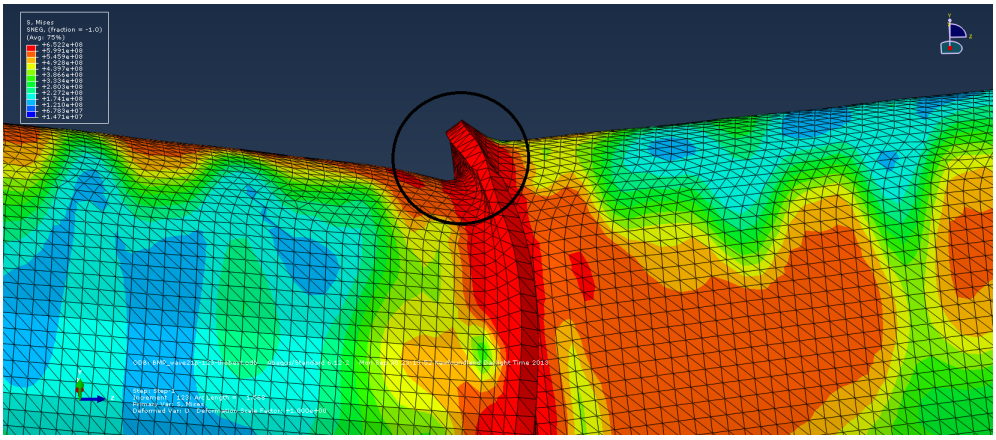


(b) With partition

Figure 4.1: The deflection of the BMP model compared to the F&K model.

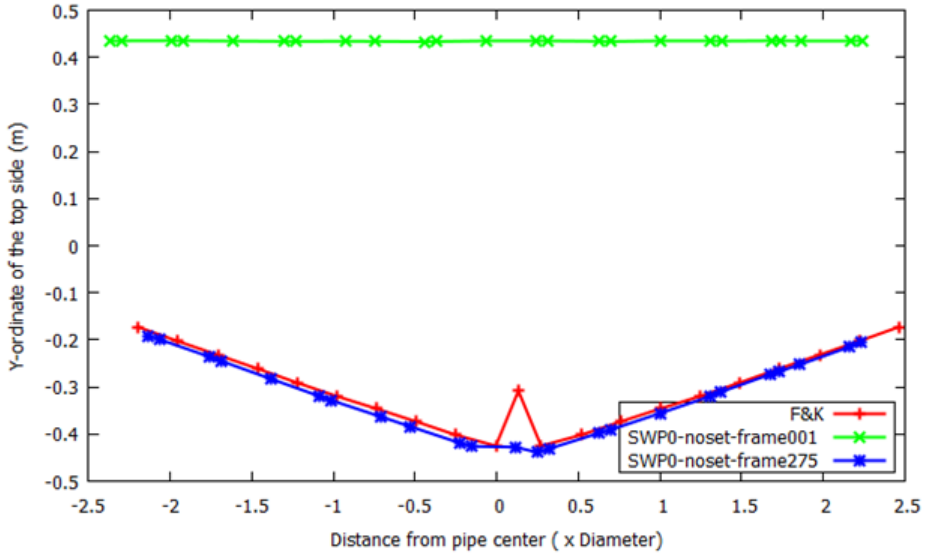


(a) Without partition

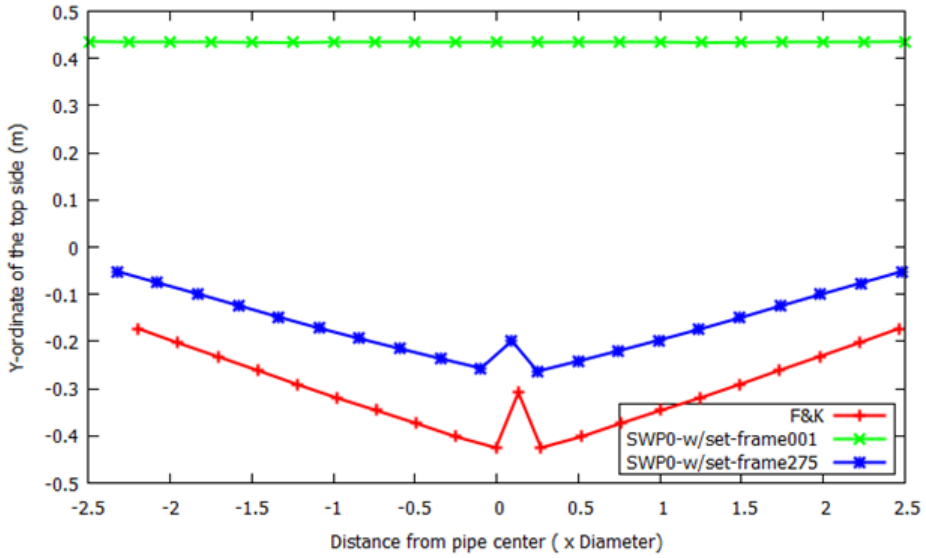


(b) With partition

Figure 4.2: The deflection of the BMP model (3D image).



(a) Without partition



(b) With partition

Figure 4.3: The deflection of the SWP model compared to the F&K model.

Now, the models are run with the same step, as depicted on Figure 4.3b, the model with a set can not reach the same deflection as the model without a set (Figure 4.3a). This test shows that a model with partitions is stiffer than a model without partitions, because mesh density or distribution is different.

4.2 The Material Test

The second test will see the influence of the material of the pipe. Two SWPs are modeled, one has uniform material and the other contains two material; material of welding is different from the shell as depicted on Figure 2.11. Both models are using the same geometric imperfection and having the same mesh density/distribution.

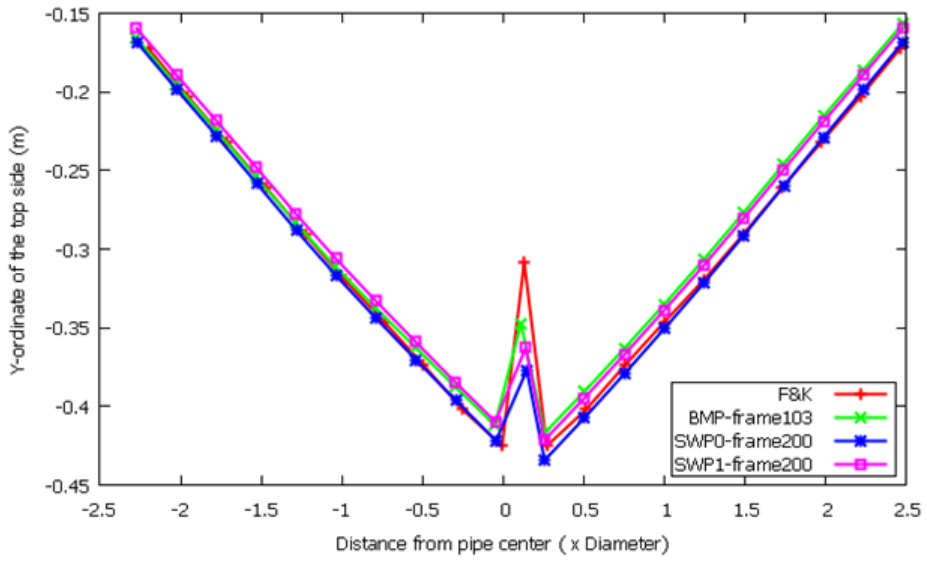
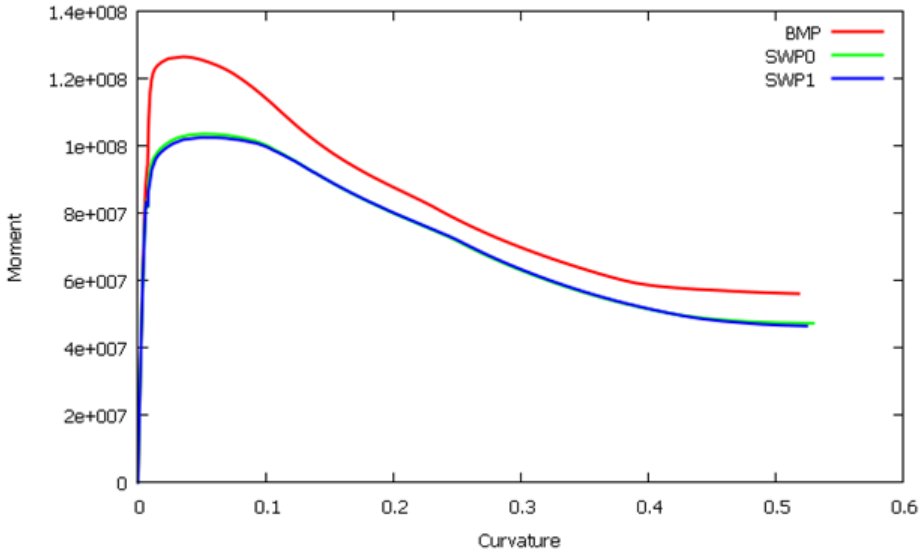
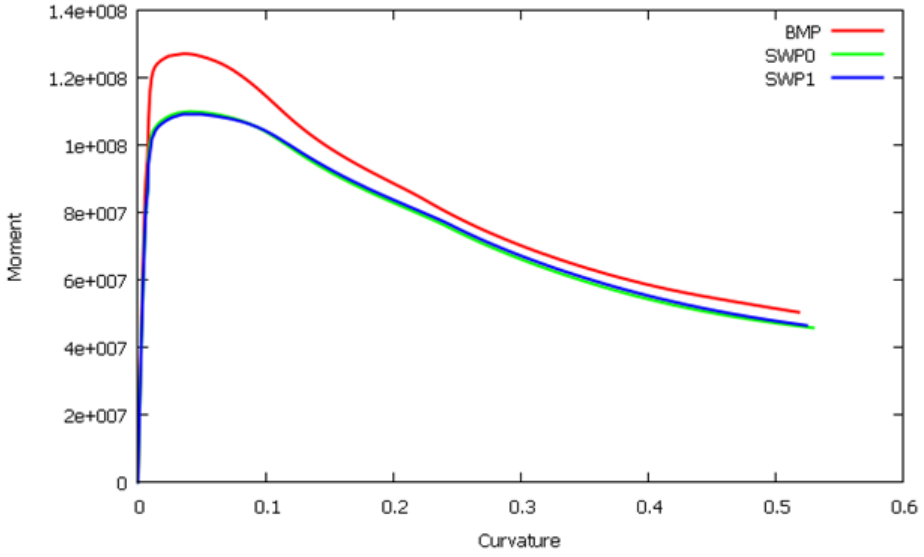


Figure 4.4: The deflection of the SWP models (uniform and non-uniform).

Figure 4.4 through 4.6 show that having different material (weld yield stress 1.1x than shell yield stress) give no much difference. Both SWPs produce similar plot.

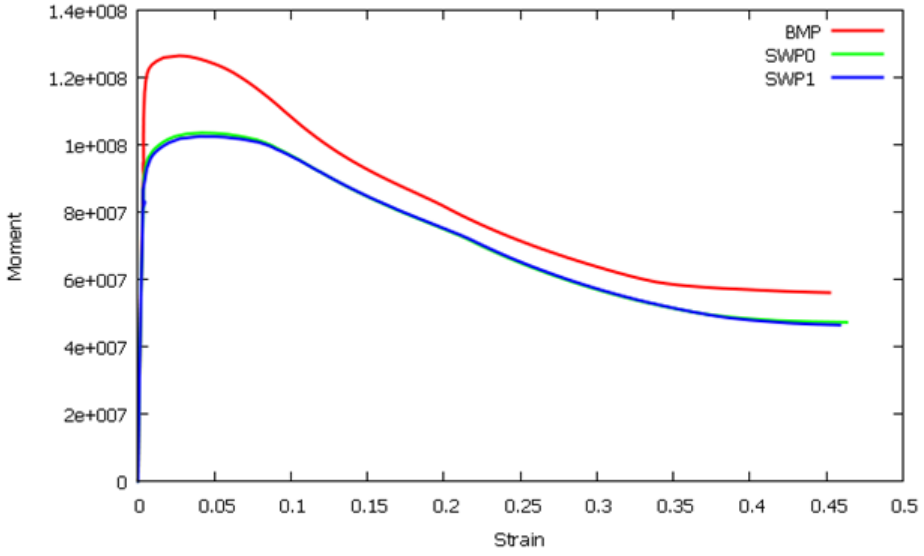


(a) at 0.50D

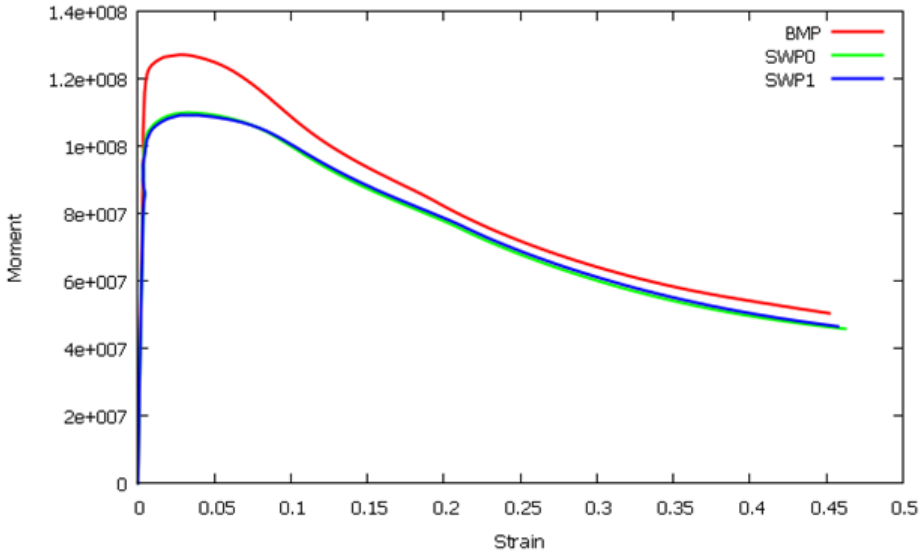


(b) at 0.25D

Figure 4.5: Moment versus curvature.



(a) at 0.50D



(b) at 0.25D

Figure 4.6: Moment versus strain.

They reach the same deflection with the same number of computation (twice than BMP model), see Figure 4.4. Both models also show the same trend on the moment-curvature and moment-strain graph, as shown on Figure 4.5 and Figure 4.6 respectively.

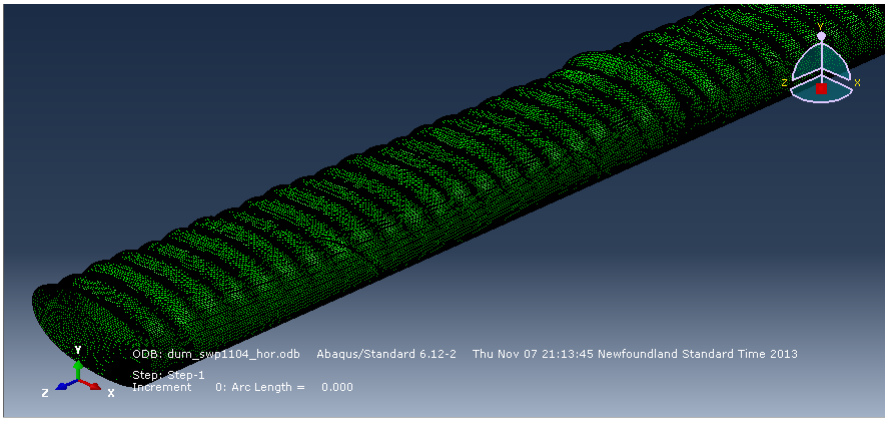
The numerical simulations provide evidence on the influence of the pipe pitch angle and distribution of material properties along the spiral weld seam influence the mechanical performance relative to the UOE pipe. The numerical simulations suggest the UOE pipe has slightly greater moment capacity with nominally similar strain (i.e. curvature) at peak moment.

4.3 The Ovality Test

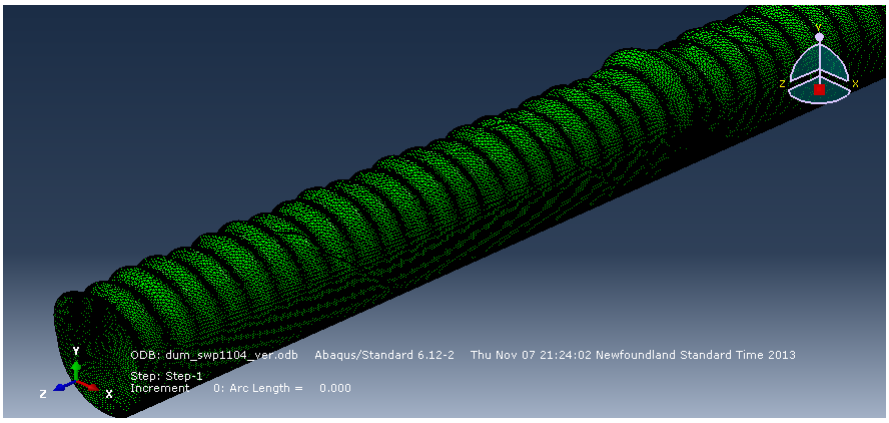
The third test is dealing with the cross section of the pipe. There are three cross section types; circle (a/b = 1), ellipse horizontally (a/b > 1), and ellipse vertically (a/b < 1). These three shapes have the same imperfection amplitude and wavelength, as depicted on Figure 4.7 (only ellipse pipes shown). For this test, we have 5 SWPs, a pipe with a/b=1.05, 1.01, 1.00, 0.99, and 0.95 respectively.

Figure 4.8 shows deflection and local buckling of pipes with a/b > 1. These figures show similar result from a circle pipe, the local buckling occurs on the middle of the pipe. Figure 4.9 shows loci on different section of the pipe; loci at 0.25D and 0.50D respectively, at beginning and ending of the computation. At beginning, although there is imperfection, but a small imperfection, all pipes seem to have a circle shape through the pipe length (see Figure 4.9a & 4.9c). Then, at the end of computation we can see that the section close to the middle length (0.25D) deforms more than the

section 0.50D (see Figure 4.9b & 4.9d).



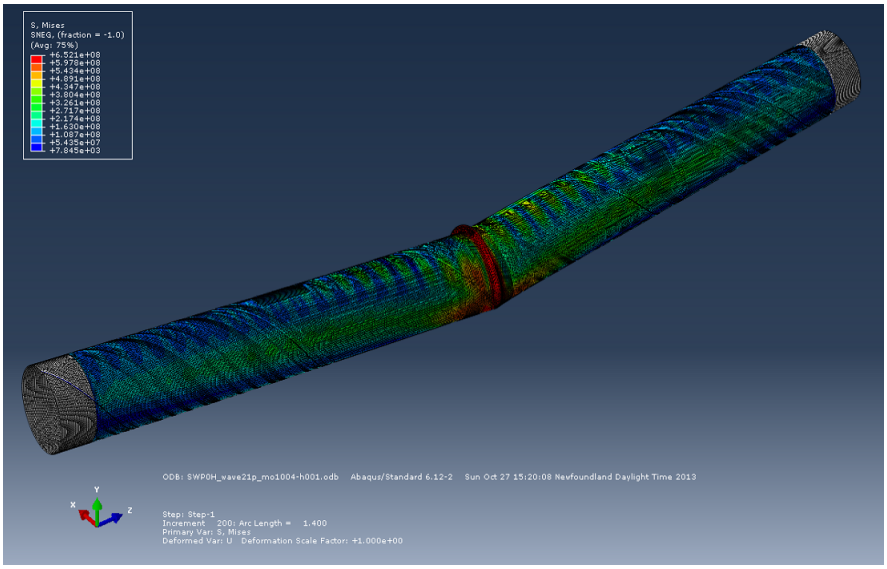
(a) Ellipse horizotally



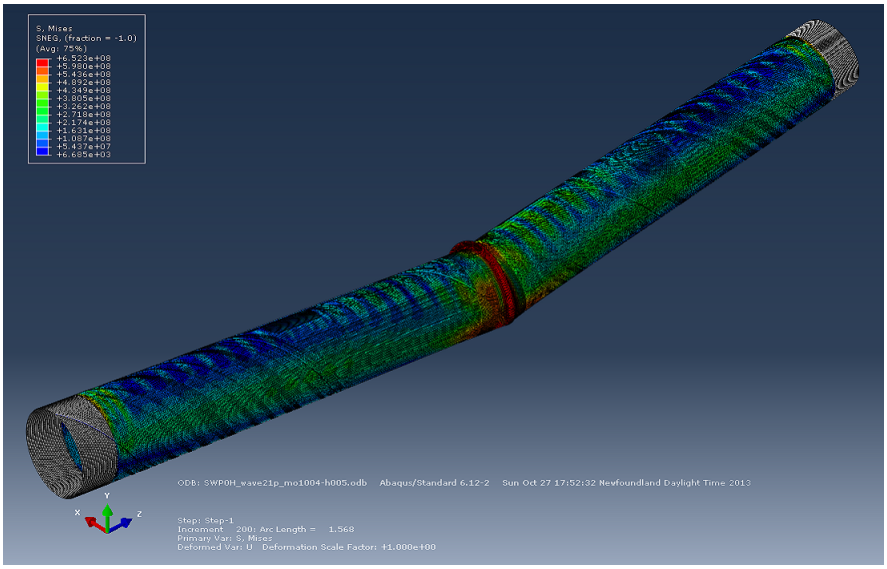
(b) Ellipse vertically

Figure 4.7: Ellipse shape and its imperfection (exaggerated).

Figure 4.11 shows detail of loci on different section of the pipe (loci at 0.25D and 0.50D). For a/b=0.99, the local buckling happens on the middle (see Figure 4.10a), and we can see that the section 0.25D deforms more than the section 0.50D (see Figure 4.11b). However, this pattern is not found on the pipe with a/b = 0.95, this pipe produces local buckling on the quarter (more or less) length of the pipe, as shown on Figure 4.10b, and we can not find much difference on those sections. Those sections give a circle shape on the end of computation.



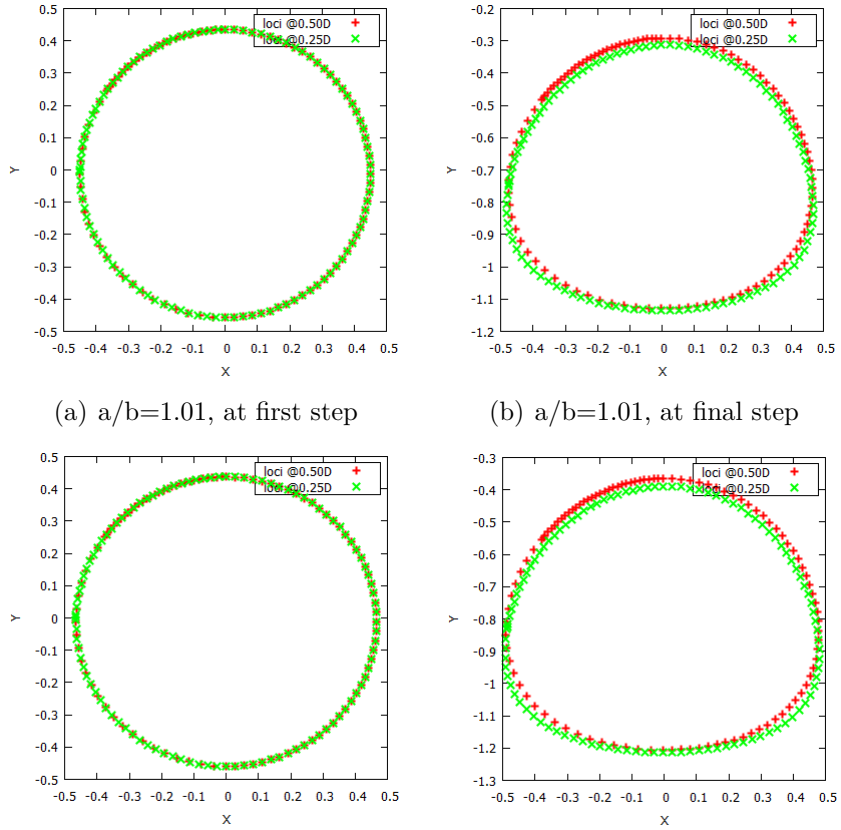
(a) a/b=1.01



(b) a/b=1.05

Figure 4.8: Isometric view of the deflection of the horizontal ellipse pipe.

Something interesting is found in Figure 4.12. For horizontally ellipse pipes, the 1% ovality (a/b = 1.01) seems to smooth the imperfection, the pipe becomes stronger, for the same number of computation, it bends less than the circle pipe. Increasing



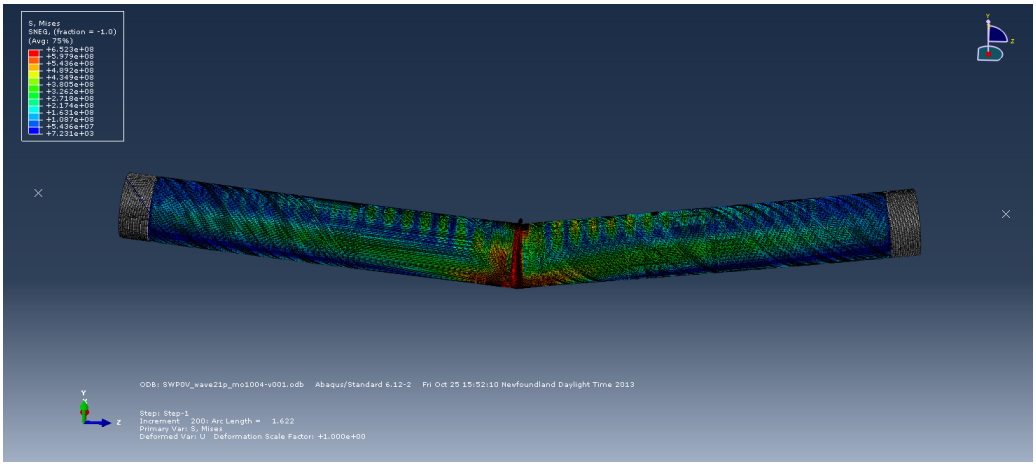
(a) a/b=1.01, at first step

(b) a/b=1.01, at final step

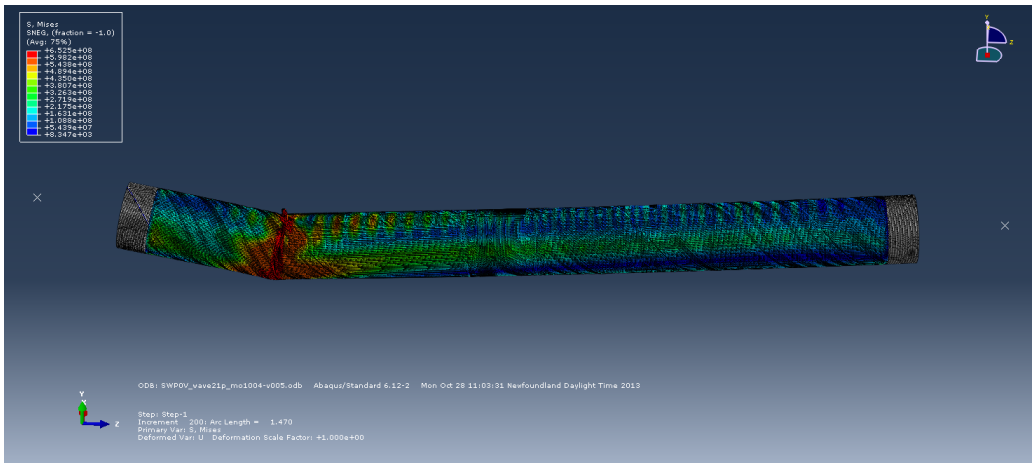
(c) a/b=1.05, at first step

(d) a/b=1.05, at final step

Figure 4.9: Horizontally ellipse shape pipe at different section.

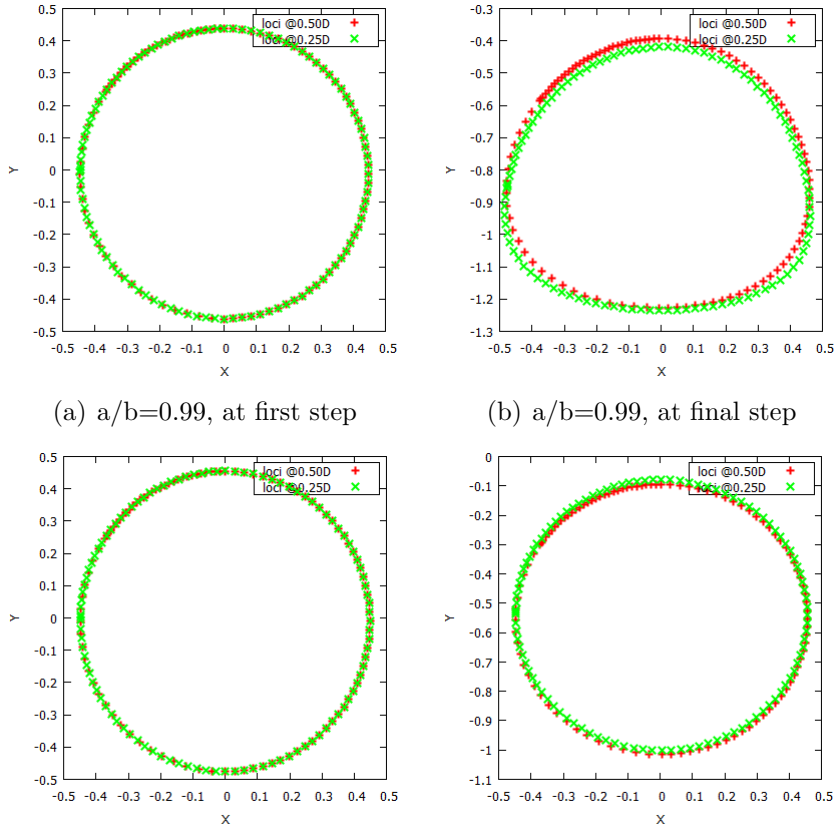


(a) a/b=0.99



(b) a/b=0.95

Figure 4.10: Deflection of the vertical ellipse pipe.



(a) a/b=0.99, at first step

(b) a/b=0.99, at final step

(c) a/b=0.95, at first step

(d) a/b=0.95, at final step

Figure 4.11: Vertically ellipse shape pipe at different section.

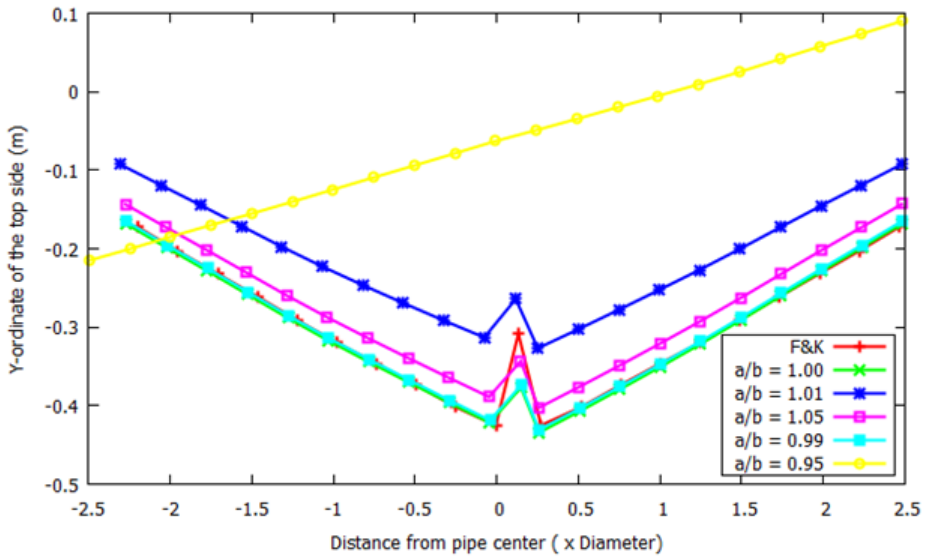


Figure 4.12: The deflection of the SWP models (circle and ellipses).

the ovality to 5% (a/b = 1.05), it makes the pipe weaker, it bends more than the 1% ovality pipe. For vertically ellipse pipes, the 1% ovality (a/b=0.99) influences little to the pipe, the pipe behaves like the circle pipe. However, the 5% ovality (a/b=0.95) does change the pipe behaviour a lots. The weakest point is not on the middle pipe anymore. The local buckling happens near the collar.

4.4 The Pitch Test

The fourth test is testing the pitch angle of the SWPs. There are two models, both models have two material (weld + shell) and have the same imperfection, only the pitch angle is different, one is 40° and the other is 60° respectively.

The results are shown on Figure 4.13 through 4.15. It is a surprising result, even though we have the same imperfection and material properties, the changing pitch angle can change the local buckling location. We expect the buckling will occur at

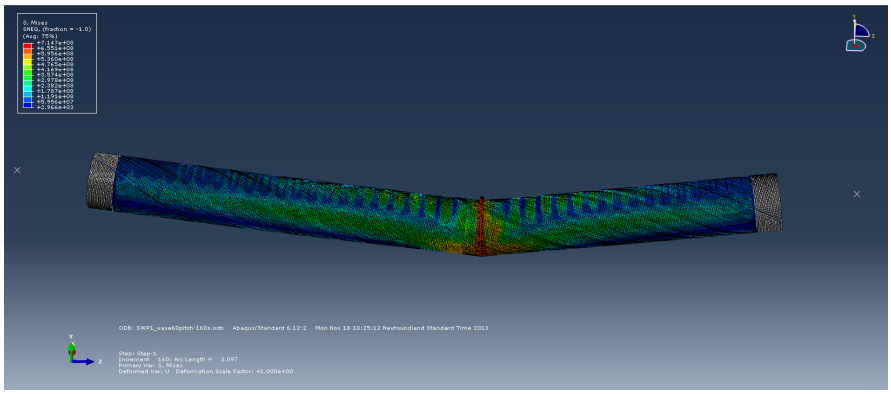
the middle, but apparently it appears somewhere 1D from the middle (see Figure 4.13). This graph also tells the 60° pitch requires less computation than the 40° pipe to reach the same deflection, around 50% less.

The moment-curvature and moment-strain graphs are computed from the set 0.25D and 0.50D from the middle of the pipe and it is assumed the local buckling will be on the middle, but apparently the local buckling of is not on the middle for the 60° pitch pipe, therefore the curve from the 60° pitch pipe is different from the 40° pitch pipe (see Figure 4.14 and see 4.15).

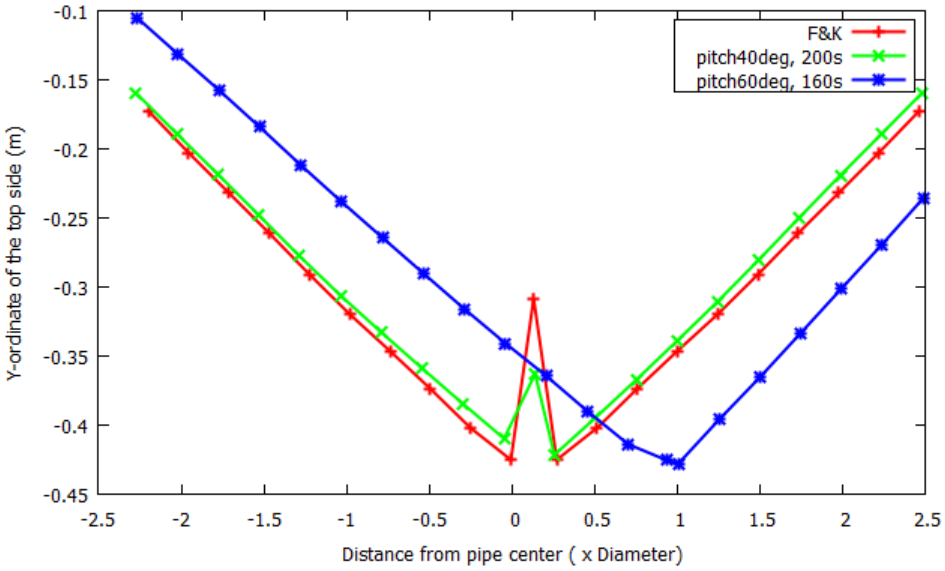
These results suggest the tests conducted on spiral pipe with segments lengths of 3D will be influenced by the boundary conditions and test specific parameters such as pipe pitch and material properties.

Consequently the qualification of spiral pipe for use in strain based design application requires physical testing to calibrate and verify the modelling procedures. The numerical simulations indicate factors that will influence the moment and strain capacity response include pipe pitch, pipe imperfections, and weld/base metal overmatch.

Other factors not addressed in this study that may influence the moment and strain capacity response of spiral pipe include material characteristics such as anisotropy for high strength pipe, Bauschinger effect, Luder's banding, and girth weld processes.

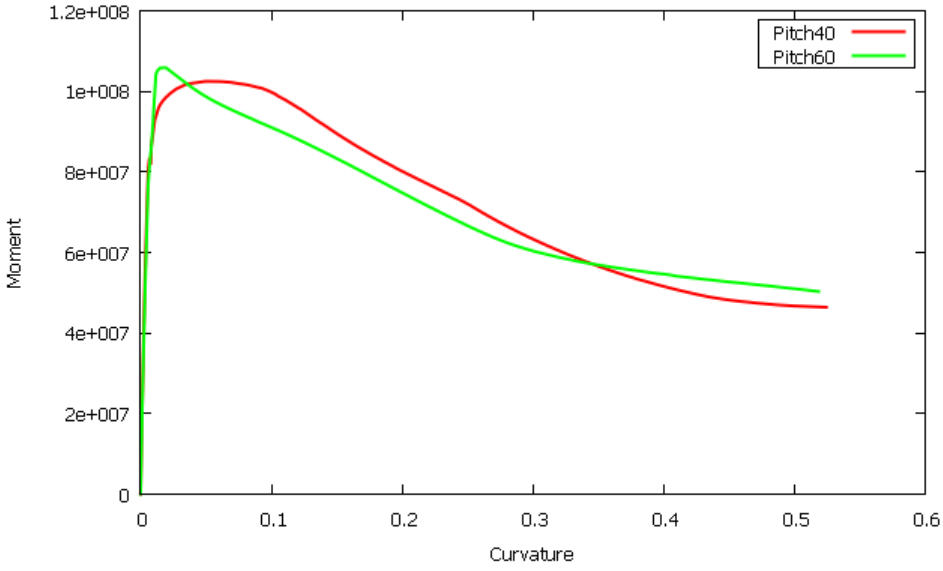


(a) Visual

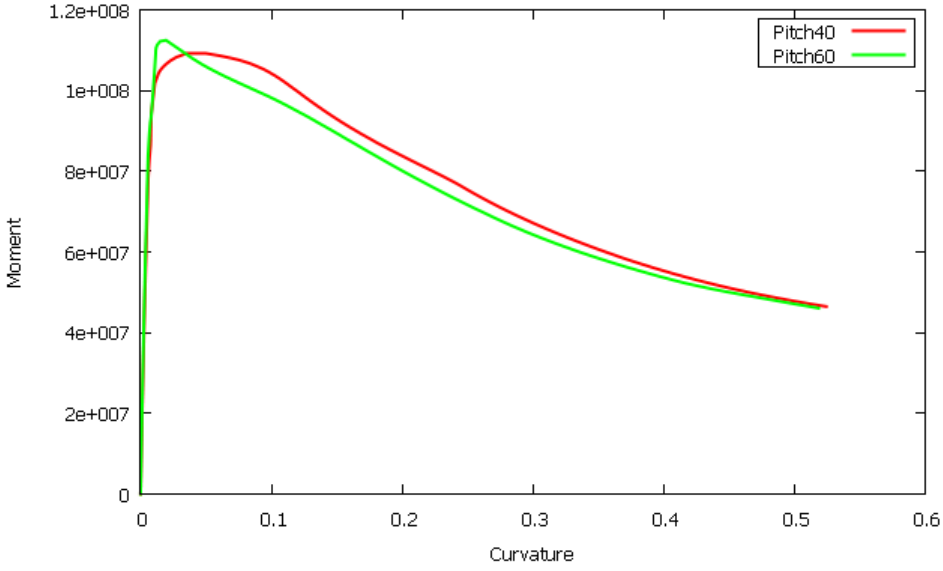


(b) 2D Plot

Figure 4.13: Deflection of the 60° pitch pipe.

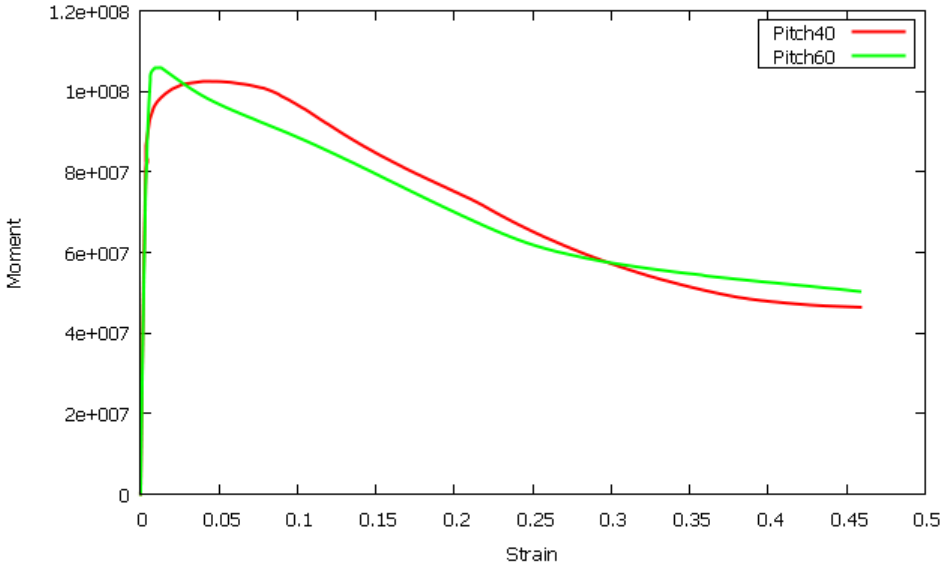


(a) at 0.50D

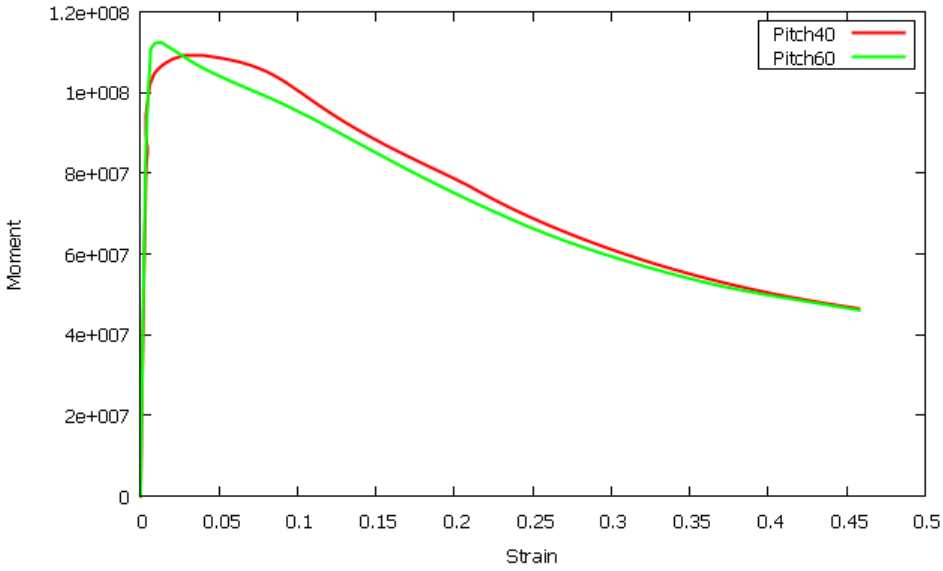


(b) at 0.25D

Figure 4.14: Moment versus curvature for different pitch pipes.



(a) at 0.50D



(b) at 0.25D

Figure 4.15: Moment versus strain for different pitch pipes.

4.5 The Internal Load Test

The fifth test is testing the internal load. First, we have a model with two kind of load applied which are bending moment and internal pressure. Second, a model with one kind of load applied which is bending moment only. The second model will use the `inp` file from the first model with small editing. The second model requires no pressure, so we will switch off the pressure by adding two asterisks (**) on unwanted lines or we delete these lines. Following shows part of the `inp` file on the load for each model:

- model with pressure

```
** LOADS
**
** Name: Moment-1   Type: Moment
*Cload
_PickedSet17, 4, 1.4e+07
** Name: Moment-2   Type: Moment
*Cload
_PickedSet18, 4, -1.4e+07
** Name: Pressure   Type: Pressure
*Dsload
Pipe-1.Surf-Pressure, P, 1.13143e+07
**
```

- model without pressure

```
** LOADS
**
** Name: Moment-1   Type: Moment
*Cload
_PickedSet17, 4, 1.4e+07
** Name: Moment-2   Type: Moment
*Cload
_PickedSet18, 4, -1.4e+07
**** Name: Pressure   Type: Pressure
```

```
***Dsload
**Pipe-1.Surf-Pressure, P, 1.13143e+07
**
```

However, when running the second model, it does not converge. Editing Riks' parameters does not solve this problem. It took days to figure out this problem. Apparently, omitting the pressure and only applying moment creates rotation about 3-direction. Using the same boundary conditions as the first model makes the computation diverge. Therefore we have to edit the boundary conditions for the second model. We have to nullify the DOF (degree of freedom) number 6. Following shows the difference of boundary conditions applied to each model:

- boundary conditions for the first model (model with pressure)

```
** BOUNDARY CONDITIONS
**
** Name: BC-1 Type: Displacement/Rotation
*Boundary
_PickedSet19, 1, 1
_PickedSet19, 2, 2
** Name: BC-2 Type: Displacement/Rotation
*Boundary
_PickedSet20, 1, 1
_PickedSet20, 2, 2
_PickedSet20, 3, 3
**
```

- boundary conditions for the second model (model without pressure)

```
** BOUNDARY CONDITIONS
**
** Name: BC-1 Type: Displacement/Rotation
*Boundary
_PickedSet19, 1, 1
_PickedSet19, 2, 2
** Name: BC-2 Type: Displacement/Rotation
```

```
*Boundary
_PickedSet20, 1, 1
_PickedSet20, 2, 2
_PickedSet20, 3, 3
_PickedSet20, 6, 6
**
```

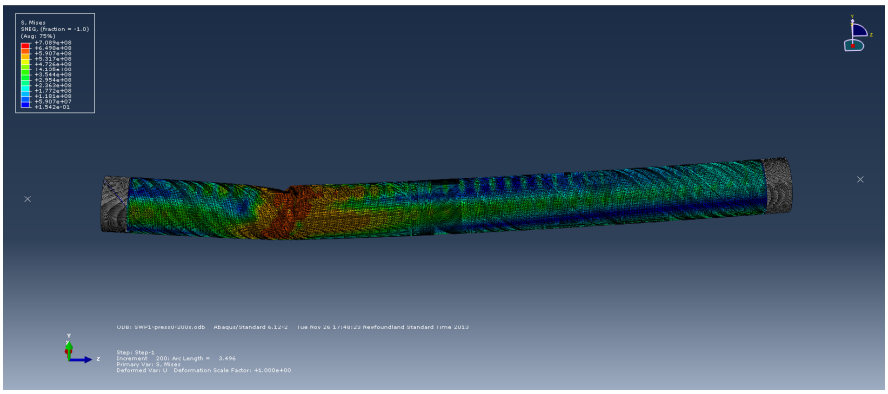
In this test we are going to see the deflection and the local buckling due to omitting the pressure. The imperfection is designed to have a local buckling on the middle length of the pipe if the internal pressure is involved. With the same imperfection, it is found that the local buckling of the pipe without pressure is not on the middle of the pipe. Figure 4.16a shows the local buckling somewhere around quarter length of the pipe and Figure 4.16b can not plot the local buckling because it is beyond the set.

As expected, without internal pressure, the local buckling is depicted by dents, as shown on Figure 4.17. It will be a different shape if internal pressure applied, a bump will appear to represent the local buckling as depicted on Figure 4.16b or 4.18.

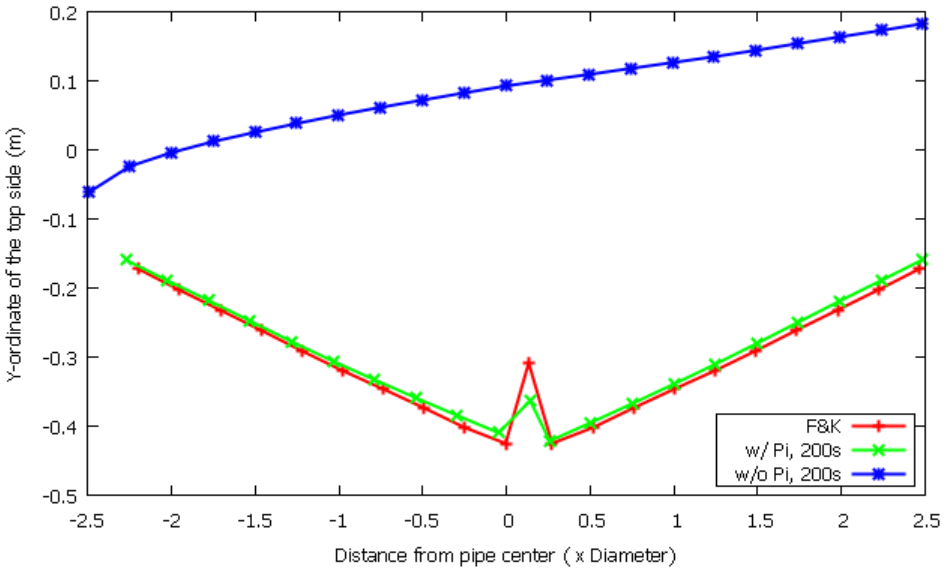
4.6 The Visualization

It is necessary to show the result of the visualization, because we found unexpected result plot on the visualization. As shown on Figure 4.18, the contour plot of the BMP model is good, but the contour plot of the SWP model is broken. Some regions that suppose to have some values give 'white' colour or empty.

Further investigation found that it was just a visual error. Figure 4.19 plots result of the beginning computation and the next step (Step-0 and Step-1). Let's see elements 30060 and 30102. The element 30060 has value for all steps, even though on the

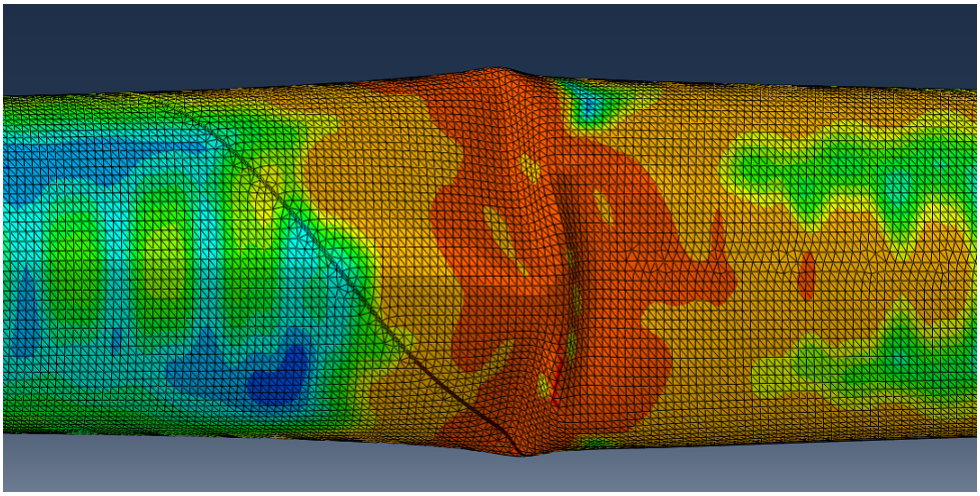


(a) Visual

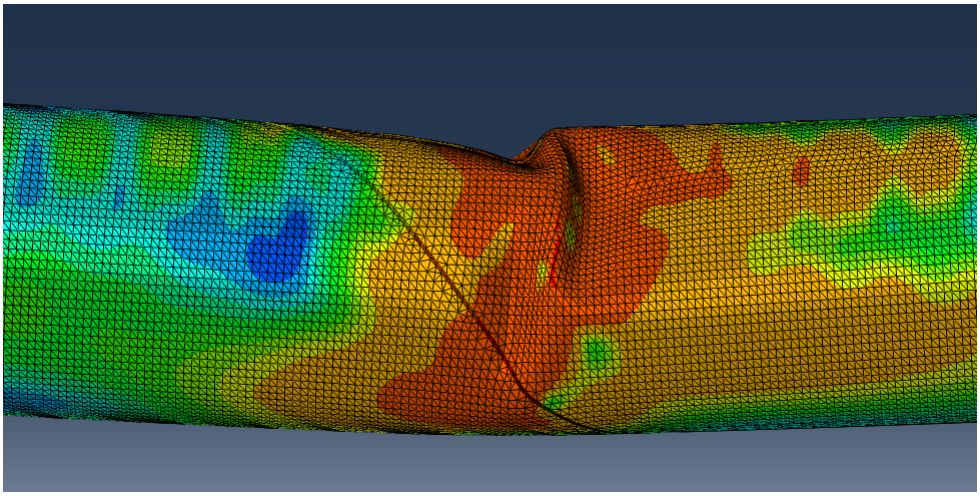


(b) 2D Plot

Figure 4.16: Deflection of the zero / non zero pressure pipe.

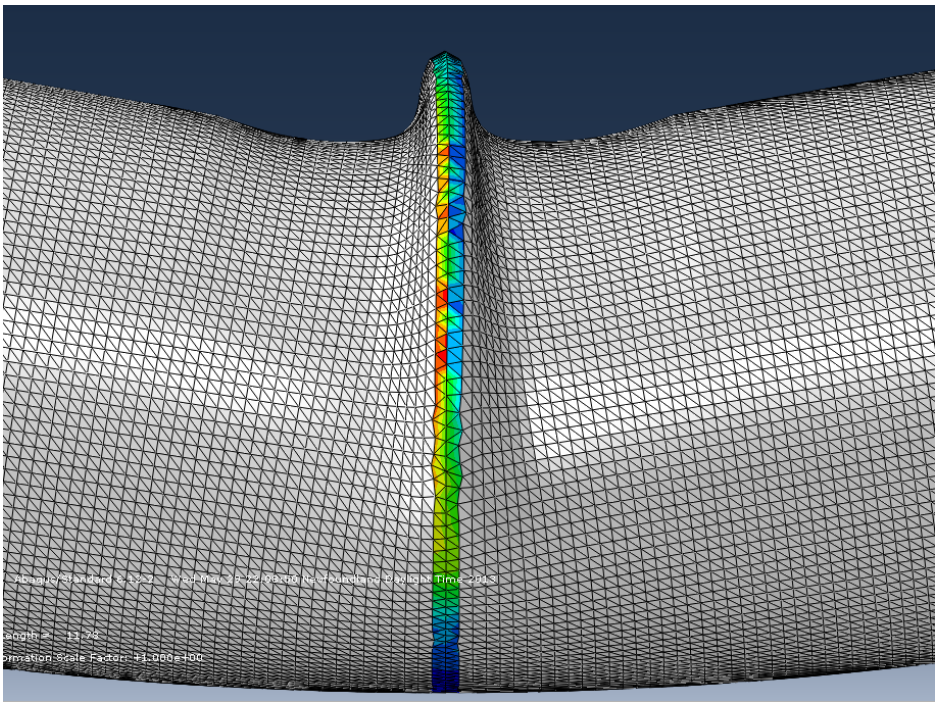


(a) top view

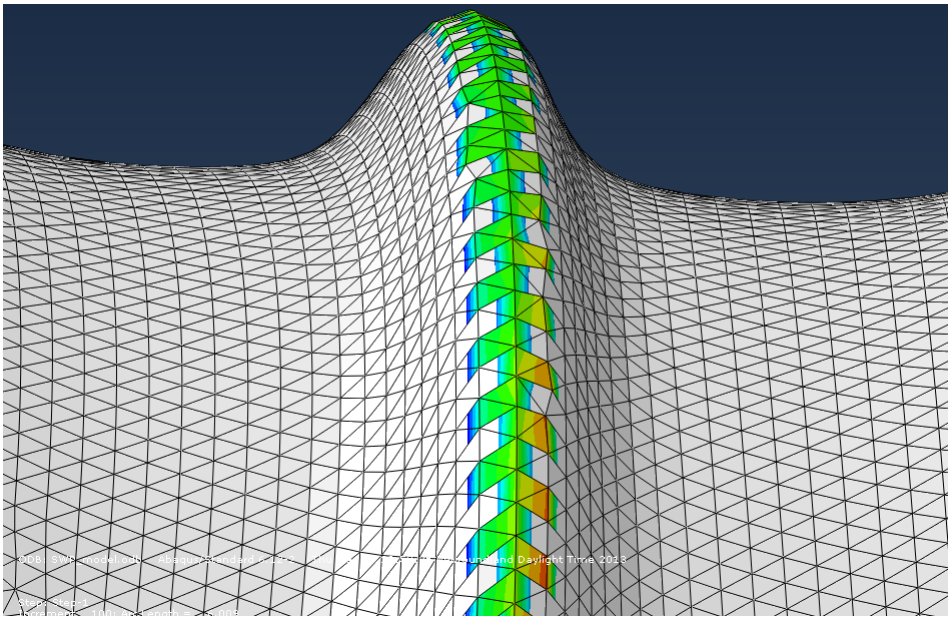


(b) side view

Figure 4.17: Local buckling is zoomed in (consistent with [12]).

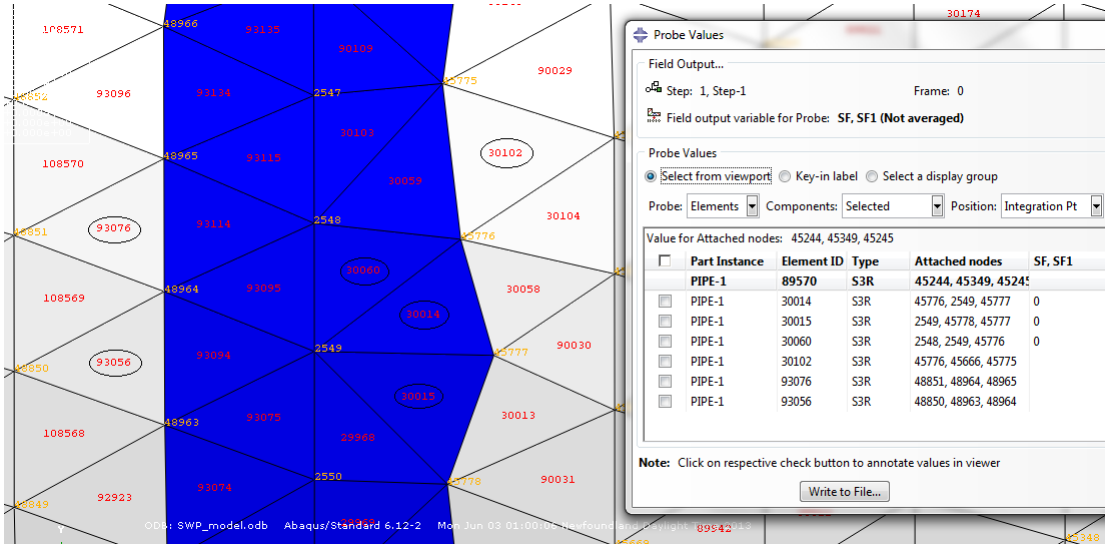


(a) BMP contour

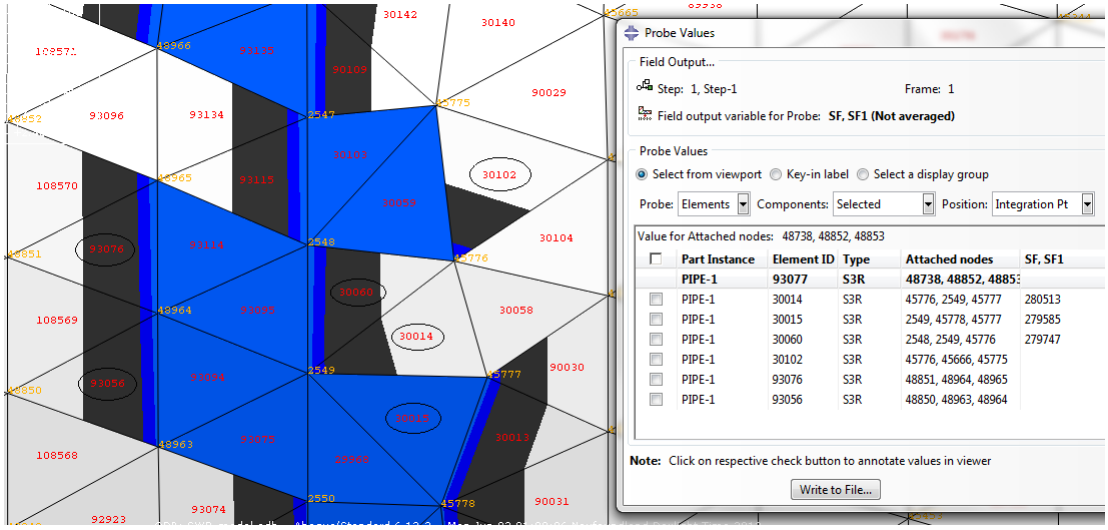


(b) SWP contour

Figure 4.18: Visualization of the model (consistent with [6]).

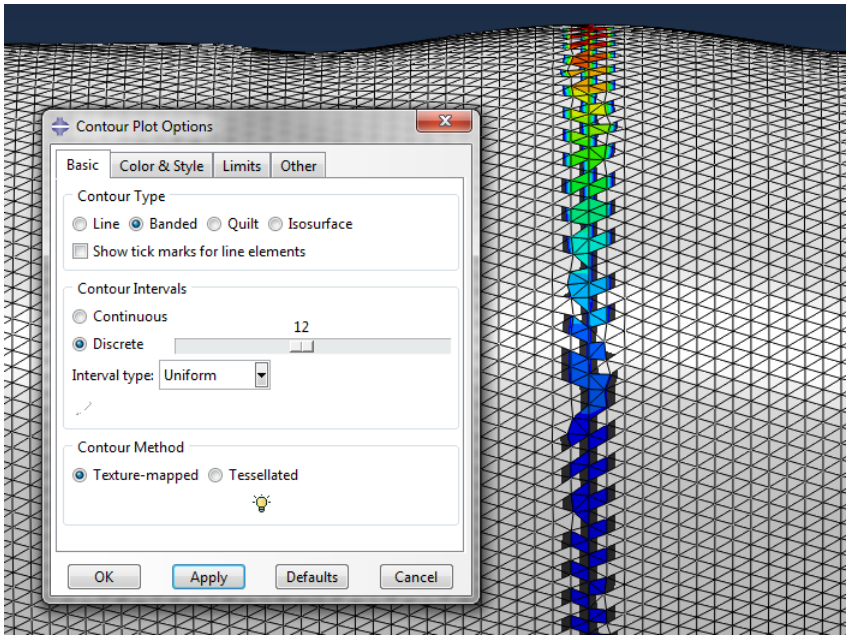


(a) Step-0

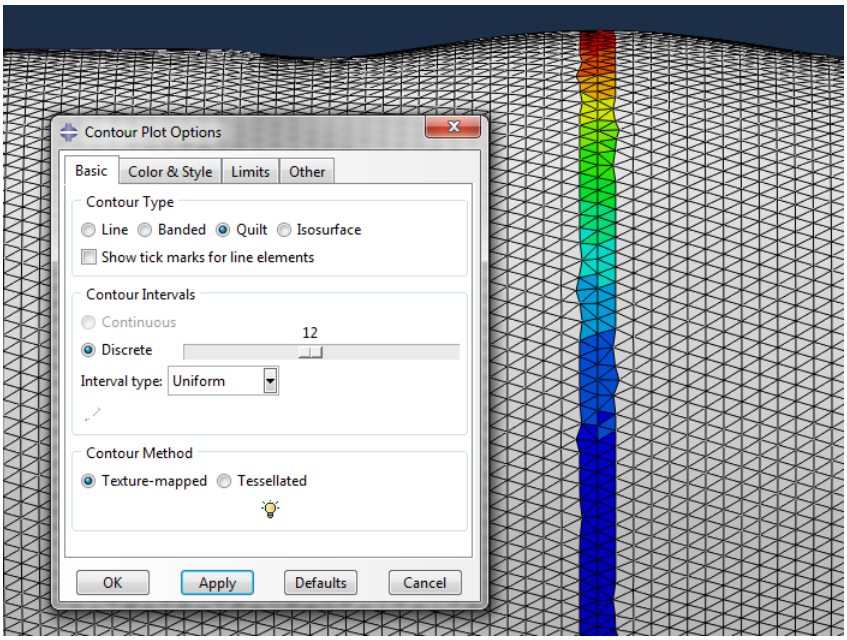


(b) Step-1

Figure 4.19: Contour plot on the SWP model.



(a) Banded



(b) Quilt

Figure 4.20: Contour type.

Step-1, its colour is not full. The element 30102 has no value for all step, but it has colour on the Step-1.

The other element of the SWP model shows the same behaviour, it will not give any value when we do not request it, but the element may have colour on the visualization. This odd is found on the SWP model only, the BMP gives good visualization. This problem can be eliminated by switching the default contour type, from 'banded' to 'quilt', as shown on Figure 4.20.

Chapter 5

Conclusions and Recommendations

This chapter consists of conclusions and recommendations. First, it will conclude results found in this work, result from the tests and also result from building the model. Then, it gives recommendations for future work to be done.

5.1 Conclusions

Based on those results and experiencing the process (from preparing the input, executing the model, and getting the result), it can be concluded:

- The scripting is very useful in Abaqus, it can build and edit a model fast, it can create an imperfection, and it works for all stages (pre- to post-processing).
- Working on a nonlinear problem, the following parameters -imperfection, mesh size, mesh distribution, and Riks step- must be considered.
- Creating set by partition to collect the output is good, but it changes mesh density and can influence the stiffness of the model numerically.
- The SWP with weld material 1.1x stronger than the shell gives similar result as

the SWP with uniform material.

- The SWP, due to extruding shape with a pitch, requires more computation step than then UOE pipe.
- The orientation of the cross section shape can change the imperfection influence, it is able to move the local buckling expected location.
- The pitch of the SWP can change the imperfection influence too. It is found, for higher pitch, the SWP tends to behave like UOE pipe.
- The shape of local buckling is function of the internal pressure. The internal pressure will create a bump, the zero pressure will create a dent.
- In this work, without internal pressure can make the model rotate on 3-direction, therefore we have to adjust the boundary condition.
- Some bugs are found in this Abaqus, one is in pre-processing and the other is in post-processing. The requested nodes missing some of their attached elements and in visualization the default contour plot, banded type, is distorted in extruded shape with pitch.

5.2 Recommendations

Based on the experience doing this work and accommodating real condition on the field, for future work, it is recommended:

- The Abaqus must fix the bugs, especially the missing element in the triangle mesh (the contour plot on twisted surface can be can be fixed later).

- Using the actual materials (for weld and base plate) that have been tested in lab to define the stress-strain relationships. This testing accounts for material orientation and effects (e.g. Bauschinger, Luder's) on strength behaviour.
- Using the actual imperfections that accommodates the real surface of the pipe, this imperfection includes cross section ovality, thickness, local dent or bump, and spiral or girth weld offset.
- Finally, large-scale tests on spiral welded pipe are required to calibrate and verify the results from the numerical simulation.

References

[1] ArcelorMittal Projects,*Spirally Welded Steel Pipes*, company brochure, 2010.

[2] Dassault Systèmes, *Getting Started with Abaqus*, 2010.

[3] Dassault Systèmes, *Abaqus User's Manual*, 2010.

[4] Dassault Systèmes, *Abaqus Scripting User's Manual*, 2010.

[5] Eltaher, A., Jafri, S., Jukes, P., and Heiberg, G., *Advanced Finite Element Analysis for Qualification of Spiral Welded Pipe for Offshore Application*, Proceedings of the Twenty-second (2012) International Offshore and Polar Engineering Conference, Rhodes, Greece, pp 303-310, June 17–22, 2012.

[6] Fatemi, A., Kenny, S., *Characterization of Initial Geometric Imperfections for Pipelines and Influence on Compressive Strain Capacity*, 7 pages, ISOPE, Rhodes, Greece, 2012,.

[7] Fatemi, A., Kenny, S., *Ovality of High-Strength Linepipes Subject to Combined Loads*, presented at the Arctic Technology Conference, Houston, Texas, USA, 11 pages, 3–5 December, 2012.

[8] Fonzo, A., Ferino, J., and Spinelli, C.M., *Pipeline Strain-Based Design Approach: FEM through Full-Scale Bending Tests*, Proceeding of the Twenty-second (2012) International Offshore and Polar Engineering, Rhodes, Greece, p. 497-503, 2012.

[9] Herynk, M.D., Kyriakides, S., Onoufriou, A., and Yun, H.D., *Effects of the UOE/UOC Pipe Manufacturing Processes on Pipe Collapse Pressure*, International Journal of Mechanical Sciences, 49, pp.533–553, 2007.

[10] The Iron Age, *Spiral Weld Tube Machine*, p 359-360, March 1, 1888.

[11] Knoop, F.M., Sommer, B., *Manufacturing and Use of Spiral Welded Pipes for High Pressure Service State of the Art*, Proceeding of IPC 2004 International Pipeline Conference, Calgary, Alberta, Canada, 9 pages, October 4-8, 2004.

[12] Kyriakides, S., Ju, G.T., *Bifurcation and Localization Instabilities in Cylindrical Shells under Bending - I. Experiments*, Int. J. Solids Structures Vol. 29, No. 9, pp. 1117-1142, 1992.

[13] Kyriakides, S., Ju, G.T., *Bifurcation and Localization Instabilities in Cylindrical Shells under Bending - II. Predictions*, Int. J. Solids Structures Vol. 29, No. 9, pp. 1143-1171, 1992.

[14] Langtangen, H.P., *Python Scripting for Computational Science*, Springer-Verlag Berlin Heidelberg, 2008.

[15] Limam, A., Lee, L.H., Corona, E., and Kyriakides, S., *Plastic Buckling and Collapse of Tubes Under Bending and Internal Pressure*, Proceedings of the ASME 27th International Conference on Offshore Mechanics and Artic Engineering OMAE2008, 9 pages, Estoril, Portugal, June 15-20, 2008.

[16] Limam, A. et al., *Inelastic Wrinkling and Collapse of Tubes Under Combined Bending and Internal Pressure*, International Journal of Mechanical Science, doi:10.1016/j.ijmesci.2009.06.008, 11 pages, 2009.

[17] Limam, A., Lee, L.H., and Kyriakides, S., *Effect of Local Imperfections of The Collapse of Tubes under Bending and Internal Pressure*, Proceeding of the 8th International Pipeline Conference IPC2010, 8 pages, Calgary, Alberta, Canada, Sept 27-Oct 1, 2010.

[18] Lutz, M., and Ascher, D., *Learning Python, Second Edition*, O'Reilly & Associates Inc., Sebastopol, California, USA, 2004.

[19] PRWEB, *Energy Sector to Drive Demand for Spiral Welded Pipes and Tubes, According to New Report by Global Industry Analysts*. Available at: `http://www.prweb.com/releases/spiral_welded_pipes_tubes/DSAW_HSAW_pipes/prweb10402550.htm`, retrieved February 2013.

[20] Puri, G., *Python Scripts for Abaqus, Learn by Example*. Available at: `http://www.abaquspython.com`, retrieved February 2013.

[21] Schmidt, W., *Why Spiral Weld Pipe?*, World Petroleum, vol 43., no 4., April 1972.

[22] TWI Report, *Spiral Welded Pipe for Oil & Gas: State-of-the-Art*, PR6161, August 2002.

[23] Van Es, S.H.J., Gresnigt, A.M., Kolstein, M.H., and Bijlaard, F.S.K., *Local Buckling of Spirally Welded Tubes - Analysis of Imperfections and Physical Testing*, Proceeding of the Twenty-third (2013) International Offshore and Polar Engineering, Anchorage, Alaska, USA, p. 248-259, 2013.

[24] Van Minnebruggen, K., De Waele, W., Denys, R., and Thibaux, P., *Strain Based Design Considerations for Spiral Welded Pipelines*, In: Hertelé S, editor, Sustainable Construction and Design, Ghent, Belgium: Ghent University, Laboratory Soete; p. 44–51, 2012.

[25] Wikibooks, *Python Programming*. Available at: `http://en.wikibooks.org/wiki/Python_Programming`, retrieved June 2013.

[26] Wikipedia, *Abaqus*. Available at: `http://en.wikipedia.org/wiki/Abaqus`, retrieved June 2013.

[27] Wikipedia, *Python (programming language)*. Available at: `http://en.wikipedia.org/wiki/Python_(programming_language)`, retrieved June 2013.

[28] Zimmerman, T., Timms, C., Xie, J., and Asante, J., *Buckling Resistance of Large Diameter Spiral Welded Linepipe*, Proceeding of IPC 2004 International Pipeline Conference, Calgary, Alberta, Canada, 9 pages, October 4-8, 2004.

Appendix

A1. BMP_model.py

```
#==========================================================================#
# BENCHMARK PIPE (BMP_model.py)
#
# This program simulates buckling on a pipe under an internal pressure
# and moment load.
#
# The SI system of units is used in this program.
#
# This program is create with the CAE's journal file and then
# it is modified by A Susilo.
#
# Created on: 01 Oct 2013
#==========================================================================#


from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *


# Dimmesions required
pipeLength = 11.0
radius = 0.44577
pipeODia = 2*radius
pipeThickness = 0.01143
collar = 0.4

# Reference points
zm = pipeLength/2.

zp1 = zm + (0.25*pipeODia)
zp2 = zm + (0.50*pipeODia)
zp3 = zm + (0.75*pipeODia)
```

```
zp4 = zm + (1.00*pipeODia)
zp5 = zm + (1.25*pipeODia)
zp6 = zm + (1.50*pipeODia)
zp7 = zm + (1.75*pipeODia)
zp8 = zm + (2.00*pipeODia)
zp9 = zm + (2.25*pipeODia)
zp10 = zm + (2.50*pipeODia)

zn10 = zm - (0.25*pipeODia)
zn9 = zm - (0.50*pipeODia)
zn8 = zm - (0.75*pipeODia)
zn7 = zm - (1.00*pipeODia)
zn6 = zm - (1.25*pipeODia)
zn5 = zm - (1.50*pipeODia)
zn4 = zm - (1.75*pipeODia)
zn3 = zm - (2.00*pipeODia)
zn2 = zm - (2.25*pipeODia)
zn1 = zm - (2.50*pipeODia)


ypos = radius
yneg = -radius

epsilon = 0.5*(zm-zn1)

# 1. MODULE: PART
#
# 1.1. Create the model
#==========================================================================#
#
# The model is called 'Benchmark Pipe'

mdb.models.changeKey(fromName='Model-1', toName='Benchmark Pipe')
bmpModel = mdb.models['Benchmark Pipe']

# 1.2. Create the part
#==========================================================================#
#
# The part is a circle. It is created by a circle uses 2 points (center, radius).
# Then this part is extruded to 'pipeLength' long.
#
# The part is called 'Pipe'


bmpModel.ConstrainedSketch(name='__profile__', sheetSize=200.0)
bmpModel.sketches['__profile__'].CircleByCenterPerimeter(center=(
```

```
    0.0, 0.0), point1=(0.0, radius))
bmpModel.Part(dimensionality=THREE_D, name='Pipe', type=
    DEFORMABLE_BODY)
bmpModel.parts['Pipe'].BaseShellExtrude(depth=pipeLength, sketch=
    bmpModel.sketches['__profile__'])

del bmpModel.sketches['__profile__']


# 2. MODULE: PROPERTY
#
# 2.1. Create material
#==========================================================================#
#
# There are one material in this model, material for the pipe

matP = 'Material-Pipe'
young = 2.05e11
poisson = 0.3

pipeMaterial = bmpModel.Material(name=matP)
pipeMaterial.Elastic(table=((young, poisson), ))
pipeMaterial.Plastic(table=(
(430500000,0),
(433125000,1.22E-05),
(435750000,2.44E-05),
(438375000,3.66034E-05),
(441000000,4.8808E-05),
(443625000,6.1015E-05),
(446250000,7.32251E-05),
(448875000,8.54391E-05),
(451500000,9.76579E-05),
(454125000,0.000109883),
(456750000,0.000122115),
(459375000,0.000134357),
(462000000,0.00014661),
(464625000,0.000158878),
(467250000,0.000171164),
(469875000,0.000183472),
(472500000,0.000195807),
(475125000,0.000208176),
(477750000,0.000220587),
(480375000,0.00023305),
(483000000,0.000245576),
(485625000,0.000258181),
(488250000,0.000270882),
```

```
(490875000,0.0002837),
(493500000,0.000296663),
(496125000,0.000309803),
(498750000,0.00032316),
(501375000,0.000336781),
(504000000,0.000350725),
(506625000,0.000365063),
(509250000,0.000379881),
(511875000,0.000395283),
(514500000,0.000411395),
(517125000,0.000428369),
(519750000,0.000446389),
(522375000,0.000465677),
(525000000,0.000486501),
(527625000,0.000509182),
(530250000,0.000534109),
(532875000,0.000561747),
(535500000,0.000592655),
(538125000,0.000627506),
(540750000,0.000667103),
(543375000,0.000712412),
(546000000,0.000764585),
(548625000,0.000825002),
(551250000,0.000895311),
(553875000,0.000977479),
(556500000,0.001073852),
(559125000,0.001187224),
(561750000,0.001320926),
(564375000,0.001478915),
(567000000,0.001665899),
(569625000,0.001887468),
(572250000,0.002150257),
(574875000,0.002462134),
(577500000,0.002832422),
(580125000,0.003272159),
(582750000,0.003794406),
(585375000,0.004414603),
(588000000,0.005150989),
(590625000,0.006025094),
(593250000,0.007062316),
(595875000,0.008292589),
(598500000,0.009751174),
(601125000,0.011479569),
(603750000,0.013526587),
(606375000,0.015949594),
(609000000,0.018815971),
```

```
(611625000,0.0222048),
(614250000,0.026208832),
(616875000,0.030936781),
(619500000,0.036515981),
(622125000,0.043095472),
(624750000,0.05084959),
(627375000,0.059982121),
(630000000,0.070731124),
(632625000,0.083374509),
(635250000,0.098236502),
(637875000,0.115695123),
(640500000,0.13619083),
(643125000,0.16023651),
(645750000,0.188429017),
(648375000,0.221462487)))
pipeMaterial.plastic.Potential(table=
    ((0.981, 1.0, 1.0, 1.0, 1.0, 1.0), ))

# 2.2. Create section
#======================================================================#
#
# Only one section, the pipe

bmpModel.HomogeneousShellSection(idealization=NO_IDEALIZATION,
    integrationRule=SIMPSON, material=matP, name='Section-Pipe',
    numIntPts=9, poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=
    GRADIENT, thickness=pipeThickness, thicknessField='', thicknessModulus=None,
    thicknessType=UNIFORM, useDensity=OFF)

# 2.3. Assign section
#======================================================================#

bmpModel.parts['Pipe'].Set(faces=
    bmpModel.parts['Pipe'].faces.findAt(((0.0,
    radius, pipeLength/2.0),  )), name='Set-Section')

bmpModel.parts['Pipe'].SectionAssignment(offset=0.0, offsetField=
    '', offsetType=TOP_SURFACE, region=
    bmpModel.parts['Pipe'].sets['Set-Section'], sectionName=
    'Section-Pipe', thicknessAssignment=FROM_SECTION)

# NOTE:
# For a model with the conventional shell element,
# it is suggested that the X-axis is the axial axis.
#
# The model uses Z-axis is the main axis so then the local axis must be defined.
```

```
# The local axis is cylindrical that states:
# x = radial direction, y = axial direction.

bmpModel.parts['Pipe'].DatumCsysByThreePoints(coordSysType=
    CYLINDRICAL, line1=(1.0, 0.0, 0.0), line2=(0.0, 1.0, 0.0), name=
    'Datum csys-1', origin=(0.0, 0.0, 0.0))

bmpModel.parts['Pipe'].MaterialOrientation(
    additionalRotationField='', additionalRotationType=ROTATION_NONE, angle=0.0
    , axis=AXIS_1, fieldName='', localCsys=
    bmpModel.parts['Pipe'].datums[3], orientationType=SYSTEM,
    region=Region(faces=bmpModel.parts['Pipe'].faces.findAt(((
    0.0, radius, pipeLength), (0.0, radius, 0.0)), )))


# 3. MODULE: ASSEMBLY
#
# 3.1. Create the assembly
#==========================================================================#

bmpModel.rootAssembly.DatumCsysByDefault(CARTESIAN)
bmpModel.rootAssembly.Instance(dependent=OFF, name='Pipe-1',
    part=bmpModel.parts['Pipe'])


# Creating collars

bmpModel.rootAssembly.DatumAxisByPrincipalAxis(principalAxis=
    ZAXIS)
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
    (0.0, 0.0, collar))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
    (0.0, 0.0, pipeLength-collar))

bmpModel.rootAssembly.DatumPlaneByPointNormal(normal=
    bmpModel.rootAssembly.datums[4], point=
    bmpModel.rootAssembly.datums[5])
bmpModel.rootAssembly.DatumPlaneByPointNormal(normal=
    bmpModel.rootAssembly.datums[4], point=
    bmpModel.rootAssembly.datums[6])

bmpModel.rootAssembly.PartitionFaceByDatumPlane(datumPlane=
    bmpModel.rootAssembly.datums[7], faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(((
    0.0, radius, pipeLength/2.0), )))
bmpModel.rootAssembly.PartitionFaceByDatumPlane(datumPlane=
```

```
    bmpModel.rootAssembly.datums[8], faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(((
    0.0, radius, pipeLength/2.0), )))


# 4. MODULE: STEP
#
# 4.1. Create the step
#=====================================================================#

bmpModel.StaticRiksStep(name='Step-1', nlgeom=ON, previous=
    'Initial')

# Requests for Field and History Output are placed after setting sets.


# 5. MODULE: INTERACTION
#
# Connect the pipe to the reference points with 'Coupling'
#=====================================================================#

disRP = 1.2
z1RP = -disRP
z2RP = pipeLength + disRP

bmpModel.rootAssembly.ReferencePoint(point=(0.0, 0.0, z1RP))
bmpModel.rootAssembly.ReferencePoint(point=(0.0, 0.0, z2RP))

bmpModel.RigidBody(bodyRegion=Region(
    faces=bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0.0, -radius, 0.0), (0.0, radius, 0.0)), )),
    name='Constraint-1', refPointRegion=Region(referencePoints=(
    bmpModel.rootAssembly.referencePoints[11], )))

bmpModel.RigidBody(bodyRegion=Region(
    faces=bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0.0, -radius, pipeLength), (0.0, radius, pipeLength)), )),
    name='Constraint-2', refPointRegion=Region(referencePoints=(
    bmpModel.rootAssembly.referencePoints[12], )))


# 6. MODULE: LOAD
#
# 6.1. Create the loads, internal pressure and bending moment
#=====================================================================#
```

```
pressInt = 11314297.0286
momentLoad = 2*7000000

#NOTE: internal pressure is defined by 'side2Faces'

bmpModel.Pressure(createStepName='Step-1', distributionType=
UNIFORM, field='', magnitude=pressInt, name='Pressure', region=Region(
side2Faces=bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
((0.0, -radius, 0.0), (0.0, radius, 0.0)),
((0.0, -radius, pipeLength/2.0), (0.0, radius, pipeLength/2.0)),
((0.0, -radius, pipeLength), (0.0, radius, pipeLength)), )))

bmpModel.Moment(cm1= momentLoad, createStepName='Step-1',
    distributionType=UNIFORM, field='', localCsys=None, name='Moment-1',
    region=Region(referencePoints=(
    bmpModel.rootAssembly.referencePoints[11], )))

bmpModel.Moment(cm1= -momentLoad, createStepName='Step-1',
    distributionType=UNIFORM, field='', localCsys=None, name='Moment-2',
    region=Region(referencePoints=(
    bmpModel.rootAssembly.referencePoints[12], )))


# 6.2. Create boundary conditions
#======================================================================#

bmpModel.DisplacementBC(amplitude=UNSET, createStepName='Step-1',
    distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=None,
    name='BC-1', region=Region(referencePoints=(
    bmpModel.rootAssembly.referencePoints[11], )), u1=0.0, u2=0.0,
    u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)

bmpModel.DisplacementBC(amplitude=UNSET, createStepName='Step-1',
    distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=None,
    name='BC-2', region=Region(referencePoints=(
    bmpModel.rootAssembly.referencePoints[12], )), u1=0.0, u2=
    0.0, u3=0.0, ur1=UNSET, ur2=UNSET, ur3=UNSET)


# 7. MODULE: MESH
#
# 7.1. Create partitions
#======================================================================#
#
# Creating sets (partitions)
```

```
# Create the reference points

# Create the reference points
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn1))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn2))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn3))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn4))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn5))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn6))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn7))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn8))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn9))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn10))

bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zm))

bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp1))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp2))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp3))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp4))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp5))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp6))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp7))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp8))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp9))
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
```

```
(0.0, ypos, zp10))

# Partitions

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[22], point2=
    bmpModel.rootAssembly.datums[23])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[23], point2=
    bmpModel.rootAssembly.datums[24])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[24], point2=
    bmpModel.rootAssembly.datums[25])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[25], point2=
    bmpModel.rootAssembly.datums[26])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[26], point2=
    bmpModel.rootAssembly.datums[27])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[27], point2=
    bmpModel.rootAssembly.datums[28])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[28], point2=
    bmpModel.rootAssembly.datums[29])
```

```
bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[29], point2=
    bmpModel.rootAssembly.datums[30])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[30], point2=
    bmpModel.rootAssembly.datums[31])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[31], point2=
    bmpModel.rootAssembly.datums[32])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[32], point2=
    bmpModel.rootAssembly.datums[33])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[33], point2=
    bmpModel.rootAssembly.datums[34])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[34], point2=
    bmpModel.rootAssembly.datums[35])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[35], point2=
    bmpModel.rootAssembly.datums[36])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[36], point2=
```

```
    bmpModel.rootAssembly.datums[37])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[37], point2=
    bmpModel.rootAssembly.datums[38])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[38], point2=
    bmpModel.rootAssembly.datums[39])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[39], point2=
    bmpModel.rootAssembly.datums[40])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[40], point2=
    bmpModel.rootAssembly.datums[41])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    bmpModel.rootAssembly.datums[41], point2=
    bmpModel.rootAssembly.datums[42])

bmpModel.rootAssembly.Set(name='Set-Top', vertices=
 bmpModel.rootAssembly.instances['Pipe-1'].vertices.findAt(
 ((0.0, ypos, zn1), ), ((0.0, ypos, zn2), ), ((0.0, ypos, zn3), ),
 ((0.0, ypos, zn4), ), ((0.0, ypos, zn5), ), ((0.0, ypos, zn6), ),
 ((0.0, ypos, zn7), ), ((0.0, ypos, zn8), ), ((0.0, ypos, zn9), ),
 ((0.0, ypos, zn10), ), ((0.0, ypos, zm), ), ((0.0, ypos, zp1), ),
 ((0.0, ypos, zp2), ), ((0.0, ypos, zp3), ), ((0.0, ypos, zp4), ),
 ((0.0, ypos, zp5), ), ((0.0, ypos, zp6), ), ((0.0, ypos, zp7), ),
 ((0.0, ypos, zp8), ), ((0.0, ypos, zp9), ), ((0.0, ypos, zp10), ), ))


bmpModel.rootAssembly.DatumPlaneByPointNormal(normal=
    bmpModel.rootAssembly.datums[4], point=
    bmpModel.rootAssembly.datums[30])
```

```
bmpModel.rootAssembly.DatumPlaneByPointNormal(normal=
    bmpModel.rootAssembly.datums[4], point=
    bmpModel.rootAssembly.datums[31])

bmpModel.rootAssembly.DatumPlaneByPointNormal(normal=
    bmpModel.rootAssembly.datums[4], point=
    bmpModel.rootAssembly.datums[33])
bmpModel.rootAssembly.DatumPlaneByPointNormal(normal=
    bmpModel.rootAssembly.datums[4], point=
    bmpModel.rootAssembly.datums[34])

bmpModel.rootAssembly.PartitionFaceByDatumPlane(datumPlane=
    bmpModel.rootAssembly.datums[64], faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zm), )))
bmpModel.rootAssembly.PartitionFaceByDatumPlane(datumPlane=
    bmpModel.rootAssembly.datums[65], faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zm), )))
bmpModel.rootAssembly.PartitionFaceByDatumPlane(datumPlane=
    bmpModel.rootAssembly.datums[66], faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zm), )))
bmpModel.rootAssembly.PartitionFaceByDatumPlane(datumPlane=
    bmpModel.rootAssembly.datums[67], faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp3), )))

bmpModel.rootAssembly.Set(edges=
    bmpModel.rootAssembly.instances['Pipe-1'].edges.findAt(
    ((0, yneg, zn9), )), name='Set-L50')
bmpModel.rootAssembly.Set(edges=
    bmpModel.rootAssembly.instances['Pipe-1'].edges.findAt(
    ((0, yneg, zn10), )), name='Set-L25')
bmpModel.rootAssembly.Set(edges=
    bmpModel.rootAssembly.instances['Pipe-1'].edges.findAt(
    ((0, yneg, zp1), )), name='Set-R25')
bmpModel.rootAssembly.Set(edges=
    bmpModel.rootAssembly.instances['Pipe-1'].edges.findAt(
    ((0, yneg, zp2), )), name='Set-R50')

bmpModel.rootAssembly.Set(name='Set-T1', vertices=
 bmpModel.rootAssembly.instances['Pipe-1'].vertices.findAt(
 ((0.0, ypos, zn9), ),  ))

bmpModel.rootAssembly.Set(name='Set-T2', vertices=
```

```
 bmpModel.rootAssembly.instances['Pipe-1'].vertices.findAt(
 ((0.0, ypos, zn10), ),  ))

bmpModel.rootAssembly.Set(name='Set-T3', vertices=
 bmpModel.rootAssembly.instances['Pipe-1'].vertices.findAt(
 ((0.0, ypos, zp1), ),  ))

bmpModel.rootAssembly.Set(name='Set-T4', vertices=
 bmpModel.rootAssembly.instances['Pipe-1'].vertices.findAt(
 ((0.0, ypos, zp2), ),  ))


# Creating set on the bottom

bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, yneg, zn9)) #80

bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, yneg, zn10))  #81

bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, yneg, zp1))   #82
bmpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, yneg, zp2))   #83

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0.0, yneg, zn9+0.01), )), point1=
    bmpModel.rootAssembly.datums[80], point2=
    bmpModel.rootAssembly.datums[81])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zm), )), point1=
    bmpModel.rootAssembly.datums[81], point2=
    bmpModel.rootAssembly.datums[82])

bmpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp1+0.01), )), point1=
    bmpModel.rootAssembly.datums[82], point2=
    bmpModel.rootAssembly.datums[83])

bmpModel.rootAssembly.Set(name='Set-B1', vertices=
    bmpModel.rootAssembly.instances['Pipe-1'].vertices.findAt(
    ((0.0, yneg, zn9), ),  ))
```

```
bmpModel.rootAssembly.Set(name='Set-B2', vertices=
    bmpModel.rootAssembly.instances['Pipe-1'].vertices.findAt(
    ((0.0, yneg, zn10), ),  ))

# 7.2. Create mesh
#====================================================================#

sizeMesh = 0.025  #(for the simulation)


# NOTE:
# TRI = triangle
# QUAD = guadrilateral
# technique = FREE / STRUCTURED

bmpModel.rootAssembly.setMeshControls(elemShape=TRI,
 regions=bmpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
 ((0.0, ypos, 0.0), ), ((0.0, yneg, zn5), ), ((yneg, 0.0, zn9+0.01), ),((ypos,
0.0, zn9+0.01), ),
 ((yneg, 0.0, zm), ), ((ypos, 0.0, zm), ), ((yneg, 0.0, zp1+0.01), ),((ypos,
0.0, zp1+0.01), ),
 ((0.0, yneg, zp5), ), ((0.0, ypos, pipeLength), ),  ),technique=FREE)

bmpModel.rootAssembly.seedPartInstance(deviationFactor=0.1,
    regions=(bmpModel.rootAssembly.instances['Pipe-1'], ),
    size=sizeMesh)
bmpModel.rootAssembly.generateMesh(regions=(
    bmpModel.rootAssembly.instances['Pipe-1'], ))


# For additional requested Field and History output

bmpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-2', rebar=EXCLUDE, region=
    bmpModel.rootAssembly.sets['Set-Top'], sectionPoints=
    DEFAULT, variables=('COORD',))

bmpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-3', rebar=EXCLUDE, region=
    bmpModel.rootAssembly.sets['Set-T1'], sectionPoints=
    DEFAULT, variables=('UR','COORD',))

bmpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-4', rebar=EXCLUDE, region=
    bmpModel.rootAssembly.sets['Set-T2'], sectionPoints=
```

```
    DEFAULT, variables=('UR','COORD',))

bmpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-5', rebar=EXCLUDE, region=
    bmpModel.rootAssembly.sets['Set-T3'], sectionPoints=
    DEFAULT, variables=('UR','COORD',))

bmpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-6', rebar=EXCLUDE, region=
    bmpModel.rootAssembly.sets['Set-T4'], sectionPoints=
    DEFAULT, variables=('UR','COORD',))

bmpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-7', rebar=EXCLUDE, region=
    bmpModel.rootAssembly.sets['Set-B1'], sectionPoints=
    DEFAULT, variables=('UR','COORD',))

bmpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-8', rebar=EXCLUDE, region=
    bmpModel.rootAssembly.sets['Set-B2'], sectionPoints=
    DEFAULT, variables=('UR','COORD',))

bmpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-9', rebar=EXCLUDE, region=
    bmpModel.rootAssembly.sets['Set-L25'], sectionPoints=
    DEFAULT, variables=('S', 'SE', 'UR', 'SF', 'COORD'))

bmpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-10', rebar=EXCLUDE, region=
    bmpModel.rootAssembly.sets['Set-L50'], sectionPoints=
    DEFAULT, variables=('S', 'SE', 'UR', 'SF', 'COORD'))



# 8. MODULE: JOB
#
# 8.1. Create job
#====================================================================#

mdb.Job(atTime=None, contactPrint=OFF, description='', echoPrint=OFF,
    explicitPrecision=SINGLE, getMemoryFromAnalysis=True, historyPrint=OFF,
    memory=90, memoryUnits=PERCENTAGE, model='Benchmark Pipe', modelPrint=
    OFF, multiprocessingMode=DEFAULT, name='BMP_model', nodalOutputPrecision=
    SINGLE, numCpus=1, numGPUs=0, queue=None, scratch='', type=ANALYSIS,
    userSubroutine='', waitHours=0, waitMinutes=0)
```

```
# 8.2. Write BMP_model.inp
#==========================================================================#
# Click 'Write Input' on Module:Job, the file 'BMP_model.inp' will be
# generated. After adding IMPERFECTION and edit the output element set
# on that file, then run the job by typing:
# 'abaqus job=BMP_model interactive' on Windows command prompt.
# The result will be saved on the file BMP_model.odb.
```

A2. SWP_model.py

```
#==========================================================================#
# SPIRAL WELDED PIPE (SWP_model.py)
#
# This program simulates buckling on a sipral welded pipe under
# an internal pressure and moment load.
#
# The SI system of units is used in this program.
#
# This program is create with the CAE's journal file and then
# it is modified by A Susilo.
#
# Created on: 04 Oct 2013
#==========================================================================#


from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
import math

# Dimmesions required
pipeLength = 11.0
radius = 0.44577
pipeODia = 2*radius
pipeThickness = 0.01143
collar = 0.4
pipePitch = 40.
weldGap = 0.01

# Constants / inputs
Pi = math.pi
alpha  = pipePitch/180.*Pi
radius = pipeODia/2.
```

```
# Reference points
zm = pipeLength/2.

zp1 = zm + (0.25*pipeODia)
zp2 = zm + (0.50*pipeODia)
zp3 = zm + (0.75*pipeODia)
zp4 = zm + (1.00*pipeODia)
zp5 = zm + (1.25*pipeODia)
zp6 = zm + (1.50*pipeODia)
zp7 = zm + (1.75*pipeODia)
zp8 = zm + (2.00*pipeODia)
zp9 = zm + (2.25*pipeODia)
zp10 = zm + (2.50*pipeODia)

zn10 = zm - (0.25*pipeODia)
zn9 = zm - (0.50*pipeODia)
zn8 = zm - (0.75*pipeODia)
zn7 = zm - (1.00*pipeODia)
zn6 = zm - (1.25*pipeODia)
zn5 = zm - (1.50*pipeODia)
zn4 = zm - (1.75*pipeODia)
zn3 = zm - (2.00*pipeODia)
zn2 = zm - (2.25*pipeODia)
zn1 = zm - (2.50*pipeODia)

ypos = radius
yneg = -radius

epsilon = 0.5*(zp2-zp1)


# 1. MODULE: PART
#
# 1.1. Create the model
#==========================================================================#
#
# The model is called 'Spiral Welded Pipe'

mdb.models.changeKey(fromName='Model-1', toName='Spiral Welded Pipe')
swpModel = mdb.models['Spiral Welded Pipe']

# 1.2. Create the part
#==========================================================================#
#
# The part is a circle. It is created by a circle uses 2 points (center, radius).
```

```
# Then this part is extruded 'pipeLength' long.
#
# The part has 2 sections, the shell and the weld. The part is a circle.
# It is created by an arc that uses 3 points (center, point1 and point 2).
# Then this part is extruded to 'pipeLength' long and twisted
# 'pipePitch' long/revolution.
#
# The part is called 'Pipe'

# Define points for creating the arc
#==================================#

weldGapBase = weldGap/math.sin(alpha)
xArc1 = weldGapBase/2.
yArc1 = (radius*radius - xArc1*xArc1)**0.5
xArc2 = -xArc1
yArc2 = yArc1

pipeCirc = Pi * pipeODia
pipePitch = pipeCirc*math.tan(alpha)

swpModel.ConstrainedSketch(name='__profile__', sheetSize=1.0)
swpModel.sketches['__profile__'].Spot(point=(0.0, 0.0))

# Arches are drawn counterclockwise so it will give normal outward
swpModel.sketches['__profile__'].ArcByCenterEnds(center=(0.0, 0.0)
, direction=COUNTERCLOCKWISE, point1=(xArc1, yArc1), point2=(xArc2, yArc2))
swpModel.sketches['__profile__'].ArcByCenterEnds(center=(0.0, 0.0)
, direction=COUNTERCLOCKWISE, point1=(xArc2, yArc2), point2=(xArc1, yArc1))
swpModel.Part(dimensionality=THREE_D, name='Pipe', type=
DEFORMABLE_BODY)
swpModel.parts['Pipe'].BaseShellExtrude(depth=pipeLength, pitch=pipePitch,
sketch=swpModel.sketches['__profile__'])

del swpModel.sketches['__profile__']


# 2. MODULE: PROPERTY
#
# 2.1. Create material
#====================================================================#
#
# There are 2 materials in this model, material of the shell and
# material of the weld. The values are supplied by user.

mat1 = 'Material-Shell'
```

```
mat2 = 'Material-Weld'

# Material 1 = the shell
young1 = 2.05e11
poisson1 = 0.3

# Material 2 = the weld
young2 = 2.05e11
poisson2 = 0.3

pipeMaterial = swpModel.Material(name=mat1)
pipeMaterial.Elastic(table=((young1, poisson1), ))
pipeMaterial.Plastic(table=(
(430500000,0),
(433125000,1.22E-05),
(435750000,2.44E-05),
(438375000,3.66034E-05),
(441000000,4.8808E-05),
(443625000,6.1015E-05),
(446250000,7.32251E-05),
(448875000,8.54391E-05),
(451500000,9.76579E-05),
(454125000,0.000109883),
(456750000,0.000122115),
(459375000,0.000134357),
(462000000,0.00014661),
(464625000,0.000158878),
(467250000,0.000171164),
(469875000,0.000183472),
(472500000,0.000195807),
(475125000,0.000208176),
(477750000,0.000220587),
(480375000,0.00023305),
(483000000,0.000245576),
(485625000,0.000258181),
(488250000,0.000270882),
(490875000,0.0002837),
(493500000,0.000296663),
(496125000,0.000309803),
(498750000,0.00032316),
(501375000,0.000336781),
(504000000,0.000350725),
(506625000,0.000365063),
(509250000,0.000379881),
(511875000,0.000395283),
(514500000,0.000411395),
```

```
(517125000,0.000428369),
(519750000,0.000446389),
(522375000,0.000465677),
(525000000,0.000486501),
(527625000,0.000509182),
(530250000,0.000534109),
(532875000,0.000561747),
(535500000,0.000592655),
(538125000,0.000627506),
(540750000,0.000667103),
(543375000,0.000712412),
(546000000,0.000764585),
(548625000,0.000825002),
(551250000,0.000895311),
(553875000,0.000977479),
(556500000,0.001073852),
(559125000,0.001187224),
(561750000,0.001320926),
(564375000,0.001478915),
(567000000,0.001665899),
(569625000,0.001887468),
(572250000,0.002150257),
(574875000,0.002462134),
(577500000,0.002832422),
(580125000,0.003272159),
(582750000,0.003794406),
(585375000,0.004414603),
(588000000,0.005150989),
(590625000,0.006025094),
(593250000,0.007062316),
(595875000,0.008292589),
(598500000,0.009751174),
(601125000,0.011479569),
(603750000,0.013526587),
(606375000,0.015949594),
(609000000,0.018815971),
(611625000,0.0222048),
(614250000,0.026208832),
(616875000,0.030936781),
(619500000,0.036515981),
(622125000,0.043095472),
(624750000,0.05084959),
(627375000,0.059982121),
(630000000,0.070731124),
(632625000,0.083374509),
(635250000,0.098236502),
```

```
(637875000,0.115695123),
(640500000,0.13619083),
(643125000,0.16023651),
(645750000,0.188429017),
(648375000,0.221462487)))
pipeMaterial.plastic.Potential(table=
    ((0.981, 1.0, 1.0, 1.0, 1.0, 1.0), ))

weldMaterial = swpModel.Material(name=mat2)
weldMaterial.Elastic(table=((young2, poisson2), ))
weldMaterial.Plastic(table=(
(430500000,0),
(476437500,1.22E-05),
(479325000,2.44E-05),
(482212500,3.66E-05),
(485100000,4.88E-05),
(487987500,6.10E-05),
(490875000,7.32E-05),
(493762500,8.54E-05),
(496650000,9.77E-05),
(499537500,0.000109883),
(502425000,0.000122115),
(505312500,0.000134357),
(508200000,0.00014661),
(511087500,0.000158878),
(513975000,0.000171164),
(516862500,0.000183472),
(519750000,0.000195807),
(522637500,0.000208176),
(525525000,0.000220587),
(528412500,0.00023305),
(531300000,0.000245576),
(534187500,0.000258181),
(537075000,0.000270882),
(539962500,0.0002837),
(542850000,0.000296663),
(545737500,0.000309803),
(548625000,0.00032316),
(551512500,0.000336781),
(554400000,0.000350725),
(557287500,0.000365063),
(560175000,0.000379881),
(563062500,0.000395283),
(565950000,0.000411395),
(568837500,0.000428369),
(571725000,0.000446389),
```

```
(574612500,0.000465677),
(577500000,0.000486501),
(580387500,0.000509182),
(583275000,0.000534109),
(586162500,0.000561747),
(589050000,0.000592655),
(591937500,0.000627506),
(594825000,0.000667103),
(597712500,0.000712412),
(600600000,0.000764585),
(603487500,0.000825002),
(606375000,0.000895311),
(609262500,0.000977479),
(612150000,0.001073852),
(615037500,0.001187224),
(617925000,0.001320926),
(620812500,0.001478915),
(623700000,0.001665899),
(626587500,0.001887468),
(629475000,0.002150257),
(632362500,0.002462134),
(635250000,0.002832422),
(638137500,0.003272159),
(641025000,0.003794406),
(643912500,0.004414603),
(646800000,0.005150989),
(649687500,0.006025094),
(652575000,0.007062316),
(655462500,0.008292589),
(658350000,0.009751174),
(661237500,0.011479569),
(664125000,0.013526587),
(667012500,0.015949594),
(669900000,0.018815971),
(672787500,0.0222048),
(675675000,0.026208832),
(678562500,0.030936781),
(681450000,0.036515981),
(684337500,0.043095472),
(687225000,0.05084959),
(690112500,0.059982121),
(693000000,0.070731124),
(695887500,0.083374509),
(698775000,0.098236502),
(701662500,0.115695123),
(704550000,0.13619083),
```

```
(707437500,0.16023651),
(710325000,0.188429017),
(713212500,0.221462487)))
weldMaterial.plastic.Potential(table=
    ((0.981, 1.0, 1.0, 1.0, 1.0, 1.0), ))

# 2.2. Create section
#==========================================================================#
#
# There are 2 SECTIONS in this model, shell and weld

swpModel.HomogeneousShellSection(idealization=NO_IDEALIZATION,
integrationRule=SIMPSON, material=mat1, name='Section-Shell',
numIntPts=9, poissonDefinition=DEFAULT, preIntegrate=OFF,
temperature=GRADIENT, thickness=pipeThickness, thicknessField='',
thicknessModulus=None, thicknessType=UNIFORM, useDensity=OFF)

swpModel.HomogeneousShellSection(idealization=NO_IDEALIZATION,
integrationRule=SIMPSON, material=mat2, name='Section-Weld',
numIntPts=9, poissonDefinition=DEFAULT, preIntegrate=OFF,
temperature=GRADIENT, thickness=pipeThickness, thicknessField='',
thicknessModulus=None, thicknessType=UNIFORM, useDensity=OFF)

# 2.3. Assign section
#==========================================================================#

swpModel.parts['Pipe'].Set(faces=
    swpModel.parts['Pipe'].faces.findAt(((-0.397661,
    -0.201437, 0.654762), )), name='Set-1-shell')

swpModel.parts['Pipe'].SectionAssignment(offset=0.0,
    offsetField='', offsetType=TOP_SURFACE, region=
    swpModel.parts['Pipe'].sets['Set-1-shell'],
    sectionName='Section-Shell', thicknessAssignment=FROM_SECTION)

swpModel.parts['Pipe'].Set(faces=
    swpModel.parts['Pipe'].faces.findAt(((-0.437571,
    -0.085104, 0.662651), )), name='Set-2-weld')

swpModel.parts['Pipe'].SectionAssignment(offset=0.0,
    offsetField='', offsetType=TOP_SURFACE, region=
    swpModel.parts['Pipe'].sets['Set-2-weld'],
    sectionName='Section-Weld', thicknessAssignment=FROM_SECTION)

# NOTE:
# For a model with the conventional shell element,
```

```
# it is suggested that the X-axis is the axial axis.
#
# The model uses Z-axis is the main axis so then the local axis must be defined.
# The local axis is cylindrical that states:
# x = radial direction, y = axial direction.

swpModel.parts['Pipe'].DatumCsysByThreePoints(coordSysType=
    CYLINDRICAL, line1=(1.0, 0.0, 0.0), line2=(0.0, 1.0, 0.0), name=
    'Datum csys-1', origin=(0.0, 0.0, 0.0))

swpModel.parts['Pipe'].MaterialOrientation(
    additionalRotationField='', additionalRotationType=ROTATION_NONE, angle=0.0
    , axis=AXIS_1, fieldName='', localCsys=
    swpModel.parts['Pipe'].datums[4], orientationType=SYSTEM,
    region=Region(faces=swpModel.parts['Pipe'].faces.findAt(((0.0, -radius,
pipeLength),(0.0, -radius, 0.0)), ((0.0, radius, 0.0), (xArc1, yArc1, 0.0)),
)))



# 3. MODULE: ASSEMBLY
#
# 3.1. Create the assembly
#=============================================================================#

swpModel.rootAssembly.DatumCsysByDefault(CARTESIAN)
swpModel.rootAssembly.Instance(dependent=OFF, name='Pipe-1',
    part=swpModel.parts['Pipe'])


# Creating collars
swpModel.rootAssembly.DatumAxisByPrincipalAxis(principalAxis=
    ZAXIS)
swpModel.rootAssembly.DatumPointByCoordinate(coords=
    (0.0, 0.0, collar))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
    (0.0, 0.0, pipeLength-collar))

swpModel.rootAssembly.DatumPlaneByPointNormal(normal=
    swpModel.rootAssembly.datums[4], point=
    swpModel.rootAssembly.datums[5])
swpModel.rootAssembly.DatumPlaneByPointNormal(normal=
    swpModel.rootAssembly.datums[4], point=
    swpModel.rootAssembly.datums[6])

swpModel.rootAssembly.PartitionFaceByDatumPlane(datumPlane=
```

```
    swpModel.rootAssembly.datums[7], faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(((
    0.0, radius, pipeLength/2.0), )))
swpModel.rootAssembly.PartitionFaceByDatumPlane(datumPlane=
    swpModel.rootAssembly.datums[8], faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(((
    0.0, radius, pipeLength/2.0), )))


# 4. MODULE: STEP
#
# 4.1. Create the step
#==========================================================================#

swpModel.StaticRiksStep(name='Step-1', nlgeom=ON, previous=
    'Initial')

# Requests for Field and History Output are placed after setting sets.


# 5. MODULE: INTERACTION
#
# Connect the pipe to the reference points with 'rigid body'
#==========================================================================#

disRP = 1.2
z1RP = -disRP
z2RP = pipeLength + disRP

swpModel.rootAssembly.ReferencePoint(point=(0.0, 0.0, z1RP))
swpModel.rootAssembly.ReferencePoint(point=(0.0, 0.0, z2RP))

swpModel.RigidBody(bodyRegion=Region(
    faces=swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0.0, -radius, 0.0), (0.0, radius, 0.0)), )),
    name='Constraint-1', refPointRegion=Region(referencePoints=(
    swpModel.rootAssembly.referencePoints[11], )))

swpModel.RigidBody(bodyRegion=Region(
    faces=swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0.0, -radius, pipeLength), (0.0, radius, pipeLength)), )),
    name='Constraint-2', refPointRegion=Region(referencePoints=(
    swpModel.rootAssembly.referencePoints[12], )))


# 6. MODULE: LOAD
```

```
#
# 6.1. Create the loads, internal pressure and bending moment
#=====================================================================#

pressInt = 11314297.0286
momentLoad = 2*7000000

#NOTE: internal pressure is defined by 'side2Faces'

swpModel.parts['Pipe'].Surface(name='Surf-Pressure',
    side2Faces=swpModel.parts['Pipe'].faces.findAt(((
    -0.437571, -0.085104, 0.662651), ), ((-0.397661, -0.201437, 0.654762), ),
    ))

swpModel.Pressure(createStepName='Step-1',
    distributionType=UNIFORM, field='', magnitude=pressInt, name='Pressure',
    region=swpModel.rootAssembly.instances['Pipe-1'].surfaces['Surf-Pressure'])

swpModel.Moment(cm1= momentLoad, createStepName='Step-1',
    distributionType=UNIFORM, field='', localCsys=None, name='Moment-1',
    region=Region(referencePoints=(
    swpModel.rootAssembly.referencePoints[11], )))

swpModel.Moment(cm1= -momentLoad, createStepName='Step-1',
    distributionType=UNIFORM, field='', localCsys=None, name='Moment-2',
    region=Region(referencePoints=(
    swpModel.rootAssembly.referencePoints[12], )))


# 6.2. Create boundary conditions
#=====================================================================#

swpModel.DisplacementBC(amplitude=UNSET, createStepName='Step-1',
    distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=None,
    name='BC-1', region=Region(referencePoints=(
    swpModel.rootAssembly.referencePoints[11], )), u1=0.0, u2=0.0,
    u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)

swpModel.DisplacementBC(amplitude=UNSET, createStepName='Step-1',
    distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=None,
    name='BC-2', region=Region(referencePoints=(
    swpModel.rootAssembly.referencePoints[12], )), u1=0.0, u2=
    0.0, u3=0.0, ur1=UNSET, ur2=UNSET, ur3=UNSET)


# 7. MODULE: MESH
```

```
#
# 7.1. Create partitions
#====================================================================#
#
# Creating sets (partitions)

# Create the reference points
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn1))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn2))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn3))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn4))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn5))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn6))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn7))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn8))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn9))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zn10))

swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zm))

swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp1))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp2))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp3))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp4))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp5))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp6))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp7))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
```

```
(0.0, ypos, zp8))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp9))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, ypos, zp10))

# Partitions

swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[21], point2=
    swpModel.rootAssembly.datums[22])

swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[22], point2=
    swpModel.rootAssembly.datums[23])

swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[23], point2=
    swpModel.rootAssembly.datums[24])

swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[24], point2=
    swpModel.rootAssembly.datums[25])

swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[25], point2=
    swpModel.rootAssembly.datums[26])

swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[26], point2=
    swpModel.rootAssembly.datums[27])

# No shortest path partition between 27-28
```

```
swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[28], point2=
    swpModel.rootAssembly.datums[29])

swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[29], point2=
    swpModel.rootAssembly.datums[30])

swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[30], point2=
    swpModel.rootAssembly.datums[31])

swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[31], point2=
    swpModel.rootAssembly.datums[32])

swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[32], point2=
    swpModel.rootAssembly.datums[33])

swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[33], point2=
    swpModel.rootAssembly.datums[34])

swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[34], point2=
    swpModel.rootAssembly.datums[35])

swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[35], point2=
```

```
    swpModel.rootAssembly.datums[36])

swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[36], point2=
    swpModel.rootAssembly.datums[37])

# No shortest path partition between 37-38

swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[38], point2=
    swpModel.rootAssembly.datums[39])

swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[39], point2=
    swpModel.rootAssembly.datums[40])

swpModel.rootAssembly.PartitionFaceByShortestPath(faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0, yneg, zp10+epsilon), )), point1=
    swpModel.rootAssembly.datums[40], point2=
    swpModel.rootAssembly.datums[41])

# Create the set of top points

swpModel.rootAssembly.Set(name='Set-Top', vertices=
 swpModel.rootAssembly.instances['Pipe-1'].vertices.findAt(
 ((0.0, ypos, zn1), ), ((0.0, ypos, zn2), ), ((0.0, ypos, zn3), ),
 ((0.0, ypos, zn4), ), ((0.0, ypos, zn5), ), ((0.0, ypos, zn6), ),
 ((0.0, ypos, zn7), ), ((0.0, ypos, zn8), ), ((0.0, ypos, zn9), ),
 ((0.0, ypos, zn10), ), ((0.0, ypos, zm), ), ((0.0, ypos, zp1), ),
 ((0.0, ypos, zp2), ), ((0.0, ypos, zp3), ), ((0.0, ypos, zp4), ),
 ((0.0, ypos, zp5), ), ((0.0, ypos, zp6), ), ((0.0, ypos, zp7), ),
 ((0.0, ypos, zp8), ), ((0.0, ypos, zp9), ), ((0.0, ypos, zp10), ), ))

swpModel.rootAssembly.DatumPlaneByPointNormal(normal=
    swpModel.rootAssembly.datums[4], point=
    swpModel.rootAssembly.datums[29])
swpModel.rootAssembly.DatumPlaneByPointNormal(normal=
    swpModel.rootAssembly.datums[4], point=
    swpModel.rootAssembly.datums[30])
```

```
swpModel.rootAssembly.DatumPlaneByPointNormal(normal=
    swpModel.rootAssembly.datums[4], point=
    swpModel.rootAssembly.datums[32])
swpModel.rootAssembly.DatumPlaneByPointNormal(normal=
    swpModel.rootAssembly.datums[4], point=
    swpModel.rootAssembly.datums[33])

# Create the circle set

swpModel.rootAssembly.PartitionFaceByDatumPlane(
    datumPlane=swpModel.rootAssembly.datums[61], faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((-0.122593, 0.428581, 3.27381), ), ((-0.13849, -0.423712, 1.060241), ),
))
swpModel.rootAssembly.PartitionFaceByDatumPlane(
    datumPlane=swpModel.rootAssembly.datums[62], faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((-0.413812, -0.165743, 5.433735), ), ((-0.389887, -0.2161, 5.369048), ),
    ))
swpModel.rootAssembly.PartitionFaceByDatumPlane(
    datumPlane=swpModel.rootAssembly.datums[63], faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((-0.006518, -0.445722, 5.761905), ), ((-0.44561, -0.011928, 5.301205), ),
    ))
swpModel.rootAssembly.PartitionFaceByDatumPlane(
    datumPlane=swpModel.rootAssembly.datums[64], faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0.405486, 0.185181, 6.626506), ), ((-0.006518, -0.445722, 5.761905), ),
    ))

swpModel.rootAssembly.Set(edges=
    swpModel.rootAssembly.instances['Pipe-1'].edges.findAt(
    ((-0.334379, -0.294791, 5.05423), ), ((-0.35926, 0.2639, 5.05423), ), ((
    -0.102578, 0.433807, 5.05423), ), ), name='Set-L50')
swpModel.rootAssembly.Set(edges=
    swpModel.rootAssembly.instances['Pipe-1'].edges.findAt(
    ((-0.174145, -0.410347, 5.277115), ), ((-0.165791, 0.413792, 5.277115), ),
    ((-0.445454, 0.01679, 5.277115), ), ), name='Set-L25')
swpModel.rootAssembly.Set(edges=
    swpModel.rootAssembly.instances['Pipe-1'].edges.findAt(
    ((0.210758, -0.3928, 5.722885), ), ((-0.18043, -0.407622, 5.722885), ), ((
    -0.27998, 0.346874, 5.722885), ), ), name='Set-R25')
swpModel.rootAssembly.Set(edges=
    swpModel.rootAssembly.instances['Pipe-1'].edges.findAt(
    ((0.35983, -0.263122, 5.94577), ), ((-0.328356, 0.301485, 5.94577), ), ((
```

```
    0.079438, -0.438635, 5.94577), ), ), name='Set-R50')

# Create set of 4 points on the top

swpModel.rootAssembly.Set(name='Set-T1', vertices=
 swpModel.rootAssembly.instances['Pipe-1'].vertices.findAt(
 ((0.0, ypos, zn9), ),  ))

swpModel.rootAssembly.Set(name='Set-T2', vertices=
 swpModel.rootAssembly.instances['Pipe-1'].vertices.findAt(
 ((0.0, ypos, zn10), ),  ))

swpModel.rootAssembly.Set(name='Set-T3', vertices=
 swpModel.rootAssembly.instances['Pipe-1'].vertices.findAt(
 ((0.0, ypos, zp1), ),  ))

swpModel.rootAssembly.Set(name='Set-T4', vertices=
 swpModel.rootAssembly.instances['Pipe-1'].vertices.findAt(
 ((0.0, ypos, zp2), ),  ))


# Create set of 4 points on the bottom

swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, yneg, zn9))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, yneg, zn10))

swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, yneg, zp1))
swpModel.rootAssembly.DatumPointByCoordinate(coords=
(0.0, yneg, zp2))

swpModel.rootAssembly.PartitionFaceByShortestPath(
    faces=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((-0.440373, -0.069157, 5.238095), )), point1=
    swpModel.rootAssembly.datums[77], point2=
    swpModel.rootAssembly.datums[78])

swpModel.rootAssembly.DatumPlaneByLinePoint(line=
    swpModel.rootAssembly.datums[4], point=
    swpModel.rootAssembly.datums[79])

swpModel.rootAssembly.PartitionFaceByDatumPlane(
    datumPlane=swpModel.rootAssembly.datums[82], faces=
```

```
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0.146846, -0.420889, 5.892857), ), ((-0.056343, -0.442195, 5.831325), ),
    ((-0.080843, 0.438378, 5.892857), ), ))

swpModel.rootAssembly.Set(name='Set-B1', vertices=
    swpModel.rootAssembly.instances['Pipe-1'].vertices.findAt(
    ((0.0, yneg, zn9), )))
swpModel.rootAssembly.Set(name='Set-B2', vertices=
    swpModel.rootAssembly.instances['Pipe-1'].vertices.findAt(
    ((0.0, yneg, zn10), )))
swpModel.rootAssembly.Set(name='Set-B3', vertices=
    swpModel.rootAssembly.instances['Pipe-1'].vertices.findAt(
    ((0.0, yneg, zp1), )))
swpModel.rootAssembly.Set(name='Set-B4', vertices=
    swpModel.rootAssembly.instances['Pipe-1'].vertices.findAt(
    ((0.0, yneg, zp2), )))


# 7.2. Create mesh
#======================================================================#

sizeMesh = 0.025  #(for the simulation)

# NOTE:
# TRI = triangle
# QUAD = guadrilateral
# technique = FREE / STRUCTURED

swpModel.rootAssembly.setMeshControls(elemShape=TRI,
    regions=
    swpModel.rootAssembly.instances['Pipe-1'].faces.findAt(
    ((0.012643, -0.445591, 5.935145), ), ((0.012661, -0.44559, 5.890469), ),
((
    -0.006518, -0.445722, 5.761905), ), ((-0.060001, -0.441713, 5.107143), ),
(
    (0.03978, 0.443991, 5.892857), ), ((0.155111, 0.417913, 6.809524), ), ((
    -0.056343, -0.442195, 5.831325), ), ((0.358932, -0.264345, 6.228916), ),
((
    -0.158908, -0.416484, 5.630952), ), ((-0.011358, 0.445625, 5.630952), ),
((
    0.095033, -0.435522, 5.238095), ), ((-0.046136, 0.443376, 5.761905), ), ((
    -0.422056, 0.143456, 5.168675), ), ((-0.206331, -0.395143, 5.698795), ),
((
    -0.122593, 0.428581, 3.27381), ), ((-0.027426, 0.444925, 5.238095), ), ((
    -0.433896, 0.102202, 0.392857), ), ((0.369481, 0.249389, 1.987952), ), ((
    -0.091119, -0.436358, 10.607143), ), ))
```

```
swpModel.rootAssembly.seedPartInstance(deviationFactor=0.1,
    regions=(swpModel.rootAssembly.instances['Pipe-1'], ),
    size=sizeMesh)
swpModel.rootAssembly.generateMesh(regions=(
    swpModel.rootAssembly.instances['Pipe-1'], ))


# For additional requested Field and History output

swpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-2', rebar=EXCLUDE, region=
    swpModel.rootAssembly.sets['Set-Top'], sectionPoints=
    DEFAULT, variables=('COORD',))

swpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-3', rebar=EXCLUDE, region=
    swpModel.rootAssembly.sets['Set-T1'], sectionPoints=
    DEFAULT, variables=('UR','COORD',))

swpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-4', rebar=EXCLUDE, region=
    swpModel.rootAssembly.sets['Set-T2'], sectionPoints=
    DEFAULT, variables=('UR','COORD',))

swpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-5', rebar=EXCLUDE, region=
    swpModel.rootAssembly.sets['Set-T3'], sectionPoints=
    DEFAULT, variables=('UR','COORD',))

swpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-6', rebar=EXCLUDE, region=
    swpModel.rootAssembly.sets['Set-T4'], sectionPoints=
    DEFAULT, variables=('UR','COORD',))

swpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-7', rebar=EXCLUDE, region=
    swpModel.rootAssembly.sets['Set-B1'], sectionPoints=
    DEFAULT, variables=('UR','COORD',))

swpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-8', rebar=EXCLUDE, region=
    swpModel.rootAssembly.sets['Set-B2'], sectionPoints=
    DEFAULT, variables=('UR','COORD',))

swpModel.FieldOutputRequest(createStepName='Step-1', name=
```

```
    'F-Output-9', rebar=EXCLUDE, region=
    swpModel.rootAssembly.sets['Set-L25'], sectionPoints=
    DEFAULT, variables=('S', 'SE', 'UR', 'SF', 'COORD'))

swpModel.FieldOutputRequest(createStepName='Step-1', name=
    'F-Output-10', rebar=EXCLUDE, region=
    swpModel.rootAssembly.sets['Set-L50'], sectionPoints=
    DEFAULT, variables=('S', 'SE', 'UR', 'SF', 'COORD'))



# 8. MODULE: JOB
#
# 8.1. Create job
#====================================================================#

mdb.Job(atTime=None, contactPrint=OFF, description='', echoPrint=OFF,
    explicitPrecision=SINGLE, getMemoryFromAnalysis=True, historyPrint=OFF,
    memory=90, memoryUnits=PERCENTAGE, model='Spiral Welded Pipe', modelPrint=
    OFF, multiprocessingMode=DEFAULT, name='SWP_model', nodalOutputPrecision=
    SINGLE, numCpus=1, numGPUs=0, queue=None, scratch='', type=ANALYSIS,
    userSubroutine='', waitHours=0, waitMinutes=0)

# 8.2. Write SWP_model.inp
#====================================================================#
# Click 'Write Input' on Module:Job, the file 'SWP_model.inp' will be
# generated. After adding IMPERFECTION and edit the output element set
# on that file, then run the job by typing:
# 'abaqus job = SWP_model interactive' on Windows command prompt.
# The result will be saved on the file SWP_model.odb.
```

A3. BMP/SWP_input.dat[1]

```
    1,           0.,  0.609499991,           0.
    2,           0.,  0.609499991,          0.5
    3,           0.,  0.609499991,   14.1280003
    4,           0.,  0.609499991,   14.6280003
    5,           0., -0.609499991,   7.31400013
...
...
...
81131, -0.0173710529, -0.609252393,   7.40107536
81132, -0.0168078318, -0.609268188,   7.38118792
```

[1]The format is: **nodenumber, x, y, z** (in global axis).

A4. createIMP.f

```
        program readINP

c!=========================================================================!
c! This program reads 'XXX_input.dat', this file is produced in a text editor.
c! It contains a list of all nodes of the model found in 'XXX_model.inp'.
c!
c! Then the program generates 'dx,dy,0' list to represent the imperfection
c! on the middle of the pipe. The imperfection has wave shape with 2 troughs
c! and 1 crest.
c!
c! The imperfection is written on the file 'XXX_imper.dat'. Then this file
c! will be used by 'XXX_model.inp' to run the model with the imperfection.
c!
c! Prepared by A Susilo
c! Created on: 01 Oct 2013
c!=========================================================================!

       implicit none
       integer i,n

c! n = total nodes for each model, BMP and SWP respectively
       parameter (n=45424)
c       parameter (n=56251)


       integer inode(n)
       real x(n),y(n),z(n), rm, lambda,lam025,lambda2
       real thk,delta,pi,dx,dy,tiny,delta1,deltaZ,angle
   real thc,amp1,amp2,length,dlz,ampblis


       pi = 4*atan(1.)

   thk = 0.01143
   amp1 = 0.02*thk
       amp2 = 1.25*amp1
   length = 11
   rm = length/2.

c! To prevent division by zero
        tiny = 0.00001

lambda = 0.815
```

```
lam025 = 0.25*lambda

lambda2 = 0.25
ampblis = 0.0005


c!============================!
c! SELECT ONE: ('c' = disable)
c!============================!

      open (7,file="BMP_input.dat")
c     open (7,file="SWP_input.dat")


       do i = 1,n
        read(7,*)inode(i),x(i),y(i),z(i)
       end do

c!============================!
c! SELECT ONE: ('c' = disable)
c!============================!

      open (7,file="BMP_imper.dat")
c     open (7,file="SWP_imper.dat")


c! Start computation
       do i = 1,n
        dlz = z(i)-rm


       deltaZ = amp1*cos(dlz/lambda * 2*pi)
     &          + ampblis*cos(dlz/lambda2 * 2*pi)
         if (abs(dlz).le.lam025) deltaZ = amp2*cos(dlz/lambda * 2*pi)
     &                           + ampblis*cos(dlz/lambda2 * 2*pi)
          dx = 0.
          dy = 0.

          if(x(i).ge.0.0.and.y(i).ge.0.0) then
             if (abs(x(i)).lt.tiny) x(i) = tiny
             angle = atan(y(i)/x(i))
             delta = deltaZ*sin(angle)
             dx = delta*cos(angle)
             dy = delta*sin(angle)
          end if
```

```
          if(x(i).lt.0.0.and.y(i).ge.0.0) then
             angle = pi-atan(abs(y(i)/x(i)))
             delta = deltaZ*sin(angle)
             dx = delta*cos(angle)
             dy = delta*sin(angle)
          end if

c! Write the non zero only
          if(dx.eq.0.0.and.dy.eq.0.0) then
           continue
          else
           write(7,*)"Pipe-1.",inode(i),",",dx,",",dy,",",0.
          end if


c! End computation

       end do

       close(7)

       end program readINP
```

A5. BMP/SWP_imper.dat[2]

```
Pipe-1.6,  5.00418707E-07,  0.0305000003,  0.
Pipe-1.657, -0.000622026157,  1.26910691E-05,  0.
Pipe-1.658, -0.00186193699,  0.000114092676,  0.
...
...
...
Pipe-1.81121,  0.00188690808,  0.0292593371,  0.
Pipe-1.81122,  0.00165189803,  0.0301281363,  0.
```

[2]The format is: **partname.nodenumber, Δx, Δy, Δz** (in global axis).

A6. The_MiddElm.py

```
#=============================================================================#
# THE MIDDLE ELEMENT (The_MiddElm.py)
#
# This program finds the correct element set of the requested set.
#
# It seems Abaqus needs minimal 2 points to define
# an attached element on the set, but for triangle elements,
# there are some elements have 1 point attached to the set only.
# Therefore those elements are not included in the set.
# This program will list all elements that attached to the set.
#
# The program reads data from 'XXX_model.odb' then determines the correct
# element set.
# The program produces 2 files:
# 1. xxx_attelib.py
#    For processing the final output ('XXX_model.odb')
# 2. xxx_mymid.dat
#  For editting the input file ('XXX_model.inp')
#
# Prepared by A Susilo
# Created on: 01 Oct 2013
#=============================================================================#

from odbAccess import *

#============================#
# SELECT ONE: ('#' = disable)
#============================#
#odbName = 'BMP_wave.odb'
#odbName = 'SWP_wave.odb'

stepName = 'Step-1'
setName1 = 'SET-L25'
setName2 = 'SET-L50'

odb = openOdb(path=odbName)
step = odb.steps[stepName]
totFrame = len(step.frames)

# Creating connectivity element library of all elements
allEl = odb.rootAssembly.instances['PIPE-1'].elements
totEl = len(allEl)

# In case a node has more than 10 attached elements,
```

```
# additional 11 letters are provided.
# Now, the maximum attached elements is 20
indexGe10 = ('a','b','c','d','e','f','g','i','j','k','l')

connLib={}
for e in range(totEl):
elLab = odb.rootAssembly.instances['PIPE-1'].elements[e].label
conn = odb.rootAssembly.instances['PIPE-1'].elements[e].connectivity
lconn = len(conn)
counter = 0
for i in range(lconn):
counter = counter + 1
dumAppx = counter
sdumAppx = str(dumAppx)

if counter >= 10:
theIndx = counter-10
sdumAppx = indexGe10[theIndx]

key = str(elLab) + 'x' + sdumAppx
val = conn[i]
connLib.update({key:val})


#FIRST SET

# Creating attachment element library of the 'setName1' nodes
midSet = odb.rootAssembly.elementSets[setName1]
nodesMid = odb.rootAssembly.nodeSets[setName1]
totNodes = len(nodesMid.nodes[0])

atteLib1={}
elMid1=[] # This is for the list of element in the 'setName1'
for n in range(totNodes):
theNode = nodesMid.nodes[0][n].label
elList = []   # This is for the library of node & its attached elements in the
'setName1'

for k,v in connLib.items():
elmidLib={}
if v == theNode:
theElement = k[:-2]
elList.append(int(theElement))
elMid1.append(int(theElement))
key = str(theNode)
val = elList
```

```
atteLib1.update({key:val})

#SECOND SET

# Creating attachment element library of the 'setName2' nodes
midSet = odb.rootAssembly.elementSets[setName2]
nodesMid = odb.rootAssembly.nodeSets[setName2]
totNodes = len(nodesMid.nodes[0])

atteLib2={}
elMid2=[] # This is for the list of element in the 'setName2'
for n in range(totNodes):
theNode = nodesMid.nodes[0][n].label
elList = []   # This is for the library of node & its attached elements in the
'setName1'

for k,v in connLib.items():
elmidLib={}
if v == theNode:
theElement = k[:-2]
elList.append(int(theElement))
elMid2.append(int(theElement))
key = str(theNode)
val = elList

atteLib2.update({key:val})

# Print the attached element library

#============================#
# SELECT ONE: ('#' = disable)
#============================#
#outFile1 = open('bmp_attelib.py','w')
outFile1 = open('swp_attelib.py','w')

print >>outFile1, 'atteLib1 = ',atteLib1
print >>outFile1, 'atteLib2 = ',atteLib2
outFile1.close()

# Print the middle element list

#============================#
# SELECT ONE: ('#' = disable)
#============================#
#outFile2 = open('bmp_mymid.dat','w')
```

```
outFile2 = open('swp_mymid.dat','w')

#FIRST SET
myMid1 = tuple(sorted(set(elMid1)))
totN = len(myMid1)
repeatP = totN//16

n0 = 0
n1 = 16

# Print to file with 'print' on Python 2.5 or earlier
# 16 is number of column of the element set on the INP file

print >>outFile2, setName1
for x in range(repeatP):
print >>outFile2, myMid1[n0:n1]
n0 = n0 + 16
n1 = n1 + 16
print >>outFile2, myMid1[n0:totN]

print >>outFile2, setName2

#SECOND SET
myMid2 = tuple(sorted(set(elMid2)))
totN = len(myMid2)
repeatP = totN//16

n0 = 0
n1 = 16

# Print to file with 'print' on Python 2.5 or earlier
# 16 is number of column of the element set on the INP file
for x in range(repeatP):
print >>outFile2, myMid2[n0:n1]
n0 = n0 + 16
n1 = n1 + 16
print >>outFile2, myMid2[n0:totN]

outFile2.close()

print 'DONE'
```

A7. bmp/swp_mymid.dat[3]

```
(1, 2, 3, 4, 11307, 11308, 11309, 11310, 22613, 22614, 22906, 22907, 22908, 22909,
22910, 22940)
(22941, 23087, 23088, 23372, 23375, 23376, 23377, 23378, 23379, 23380, 23381,
23382, 23383, 23384, 23385, 23386)
...
...
...
(108625, 108626, 108627, 108628, 158049, 158050, 158061, 158062, 158063, 158064,
158065, 158066, 158067, 158068, 158069, 158070)
(158071, 158072, 158073, 158074, 158075, 158076, 158077, 158078, 158079, 158080,
158081, 158082, 158083)
```

[3]There are 16 columns of number.

A8. bmp-swp_attelib.py

```
atteLib1 =  {'65': [3069, 44, 3020, 43, 3019, 63], '66': [83, 64, 2969, 2970,
3019, 63], '67': [103, 2920, 83, 84, 2969, 2919], '68': [103, 2869, 104, 123,
2870, 2919], '69': [2820, 124, 2819, 2869, 123, 143], '977': [8188, 8189, 6990,
6992, 8187, 6991], '976': [8189, 8191, 6994, 6993, 6992, 8190], '975': [6996,
8191, 8192, 6994, 8193, 6995], '974': [6996, 8195, 6997, 8193, 6998, 8194], ...
... - not all attached elements shown here -

atteLib2 =  {'1106': [112079, 8696, 67727, 111954, 8694, 8695], '1033': [7399,
7419, 7400, 68607, 68594, 68614, 82200], '98': [68773, 505, 112060, 506, 486,
68786], '20': [8020, 8740, 68467, 8019, 111944], '21': [14293, 67684, 8683, 8684,
14294, 8060], '1038': [81813, 7519, 112092, 7500, 81812, 7499, 82176], '1039':
[81813, 7539, 7519, 68593, 7520], ...
... - not all attached elements shown here -
```

A9. Not completed element set (BMP)

```
====================================
ORIGINAL: #Node = 112  #Element = 224
====================================

*Nset, nset=Set-L25, instance=Pipe-1
   5,   7, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202
 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218
 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234
 235, 236, 237, 238, 239, 240, 241, 242, 243, 425, 426, 427, 428, 429, 430, 431
 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447
 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463
 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479
*Elset, elset=Set-L25, instance=Pipe-1
 1010, 1046, 1082, 1118, 1154, 1190, 1226, 1262, 1298, 1334, 1370, 1406, 1442,
1478, 1514, 1550
 1586, 1622, 1658, 1694, 1730, 1766, 1802, 1838, 1874, 1910, 1946, 1982, 2018,
2054, 2090, 2126
 2162, 2198, 2234, 2270, 2306, 2342, 2378, 2414, 2450, 2486, 2522, 2558, 2594,
2630, 2666, 2702
 2738, 2774, 2810, 2846, 2882, 2918, 2954, 2990, 3041, 3059, 3077, 3095, 3113,
3131, 3149, 3167
 3185, 3203, 3221, 3239, 3257, 3275, 3293, 3311, 3329, 3347, 3365, 3383, 3401,
3419, 3437, 3455
 3473, 3491, 3509, 3527, 3545, 3563, 3581, 3599, 3617, 3635, 3653, 3671, 3689,
3707, 3725, 3743
 3761, 3779, 3797, 3815, 3833, 3851, 3869, 3887, 3905, 3923, 3941, 3959, 3977,
3995, 4013, 4031
 5076, 5112, 5148, 5184, 5220, 5256, 5292, 5328, 5364, 5400, 5436, 5472, 5508,
5544, 5580, 5616
 5652, 5688, 5724, 5760, 5796, 5832, 5868, 5904, 5940, 5976, 6012, 6048, 6084,
6120, 6156, 6192
 6228, 6264, 6300, 6336, 6372, 6408, 6444, 6480, 6516, 6552, 6588, 6624, 6660,
6696, 6732, 6768
 6804, 6840, 6876, 6912, 6948, 6984, 7020, 7056, 7057, 7075, 7093, 7111, 7129,
7147, 7165, 7183
 7201, 7219, 7237, 7255, 7273, 7291, 7309, 7327, 7345, 7363, 7381, 7399, 7417,
7435, 7453, 7471
 7489, 7507, 7525, 7543, 7561, 7579, 7597, 7615, 7633, 7651, 7669, 7687, 7705,
7723, 7741, 7759
 7777, 7795, 7813, 7831, 7849, 7867, 7885, 7903, 7921, 7939, 7957, 7975, 7993,
8011, 8029, 8047
*Nset, nset=Set-R25, instance=Pipe-1
```

```
==================================
EDITED: #Node = 112  #Element = 448
==================================

*Nset, nset=Set-L25, instance=Pipe-1
   5,   7, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202
 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218
 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234
 235, 236, 237, 238, 239, 240, 241, 242, 243, 425, 426, 427, 428, 429, 430, 431
 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447
 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463
 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479
*Elset, elset=Set-L25, instance=Pipe-1
1009, 1010, 1045, 1046, 1081, 1082, 1117, 1118, 1153, 1154, 1189, 1190, 1225,
1226, 1261, 1262
1297, 1298, 1333, 1334, 1369, 1370, 1405, 1406, 1441, 1442, 1477, 1478, 1513,
1514, 1549, 1550
1585, 1586, 1621, 1622, 1657, 1658, 1693, 1694, 1729, 1730, 1765, 1766, 1801,
1802, 1837, 1838
1873, 1874, 1909, 1910, 1945, 1946, 1981, 1982, 2017, 2018, 2053, 2054, 2089,
2090, 2125, 2126
2161, 2162, 2197, 2198, 2233, 2234, 2269, 2270, 2305, 2306, 2341, 2342, 2377,
2378, 2413, 2414
2449, 2450, 2485, 2486, 2521, 2522, 2557, 2558, 2593, 2594, 2629, 2630, 2665,
2666, 2701, 2702
2737, 2738, 2773, 2774, 2809, 2810, 2845, 2846, 2881, 2882, 2917, 2918, 2953,
2954, 2989, 2990
3041, 3042, 3059, 3060, 3077, 3078, 3095, 3096, 3113, 3114, 3131, 3132, 3149,
3150, 3167, 3168
3185, 3186, 3203, 3204, 3221, 3222, 3239, 3240, 3257, 3258, 3275, 3276, 3293,
3294, 3311, 3312
3329, 3330, 3347, 3348, 3365, 3366, 3383, 3384, 3401, 3402, 3419, 3420, 3437,
3438, 3455, 3456
3473, 3474, 3491, 3492, 3509, 3510, 3527, 3528, 3545, 3546, 3563, 3564, 3581,
3582, 3599, 3600
3617, 3618, 3635, 3636, 3653, 3654, 3671, 3672, 3689, 3690, 3707, 3708, 3725,
3726, 3743, 3744
3761, 3762, 3779, 3780, 3797, 3798, 3815, 3816, 3833, 3834, 3851, 3852, 3869,
3870, 3887, 3888
3905, 3906, 3923, 3924, 3941, 3942, 3959, 3960, 3977, 3978, 3995, 3996, 4013,
4014, 4031, 4032
5075, 5076, 5111, 5112, 5147, 5148, 5183, 5184, 5219, 5220, 5255, 5256, 5291,
5292, 5327, 5328
5363, 5364, 5399, 5400, 5435, 5436, 5471, 5472, 5507, 5508, 5543, 5544, 5579,
5580, 5615, 5616
5651, 5652, 5687, 5688, 5723, 5724, 5759, 5760, 5795, 5796, 5831, 5832, 5867,
```

```
5868, 5903, 5904
5939, 5940, 5975, 5976, 6011, 6012, 6047, 6048, 6083, 6084, 6119, 6120, 6155,
6156, 6191, 6192
6227, 6228, 6263, 6264, 6299, 6300, 6335, 6336, 6371, 6372, 6407, 6408, 6443,
6444, 6479, 6480
6515, 6516, 6551, 6552, 6587, 6588, 6623, 6624, 6659, 6660, 6695, 6696, 6731,
6732, 6767, 6768
6803, 6804, 6839, 6840, 6875, 6876, 6911, 6912, 6947, 6948, 6983, 6984, 7019,
7020, 7055, 7056
7057, 7058, 7075, 7076, 7093, 7094, 7111, 7112, 7129, 7130, 7147, 7148, 7165,
7166, 7183, 7184
7201, 7202, 7219, 7220, 7237, 7238, 7255, 7256, 7273, 7274, 7291, 7292, 7309,
7310, 7327, 7328
7345, 7346, 7363, 7364, 7381, 7382, 7399, 7400, 7417, 7418, 7435, 7436, 7453,
7454, 7471, 7472
7489, 7490, 7507, 7508, 7525, 7526, 7543, 7544, 7561, 7562, 7579, 7580, 7597,
7598, 7615, 7616
7633, 7634, 7651, 7652, 7669, 7670, 7687, 7688, 7705, 7706, 7723, 7724, 7741,
7742, 7759, 7760
7777, 7778, 7795, 7796, 7813, 7814, 7831, 7832, 7849, 7850, 7867, 7868, 7885,
7886, 7903, 7904
7921, 7922, 7939, 7940, 7957, 7958, 7975, 7976, 7993, 7994, 8011, 8012, 8029,
8030, 8047, 8048
*Nset, nset=Set-R25, instance=Pipe-1
```

A10. BMP_model.inp (edited)

```
*Heading
** Job name: BMP_model Model name: Benchmark Pipe
** Generated by: Abaqus/CAE 6.12-2
*Preprint, echo=NO, model=NO, history=NO, contact=NO
**
** PARTS
**
*Part, name=Pipe
*End Part
**
**
** ASSEMBLY
**
*Assembly, name=Assembly
**
*Instance, name=Pipe-1, part=Pipe
*Node
      1,           0., -0.445769995,   5.72288513
      2,           0., -0.445769995,   5.94576979
      3,           0.,  0.445769995,   5.94576979
...
...
...
  45423, 0.0208802875,  0.445280701,   7.73659325
  45424, -0.025456138,  0.445042551,   7.74229479
*Element, type=S3
   1,    1,   30, 1107
   2, 1107,  155,    1
   3,   30,   31, 1108
...
...
...
90623, 45424, 27001,    29
90624, 45424,    29, 45306
*Nset, nset=Set-Section, generate
     1,  45424,      1
*Elset, elset=Set-Section, generate
     1,  90624,      1
*Nset, nset=_PickedSet4, internal, generate
     1,  45424,      1
*Elset, elset=_PickedSet4, internal, generate
     1,  90624,      1
*Orientation, name=Ori-1, system=CYLINDRICAL
          0.,           0.,           0.,           0.,           0.,
```

```
   1.
1, 0.
** Section: Section-Pipe
*Shell Section, elset=Set-Section, material=Material-Pipe, orientation=Ori-1,
offset=SPOS
0.01143, 9
*End Instance
**
*Node
      1,           0.,           0.,  -1.20000005
*Node
      2,           0.,           0.,   12.1999998
*Nset, nset=_PickedSet13, internal
 1,
*Nset, nset=_PickedSet14, internal, instance=Pipe-1
   10,   11,  535,  536,  537,  538,  539,  540,  541,  542,  543,  544,  545,
 546,  547,  548
...
...
...
 6385, 6386, 6387, 6388, 6389, 6390, 6391, 6392, 6393, 6394, 6395, 6396, 6397,
6398, 6399, 6400
 6401, 6402, 6403, 6404, 6405, 6406, 6407, 6408, 6409, 6410, 6411, 6412, 6413,
6414, 6415, 6416
*Elset, elset=_PickedSet14, internal, instance=Pipe-1, generate
  8065,  11648,      1
*Nset, nset=_PickedSet15, internal
 2,
*Nset, nset=_PickedSet16, internal, instance=Pipe-1
   12,   13,  757,  758,  759,  760,  761,  762,  763,  764,  765,  766,  767,
 768,  769,  770
...
...
...
 8065, 8066, 8067, 8068, 8069, 8070, 8071, 8072, 8073, 8074, 8075, 8076, 8077,
8078, 8079, 8080
 8081, 8082, 8083, 8084, 8085, 8086, 8087, 8088, 8089, 8090, 8091, 8092, 8093,
8094, 8095, 8096
*Elset, elset=_PickedSet16, internal, instance=Pipe-1, generate
 11649,  15232,      1
*Nset, nset=_PickedSet18, internal
 1,
*Nset, nset=_PickedSet19, internal
 2,
*Nset, nset=_PickedSet20, internal
 1,
```

```
*Nset, nset=_PickedSet21, internal
 2,
*Nset, nset=Set-Top, instance=Pipe-1
  3,  4,  6,  7,  9, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
 25, 26, 27, 28, 29
*Nset, nset=Set-L50, instance=Pipe-1
   8,   9, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273
 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289
 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305
 306, 307, 308, 309, 310, 311, 312, 313, 314, 480, 481, 482, 483, 484, 485, 486
 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502
 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518
 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534
*Elset, elset=Set-L50, instance=Pipe-1
3025, 3026, 3043, 3044, 3061, 3062, 3079, 3080, 3097, 3098, 3115, 3116, 3133,
3134, 3151, 3152
...
...
...
52625, 52626, 52627, 52628, 52629, 52630, 52631, 52632, 52649, 52650, 52651,
52652, 52653, 52654, 52655, 52656
52657, 52658, 52659, 52660, 52661, 52662, 52663, 52664, 52665, 52666, 52741
*Nset, nset=Set-L25, instance=Pipe-1
   5,   7, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202
 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218
 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234
 235, 236, 237, 238, 239, 240, 241, 242, 243, 425, 426, 427, 428, 429, 430, 431
 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447
 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463
 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479
*Elset, elset=Set-L25, instance=Pipe-1
1009, 1010, 1045, 1046, 1081, 1082, 1117, 1118, 1153, 1154, 1189, 1190, 1225,
1226, 1261, 1262
...
...
...
7777, 7778, 7795, 7796, 7813, 7814, 7831, 7832, 7849, 7850, 7867, 7868, 7885,
7886, 7903, 7904
7921, 7922, 7939, 7940, 7957, 7958, 7975, 7976, 7993, 7994, 8011, 8012, 8029,
8030, 8047, 8048
*Nset, nset=Set-R25, instance=Pipe-1
   1,   4, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114
 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130
 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146
 147, 148, 149, 150, 151, 152, 153, 154, 155, 315, 316, 317, 318, 319, 320, 321
 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337
```

```
 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353
 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369
*Elset, elset=Set-R25, instance=Pipe-1
    2,   20,   38,   56,   74,   92,  110,  128,  146,  164,  182,  200,  218,
 236,  254,  272
...
...
...
 5905, 5941, 5977, 6013, 6049, 6085, 6121, 6157, 6193, 6229, 6265, 6301, 6337,
6373, 6409, 6445
 6481, 6517, 6553, 6589, 6625, 6661, 6697, 6733, 6769, 6805, 6841, 6877, 6913,
6949, 6985, 7021
*Nset, nset=Set-R50, instance=Pipe-1
   2,   3,  38,  39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,
51
  52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,  65,  66,
67
  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,  78,  79,  80,  81,  82,
83
  84,  85,  86,  87,  88,  89,  90,  91,  92, 370, 371, 372, 373, 374, 375, 376
 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392
 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408
 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424
*Elset, elset=Set-R50, instance=Pipe-1
    17,    35,    53,    71,    89,   107,   125,   143,   161,   179,   197,
  215,   233,   251,   269,   287
...
...
...
 90278, 90279, 90280, 90281, 90282, 90283, 90284, 90285, 90286, 90287, 90288,
90289, 90290, 90291, 90292, 90293
 90294, 90295, 90296, 90297, 90298, 90299, 90300, 90301, 90302, 90303, 90304,
90305, 90306, 90307, 90308, 90309
*Nset, nset=Set-T1, instance=Pipe-1
 9,
*Nset, nset=Set-T2, instance=Pipe-1
 7,
*Nset, nset=Set-B1, instance=Pipe-1
 8,
*Nset, nset=Set-B2, instance=Pipe-1
 5,
*Elset, elset=__PickedSurf17_SNEG, internal, instance=Pipe-1, generate
     1,  90624,      1
*Surface, type=ELEMENT, name=_PickedSurf17, internal
__PickedSurf17_SNEG, SNEG
** Constraint: Constraint-1
```

```
*Rigid Body, ref node=_PickedSet13, elset=_PickedSet14
** Constraint: Constraint-2
*Rigid Body, ref node=_PickedSet15, elset=_PickedSet16
*End Assembly
**
**
*IMPERFECTION, INPUT=BMP_imper.dat
**
** MATERIALS
**
*Material, name=Material-Pipe
*Elastic
 2.05e+11, 0.3
*Plastic
   4.305e+08,          0.
 4.33125e+08,    1.22e-05
  4.3575e+08,    2.44e-05
 4.38375e+08, 3.66034e-05
    4.41e+08,  4.8808e-05
 4.43625e+08,  6.1015e-05
  4.4625e+08, 7.32251e-05
 4.48875e+08, 8.54391e-05
   4.515e+08, 9.76579e-05
 4.54125e+08, 0.000109883
  4.5675e+08, 0.000122115
 4.59375e+08, 0.000134357
    4.62e+08,  0.00014661
 4.64625e+08, 0.000158878
  4.6725e+08, 0.000171164
 4.69875e+08, 0.000183472
   4.725e+08, 0.000195807
 4.75125e+08, 0.000208176
  4.7775e+08, 0.000220587
 4.80375e+08,  0.00023305
    4.83e+08, 0.000245576
 4.85625e+08, 0.000258181
  4.8825e+08, 0.000270882
 4.90875e+08,   0.0002837
   4.935e+08, 0.000296663
 4.96125e+08, 0.000309803
  4.9875e+08,  0.00032316
 5.01375e+08, 0.000336781
    5.04e+08, 0.000350725
 5.06625e+08, 0.000365063
  5.0925e+08, 0.000379881
 5.11875e+08, 0.000395283
```

```
  5.145e+08, 0.000411395
5.17125e+08, 0.000428369
 5.1975e+08, 0.000446389
5.22375e+08, 0.000465677
   5.25e+08, 0.000486501
5.27625e+08, 0.000509182
 5.3025e+08, 0.000534109
5.32875e+08, 0.000561747
  5.355e+08, 0.000592655
5.38125e+08, 0.000627506
 5.4075e+08, 0.000667103
5.43375e+08, 0.000712412
   5.46e+08, 0.000764585
5.48625e+08, 0.000825002
 5.5125e+08, 0.000895311
5.53875e+08, 0.000977479
  5.565e+08,  0.00107385
5.59125e+08,  0.00118722
 5.6175e+08,  0.00132093
5.64375e+08,  0.00147891
   5.67e+08,   0.0016659
5.69625e+08,  0.00188747
 5.7225e+08,  0.00215026
5.74875e+08,  0.00246213
  5.775e+08,  0.00283242
5.80125e+08,  0.00327216
 5.8275e+08,  0.00379441
5.85375e+08,   0.0044146
   5.88e+08,  0.00515099
5.90625e+08,  0.00602509
 5.9325e+08,  0.00706232
5.95875e+08,  0.00829259
  5.985e+08,  0.00975117
6.01125e+08,   0.0114796
 6.0375e+08,   0.0135266
6.06375e+08,   0.0159496
   6.09e+08,    0.018816
6.11625e+08,   0.0222048
 6.1425e+08,   0.0262088
6.16875e+08,   0.0309368
  6.195e+08,    0.036516
6.22125e+08,   0.0430955
 6.2475e+08,   0.0508496
6.27375e+08,   0.0599821
    6.3e+08,   0.0707311
6.32625e+08,   0.0833745
```

```
  6.3525e+08,   0.0982365
 6.37875e+08,    0.115695
   6.405e+08,    0.136191
 6.43125e+08,    0.160237
  6.4575e+08,    0.188429
 6.48375e+08,    0.221462
*Potential
 0.981,1.,1.,1.,1.,1.
** ----------------------------------------------------------------
**
** STEP: Step-1
**
*Step, name=Step-1, nlgeom=YES, inc=103
*Static, riks
0.05,1.0,0.00000000000000005,0.025,
**
** BOUNDARY CONDITIONS
**
** Name: BC-1 Type: Displacement/Rotation
*Boundary
_PickedSet20, 1, 1
_PickedSet20, 2, 2
** Name: BC-2 Type: Displacement/Rotation
*Boundary
_PickedSet21, 1, 1
_PickedSet21, 2, 2
_PickedSet21, 3, 3
**
** LOADS
**
** Name: Moment-1   Type: Moment
*Cload
_PickedSet18, 4, 1.4e+07
** Name: Moment-2   Type: Moment
*Cload
_PickedSet19, 4, -1.4e+07
** Name: Pressure   Type: Pressure
*Dsload
_PickedSurf17, P, 1.13143e+07
**
** OUTPUT REQUESTS
**
*Restart, write, frequency=0
**
** FIELD OUTPUT: F-Output-5
**
```

```
*Output, field
*Node Output, nset=Set-B1
COORD, UR
**
** FIELD OUTPUT: F-Output-6
**
*Node Output, nset=Set-B2
COORD, UR
**
** FIELD OUTPUT: F-Output-7
**
*Node Output, nset=Set-L25
COORD, UR
*Element Output, elset=Set-L25, directions=YES
S, SE, SF
**
** FIELD OUTPUT: F-Output-8
**
*Node Output, nset=Set-L50
COORD, UR
*Element Output, elset=Set-L50, directions=YES
S, SE, SF
**
** FIELD OUTPUT: F-Output-3
**
*Node Output, nset=Set-T1
COORD, UR
**
** FIELD OUTPUT: F-Output-4
**
*Node Output, nset=Set-T2
COORD, UR
**
** FIELD OUTPUT: F-Output-2
**
*Node Output, nset=Set-Top
COORD,
**
** FIELD OUTPUT: F-Output-1
**
*Output, field, variable=PRESELECT
**
** HISTORY OUTPUT: H-Output-1
**
*Output, history, variable=PRESELECT
*End Step
```

A11. Not completed element set (SWP)

```
====================================
ORIGINAL: #Node = 106  #Element = 212
====================================

*Nset, nset=Set-L25, instance=Pipe-1
    6,    7,   16,   19,   65,   66,   67,   68,   69,   70,   71,   72,   73,
  74,   75,   76
   77,   78,   79,   80,   81,   82,   83,   84,   85,   86,   87,   88,  790,
 791,  792,  793
  794,  795,  796,  797,  798,  799,  800,  801,  802,  803,  804,  805,  806,
 807,  808,  809
  810,  811,  812,  813,  814,  815,  816,  817,  818,  819,  820,  821,  822,
 823,  824,  825
  826,  827,  828,  829,  830,  831,  832,  833,  834,  835,  836,  837,  838,
 963,  964,  965
  966,  967,  968,  969,  970,  971,  972,  973,  974,  975,  976,  977,  978,
 979,  980,  981
  982,  983,  984,  985,  986,  987,  988,  989,  990, 1073
*Elset, elset=Set-L25, instance=Pipe-1
   43,   63,   83,  103,  123,  143,  163,  183,  203,  223,  243,  263,  283,
 303,  323,  343
  363,  383,  403,  423,  443,  463,  483,  503,  523, 1869, 1919, 1969, 2019,
2069, 2119, 2169
 ...
 ...
 7020, 7022, 7042, 7062, 7082, 7102, 7122, 7142, 7162, 7182, 7202, 7222, 7242,
7262, 7282, 7302
 7322, 7342, 7362, 7382, 7402, 7422, 7442, 7462, 7482, 7502, 7522, 7542, 7562,
7582, 7602, 7622
 7642, 7662, 7682, 7702, 7722, 7742, 7762, 7782, 7802, 7822, 7842, 7862, 7882,
7902, 7922, 7942
 7962, 7982, 8002, 8021, 8023, 8158, 8160, 8161, 8163, 8165, 8167, 8169, 8171,
8173, 8175, 8177
 8179, 8181, 8183, 8185, 8187, 8189, 8191, 8193, 8195, 8197, 8199, 8201, 8203,
8205, 8207, 8209
 8211, 8213, 8215, 8217
*Nset, nset=Set-R25, instance=Pipe-1
\newpage

==================================
EDITED: #Node = 106  #Element = 424
==================================

*Nset, nset=Set-L25, instance=Pipe-1
```

```
    6,    7,   16,   19,   65,   66,   67,   68,   69,   70,   71,   72,   73,
  74,   75,   76
   77,   78,   79,   80,   81,   82,   83,   84,   85,   86,   87,   88,  790,
 791,  792,  793
  794,  795,  796,  797,  798,  799,  800,  801,  802,  803,  804,  805,  806,
 807,  808,  809
  810,  811,  812,  813,  814,  815,  816,  817,  818,  819,  820,  821,  822,
 823,  824,  825
  826,  827,  828,  829,  830,  831,  832,  833,  834,  835,  836,  837,  838,
 963,  964,  965
  966,  967,  968,  969,  970,  971,  972,  973,  974,  975,  976,  977,  978,
 979,  980,  981
  982,  983,  984,  985,  986,  987,  988,  989,  990, 1073
*Elset, elset=Set-L25, instance=Pipe-1
43, 44, 63, 64, 83, 84, 103, 104, 123, 124, 143, 144, 163, 164, 183, 184
203, 204, 223, 224, 243, 244, 263, 264, 283, 284, 303, 304, 323, 324, 343, 344
363, 364, 383, 384, 403, 404, 423, 424, 443, 444, 463, 464, 483, 484, 503, 504
523, 524, 1869, 1870, 1919, 1920, 1969, 1970, 2019, 2020, 2069, 2070, 2119, 2120,
2169, 2170
2219, 2220, 2269, 2270, 2319, 2320, 2369, 2370, 2419, 2420, 2469, 2470, 2519,
2520, 2569, 2570
2619, 2620, 2669, 2670, 2719, 2720, 2769, 2770, 2819, 2820, 2869, 2870, 2919,
2920, 2969, 2970
3019, 3020, 3069, 3070, 3119, 3120, 3169, 3170, 3219, 3220, 3269, 3270, 3319,
3320, 3369, 3370
3419, 3420, 3469, 3470, 3519, 3520, 3569, 3570, 3619, 3620, 3669, 3670, 3719,
3720, 3769, 3770
3819, 3820, 3869, 3870, 3919, 3920, 3969, 3970, 4019, 4020, 4069, 4070, 4119,
4120, 4169, 4170
4219, 4220, 4269, 4270, 4319, 4320, 4369, 4370, 4419, 4420, 4469, 4470, 4519,
4520, 4569, 4570
4619, 4620, 4669, 4670, 4719, 4720, 4769, 4770, 4819, 4820, 4869, 4870, 4919,
4920, 4969, 4970
5019, 5020, 5069, 5070, 5119, 5120, 5169, 5170, 5219, 5220, 5269, 5270, 5319,
5320, 5369, 5370
5419, 5420, 5469, 5470, 5519, 5520, 5569, 5570, 6963, 6964, 6965, 6966, 6967,
6968, 6969, 6970
6971, 6972, 6973, 6974, 6975, 6976, 6977, 6978, 6979, 6980, 6981, 6982, 6983,
6984, 6985, 6986
6987, 6988, 6989, 6990, 6991, 6992, 6993, 6994, 6995, 6996, 6997, 6998, 6999,
7000, 7001, 7002
7003, 7004, 7005, 7006, 7007, 7008, 7009, 7010, 7011, 7012, 7013, 7014, 7015,
7016, 7017, 7018
7019, 7020, 7021, 7022, 7041, 7042, 7061, 7062, 7081, 7082, 7101, 7102, 7121,
7122, 7141, 7142
7161, 7162, 7181, 7182, 7201, 7202, 7221, 7222, 7241, 7242, 7261, 7262, 7281,
```

```
7282, 7301, 7302
7321, 7322, 7341, 7342, 7361, 7362, 7381, 7382, 7401, 7402, 7421, 7422, 7441,
7442, 7461, 7462
7481, 7482, 7501, 7502, 7521, 7522, 7541, 7542, 7561, 7562, 7581, 7582, 7601,
7602, 7621, 7622
7641, 7642, 7661, 7662, 7681, 7682, 7701, 7702, 7721, 7722, 7741, 7742, 7761,
7762, 7781, 7782
7801, 7802, 7821, 7822, 7841, 7842, 7861, 7862, 7881, 7882, 7901, 7902, 7921,
7922, 7941, 7942
7961, 7962, 7981, 7982, 8001, 8002, 8021, 8022, 8023, 8024, 8157, 8158, 8159,
8160, 8161, 8162
8163, 8164, 8165, 8166, 8167, 8168, 8169, 8170, 8171, 8172, 8173, 8174, 8175,
8176, 8177, 8178
8179, 8180, 8181, 8182, 8183, 8184, 8185, 8186, 8187, 8188, 8189, 8190, 8191,
8192, 8193, 8194
8195, 8196, 8197, 8198, 8199, 8200, 8201, 8202, 8203, 8204, 8205, 8206, 8207,
8208, 8209, 8210
8211, 8212, 8213, 8214, 8215, 8216, 8217, 8218
*Nset, nset=Set-R25, instance=Pipe-1
```

A12. SWP_model.inp (edited)

```
*Heading
** Job name: SWP1_model Model name: Spiral Welded Pipe
** Generated by: Abaqus/CAE 6.12-2
*Preprint, echo=NO, model=NO, history=NO, contact=NO
**
** PARTS
**
*Part, name=Pipe
*End Part
**
**
** ASSEMBLY
**
*Assembly, name=Assembly
**
*Instance, name=Pipe-1, part=Pipe
*Node
      1,           0., -0.445769995,   5.88201952
      2,           0., -0.445769995,   5.86896467
      3, 0.0908911228,  -0.43640542,   5.94576979
...
...
...
  56250,  0.429572344,  0.119073518,   5.03454542
  56251,  0.128569379, -0.426826447,   5.03838968
*Element, type=S3
 1,    1,   44,   55
 2,   44, 2414,   55
 3,   44,    2, 2414
...
...
...
112275,  1019, 35373,  1020
112276, 56251,  1019,  1018
*Nset, nset=Set-1-shell
     1,     2,     3,     4,     5,     6,     7,     8,     9,    10,    11,
   12,    13,    14,    15,    16
...
...
...
 56234, 56235, 56236, 56237, 56238, 56239, 56240, 56241, 56242, 56243, 56244,
56245, 56246, 56247, 56248, 56249
 56250, 56251
```

```
*Elset, elset=Set-1-shell
     25,     26,     27,     28,     29,     30,     31,     32,     33,
34,     35,     36,     37,     38,     39,     40
...
...
...
 112257, 112258, 112259, 112260, 112261, 112262, 112263, 112264, 112265, 112266,
112267, 112268, 112269, 112270, 112271, 112272
 112273, 112274, 112275, 112276
*Nset, nset=Set-2-weld
    1,    2,    3,    4,    7,    8,    9,   10,   11,   12,   13,   14,   19,
  21,   22,   23
...
...
...
 9058, 9059, 9060, 9061, 9062, 9063, 9064, 9065, 9066, 9067, 9068, 9069, 9070,
9071, 9072, 9073
 9074, 9075, 9076, 9077, 9078, 9079, 9080, 9081, 9082, 9083, 9084
*Elset, elset=Set-2-weld
     1,     2,     3,     4,     5,     6,     7,     8,     9,    10,    11,
   12,    13,    14,    15,    16
...
...
...
 15519, 15520, 15521, 15522, 15523, 15524, 15525, 15526, 15527, 15528, 15529,
15530, 15531, 15532, 15533, 15534
 15535, 15536, 15537, 15538, 15539, 15540, 15541, 15542, 15543, 15544, 15545,
15546, 15547, 15548, 15549, 15550
*Nset, nset=_PickedSet5, internal, generate
     1,  56251,      1
*Elset, elset=_PickedSet5, internal, generate
      1,  112276,       1
*Elset, elset=_Surf-Pressure_SNEG, internal, generate
      1,  112276,       1
*Surface, type=ELEMENT, name=Surf-Pressure
_Surf-Pressure_SNEG, SNEG
*Orientation, name=Ori-1, system=CYLINDRICAL
          0.,           0.,           0.,           0.,           0.,
   1.
1, 0.
** Section: Section-Weld
*Shell Section, elset=Set-2-weld, material=Material-Weld, orientation=Ori-1,
offset=SPOS
0.01143, 9
** Section: Section-Shell
*Shell Section, elset=Set-1-shell, material=Material-Shell, orientation=Ori-1,
```

```
offset=SPOS
0.01143, 9
*End Instance
**
*Node
      1,           0.,           0.,  -1.20000005
*Node
      2,           0.,           0.,   12.1999998
*Nset, nset=_PickedSet13, internal
 1,
*Nset, nset=_PickedSet14, internal, instance=Pipe-1
   22,   23,   24,   25, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1120,
1121, 1122, 1123
...
...
...
 8739, 8740, 8741, 8742, 8743, 8744, 8745, 8746, 8747, 8748, 8749, 8750, 8751,
8752, 8753, 8754
 8755, 8756, 8757, 8758, 8759, 8760, 8761, 8762, 8763, 8764, 8765, 8766, 8767,
8768, 8769, 8770
*Elset, elset=_PickedSet14, internal, instance=Pipe-1, generate
  8741,  14290,      1
*Nset, nset=_PickedSet15, internal
 2,
*Nset, nset=_PickedSet16, internal, instance=Pipe-1
    11,    12,    13,    14,   428,   429,   430,   431,   432,   433,   434,
  435,   436,   437,   438,   439
...
...
...
 11693, 11694, 11695, 11696, 11697, 11698, 11699, 11700, 11701, 11702, 11703,
11704, 11705, 11706, 11707, 11708
 11709, 11710, 11711, 11712, 11713, 11714, 11715, 11716, 11717, 11718, 11719,
11720, 11721, 11722, 11723, 11724
*Elset, elset=_PickedSet16, internal, instance=Pipe-1, generate
 15551,  21100,      1
*Nset, nset=_PickedSet17, internal
 1,
*Nset, nset=_PickedSet18, internal
 2,
*Nset, nset=_PickedSet19, internal
 1,
*Nset, nset=_PickedSet20, internal
 2,
*Nset, nset=Set-Top, instance=Pipe-1
 15, 16, 18, 20, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38
```

```
 39, 40, 41, 42, 43
*Nset, nset=Set-L50, instance=Pipe-1
    5,    8,   20,   21,   98,   99,  100,  101,  102,  103,  104,  105,  106,
 107,  108,  109
  110,  111,  112,  113,  114,  115,  116,  117,  118,  119,  120,  121, 1015,
1016, 1017, 1018
 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031,
1032, 1033, 1034
 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047,
1048, 1049, 1050
 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063,
1083, 1084, 1085
 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098,
1099, 1100, 1101
 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111
*Elset, elset=Set-L50, instance=Pipe-1
25, 26, 45, 46, 65, 66, 85, 86, 105, 106, 125, 126, 145, 146, 165, 166
...
...
...
112033, 112035, 112060, 112079, 112090, 112092, 112109, 112133, 112147, 112166,
112252, 112254, 112256, 112258, 112259, 112262
112264, 112268, 112270, 112272, 112274, 112275, 112276
*Nset, nset=Set-L25, instance=Pipe-1
    6,    7,   16,   19,   65,   66,   67,   68,   69,   70,   71,   72,   73,
  74,   75,   76
   77,   78,   79,   80,   81,   82,   83,   84,   85,   86,   87,   88,  790,
 791,  792,  793
  794,  795,  796,  797,  798,  799,  800,  801,  802,  803,  804,  805,  806,
 807,  808,  809
  810,  811,  812,  813,  814,  815,  816,  817,  818,  819,  820,  821,  822,
 823,  824,  825
  826,  827,  828,  829,  830,  831,  832,  833,  834,  835,  836,  837,  838,
 963,  964,  965
  966,  967,  968,  969,  970,  971,  972,  973,  974,  975,  976,  977,  978,
 979,  980,  981
  982,  983,  984,  985,  986,  987,  988,  989,  990, 1073
*Elset, elset=Set-L25, instance=Pipe-1
43, 44, 63, 64, 83, 84, 103, 104, 123, 124, 143, 144, 163, 164, 183, 184
...
...
...
8195, 8196, 8197, 8198, 8199, 8200, 8201, 8202, 8203, 8204, 8205, 8206, 8207,
8208, 8209, 8210
8211, 8212, 8213, 8214, 8215, 8216, 8217, 8218
*Nset, nset=Set-R25, instance=Pipe-1
```

```
   9,  10,  17,  18, 130, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872
 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888
 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904
 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920
 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936
 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952
 953, 954, 955, 956, 957, 958, 959, 960, 961, 962
*Elset, elset=Set-R25, instance=Pipe-1
   558,   560,  1822,  1872,  1922,  1972,  2022,  2072,  2122,  2172,  2222,
 2272,  2322,  2372,  2422,  2472
...
...
...
 67104, 67105, 67110, 67117, 67122, 67126, 67133, 67601, 67617, 67621, 67623,
67625, 67628, 67635, 67638, 67641
 67654, 67656, 67663, 67680
*Nset, nset=Set-R50, instance=Pipe-1
    3,    4,   26,   27,   50, 2181, 2182, 2196, 2197, 2198, 2199, 2200, 2201,
2202, 2203, 2204
 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217,
2218, 2219, 2220
 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233,
2234, 2235, 2236
 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2271, 2272, 2273,
2274, 2275, 2276
 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289,
2290, 2291, 2292
 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305,
2306, 2307, 2308
 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321,
2322, 2323, 2324
 2325,
*Elset, elset=Set-R50, instance=Pipe-1
    22,    24,   561,   563, 21101, 21102, 21104, 21189, 21190, 21191, 21192,
21312, 21313, 21334, 21345, 21356
...
...
...
 67309, 67320, 67321, 67324, 67334, 67335, 67336, 67342, 67348, 67349, 67350,
67351, 67352, 67353, 67354, 67357
 67358, 67359
*Nset, nset=Set-T1, instance=Pipe-1
 20,
*Nset, nset=Set-T2, instance=Pipe-1
 16,
*Nset, nset=Set-T3, instance=Pipe-1
```

```
 18,
*Nset, nset=Set-T4, instance=Pipe-1
 27,
*Nset, nset=Set-B1, instance=Pipe-1
 5,
*Nset, nset=Set-B2, instance=Pipe-1
 6,
*Nset, nset=Set-B3, instance=Pipe-1
 17,
*Nset, nset=Set-B4, instance=Pipe-1
 26,
** Constraint: Constraint-1
*Rigid Body, ref node=_PickedSet13, elset=_PickedSet14
** Constraint: Constraint-2
*Rigid Body, ref node=_PickedSet15, elset=_PickedSet16
*End Assembly
**
**
*IMPERFECTION, INPUT=SWP_imper.dat
**
** MATERIALS
**
*Material, name=Material-Shell
*Elastic
 2.05e+11, 0.3
*Plastic
   4.305e+08,          0.
 4.33125e+08,    1.22e-05
  4.3575e+08,    2.44e-05
 4.38375e+08, 3.66034e-05
    4.41e+08,  4.8808e-05
 4.43625e+08,  6.1015e-05
  4.4625e+08, 7.32251e-05
 4.48875e+08, 8.54391e-05
   4.515e+08, 9.76579e-05
 4.54125e+08, 0.000109883
  4.5675e+08, 0.000122115
 4.59375e+08, 0.000134357
    4.62e+08,  0.00014661
 4.64625e+08, 0.000158878
  4.6725e+08, 0.000171164
 4.69875e+08, 0.000183472
   4.725e+08, 0.000195807
 4.75125e+08, 0.000208176
  4.7775e+08, 0.000220587
 4.80375e+08,  0.00023305
```

```
   4.83e+08, 0.000245576
4.85625e+08, 0.000258181
 4.8825e+08, 0.000270882
4.90875e+08,   0.0002837
  4.935e+08, 0.000296663
4.96125e+08, 0.000309803
 4.9875e+08,  0.00032316
5.01375e+08, 0.000336781
   5.04e+08, 0.000350725
5.06625e+08, 0.000365063
 5.0925e+08, 0.000379881
5.11875e+08, 0.000395283
  5.145e+08, 0.000411395
5.17125e+08, 0.000428369
 5.1975e+08, 0.000446389
5.22375e+08, 0.000465677
   5.25e+08, 0.000486501
5.27625e+08, 0.000509182
 5.3025e+08, 0.000534109
5.32875e+08, 0.000561747
  5.355e+08, 0.000592655
5.38125e+08, 0.000627506
 5.4075e+08, 0.000667103
5.43375e+08, 0.000712412
   5.46e+08, 0.000764585
5.48625e+08, 0.000825002
 5.5125e+08, 0.000895311
5.53875e+08, 0.000977479
  5.565e+08,  0.00107385
5.59125e+08,  0.00118722
 5.6175e+08,  0.00132093
5.64375e+08,  0.00147891
   5.67e+08,   0.0016659
5.69625e+08,  0.00188747
 5.7225e+08,  0.00215026
5.74875e+08,  0.00246213
  5.775e+08,  0.00283242
5.80125e+08,  0.00327216
 5.8275e+08,  0.00379441
5.85375e+08,   0.0044146
   5.88e+08,  0.00515099
5.90625e+08,  0.00602509
 5.9325e+08,  0.00706232
5.95875e+08,  0.00829259
  5.985e+08,  0.00975117
6.01125e+08,   0.0114796
```

```
  6.0375e+08,   0.0135266
 6.06375e+08,   0.0159496
    6.09e+08,    0.018816
 6.11625e+08,   0.0222048
  6.1425e+08,   0.0262088
 6.16875e+08,   0.0309368
   6.195e+08,    0.036516
 6.22125e+08,   0.0430955
  6.2475e+08,   0.0508496
 6.27375e+08,   0.0599821
     6.3e+08,   0.0707311
 6.32625e+08,   0.0833745
  6.3525e+08,   0.0982365
 6.37875e+08,    0.115695
   6.405e+08,    0.136191
 6.43125e+08,    0.160237
  6.4575e+08,    0.188429
 6.48375e+08,    0.221462
*Potential
 0.981,1.,1.,1.,1.,1.
*Material, name=Material-Weld
*Elastic
 2.05e+11, 0.3
*Plastic
   4.305e+08,          0.
 4.76438e+08,    1.22e-05
 4.79325e+08,    2.44e-05
 4.82213e+08,    3.66e-05
   4.851e+08,    4.88e-05
 4.87988e+08,     6.1e-05
 4.90875e+08,    7.32e-05
 4.93763e+08,    8.54e-05
  4.9665e+08,    9.77e-05
 4.99538e+08, 0.000109883
 5.02425e+08, 0.000122115
 5.05313e+08, 0.000134357
   5.082e+08,  0.00014661
 5.11088e+08, 0.000158878
 5.13975e+08, 0.000171164
 5.16863e+08, 0.000183472
  5.1975e+08, 0.000195807
 5.22638e+08, 0.000208176
 5.25525e+08, 0.000220587
 5.28413e+08,  0.00023305
   5.313e+08, 0.000245576
 5.34188e+08, 0.000258181
```

```
5.37075e+08, 0.000270882
5.39963e+08,   0.0002837
 5.4285e+08, 0.000296663
5.45738e+08, 0.000309803
5.48625e+08,  0.00032316
5.51513e+08, 0.000336781
  5.544e+08, 0.000350725
5.57288e+08, 0.000365063
5.60175e+08, 0.000379881
5.63063e+08, 0.000395283
 5.6595e+08, 0.000411395
5.68838e+08, 0.000428369
5.71725e+08, 0.000446389
5.74613e+08, 0.000465677
  5.775e+08, 0.000486501
5.80388e+08, 0.000509182
5.83275e+08, 0.000534109
5.86163e+08, 0.000561747
 5.8905e+08, 0.000592655
5.91938e+08, 0.000627506
5.94825e+08, 0.000667103
5.97713e+08, 0.000712412
  6.006e+08, 0.000764585
6.03488e+08, 0.000825002
6.06375e+08, 0.000895311
6.09263e+08, 0.000977479
 6.1215e+08,  0.00107385
6.15038e+08,  0.00118722
6.17925e+08,  0.00132093
6.20813e+08,  0.00147891
  6.237e+08,   0.0016659
6.26588e+08,  0.00188747
6.29475e+08,  0.00215026
6.32363e+08,  0.00246213
 6.3525e+08,  0.00283242
6.38138e+08,  0.00327216
6.41025e+08,  0.00379441
6.43913e+08,   0.0044146
  6.468e+08,  0.00515099
6.49688e+08,  0.00602509
6.52575e+08,  0.00706232
6.55463e+08,  0.00829259
 6.5835e+08,  0.00975117
6.61238e+08,   0.0114796
6.64125e+08,   0.0135266
6.67013e+08,   0.0159496
```

```
   6.699e+08,    0.018816
 6.72788e+08,   0.0222048
 6.75675e+08,   0.0262088
 6.78563e+08,   0.0309368
  6.8145e+08,    0.036516
 6.84338e+08,   0.0430955
 6.87225e+08,   0.0508496
 6.90113e+08,   0.0599821
    6.93e+08,   0.0707311
 6.95888e+08,   0.0833745
 6.98775e+08,   0.0982365
 7.01663e+08,    0.115695
  7.0455e+08,    0.136191
 7.07438e+08,    0.160237
 7.10325e+08,    0.188429
 7.13213e+08,    0.221462
*Potential
 0.981,1.,1.,1.,1.,1.
** ----------------------------------------------------------------
**
** STEP: Step-1
**
*Step, name=Step-1, nlgeom=YES, inc=200
*Static, riks
1., 1., 1e-05, , ,
**
** BOUNDARY CONDITIONS
**
** Name: BC-1 Type: Displacement/Rotation
*Boundary
_PickedSet19, 1, 1
_PickedSet19, 2, 2
** Name: BC-2 Type: Displacement/Rotation
*Boundary
_PickedSet20, 1, 1
_PickedSet20, 2, 2
_PickedSet20, 3, 3
**
** LOADS
**
** Name: Moment-1   Type: Moment
*Cload
_PickedSet17, 4, 1.4e+07
** Name: Moment-2   Type: Moment
*Cload
_PickedSet18, 4, -1.4e+07
```

```
** Name: Pressure   Type: Pressure
*Dsload
Pipe-1.Surf-Pressure, P, 1.13143e+07
**
** OUTPUT REQUESTS
**
*Restart, write, frequency=0
**
** FIELD OUTPUT: F-Output-7
**
*Output, field
*Node Output, nset=Set-B1
COORD, UR
**
** FIELD OUTPUT: F-Output-8
**
*Node Output, nset=Set-B2
COORD, UR
**
** FIELD OUTPUT: F-Output-9
**
*Node Output, nset=Set-L25
COORD, UR
*Element Output, elset=Set-L25, directions=YES
S, SE, SF
**
** FIELD OUTPUT: F-Output-10
**
*Node Output, nset=Set-L50
COORD, UR
*Element Output, elset=Set-L50, directions=YES
S, SE, SF
**
** FIELD OUTPUT: F-Output-3
**
*Node Output, nset=Set-T1
COORD, UR
**
** FIELD OUTPUT: F-Output-4
**
*Node Output, nset=Set-T2
COORD, UR
**
** FIELD OUTPUT: F-Output-5
**
*Node Output, nset=Set-T3
```

```
COORD, UR
**
** FIELD OUTPUT: F-Output-6
**
*Node Output, nset=Set-T4
COORD, UR
**
** FIELD OUTPUT: F-Output-2
**
*Node Output, nset=Set-Top
COORD,
**
** FIELD OUTPUT: F-Output-1
**
*Output, field, variable=PRESELECT
**
** HISTORY OUTPUT: H-Output-1
**
*Output, history, variable=PRESELECT
*End Step
```

A13.1. Coor_Out.py

```
#==========================================================================#
# Coor_Out.py
# The program reads the field output from an ODB file
# and then prints the result to the file "nodecoor.out".
#
#
# Prepared by A Susilo
# Date: 02 Oct 2013
#==========================================================================#


from odbAccess import *


#=======================#
# Open files, choose one
#=======================#

odbName = 'BMP_model.odb'
#odbName = 'SWP_model.odb'


#===================#
# Setting variables
#===================#

stepName = 'Step-1'
setName = 'SET-TOP'

odb = openOdb(path=odbName)
step = odb.steps[stepName]

totFrame = len(step.frames)

nodesMid = odb.rootAssembly.nodeSets[setName]
totNodes = len(nodesMid.nodes[0])


#==============================#
# Output file name and heading
#==============================#

outFile = open('bmp_coorfield.out','w')
```

```
#outFile = open('swp_coorfield.out','w')

outFile.write('%s \n' %('# Node  |      x      |      y     |      z     |'))
outFile.write('%s \n' %('# ============================================='))

iFrame = totFrame-1

print '# Step : ',iFrame

#==================#
# Coordinate Field
#==================#
theFrame = step.frames[iFrame]
coorField = theFrame.fieldOutputs['COORD']
nodeCoor = coorField.getSubset(region=nodesMid)

# Library for field of elements
coorLib = {}

for v in nodeCoor.values:
frameVal = theFrame.frameValue
key = v.nodeLabel
val = (v.data[0], v.data[1], v.data[2])
coorLib.update({key:val})

# Sorting the coordinate library based on y-value ---> x[1][1]
# Sorting the coordinate library based on z-value ---> x[1][2]

sortList = sorted(coorLib.items(),key=lambda x: x[1][2])


#===============#
# Print Results
#===============#

for n in range(totNodes):
keyNode = str(sortList[n][0])
outFile.write('%5s %10.4E %10.4E %10.4E\n' % \
(keyNode,sortList[n][1][0],sortList[n][1][1],sortList[n][1][2]))

outFile.close()

print 'DONE'
```

A13.2. NodeField_Out.py

```
#==========================================================================#
# FIELD OUTPUT (NodeField_Out.py)
#
# This program reads the field output of an ODB file (XXX_model.odb)
# and then prints the required variables result to the file "xxx_nodefield.out".
#
# Variables needed:
# 1. COORD, UR
# 2. S, SE, SF
#
# The 1st group is node value variable, so it is just collected from ODB file.
# The 2nd group is element value variable, therefore the node value is obtained
# by extrapolating its attached elements.
# The element field are collected from elements and extrapolated to the nodes
# F_node = (SIGMA F_elements) / (Number_of_Element)
#
# The program needs 2 input files:
# 1. an ODB file, 'XXX_model.odb'
# 2. a subprogram for getting the list of required elements, 'xxx_attelib.py'
#
# Variables printed:
# * Curvature, Strain, and Moment
#
# Prepared by A Susilo
# Created on: 02 Oct 2013
#==========================================================================#


from odbAccess import *


#============#
# Open files
#============#
# Output Data Base file and Attached Element Set

#====================================#
# SELECT BWP or SWP: ('#' = disable)
#====================================#

import bmp_attelib
odbName = 'BMP_model.odb'
atteLib = bmp_attelib.atteLib2      #for SET-L50
#atteLib = bmp_attelib.atteLib1     #for SET-L25
```

```
outFile = open('bmp_nodefield.out','w')


#import swp_attelib
#odbName = 'SWP0_wave21p_mo1004.odb'
#atteLib = swp_attelib.atteLib2     #for SET-L50
#atteLib = swp_attelib.atteLib1     #for SET-L25
#outFile = open('swp_nodefield.out','w')

#======================================#


#===================#
# Setting variables
#===================#

pipeODia = 2*0.44577

stepName = 'Step-1'

setName = 'SET-L50'  #atteLib2
setTop = 'SET-T1'
setBot = 'SET-B1'

#setName = 'SET-L25'  #atteLib1
#setTop = 'SET-T2'
#setBot = 'SET-B2'


odb = openOdb(path=odbName)
step = odb.steps[stepName]

totFrame = len(step.frames)
midSet = odb.rootAssembly.elementSets[setName]
nodesMid = odb.rootAssembly.nodeSets[setName]
totNodes = len(nodesMid.nodes[0])

# The point on the top
nodeTop = odb.rootAssembly.nodeSets[setTop].nodes[0][0]
nodeTs = str(nodeTop.label)

# The point on the bottom
nodeBot = odb.rootAssembly.nodeSets[setBot].nodes[0][0]
nodeBs = str(nodeBot.label)
```

```
#==============================#
# Output file name and heading
#==============================#


outFile.write('%s \n' %('# Step  |  Curvature  |   Top Strain  | Moment'))
outFile.write('%s \n' %('# ============================================='))


# List for 'x >= 0 and y <= 0' nodes
# This list is needed to compute the neutral axis (NA)
keyXpos  = []

# Start computing all frames
for iStep in range(totFrame):

iFrame = iStep

#======================================#
# I. Sorting nodes, from top to bottom
#======================================#

# 1. The coordinate field

theFrame = step.frames[iFrame]
coorField = theFrame.fieldOutputs['COORD']
nodeCoor = coorField.getSubset(region=nodesMid)

# Library for field of elements
coorLib = {}

for v in nodeCoor.values:
frameVal = theFrame.frameValue
key = v.nodeLabel
val = (v.data[0], v.data[1], v.data[2])
coorLib.update({key:val})

# For computing the neutral axis
if v.data[0] >= 0.0 and v.data[1] <= 0.0 and iStep == 0:
keyXpos.append(key)


# Sorting the coordinate library based on y-value ---> x[1][1]
sortList = sorted(coorLib.items(),key=lambda x: x[1][1],reverse=True)
```

```
#=========================================#
# II. Read the other variables, SE, SF, S
#=========================================#


# 1. Section Strain (SE) Field
#=============================#

#-------------------------------------------------#
# SE is defined in local coordinates.
# In the model, it is defined 1 = radial, 2 = axial
#-------------------------------------------------#

stressField = step.frames[iFrame].fieldOutputs['SE']
field = stressField.getSubset(region = midSet, position=INTEGRATION_POINT)

fieldValues = field.values

# Library for field of elements

fieldLib_se2 = {}
for v in fieldValues:
key = str(v.elementLabel)
val = v.data[1]          #.data[0/1] = SE1,SE2 (radial, axial)
fieldLib_se2.update({key:val})


# 2. Section Force (SF) Field
#============================#

#-------------------------------------------------#
# SF is defined in local coordinates.
# In the model, it is defined 1 = radial, 2 = axial
#-------------------------------------------------#

stressField = step.frames[iFrame].fieldOutputs['SF']
field = stressField.getSubset(region = midSet, position=INTEGRATION_POINT)

fieldValues = field.values

# Library for field of elements

fieldLib_sf2 = {}
for v in fieldValues:
```

```
key = str(v.elementLabel)
val = v.data[1]          #.data[0/1/2] = SF1,SF2,SF3 (radial, axial, 0)
fieldLib_sf2.update({key:val})


# 3. Stress (S) Field
#====================#

#-------------------------------------------------#
# S is defined in local coordinates.
# In the model, it is defined 1 = radial, 2 = axial
#-------------------------------------------------#

stressField = step.frames[iFrame].fieldOutputs['S']
field = stressField.getSubset(region = midSet, position=INTEGRATION_POINT)

fieldValues = field.values

# Library for field of elements
# Only section point = 9 (at the top) is considered
fieldLib_s22 = {}
for v in fieldValues:
secNumber = v.sectionPoint.number
if secNumber==9:
key = str(v.elementLabel)
val = v.data[1]          #.data[0/1] = SE1,SE2 (radial, axial)
fieldLib_s22.update({key:val})



#=========================================================#
# III. Compute strain on the bottom and moment on the middle
#=========================================================#

# 1. Extrapolate the element values
#==================================#

#-----------------------------------------------------------------#
# This section finds requested nodes and computes the field of node
# from the field of attached elements
#-----------------------------------------------------------------#

# A. The attached elements list is found in variable 'atteLib' from the file
'xxx_attelib.dat'
#    - done by 'import xxx_attelib'
```

```
# B. Compute the field of the node = (SIGMA fieldAttachedElement)/(numberOfElement)

nodeLib_s22 = {}   # 2.1. Axial stress          (S22)
nodeLib_sf2 = {}   # 2.2. Axial Section force   (SF2)
nodeLib_se2 = {}   # 2.3. Axial Section strain  (SE2)

for n in range(totNodes):
theNode = nodesMid.nodes[0][n].label

fieldNode1 = 0
fieldNode2 = 0
fieldNode3 = 0
key = str(theNode)
totatte = len(atteLib[key])

for x in range(totatte):

theElement = str(atteLib[key][x])
fieldNode1 = fieldNode1 + fieldLib_s22[theElement]
fieldNode2 = fieldNode2 + fieldLib_sf2[theElement]
fieldNode3 = fieldNode3 + fieldLib_se2[theElement]

val = fieldNode1/totatte
nodeLib_s22.update({key:val})

val = fieldNode2/totatte
nodeLib_sf2.update({key:val})

val = fieldNode3/totatte
nodeLib_se2.update({key:val})


# 2. Compute the Bending Moment
#==============================#

# A. Finding the Neutral Axis

if iStep == 0:
NAy = 0.
else:
totXpos = len(keyXpos)
posve = {}
for x in range(totXpos):
k = keyXpos[x]
v = nodeLib_sf2[str(k)]
```

```
if v >= 0.0:
posve.update({k:v})

kpos1 = min(posve, key=posve.get) # getting the 1st min positive SF
del posve[kpos1]                  # deleting the mininum value
kpos2 = min(posve, key=posve.get) # getting the 2nd min positive SF

# Determine NA axis, which defined as SF = f(NA) = 0.0

ya = coorLib[kpos1][1]  # it needs key in integer
yb = coorLib[kpos2][1]  # it needs key in integer

fa = nodeLib_sf2[str(kpos1)]  # it needs key in string
fb = nodeLib_sf2[str(kpos2)]  # it needs key in string

tanagl = (fb-fa)/(yb-ya)

NAy = ya - fa/tanagl


# B. Compute BendingMoment = SUM(SF2*arm), where arm = Y-NA

moment = 0
for n in range(totNodes):
keyNode = str(sortList[n][0])
arm = abs(abs(sortList[n][1][1])-abs(NAy))
moment = moment + abs(nodeLib_sf2[keyNode])*arm

# Displacement (U) Field at Point 1
node1 = odb.rootAssembly.nodeSets['SET-T1']

displc = theFrame.fieldOutputs['U']
nodeDis = displc.getSubset(region=node1)
u1 = nodeDis.values[0].data[1]   #data[0/1/2] = ux,uy,uz

# Coordinate Point 1
nodeCoor = coorField.getSubset(region=node1)
x1 = nodeCoor.values[0].data[2]   #data[0/1/2] = x,y,z

# Displacement (U) Field at Point 2
node2 = odb.rootAssembly.nodeSets['SET-T2']

displc = theFrame.fieldOutputs['U']
nodeDis = displc.getSubset(region=node2)
u2 = nodeDis.values[0].data[1]   #data[0/1/2] = ux,uy,uz
```

```
# Coordinate Point 2
nodeCoor = coorField.getSubset(region=node2)
x2 = nodeCoor.values[0].data[2]   #data[0/1/2] = x,y,z

d2y = u2-u1
dx2 = (x2-x1)*(x2-x1)

# Strain on the top
strainTopX = nodeLib_se2[nodeTs]

# Curvature
curv = abs(d2y/dx2)

# Strain on the bottom
strainBotX = nodeLib_se2[nodeBs]

# Strain on the top (Et = KD - Eb)
strainTop = curv*pipeODia - strainBotX

#==================#
# V. Print Results
#==================#


# Just for feedback on the CAE
print 'Step         : ',iFrame
print 'Neutral Axis : ',NAy
print ' '

# Print output to the file
outFile.write('%d %10.4E %10.4E %10.4E\n' % (iStep,curv,abs(strainTop), moment))


outFile.close()

# Just for feedback on the CAE
print 'DONE'
```

A14. bmp/swp_nodefield.out

```
# Step  |  Curvature-025D  |  Curvature-050D  |  Bottom Strain | Moment
# ========================================================================
0 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00
1 8.2826E-05 8.1047E-05 9.0651E-05 2.8394E+07
2 1.6600E-04 1.6248E-04 1.8163E-04 5.6079E+07
3 2.9135E-04 2.8530E-04 3.1869E-04 9.6284E+07
4 4.8056E-04 4.7086E-04 5.2555E-04 1.5365E+08
5 7.6658E-04 7.5176E-04 8.3846E-04 2.3321E+08
...
...
...
98 5.9316E-02 6.0641E-02 3.2467E-02 2.4017E+08
99 6.0106E-02 6.1379E-02 3.2605E-02 2.3792E+08
100 6.0895E-02 6.2117E-02 3.2741E-02 2.3571E+08
```

A15. field_plot.plt

```
#=========================================================================#
# This program plots results from xxx-nodefield.out

# Prepared by A. Susilo
# Created on: 25 Feb 2013
#=========================================================================#


# SELECT ONE: ('#' = disable)
#============================#
#set xlabel "Step"
#set xlabel "Curvature"
set xlabel "Strain"

#set ylabel "Strain"
set ylabel "Moment"


# Plotting 'Strain vs Step'
#============================#
plot "bmp_nodefield.out" using 1:4 with linespoints lw 2 title 'BMP',\
     "swp_nodefield.out" using 1:4 with linespoints lw 2 title 'SWP'

# Plotting 'Moment vs Curvature'
#==============================#
#plot "bmp_nodefield.out" using 2:5 with linespoints lw 2 title 'BMP, at 0.5D
long',\
#     "swp_nodefield.out" using 2:5 with linespoints lw 2 title 'SWP, at 0.5D
long'

#plot "bmp_nodefield.out" using 3:5 with linespoints lw 2 title 'BMP, at 1.0D
long',\
#     "swp_nodefield.out" using 3:5 with linespoints lw 2 title 'SWP, at 1.0D
long'

# Plotting 'Moment vs Strain'
#============================#
plot "bmp_nodefield.out" using 4:5 with linespoints lw 2 title 'BMP',\
     "swp_nodefield.out" using 4:5 with linespoints lw 2 title 'SWP'
```