University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Offshore Technology/ Subsea Technology	Spring semester, 2011 Restricted access
Writer: Sivert Duvsete	…… (Writer's signature)
Faculty supervisor: Ove Tobias Gudmestad, University of Stavanger External supervisor(s): Pål Foss, IKM Ocean Design AS	
Title of thesis: 'Development of Screening Software Tool for Evaluation of Pipeline Lateral Buckling'	
Credits (ECTS): 30	
Key words: Buckling, Lateral Buckling, Python, Smoothing	Pages: 132 + enclosure: CD w/ LBSS.py, Survey.txt and Design.txt (the software and two text files) Stavanger, 14.06.11

Development of Screening Software Tool for Evaluation of Pipeline Lateral Buckling

Pipeline exposed to lateral buckling [20]

Master thesis at the University of Stavanger

Spring 2011

By Sivert Duvsete

I·K·M

University of Stavanger

Summary

During its lifetime a pipeline is subjected to an increase in internal pressure and temperature from its as-installed condition. These loads cause the pipeline to expand at its "free" end, and induce stresses and forces in the pipe wall. If the pipeline is not allowed to expand, this will result in buckling if the magnitude of the loads is large enough, to overcome external restraining forces.

The scope of this thesis is to develop a *Screening Software Tool for Evaluation of Pipeline Lateral Buckling*; a tool for evaluation of pipeline integrity based on survey results. The main purpose of the software is the identification and evaluation of locations that have undergone lateral buckling. This is done by first of all creating a software tool that can import text files; files such as the design route, as-laid data and survey data. With this information the software is able to print all three pipeline routes, and a visual comparison between design/as-laid and the survey route can be performed. Further, the lateral offset between design/as-laid and the survey route is plotted in order to get a better overview on locations that may have undergone global lateral buckling. From this plot the user can select a section for further investigation. The survey data is the data set to be analyzed, it is not fully accurate; noise often occurs and needs to be reduced/ eliminated. Noise reduction is done by smoothing the data set. After this smoothing is carried out and the user feels the smoothed curve is realistic; this part of the work is finalized.

With this smoothed curve the screening tool can now calculate the curvature, and multiplying it with the bending stiffness of the pipe; the result is the bending moment. This is not a fully accurate result, but it gives a good indication on where the pipeline might be operating under a high degree of utilization, and that a better and more thorough FE-analysis should be run.

The verification work on this screening tool has been applied in the program SIMLA, where self-established routes have been tested with the Lateral Buckling Screening Software.

The Lateral Buckling Screening Software (LBSS) has shown to be a good tool for the evaluation of pipeline lateral buckling. It has developed into an easy and effective tool for import of data files. And it contains a great smoothing function that makes the results realistic and sufficient to obtain a good estimate for further curvature calculations.

The LBSS has its limitations when calculating the bending moment; it only takes the horizontal position into consideration. So any effect from an uneven seabed or upheaval buckling is not included, this is why the LBSS will always give lower stress results than a FE-analysis software.

Preface

This Master thesis has been written in the spring of 2011 as the final examination before achieving my Master degree in Offshore Technology – Subsea Technology at the University of Stavanger. The thesis has been defined in cooperation with IKM Ocean Design AS, Trondheim. IKM Ocean Design AS provided office space at their location in Trondheim, where the writing, programming and analytical work of the thesis was performed. It has been an exciting and challenging phase of my life.

Through this period many people have given me useful advice and guidance to help me complete my thesis. I would like to express my gratitude to the following people;

- Professor Ove Tobias Gudmestad, my supervisor from the University of Stavanger. He has given me much valuable guidance through this whole process.

- Engineering Manager Bjørn Lunde for given me the opportunity to write my thesis in cooperation with IKM Ocean Design AS.

- Discipline Manager (pipelines) Pål Foss, my supervisor in IKM Ocean Design. He has been my go-to-guy when problems have occurred, and he has been defining the problems for me and explained them well.

- Knut Nordanger, engineer, for helping me to understand the program Python, and he has helped me through the whole programming part of the thesis.

- Per Tommy Roten, engineer, for helping me getting the program SIMLA running.

- Audun Kristoffersen, engineer, my problem solver. He has been a very useful asset during this period, with FE-analysis in SIMLA and programming in Python.

- John Bjarne Svinvik, engineer, my office partner and *oil and gas-dictionary*. He has used a lot of time explaining words and expressions I have needed help with.

Sivert Duvsete, Stavanger, 14.06.2011

Table of Contents

Abbreviations

HP/HT – High pressure / high temperature
VIV – Vortex induced vibrations
FEM – Finite element method
DNV – Det Norske Veritas
FE – Finite Element
FEA – Finite element analysis
ULS – Ultimate Limit State
SLS – Serviceability Limit State
FLS – Fatigue Limit State
ALS – Accidental Limit State
KP – Kilometer Point (along pipeline)
ROV – Remotely Operated Vehicle
GPS – Global Positioning System
EPD – Enthought Python Distribution
ID – Internal pipe diameter
TH – Pipe wall thickness
E – Modulus of elasticity (Young's modulus)
LBSS – Lateral Buckling Screening Software
BM – Bending moment

Symbols

m	Meter	m
-	Degrees of Celsius	˚C
p_o	External hydrostatic pressure	N/m^2
WD	Water depth	m
ρ	Density	Kg/m^3
g	Gravity	m/s^2
$\sigma_H = S_H$	Hoop stress	N/m^2
$\sigma_L = S_L$	Longitudinal stress	N/m^2
p_i	Internal pressure (operating pressure)	N/m^2
D_i	Internal diameter	m
D_o	Outer diameter	m
t	Wall thickness	m
α	Linear thermal expansion coefficient	1/˚C
ΔT	Temperature difference	˚C
p_Δ	Differential pressure across pipe wall	N/m^2
ν	Poisson's ratio	-
F_a	Axial force	N
A	Cross-section area of pipe	m^2
M_b	Bending moment	Nm
I	Moment of inertia	m^4
y	Distance from the pipe bottom to the centre of the pipe; $D_o/2$	m
ε_l	Longitudinal strain	-
K	Curvature	m^{-1}
R	Radius	m
y''	Second derivative of y	-
y'	First derivative of y	-

List of Figures

List of Tables

1 – Introduction

In the offshore industry fluids have to be moved in huge quantities and over long distances; water, oil, natural gas, and carbon dioxide are examples. One option to move fluids is transporting them through pipelines. A pipeline is a fixed asset with large capital costs. Once the pipeline is in place, the operation and maintenance costs are relatively small[1], which makes the pipelines a good option for transport of oil and gas from many fields.

Due to the risk of impact by fishing gear, ship anchors, etc, pipelines should be buried under the seabed, but this is not always the case, nor does it always solve the problem.

When production starts and the produced fluid runs through the pipeline the internal temperature and pressure will increase, due to the reservoir conditions. When the internal pressure increases the hydrostatic pressure outside the pipe remains the same, which causes greater circumferential stress, also called hoop stress, in the pipe. The temperature increase will lead to thermal expansion of the steel, and result in axial compressive forces in the pipe. Combined they result in a longitudinal stress of the pipe. As a response to the longitudinal compressive force, when the pipeline is restrained (see *chapter 3.2*), global buckling may occur at a position determined by the curvature of the pipeline and the support conditions.

A pipeline will buckle in the direction where it meets the least resistance. In a free span it usually buckles downwards, on the seabed it can move sideways (lateral buckling), and for buried pipelines the easiest way to move usually is upwards. The last one is well known as ‘upheaval buckling’, which is a phenomenon that is unfavorable considering the risk of impact by fishing gear mentioned earlier.

For control of the pipeline and its movement after being set in operation, an ROV is used, equipped with several cameras and a *pipetracker*. This method of determining the new position of the pipeline will not give the exact position; it will have its errors. These errors are called *noise* (see smoothing section) in the data set (positional coordinates). The standard formats of these data sets are 5pt-files or 7pt-files (appendix A). To use this data, for example, to find the curvature of the pipeline, smoothing of the data set is necessary. This thesis is narrowed into the lateral movement of the pipeline.

2 – Objectives

The scope of this thesis is to develop a *Screening Software Tool for Evaluation of Pipeline Lateral Buckling*, a tool for evaluation of pipeline integrity based on survey results. The survey results to be evaluated will be in the form of 5-pt or 7-pt files (explained in appendix A). The main purpose of the software is evaluation of locations that have undergone lateral buckling. The software tool will be based on Python (appendix B). Python is the chosen programming tool because of its effectiveness; quick line reading, and great memory (this is a requirement because of the size of the survey data file).

In order to develop a buckling screening tool for evaluation of results from pipeline surveys, the following work should be included:

- Calibration analyses
  - FE analyses (using the program SIMLA) for selected pipeline dimensions and pipe-soil parameters will be performed in order to check buckling scenarios.
  - Existing IKM Ocean Design in-house data will also be used as an input to the result database.
  - A comparison of the *Python* and *SIMLA* results will give a good approximation as to which of the smoothing methods should be used.

- Software development
  - Identification of locations that have undergone lateral buckling will be made, and comparison between different surveys will be carried out.
  - Automated routines for import of design/ as-laid and operational data.
  - Presentation of development in lateral buckling behavior.
  - Evaluation of identified locations, smoothing of survey data, estimation of utilization factor.

- User manual development
  - The user manual shall give a description of the applied methodology.
  - Description of required input and operations in order to use the software.

3 – Theory

3.1 Buckling

3.1.1 General

Global buckling is a mode of buckling which involves a substantial length of the pipeline; usually several pipe joints without gross deformations of the cross section; upheaval buckling is an example thereof. On the other hand, local buckling is a mode of buckling that is confined to a short length of the pipeline causing gross changes in the cross section; collapse, localized wall wrinkling and kinking are examples of thereof[9]. Global buckling of a pipeline can be compared to a bar in compression; the pipeline will buckle in the direction where it meets the least resistance. In a free span it usually buckles downwards, on the seabed it can move sideways (lateral buckling), and for buried pipelines the easiest way to move usually is upwards. The last one is well known as upheaval buckling, which is a phenomenon that is unfavorable considering the risk of impact by fishing gear and anchors.

The driving force for buckling of the pipeline is the effective axial force (see *chapter 3.5.1*). It is induced by a temperature or/and pressure increase. Before production starts the internal temperature of the pipeline is about the same as its surrounding seawater. When the pipeline is put into service the temperature and pressure will increase. As a result of this the pipe will expand. A constrained pipeline will not allow expansion to occur which will result in axial compressive forces in the pipe wall. The pipeline will try to relieve the stresses by buckling[3], it will try to find a new equilibrium by moving perpendicular to the pipe axial axis. The level of axial force to initiate this global buckling depends on[10]:

- Pipe cross section properties
- Lateral resistance
- Out-of-straightness in the pipeline
- Lateral trigging force (for example trawling)

This phenomenon is most likely to happen in HP/HT reservoirs. Even pipelines with adequate wall thickness may be exposed to buckling at moderate temperatures and pressures[3].

There are several failure modes for a pipe exposed to global buckling. Global buckling is a load response and not a failure mode alone, but global buckling may lead to failures such as fracture, fatigue, local buckling, bending moments, and large plastic deformations. For pipelines lying exposed on the seabed, global buckling may be allowed as long as its displacement is predetermined[3] (controlled).

If the curvature of the buckle (upheaval) leaves a gap between the pipe and seabed, a free span is formed. The pipeline may then be vulnerable to fatigue due to VIV, vortex induced vibrations, at this region[3] or to fishing gear hooking onto the pipeline.

3.1.2 Upheaval Buckling

A buried pipeline can sometimes arch upwards out of the seabed, forming a raised loop that may project several meters[1]. This phenomenon (upheaval buckling) is induced by a longitudinal compressive force due to temperature and pressure increase, when going into operating mode. Upheaval buckling is caused by the interaction between that longitudinal compressive force and the local curvature of the pipeline axis[1]. In other words, axial compressive forces tend to make the pipe push upwards. Upheaval may occur if the combination of weight and the uplift resistance of any cover are not large enough to restrain the pipe[5].

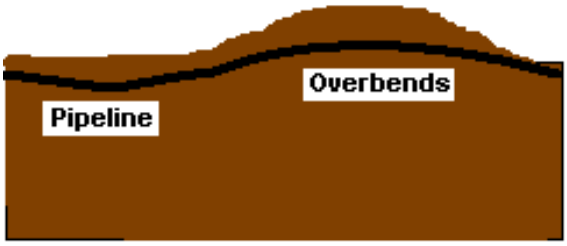

Figure 1 - Pipeline with overbend

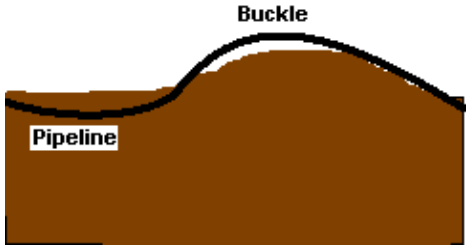

Figure 2 - Pipeline exposed to upheaval buckling

If the pipeline is buried, then there is less resistance to upwards movement compared to sideways/ lateral buckling. The pipeline therefore buckles upwards, almost invariably at overbends where the profile is convex upwards. *Figures 1 and 2* illustrate the sequence schematically[1]. The pipe can also buckle down into the ground, if the seabed stiffness is low enough; for example soft mud. The pipe will always buckle in the direction where it meets the least resistance.

3.1.3 Lateral Buckling

Lateral buckling is induced in the same way as upheaval buckling by a temperature or pressure increase. The difference between lateral and upheaval buckling is just the direction of the bending movement.

If a pipeline is not buried it is usually easier for it to buckle sideways. The resistance to sideways movement is the friction force (soil friction), which is the submerged weight of pipe multiplied by the friction coefficient. There is also a resistance when moving through seawater, but it is negligible[1]. In *figure 3*, a pipeline in service has been exposed to lateral buckling. The track of the as-laid pipeline is notable.

Many pipelines buckle laterally to some extent, but lateral movements frequently go undetected. Lateral movements are often harmless, because the lateral movement occurs over a substantial distance, the bending stresses are small, and the buckle does not localize into a sharp kink. However, lateral movements can be larger, and if one is unlucky, all the movement is concentrated in one buckle. If this movement is too large then a kink might be formed, and if the strain is large enough this can result in rupture of the pipe wall.

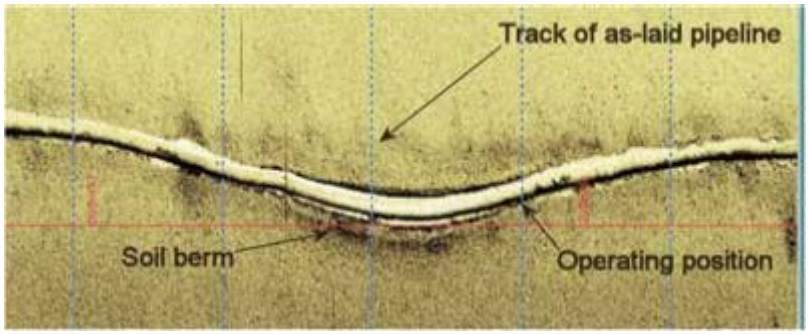

Figure 3 - Pipeline exposed to lateral buckling[13]

A lateral buckling incident in Brazil in 2000 has generated further concern. A hot pipeline buried in soft mud in a shore approach buckled sideways and kinked. The thin wall folded, and the pipe ruptured, leading to a damaging oil spill[1]. In other words, lateral buckling can be a problem and it needs to be controlled and monitored.

3.1.4 Relation between Lateral and Upheaval Buckling

When a pipeline is exposed to upheaval buckling its normal response will be to lie down on the 'side', on the seabed. This is due to the curvature of the pipeline lay and currents on the sea bottom. For illustration, one can think of a perfectly straight elastic stick on a flat table. Compressive axial force is applied from both sides of the stick; this will result in the stick bending up in a smooth curve, as long as it is not bent upwards in an angle (transverse angle). Pipelines are usually not laid in a perfectly straight line given the uneven seabed, so gravity and currents will be the forces laying the pipeline down.

If a pipeline buckling leads the pipeline into exposure on the seabed, the simplest solution would be to stabilize the pipeline at its new position. This can be done by covering the exposed pipe, for example by rock dumping, concrete mats, etc. However, if the integrity of the pipeline is reduced and the pipe wall is overstressed, this may lead to rupture. Then the damaged part will have to be replaced before stabilizing the pipeline again[3].

3.2 Restrained pipeline

So far it has been have written that the temperature increase is proportional to the expansion. This statement is only correct as long as the pipeline is unrestrained. If it is restrained or partially restrained, then the result might be global buckling (*note:* the pipe can also buckle without being restrained). The stresses acting depend on whether the pipeline is unrestrained, restrained or partially restrained.

Friction (soil friction) acts as a restraining force. A fixed object that is connected to the pipeline will have the same effect, for example; a platform.

3.3 Effects of HP/HT reservoirs

A high pressure, high temperature reservoir is formally defined by having an undisturbed bottom hole temperature of greater than 149^{o}C and a reservoir pressure higher than 690bar.[6]
High temperatures from the content of the pipeline causes expansion of the pipe.

Material properties such as yield stress, tensile strength and Young's modulus change with material temperature, and if necessary may be accounted for.

External hydrostatic pressure (p_o) is an important factor regarding the strength capacity of deep-water pipelines. The external pressure is a function of the water depth (WD), water density (ρ = 1025 kg/m^3 for seawater) and gravity (g = 9,81m/s^2).

$$p_o = WD \cdot \rho \cdot g$$

So, for example a reservoir has its wellhead on the seabed, at 3000m water depth. The content running through the pipeline has a pressure of 700bar. The hydrostatic pressure in this case would be around 300 bar, which means that the differential pressure (p_i-p_o) is 400 bar.

For a reservoir with pressures around 200 - 400bar, at the same water depth, the pipe would not be exposed to the same amount of axial forces.

This just shows that for HP/HT reservoirs buckling will always be an issue, because of the great axial forces induced.

3.4 Trawling

Trawling is a method of fishing that involves pulling fishing nets through the water behind one or more boats. The net that is used for trawling is called a trawl[2]. One method is bottom trawling, see *figure 5*, the trawl is dragged on the seabed and it can apply a pullover load on the pipeline. Trawling may affect the pipeline in several ways; trawl impact, pullover and hooking. *Trawl impact* is when the fishing gear hits the pipe, while dragged on the sea bottom, and causes deformation/damage on the pipe. Buckling has a "weakness" for deformed pipes since the deformation weakens the pipe's bending stiffness, and makes it more exposed to axial compressive forces. While a *pullover load* is when the trawl sweeps across the pipeline and exposes the pipe for a great load for a short period of time. If a pipeline curve is experiencing great tension but still not enough to cause buckling of the pipe, then the pullover load might be all that is needed triggering lateral buckling of the pipeline. At last there is *hooking*; it can inflict some serious damage to the pipeline. *Hooking* is caused by a ships anchor, or other similar types of gear, hooking on to the pipeline and dragging it along.

With upheaval buckling the risk of trawl impact increases, this is one of the reasons why upheaval buckling is a very unfavorable scenario. An example of such an incident occurred at the gas field *Kvitebjørn* in the North Sea, with HP/HT conditions. The cause of the accident was an anchor; it had hooked on to the pipeline and inflicted serious damage, *see figure 4*. The pipe was dragged 53m along the seabed, which resulted in a kink of the pipeline. A 26m long pipe section needed to be replaced[11, 12].

Figure 4 - Weight coat loss at damaged location[11] (Kvitebjørn gas pipe)

When the pipeline is installed it is placed on the seabed in tension, which makes it less vulnerable to trawl pullover loads, compared to a pipeline in service exposed to axial compressive forces. In the last case a pullover load might be the triggering factor for the pipe to collapse or be exposed to local buckling. So lateral buckling might also be a problem if not controlled.

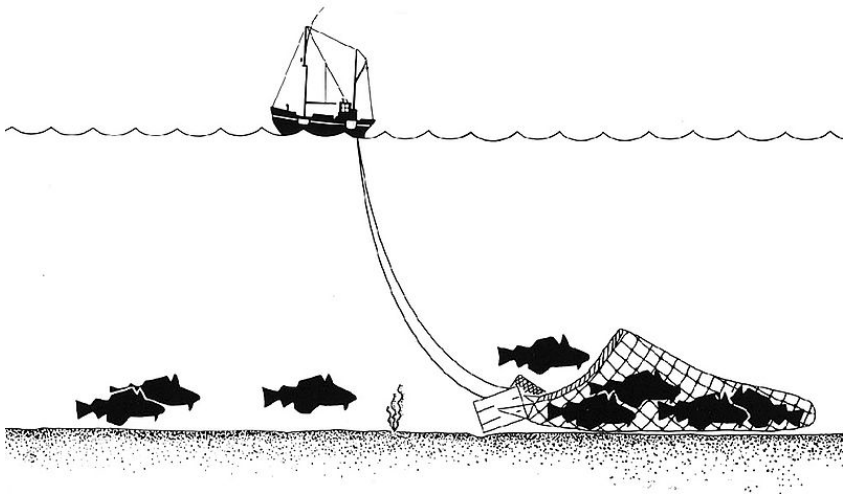

Figure 5 - Bottom trawling activity[15]

3.5 Pipeline Design Analysis

3.5.1 General stress check

This write up on theory used for the stress check is inspired by (1), (3) and (4).

A pipeline has to be designed to withstand all the loads that it will be subjected to, both during installation and operation. During installation it will be bent, pulled and twisted. When production starts and it goes into operation mode it will be loaded by; internal pressure from the fluid it carries, by external pressure from the sea (hydrostatic pressure), and by stresses induced by temperature changes. External impacts from anchors and fishing gear (trawling) can also occur.

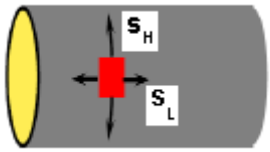

Figure 6 - Hoop stress and longitudinal stress in a cylindrical shaped part

In *figure 6*, hoop stress (S_H) and longitudinal stress (S_L) are shown. *Hoop stress* is circumferential stress; it is generated by internal pressure (being the operating pressure). If the hoop stress is too large the pipeline can yield circumferentially, and continued yielding will lead to thinning of the pipe wall and ultimately to a rupture.

Hoop stress, $\sigma_h = \frac{p_i D_i - p_o D_o}{2t}$

Where, p_i is internal pressure (operating pressure)
p_o is external pressure (hydrostatic pressure of sea)
D_i is inside diameter of pipe
D_o is outside diameter of pipe
t is wall thickness of pipe

A pipeline in operation is exposed to *longitudinal stresses* as well as hoop stress. Longitudinal stresses arise primarily from two effects: Poisson and Temperature. The first one can be explained by imagining a steel bar loaded in tension, it will extend in the tension direction and contract in transverse direction. If transverse contraction is prevented, a transverse tensile stress is set up. As for only circumferential stress (no longitudinal stress), the pipe will extend radially and contract in longitudinal direction. If friction against the seabed (soil friction) or attachments to fixed objects such as platforms prevents longitudinal contraction, a longitudinal tensile stress occurs. This is the *Poisson effect*. The second effect that induces longitudinal stress is temperature. When the temperature

inside the pipe increases the pipe expands in all directions, if the pipe is free to expand in all directions, both axially and circumferentially (radially). Circumferential expansion is usually completely unconstrained, but longitudinal expansion is constrained by the seabed (soil friction) and other objects that constrain the pipe. As for the first effect, if axial expansion is prevented, a longitudinal compressive stress will occur.

Bending moments due to, for example, free span or bending under installation can also occur and will be included in longitudinal stress. If the pipe is applied with an external axial force, this axial force will also be included in longitudinal stress.

Longitudinal stress, $\sigma_l = \alpha \cdot \Delta T \cdot E - \upsilon \frac{(p_i D_i - p_o D_o)}{2t} + \left(\frac{F_a}{A} + \frac{M_b}{I} y\right)$

Poisson effect: $\upsilon \frac{(p_i D_i - p_o D_o)}{2t}$

Thermal effect: $\alpha \cdot \Delta T \cdot E$, (thermal stress)

Where, E is young's modulus
α is linear thermal expansion coefficient
ΔT is temperature difference
p_i is internal pressure (operating pressure)
p_o is external pressure (hydrostatic pressure of sea)
D_i is inside diameter of pipe
D_o is outside diameter of pipe
t is wall thickness of pipe
p_Δ is differential pressure across pipe wall
υ is Poisson's ratio
F_a is axial force
A is cross-section area of pipe
M_b is bending moment
I is moment of inertia
y is distance from the pipe bottom to centre of pipe; $D_o/2$

Longitudinal strain,

$\varepsilon_l = \frac{\sigma_l}{E}$

The resulting axial force in curvature:

Poisson contraction effect is seen acting in the opposite direction to the end cap force. The end cap force is caused by the internal pressure of the content in the pipeline acting on an effectively "closed" end of a pipeline, such as a bend[14].

End cap force: $p_\Delta \frac{\pi D_i^2}{4}$

Expansion is due to combined effects of temperature, pressure and Poisson's effect. In operational pipelines, the three factors will usually occur in combination, which gives this expansion axial force:

$$F_a = \underbrace{\alpha \cdot \Delta T \cdot E \cdot A}_{\text{Thermal force}} - \underbrace{\upsilon \frac{(p_i D_i - p_o D_o)}{2t} A}_{\text{Poisson force}} + \underbrace{p_\Delta \frac{\pi D_i^2}{4}}_{\text{End cap force}}$$

3.5.2 Pipeline Design according to DNV codes

The information in this chapter is found in (9) and (10).

The DNV-OS-F101 standard is used to provide an internationally acceptable standard of safety for submarine pipeline systems. It serves as a guideline for designers, purchaser and contractors.

Two load conditions are used, load controlled condition and displacement controlled condition. Different design checks apply for these two conditions. An example of a displacement controlled condition is a pipeline being installed; it is bent into the shape of another curved structure, such as a reel. In this case, the curvature of the pipe is predetermined. But the circumferential bending that leads to ovalisation is determined by the interaction between the curvature and the internal forces induced by the curvature. Another case is an expansion spool on the seabed. Pipeline expansion due to a temperature or pressure increase imposes a displacement at the end of the spool. The structural response is primarily displacement controlled. However, the lateral resistance to sideways movement of the spool on the seabed also plays a significant part and induces a degree of load control.

These examples show that to choose which condition to use is not so easy, there is no distinct difference between the two conditions in several cases, so the choice should be based on a skilled judgment on which components of the combined conditions are more important.

IKM Ocean Design takes both the *Load controlled condition* and the *displacement controlled condition* into consideration during pipeline design.

Load controlled condition → **Moment criteria**

Displacement controlled condition → Strain criteria

Global buckling is a combination of these two criteria. To combine the *load controlled condition* with the *displacement controlled condition* a condition load effect factor, γ_c, needs to be calibrated.

Using the *Load controlled condition,* pipe members subjected to bending moment, effective axial force and internal overpressure shall be designed to satisfy the following condition at all cross section:

$$\gamma_{sc}\gamma_m\left(\frac{S_d}{\alpha_c S_p}\right)^2 + \gamma_{sc}\gamma_m\left(\frac{M_d}{\alpha_c M_p}\sqrt{1-\left(\frac{\Delta p_d}{\alpha_c p_b(t_2)}\right)^2}\right) + \left(\frac{\Delta p_d}{\alpha_c p_b(t_2)}\right)^2 \leq 1$$

Where,

M_d is Design bending moment

S_d is Design effective axial force

Δp_d is Design differential overpressure

γ_{sc} is Safety class resistance factor

γ_m is Material resistance factor

M_p is Plastic moment resistance

Table 1 - Material resistance factor

Limit state category	SLS/ULS/ALS	FLS
γ_m	1.15	1.00

$$M_p = f_y \cdot (D - t_2)^{2\cdot} t_2$$

S_p is Characteristic plastic axial force resistance given by:

$$S_p = f_y \cdot \pi \cdot (D - t_2) \cdot t_2$$

D is Nominal outside diameter

f_y is Yield stress to be used in design

f_u is Tensile strength to be used in design

Table 2 - Material strength

Characteristic material strength, f_y, f_u	
Property	Value
Characteristic yield stress	$f_y=(SMTS-f_{y,temp})*\alpha_U$
Characteristic tensile strength	$f_u=(SMTS-f_{u,temp})*\alpha_U*\alpha_A$

where,

$f_{y,temp}$ and $f_{u,temp}$ is the reduction value due to the temperature of the yield stress and the tensile strength.

α_U is the material strength factor,
normally 0.96 if not increased confidence in yield stress, then 1.00 is used.

α_A is the Anisotropy factor,
0.95 for axial direction due to relaxed testing requirements in line pipe specification,
1.00 for other cases.

t_2 is Pipe wall thickness*

* t_1 and t_2 are found in DNV-OS-F101, section 5-C 300, Characteristic wall thickness. t_1 is pipe wall thickness used for calculations of *pressure containment resistance,* in other words in situations like system pressure testing. Here the fabrication thickness tolerance, t_{fab}, is taken into consideration. This is because the pipe will crack at the spot where the wall thickness is the thinnest. When designing for the bending moment this is not needed, in this case the nominal wall thickness is used (the overall wall thickness). The next step is to know if the pipe is being designed for an operational condition, or for the construction phase (installation mode). For a pipe in operational condition t_{corr} is included, which is the corrosion allowance. *t* is the nominal wall thickness of the pipe(un-corroded);

$$t_2 = t - t_{corr}$$

This is due to the corrosion from the seawater, CO_2 and Hydrogen Sulfide from the reservoir fluid.

$p_b(t_2)$ is the Burst pressure

$p_b(x) = Min(p_{b,s}(x);p_{b,u}(x))$, which is the pressure containment resistance.

$p_{b,s(x)} = \frac{2 \cdot x}{D-x} \cdot f_y \cdot \frac{2}{\sqrt{3}}$, which is *yielding limit state.*

$p_{b,u(x)} = \frac{2 \cdot x}{D-x} \cdot \frac{f_u}{1{,}15} \cdot \frac{2}{\sqrt{3}}$, which is *Bursting limit state.*

Note! In the two formulas above *x* shall be replaced by t_1 or t_2 as appropriate.

α_c is Flow stress parameter accounting for strain hardening given by:

$$\alpha_c = (1-\beta) + \beta \frac{f_u}{f_y}$$

But maximum 1,20.

$$\beta = \begin{cases} 0{,}4 + q_h & \text{for } D/t_2 < 15 \\ (0{,}4 + q_h)\left(60 - {}^{D}/_{t_2}\right)/45 & \text{for } 15 \leq D/t_2 \leq 60 \\ 0 & \text{for } D/t_2 > 60 \end{cases}$$

$$q_h = \begin{cases} \dfrac{(p_{ld} - p_e)}{p_b(t_2)} \dfrac{2}{\sqrt{3}} & \text{for } p_{ld} > p_e \\ 0 & \text{for } p_{ld} \leq p_e \end{cases}$$

α_c is not to be taken larger than 1,20

Where,

p_e is External pressure
p_{ld} is Local design pressure

$$M_d = M_F \gamma_F \gamma_c + M_E \gamma_E + M_A \gamma_A \gamma_c$$

$\gamma_F, \gamma_A and\ \gamma_E$ are given in *table 1*. While γ_c is to be found separately.

M_F is Functional bending moment
M_E is Environmental bending moment
M_A is Accidental bending moment

Table 3 - Load effect factors and load combinations

Limit State / Load combination		Functional Loads[1)]	Environmental load	Accidental loads	Pressure loads
		γ_F	γ_E	γ_A	γ_p
SLS & ULS	*a*	1,2	0,7	-	1,05
	b	1,1	1,3	-	1,05
FLS		1,0	1,0	-	1,0
ALS		1,0	1,0	1,0	1,0

[1)] If the functional load effect reduces the combined load effects, γ_F shall be taken as 1/1,1.

To find γ_c, DNV-RP-F110 section 9 must be used, *Condition Load Effect For Exposed Pipelines*.

This chapter includes calculation procedures to calculate the *load condition factor*, γ_c for pipelines that buckles. The procedure applies to scenarios of even seabed, un-even seabed, with and without trawl. Depending on the scenario one or more of the parameters may be zero.
The condition factor, $\gamma_c$is based on the prevailing uncertainty in the response bending moment given by:

$$\gamma_c(p, T, F_T) = max\left[0{,}80; 0{,}72 \cdot \left(1 + 2 \cdot CoV\left(X_F(p, T, F_T)\right)\right)\right]$$

Where;

p is characteristic pressure
T is temperature
F_T is trawl load

A $\gamma_c$less than unity calculated in this section shall not be applied to the effective axial load according this Recommended Practice.

$CoV\left(X_F(p,T,F_T)\right)$ is the Coefficient of Variation of the resulting bending moment in the buckle. The uncertainty in the bending moment response from the global FE-analyses is assumed to arise from:

- uncertainty in the axial soil resistance, X_A
- uncertainty in the lateral soil resistance, X_L
- uncertainty in the applied stress-strain curve, X_B
- uncertainty in the applied trawl load, X_C

This uncertainty may be estimated from:

$$CoV\left(X_F(p,T,F_T)\right) = \sqrt{CoV\left(X_A(p,T,F_T)\right)^2 + CoV\left(X_L(p,T,F_T)\right)^2 + CoV\left(X_B(p,T,F_T)\right)^2 + CoV\left(X_C(p,T,F_T)\right)^2}$$

The CoV(X) terms in the equation above reflects the uncertainty in the impact on the bending moment response, this from the uncertainty in the soil parameters, choice of stress-strain curve, and uncertainty in the applied trawl pull-over load. The condition factor, γ_c, will then also represent the degree of *displacement control* that the pipeline experiences.

3.6 Curvature

The definition of *curvature* is the amount by which a geometric object deviates from being *flat,* or *straight* in the case from a line, but curvature is defined in different ways depending on the context[16].

$$\kappa = \frac{1}{R}.$$

Where;

K is curvature.
R is radius.

The formula above shows that in a straight line the curvature is equal to zero, and the curvature of a bend is related to its radius. The bigger the radius is the smaller the curvature is which means that for a small radius the curvature is large.

Bending moment is equal to curvature multiplied with its bending stiffness (EI). The bending stiffness of the pipeline is found by:

Bending Stiffness = material stiffness * moment of inertia

E is the Young's modulus; it is a measure of the stiffness of the material. **I** is the moment of inertia. The moment of inertia depends on the objects cross-section; in the case of pipelines a thin walled cylinder approximation is good, which is:

—

Where,

D_o is outer diameter
D_i is inner diameter

The location of an installed pipeline is defined by coordinates easting(x) and northing(y). To find the curvature of the plotted curve/route this formula is used:

$$\kappa = \frac{|y''|}{(1+y'^2)^{3/2}}$$

y' and y'' are derivatives of the curve of the pipeline route.

3.7 Smoothing

The survey data is acquired with ROV support. The data is not fully accurate; every survey report has its own tolerance regarding survey accuracy. When plotting the data points and interpolating between them a very curvy graph occurs, a curve which in reality is not possible with the transport pipeline dimensions used in the offshore industry. So to get a realistic pipe lay picture as output, this graph needs to be smoothed. *Smoothing* is to smooth a data set to create an approximating function that attempts to capture important patterns in the data, while leaving out *noise*[18]. (In common use the word noise means any unwanted sound, but in this case it is unwanted data without meaning)[19]. In this case, as seen in *figure 7*, there are multiple stages that take place to get this survey data. First there is the connection between the satellite and the ship that gives the ships location with the use of GPS. Second, by using echo the ship knows where the ROV is located at every time, and at last, the ROV is equipped with a pipetracker (see section ROV Pipetracker) to locate the pipeline. At each of the three stages of locating the pipelines position there are measurement errors, all errors together causes the *noise*.

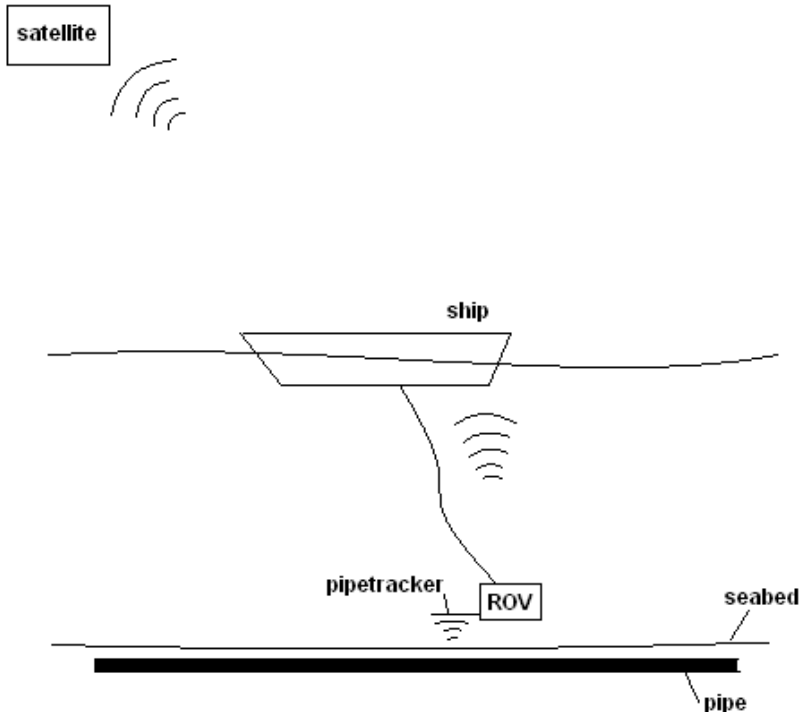

Figure 7 - Measurement errors

With this new approximating function the curvature can be calculated, and the curvature should tell where the pipeline could have buckled. The different smoothing conditions and what order of polynomial approximation is used make the result vary. This is why choosing the right smoothing condition is very important.

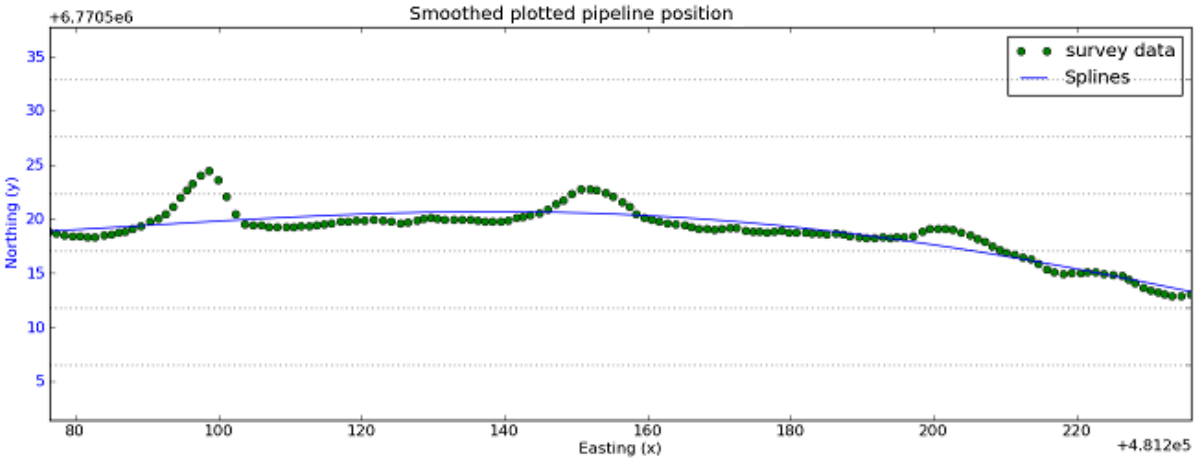

Figure 8 - An example of smoothing

Figure 8 is an example of plotted survey data set (green points) and the (blue) smoothed line (spline). For the untrained eye it looks like three possible areas have undergone local buckling. But if considering a 20 inch pipeline then this kind of movement is not possible, the resulting bending moment would be too large (unrealistic) because of the unusual high curvature.

This is the kind of *noise* that needs to be reduced. But then again where does the line go between noise and useable measurements? This is really up to the user; trust in the data given by the survey contractor, but also use common sense. If the data given is not very realistic then a quick check of an area can be all that is needed, and then if nothing is wrong with the pipe this data can be labeled as noise, and can be reduced by smoothing as done in the figure above. But again, the areas should be checked if the survey contractor says the data is good, since the contractors usually performs their own calibration, etc of survey data.

3.8 ROV Pipetracker

A ROV Pipetracker provides the capability of tracking subsea pipelines. To track the pipeline magnetometer-based sensors can be used. But it is easily affected by other forms of magnetism, which can cause measurement errors (noise). The pipetracker can also use pulse induction technology that gives the ability to locate any conductive material on or below the sea surface. The unit generates highly accurate survey data that can enable the location of any type of subsea pipeline to be recorded with exceptional accuracy[21]. It has the ability to function around subsea structures.

3.9 Finite Element Method - Analysis (FEM-A)

The finite element analysis is a way to simulate the behaviors of an installed pipeline in a realistic three-dimensional environment obtained by measurements of the seabed topography. This allows engineers to exploit any opportunities that the pipeline behavior may offer to develop both safe and cost-effective solutions[7]. The finite element model may be a tool for analyzing the *in place* behavior of a pipeline.

In this thesis the verification work is performed in SIMLA; a FE-analyze software (see Appendix C).

4 – Software development

The design criterion is the allowable bending moment, so to find the actual bending moment of a pipeline in the operational mode is very important. If the actual bending moment exceeds the allowable bending moment the axial compressive stress will be higher than expected, and this can cause buckling of the pipe. This is why finding the curvature of the pipeline can be so helpful.

In most cases a map of the pipeline route has the coordinates easting(x) and northing(y), but at the same time engineers want the different positions of the pipeline given in KP (Kilometric Point along the pipeline). The KP follows the pipeline route from KP 0 (zero) which is the starting point, to the end. The way it is written varies, sometimes KP 6 is the point on the pipeline after 6 meters, but in other cases this point will be written as KP 0.006, where KP 6 is the point after 6 kilometers. The last alternative is the correct one, since it is called kilometer point not meter point, however both are used.

To find the curvature, survey data needs to be collected so the position of the pipeline in operational mode can be determined. The survey data set is a data set of positional measurements, how many measurements that are taken varies, but normally it is at every meter following the KP. But as said these measurements differ from each other, distance wise and in accuracy. *Noise* in data set reduces the accuracy. This *noise* needs to be eliminated/ reduced, and this is done by smoothing the data set through some kind of interpolating that takes this into consideration by not interpolating through every point, instead it gives us a better approximation of the pipeline position. It is this smoothing operation that is important in this software.

For the Lateral Buckling Screening Software (LBSS) the *Python programming* will be used to obtain automated routines.

There are 2 main steps in the development of the software tool:

Step 1 will aim to present an overview of the results, based on an evaluation of the survey data:

- Lateral offset from design/as-laid data.

Import of survey data shall be performed automatically.

Step 2, based on reported lateral offset, selected locations will be evaluated individually. In order to get an accurate estimate of pipeline curvature and bending moments, different smoothing alternatives will be evaluated for use in the software. Smoothing of the survey data will reduce/eliminate "noise" in the data set.

Three different smoothing functions in the software Python was applied:

- polyfit()
- interpolation.UnivariateSpline()
- interpolate.splrep() ***with*** interpolate.splev()

The polyfit() function gives a polynomial approximation of the data set, but this is for the whole graph, not just for parts of it, so for long graphs (data sets) this is not a good alternative. The other two functions uses a method of smoothing, *fitting a smooth curve to a set of noisy observations*[17], using a *spline function*, which is piecewise polynomials with continuous derivatives to chosen degree. This allows an interpolation with a smoothing factor that determines how many of the points that should be used (interpolated through). This allows the user to reduce the "noise" in the data set, and it will most likely give a smoother/ better picture of the pipeline position.

The interpolation.UnivariateSpline() function reduces the noise very well, however the coefficients for the piecewise polynomial is needed to calculate the derivative of the curve. According to a source on the internett[22] these coefficients belong to the Bézier formula (for Bézier curves), which got too complicated compared to the next alternative, so this resulted in abondonment of the function. The choice ended on the function combination: interpolate.splrep() and interpolate.splev(). The combination is user friendly and executes excellent interpolations, and it has the .splev() function to calculate the derivatives of the graph/curve.

The resulting curvature and bending moment (according to the curvature) for the chosen alternative is presented as output in the Lateral Buckling Screening Software.

Calculation of the curvature of the smoothed curve multiplied with the pipes bending stiffness gives the bending moment along the pipeline.

The allowable bending moments for the pipeline will be plotted together with the calculated bending moments for the smoothed line. (Existing and verified in-house spreadsheets for calculation of allowable bending moments will be applied for calculation of allowable moments.)

5 – Calibration analyses

The selection of best possible smoothing method when evaluating survey data is critical. In order to increase the confidence in selection of smoothing method, a set of calibration analyses using a 3D FE in-place analysis tool, named the SIMLA is used.

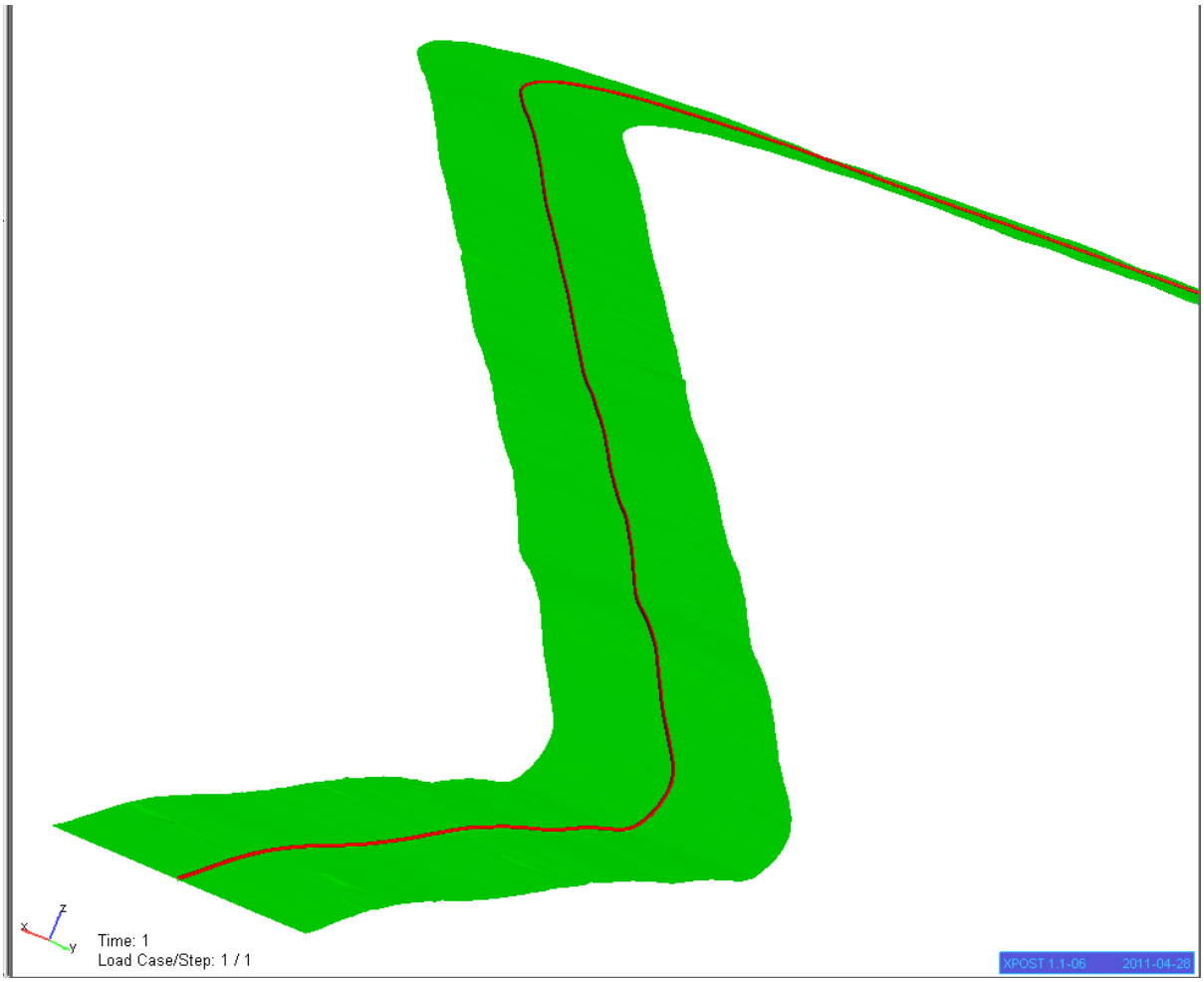

Figure 9 - A simulated pipeline on meshed seabed

SIMLA reads the seabed data from survey data to generate the three-dimensional terrain mesh. As the pipeline is laid onto the seabed mesh, the pipeline elements are free to move in all degrees of freedom, at both ends. The pipeline is therefore not restricted to movements in the vertical direction only, but may slide sideways if the slope is large perpendicular to the pipe axis.

In order to get a good estimate of pipeline utilization (bending moment) at the time of survey the following analysis is applied:

1. Create a pipeline route on a pre-made seabed (in SIMVIS).

2. Add pipeline dimensions and soil friction (in SIMLA).

3. Lay the pipeline down on the seabed (in SIMLA).

4. Add pressure and temperature, gradually (in SIMLA).

5. Run SIMPOST for the results: KP value, new x and y coordinates for the pipeline and the bending moment.

6. Create a text-file with three columns; one with the KP value, the two other columns contain x and y coordinates separately.

7. Import the new text-file in the developed Lateral Buckling Screening Software.

8. Evaluate the result, and compare it to the results given in SIMLA.

6 – Discussion on:

Selecting the program Python:

Because of its effectiveness, fast line reading and great memory, the program Python is a very good fit for the operations required in the Lateral Buckling Screening Software. Python lets the user quickly write the code needed and thanks to a highly optimized byte compiler and support libraries, Python runs more than fast enough for most applications. This makes it very compatible with the use of large data files, such as survey data.

The three smoothing functions tested in the program Python:

The *polyfit()* function gives a polynomial approximation of the data set, but this is for the whole graph, not just for parts of it, so for long graphs (data sets) this is not a good alternative. It might work for simple curves (short distances), but for longer distances with several curves this function will be too rough when smoothing the survey data.

The *interpolation.UnivariateSpline()* function reduces the noise very well, however the coefficients for the piecewise polynomial are needed in order to calculate the derivative of the curve. According to a source on the internet[22] these coefficients belong to the Bézier formula (for Bézier curves), and after testing the next function ***interpolate.splrep() with interpolate.splev()*** the choice was made. The combination is user friendly and executes excellent interpolations, and it has a function named *.splev()* to calculate the derivatives of the graph/curve. All the user needs to do is to choose a smoothing factor (see appendix D) that fits; in other words, a factor that reduces the noise and gives a realistic picture of the pipeline's position on the seabottom. The smoothing factor is a great attribute of the function since every survey data set is different with respect to size and the frequency of the data.

Problems arising through work on the thesis:

Importing easting and northing coordinates and KP values from different data set was challenging, because some survey data has the data separated with a couple of lines with field information. This field information is not required and is easy to not import, but the information can contain numbers which cause problems, for example dates. These numbers must be excluded from the import through a separate method.

The program SIMLA would never accept the flat seabed profiles created. This was never achieved during the thesis work. The solution to the problem was to use actual seabed profiles generated in the program SIMVIS from earlier IKM Ocean Design projects (no field names given). Due to this solution another problem arose, the verification work got more challenging. The LBSS tool does not take into consideration the stress effects from upheaval

buckling and an uneven seabed. This is an important factor to consider when evaluating the LBSS results, when comparing them to the results in SIMLA.

This was not the only problem when using the program SIMLA; when the material parameters were imported, SIMLA would not finalize the FE-analysis when using a *linear* material type (which is used in the Lateral Buckling Screening Software). So, in every FE-analysis run performed in SIMLA an *elasto platic* material type has been used.

Both these problems have had their effect on the results given in SIMLA, which have made the verification work complicated. This, as said, is because the Lateral Buckling Screening Software analyses are run with a linear material type, and on a flat seabed profile.

The finalized Lateral Buckling Screening Software developed:

See chapter *7 – Conclusions*.

Further work and improvements:

The effects on the stresses from an uneven seabed can be the next big step for further development of the software. This might be the solution to achieve more accurate results.

The method used for importing easting and northing coordinates and KP values from different data set should be improved. The current method reads a line in the data set and imports the elements in that line that contain a numerical value. The reason for this is that some survey data has the data separated with a couple of lines with field information. This field information is not wanted and is easy to not import, but the information can contain numbers which cause a problem, for example dates. These numbers must be excluded from the import through a separate method.

More testing of the software should be performed. Testing on a totally flat seabed would be very useful, the result will show the effects from a flat seabed compared to an uneven seabed. More testing with different pipe parameters should be performed to see how essential the pipe dimensions are for the results.

7 – Conclusions

The Lateral Buckling Screening Software (LBSS) has shown to be a good tool for the evaluation of pipeline lateral buckling. By using the program Python as programming software it has developed into an easy and effective tool for import of data (for example survey data); with quick line reading and great memory. LBSS contains a smart smoothing function. Why is it smart? Every survey data file contains different amount of data; length and frequency wise. So a permanent formula or smoothing factor will not work. LBSS gives the user the opportunity to choose a smoothing factor as many times as the user feels is necessary for the outcome/ result to be realistic and sufficient to obtain a good estimate for further curvature calculations.

Currently the LBSS has its limitations. Calculations can only be done with a linear material type, and when calculating the bending moment it only takes the horizontal position into consideration. So any effect from an uneven seabed or upheaval buckling is not included, this is why the LBSS will always give lower stress results than a FE-analysis software. To improve the tool, this is the difference maker.

Note! The LBSS smoothing function can also be used for smoothing of other types of data.

8 – References

(1) Palmer A. C., and King R. A., Subsea Pipeline Engineering. PennWell Corporation, Tulsa, USA (2006)

(2) http://en.wikipedia.org/wiki/trawl

(3) Ommundsen, M. L., *Upheaval Buckling of Buried Pipelines*, Master thesis at University of Stavanger (2009)

(4) Karunakaran, D., *Structural Design of Pipelines* [Lecture notes], University of Stavanger (2010)

(5) Karunakaran, D., *Upheaval and Lateral Buckling* [Lecture notes], University of Stavanger (2010)

(6) http://www.glossary.oilfield.slb.com, on HP/HT.

(7) Bai, Y., *Subsea Pipelines and Risers*. Elsevier Ltd, Oxford, UK (2005)

(8) ANSYS Training manual, SAS IP Inc. (2006)

(9) DNV-OS-F101, Submarine Pipeline Systems. Det Norske Veritas, Offshore Standard. DNV, Høvik, Norway (2007)

(10) DNV-RP-F110, Global Buckling of Submarine Pipelines-Structural Design due to High Temperature/High Pressure. Det Norske Veritas, Recommended Practice. DNV, Høvik, Norway (2007)

(11) Solheim, R., *Presentasjon sikkerhetsforum – alvorlige hendelser*. Petroleumstilsynet. Stavanger, Norway (2007)

(12) http://www.aftenbladet.no/energi/1253149/Anker_kan_truga_lys_og_varme.html

(13) OIL&GAS JOURNAL, www.ogj.com, *picture search* on www.google.com

(14) J P Kenny Group, *Pipeline expansion analysis*, Design guidelines (1994)

(15) http://news.mongabay.com/2006/1124-trawling.html

(16) http://en.wikipedia.org/wiki/Curvature

(17) http://en.wikipedia.org/wiki/Smoothing_spline

(18) http://en.wikipedia.org/wiki/Smoothing

(19) http://en.wikipedia.org/wiki/Noise

(20) http://www.epmag.com/archives/features/64.htm

(21) http://findarticles.com/p/articles/mi_qa5367/is_199808/ai_n21426438/

(22) http://comments.gmane.org/gmane.comp.python.scientific.user/24701

Appendix A

5pt- and 7pt-files

and

Design Route-files

7pt-files and 5pt-files

The files start with general information about the field, and are completed with a long series of numbers. Each separated part of the line contains certain information; this pattern is followed through the whole document. For the Lateral Buckling Screening Software, only three of these line elements are needed; KP, Easting and Northing. These three tell the horizontal position of the pipeline (a 2D horizontal coordinate system) and the given KP value along the pipeline.

The problem here (for the Lateral Buckling Screening Software) is that the software needs to know which line not to read, and that every line needs to be split for export of wanted data (the tree mentioned above).

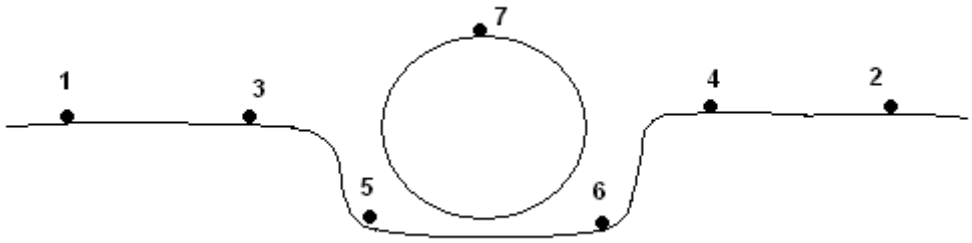

Figure A - A trenched pipeline with the seven points given in a 7pt-file

Under this text there is a section from a 7pt-file, it contains; the KP value, Easting coordinate, Northing coordinate, and then the 7 points shown in *figure A* follows. The last points after that tell the offset from the survey centre line. The difference between 7pt-files and 5pt-files is that the 5pt-files contain two less points then the 7pt-files. The two outermost points is not to be found in the 5pt-files.

KP;Easting;Northing;FarPortMSBL;NearPortMSBL;PortMSBL;StbdMSBL;NearStbdMSBL;FarStbdMSBL;TOP;FarMSBLdist;NearMSBLdist;MSBLdist (# ***This originally only one line***)

-000.0101;473081.77;6772032.49;189.91;189.94;189.53;189.45;189.62;189.31;188.70;17.88;6.06;1.82

-000.0091;473082.65;6772031.91;190.11;189.82;189.55;189.55;189.57;189.34;188.72;18.12;5.70;1.50

-000.0081;473083.56;6772031.56;190.16;189.62;189.68;189.66;189.46;189.45;188.75;18.12;5.65;1.47

-000.0071;473084.53;6772031.21;190.05;189.56;189.59;189.59;189.56;189.60;188.79;18.76;6.03;1.55

-000.0062;473085.43;6772030.89;190.10;189.71;189.71;189.71;189.64;189.71;188.86;18.41;5.73;1.55

-000.0052;473086.41;6772030.58;190.23;189.66;189.80;189.82;189.59;189.42;188.89;18.31;6.36;1.53

-000.0041;473087.38;6772030.28;190.19;189.93;189.71;189.71;189.66;189.28;188.93;17.51;5.80;1.82

Design Route-file

As the 7pt- and the 5pt-file, the Design Route-file (see example under this text) starts with general field information (not included in example), and this is followed by a bunch of numbers. In this case the station column is the KP value.

When using Design Route-files in the screening tool, it is column 1, 3 and 4 which is needed to perform the correct plotting.

Station	Point	Easting (X)	Northing (Y)	Elevation (Z)
0.0	POB	473090.8610	6772027.9130	0.0000
1.0	POT	473091.7671	6772027.4900	0.0000
2.0	POT	473092.6733	6772027.0670	0.0000
3.0	POT	473093.5794	6772026.6441	0.0000
4.0	POT	473094.4856	6772026.2211	0.0000
5.0	POT	473095.3917	6772025.7981	0.0000
6.0	POT	473096.2978	6772025.3751	0.0000
7.0	POT	473097.2040	6772024.9522	0.0000
8.0	POT	473098.1101	6772024.5292	0.0000
9.0	POT	473099.0163	6772024.1062	0.0000
10.0	POT	473099.9224	6772023.6832	0.0000

Appendix B

Python Programming

Python programming

In general

Python is an Enthought developed software. It provides scientists and engineers a comprehensive set of tools to perform rigorous data analysis and visualization[1]. Python, well known for its flexibility and ease-of-use, is becoming a more and more popular programming language for users worldwide. EPD, *Enthought Python Distribution*, backs this statement up by its impressive and powerful collection of Python libraries, including *SciPy*, *NumPy* and *matplotlib*.

The choice of selecting EPD as a programming tool is reasoned by its effectiveness; fast line reading, with a great memory. Python lets the user write the code needed, quickly. And, thanks to a highly optimized byte compiler and support libraries, Python runs more than fast enough for most applications[2]. This makes it very compatible with the use of large data files, compared to computational demanding software, like *Visual Basic*. And, it is free to use.

Some of its key distinguishing features include[2]:

- Very clear and readable language
- Intuitive object orientation
- Natural expression of procedural code
- Full modularity (*the use of individually distinct functional units*[3])
- Exception-based error handling
- Very high level dynamic data types
- Extensive standard libraries and third party modules for virtually every task

[1] http://www.enthought.com/products/epd.php
[2] http://www.python.org/about/
[3] http://dictionary.reference.com/browse/modularity

The language [4]

Python was intended to be a highly readable language. It is designed to have a clear and organized visual layout, frequently using English keywords where other languages use punctuation. The language uses certain statements and expressions.

```
def add5(x):
    return x+5

def dotwrite(ast):
    nodename = getNodename()
    label=symbol.sym_name.get(int(ast[0]),ast[0])
    print '    %s [label="%s' % (nodename,label),
    if isinstance(ast[1], str):
        if ast[1].strip():
            print '= %s"];' % ast[1]
        else:
            print '"]'
    else:
        print '"];'
        children = []
        for n, child in enumerate(ast[1:]):
            children.append(dotwrite(child))
        print '    %s -> {' % nodename,
        for name in children:
            print '%s' % name,
```

Figure B - The figure shows a Python 2.x code and the language that is used[4]

Among others, these are the main statements that have been used developing the screening tool (# marks a comment):

- The *if statement*, which conditionally executes a block of code, along with *else* and *elif* (a contraction of else if).

  A simple unrelated example:

  # A variable x is found, this is a check to see if the variable found has a value higher than 10.
  if x > 10:
    print "The variable is too high!"
  else:
    print "The variable has a value of 10 or less, accepted!"

[4] http://en.wikipedia.org/wiki/Python_(programming_language)

- The *for statement*, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.

  A simple unrelated example:

```
# Duplicate test
# A list named x1 is imported, and this is a duplicate check of the list x1
for i in range(len(x1)):            # A function that iterates as many times as the length of list x1
        if x1.count(x1[i])>1:  # A function that counts the number of similar values as x1[0], x[1], etc
          print x1[i], x1.count(x1[i])  # All duplicates are being printed out as output on the screen
        else:
          pass          # if element has no duplicates, nothing happens, it just goes on to the next element
#                         in the list
```

- The *while statement*, which executes a block of codes as long as its condition is true.

  A simple related example:

```
# A list named xx1 contains integers and is imported
# A simple test to find the closest value in the list xx1; with respect to what's being asked for
test=1
    start1 = input('What is the start easting coordinate of your check? ')
    while test==1:  # this function runs as long as the variable test is equal to 1.
      if start1 in xx1:
        test=2          # if the coordinate asked for is in the list, the while loop is quit
        print '\n'
        print str(start1)+" is the start of your interval"
        print '\n'
      else:
        start1=start1+1             # if not, it adds 1 and searches again
```

- The *try statement*, which allows exceptions raised in its attached code block to be caught and handled by *except* clauses.

  A simple related example:

```
# This is a method that, after a text file has been imported, dismisses unwanted words and certain numbers
x1=[]                # an empty list is created
try:
    if float(a1[g1-1])>30:      # 30 is just a set value, but this also indicates that the wanted value > 30
        x1.append(float(a1[g1-1]))   # A function that enters the wanted numbers into the list x1
    else:
        pass          # if the list element, a1[g1-1], is not greater than 30; nothing happens, it just passes on
except ValueError:   # this function allows exceptions in the list, so if it reads a string (a word),
     pass            # the string is not appended in the list x1
```

- The *def statement*, which defines a function or method. The *def* statement does not execute its block immediately, unlike most other statements. It executes when wanted in the main() function, as shown under:

  A related example:

```
# A function is defined outside the main() function, where its return value is executed in the main()  function.
#  4 alternatives are given
def ask_alt(prompt = 'Which of the four alternatives above do you want to look at? In other words, what files do you have? ', complaint ='1 to 4, please!'):      #the function is named ask_alt
  while True:    # a while loop is executed, it only quits when one of the elements in the list good is given
    good = [ '1', '2', '3', '4']
    alt = raw_input(prompt)        # the variable alt will contain the answer given
    if alt in good:          # if statement that asks if the answer given can be found in the list good
      print "Okey, alternative "+alt+" it is."
      return alt             # this says that alt is the return value when the function is executed
    else:
      print complaint

def main():                          # this is the main() function
  alt = ask_alt()                    # it is at this position the ask_alt() function is being executed

if __name__ == '__main__':
  main()                             # this is the end of the main() function
```

Expressions:

- In Python, == compares by value.
- Python uses the words *and*, *or*, *not* for its Boolean operators.
- Python makes a distinction between lists and tuples (a tuple is similar to a list but with parentheses instead). Lists are written [1,2,3,4,5], and changes can be made. While tuples are written (1,2,3,4,5), and cannot be changed.
- Python has slice expressions on lists, denoted as: [left:right:stride]
  Ex. If the variable *x* is assigned the list [1,2,3,4,5], then the following expressions will be true:
  x[0:2] == [1,2], ps: x[0] is the first element in the list.
  x[1:4] == [2,3,4], i.e. the slice goes up to, but not including, the right index.
  x[:] makes a copy of the list.
- As decimal separator punctuation is used, not comma. Comma is used for separation of different elements in a list.
- To use a function that belongs to a certain library, that library needs to be imported. Ex. Import scipy

Methods:

Methods on objects are functions attached to object, example, **x1.append("1")**. Where x1 is the object and append is the function. The number 1 is being appended in the list x1.

Libraries used (among others):

- Matplotlib is the python 2D plotting library
- NumPy is the fundamental package needed for scientific computing with Python. It provides convenient and fast N-dimensional array manipulation.
- Pylab is another library used for plotting.
- SciPy is open-source software for mathematics, science and engineering. It depends on NumPy, it is built to work with NumPy arrays, and provides user-friendly and efficient numerical routines such as routines for numerical integration and derivation[5]. It also have good interpolation routines.

[5] http://www.scipy.org/

Appendix C

SIMLA

SIMLA [6]

Program basis

SIMLA is a computer program for simulation of umbilical structures. It was developed (Sept., 2000) after a simulation request related to the installation of the Ormen Lange Pipelines[6].

Basic concepts

Analysis in SIMLA can be divided up in several phases; they are organized in *figure C*.

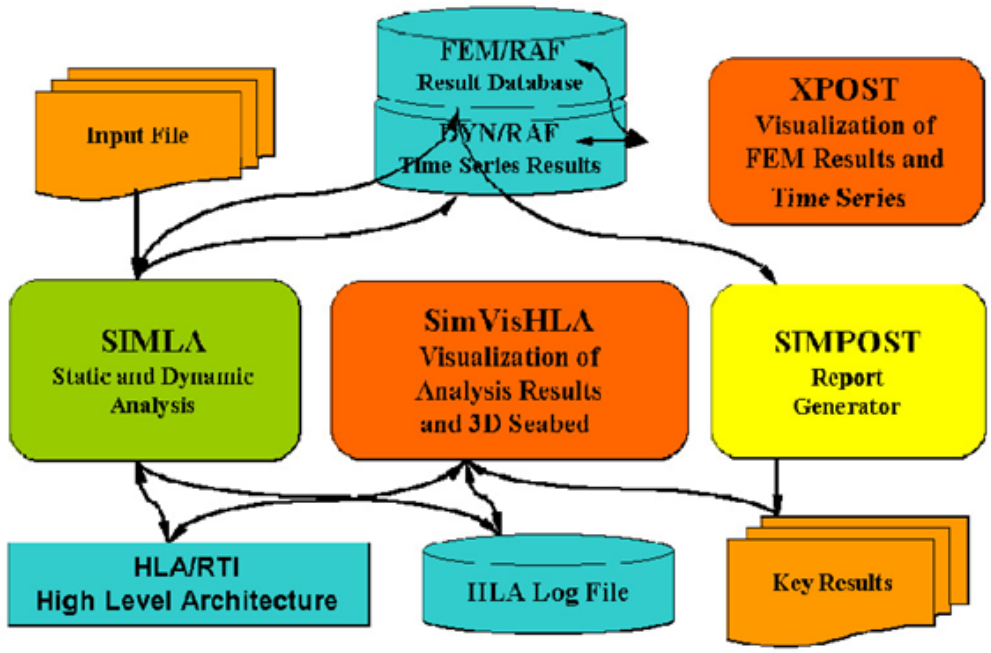

Figure C - The figure illustrates the SIMLA system architecture[6]

- *The Input File* is where the user enters his pipeline and field data. It is text string based; every input line starts with a string, also called *a card* in SIMLA. Every card has its own function, some is used to enter the pipe dimensions, other cards can be used to: decide the different loads and load histories (temperature and pressure, etc), enter material properties, seabed properties (friction wise), decide the sequence of analysis (it defines a set of time intervals where different properties may apply with respect to step length, time interval for result storage and type of analysis – static or dynamic), and among others a text file containing the seabed profile.
- When the input file is complete the next phase is to run the analysis in SIMLA. If the analysis is successful the user can use *XPOST* for visualization of pipeline and the buckling effects on it for each time step.
- To get the results such as new pipeline position after a buckle has been triggered, bending moment, etc, *SIMPOST* is the tool to use. These two results have been used

[6] Sævik, S., Økland, O. D., Baarholm, G. S., and Gjøsteen, J. K. Ø., SIMLA User Manual, MARINTEK, (2010)

in this thesis verifying the screening tool developed. This is done by calculating the curvature of the new pipeline position in the screening tool and matching the result with the result given in SIMPOST.

- Other files used are the *LOG file*, which tells the user when something is wrong and where the problem is. It also logs every step of the analysis when being run. *SIMVIS* is another program used, this is where a 3D seabed is visualized and the user can create a self-made pipeline route, to analyze further in SIMLA.

For more information, see *SIMLA User Manual*.

Appendix D

User manual

USER MANUAL

for

LBSS

A Screening Software Tool for Evaluation of Lateral Buckling

Introduction

The software is developed to reveal potential lateral buckling/snakings, and to check the degree of utilization with respect to the allowable bending moment (actual BM / allowable BM).

The main function of the software is to reduce *noise* from the survey data. This is done by smoothing the data set with a special function in python programming, and making the survey data more realistic and useable for calculations of the pipeline curvature.

INSTALLATION

Setup

To use this software (LBSS) the program Python is required. Python is an *Enthought* developed software; it provides a comprehensive set of tools to perform rigorous data analysis and visualization[7].

Python runs on Windows, Linux/Unix and Mac OS X.

To download Python (including its very much needed libraries), follow these steps:

Enter this web page: http://www.enthought.com/products/trialdownload.php

1) Choose the correct operating system, and enter personal information.
2) Follow the setup instructions.

Note! Attached to the thesis is a CD, it contains program **LBSS.py**, the master thesis and two survey data files for testing of the Lateral Buckling Screening Software (LBSS).

[7] http://www.enthought.com/products/epd.php

HOW TO USE THE SOFTWARE (LBSS)

Overview

To get a brief overview of the software; it is divided in three phases. First is the import phase where three sorts of data can be imported; the design route, as-laid data and survey data. The wanted combination is of the users own choice. Second phase is an overview (illustration) of selected data. From this a more detailed search can be done to find possible lateral buckling, and a wanted section of pipeline is to be chosen. The last phase is to find the curvature, and the related bending moment of the selected pipeline section. This is done by finding the best suited smoothing factor* through some sort of iteration. When the user feels satisfied with the chosen smoothing factor, meaning it gives the most realistic picture of the pipeline lay, the user should be able to find possible lateral bucklings. All this is described in detail in the section under.

Phase one: The purpose of the software is to smooth a data set, in other words a list of data (survey) must be imported. The software will then need to know where on your computer the file is located, for example, *C:\Program files\project2011\(file name).txt*. The software needs a certain format on the imported file. It needs to be a text file (.txt), and every column needs to be separated by space, not semi-colon or any other sign. If original data set is separated by such signs this need to be changed. This can easily be performed in Excel, by using the replace function. When this is in accordance, the next step is to get acquainted with where your wanted data is positioned, in which column. The software will ask from which column it should import the KP values from, likewise with the x and y coordinates.
As mentioned above, the main goal is to find the curvature of the pipeline. The curvature formula contains the second derivative of the polynomial approximation of the curve, this causes a small problem; there can be no duplicates in the x-column (this is a problem in the last phase, but it is best to perform the duplicate check before running the program). This is only relevant for the section chosen to be checked.

Station	Point	Easting (X)	Northing (Y)	Elevation (Z)
0.0	POB	473090.8610	6772027.9130	0.0000
1.0	POT	473091.7671	6772027.4900	0.0000
2.0	POT	473092.6733	6772027.0670	0.0000
3.0	POT	473093.5794	6772026.6441	0.0000

The list above is an example of one of many formats that are acceptable for import.

PS: Decimal separators must be punctuation, not comma.

Typical errors:

- Entering the wrong file-address
- File is not in order, comma instead of punctuation, columns are not separated with space

Phase two: After importing the necessary files the output will be a figure with graphs corresponding to the pipeline position according to, for example, the survey data. In this figure the user can zoom and explore the unsmoothed curve(s) and decide which section needs a better look, a section that might have undergone more lateral buckling than wanted. This section is from one easting (x) coordinate to another easting (x) coordinate. The length of it should not exceed 10 km, this number is relative to the number of measurements in the data set. If the data set is from every meter of pipe, 10 km is a good limitation. If less measurements, the maximum length of the interval expands.

PS: if the user enters 18001 as start en end coordinate, the software searches for the next integer that is closest to 18001. What the method does is that it just removes the current decimals, and the remaining is the new number in the "integer list".

Typical errors:

- Entering too big of an easting (x) interval.

Phase three: This is the last phase; this is where through manual iteration a reasonable result appears as output. Before the result, curvature and corresponding bending moment, appears as output the user is asked to enter a smoothing factor*. It is a parameter in the smoothing function *scipy.interpolate.splrep()* in python. The result from this function is later entered in the derivative function *scipy.interpolate.splev()*. These are the main functions in calculating the curvature.

After the smoothing factor is entered and the curvature and BM (bending moment) diagram has appeared the user can evaluate the result, the smoothing of curve. If it is realistic or not, there is no textbook answer here, this is when the ability to use logic enters. The survey can be good but at some pieces of the data set the ROV Pipetracker can collect wrong data, thinking something beside the pipe is the real pipe, and as usual there are always small measuring errors. This is why smoothing is so important.

if the user feels that the smoothing factor used is not satisfying, the user enter "no" when asked, and phase three starts over again, a new smoothing factor is entered. This is the iteration part; it should end up giving a satisfying result.

PS: A good starting point for the smoothing factor can be between 10 and 50. Then work it from there.

Typical errors:

- Duplicates in the x-coordinate column
- Too small smoothing factor

* A smoothing condition. s = smoothing factor. The amount of smoothness is determined by satisfying the conditions: sum((w * (y - g))**2,axis=0) <= s where g(x) is the smoothed interpolation of (x,y). The user can use s to control the tradeoff between closeness and smoothness of fit. Larger s means more smoothing while smaller values of s indicate less smoothing. Recommended values of s depend on the weights, w. If the weights represent the inverse of the standard-deviation of y, then a good s value should be found in the range (m-sqrt(2*m),m+sqrt(2*m)) where m is the number of data points in x, y, and w. default : s=m-sqrt(2*m) if weights are supplied. s = 0.0 (interpolating) if no weights are supplied[8].

Converting *geographical* coordinates to *easting and northing* coordinates

Geographical coordinate system is a coordinate system that enables every location on the Earth to be specified by a set of numbers. A common choice of coordinates is latitude, longitude and elevation[9]. Compared to *easting and northing* coordinates with the unit *meter*, these coordinates do not, which is a problem when calculating the curvature. However, they can be converted, having the new unit being *meter*.

Usually the 5pt-files and 7pt-files are given in easting and northing coordinates, but when they occasionally come in geographical coordinates they have to be converted. This can be performed with software, called *Geotrans*.

The Startup

To start the software (LBSS) the user must open Python idle, and open the file; (LBSS.py). To run the software when file is opened, just press *F5* or run module in the menu bar above the script.

Problems:

Contact Sivert Duvsete
E-mail: Sivert.Duvsete@IKM.no

[8] http://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.splrep.html
[9] http://en.wikipedia.org/wiki/Geographic_coordinate_system

Appendix E

Verification tests of software

Verification tests of program

Test 1.1. SMR1300 – last time step (time step 503)

SMR1300 is a self-made route constructed in SIMVIS (see Appendix C) on a pre-developed uneven seabed. It is laid with 1300m in radius. After the pipeline is installed the temperature and internal pressure is increased to trigger buckling of the pipeline. These test results are from the last time step in the load history. When visualized in XPOST (see Appendix C), two sections of the pipeline are exposed to significant buckling; both can be seen in the two result *figures D and E*.

SIMLA input:

ID: 0.254m
TH: 0.0175m
Material type: X60, elastoplastic
E: 8.26e10 N/m^2

LBSS input:

ID: 0.254m
TH: 0.0175m
Material type: x60, linear
E: 8.26e10 N/m^2

Comments/ results: Internal pipe test parameters; 93.5 Celsius and 100 bar. This test displays two possible areas for lateral buckling.

Table A - Results from test 1.1

Results	SIMLA [kNm]	SST [kNm]	Differential (SST/SIMLA) [%]
KP 500-600	≈ 175	≈ 88	≈ 50
KP 1000-1100	≈ 197	≈ 144	≈ 73

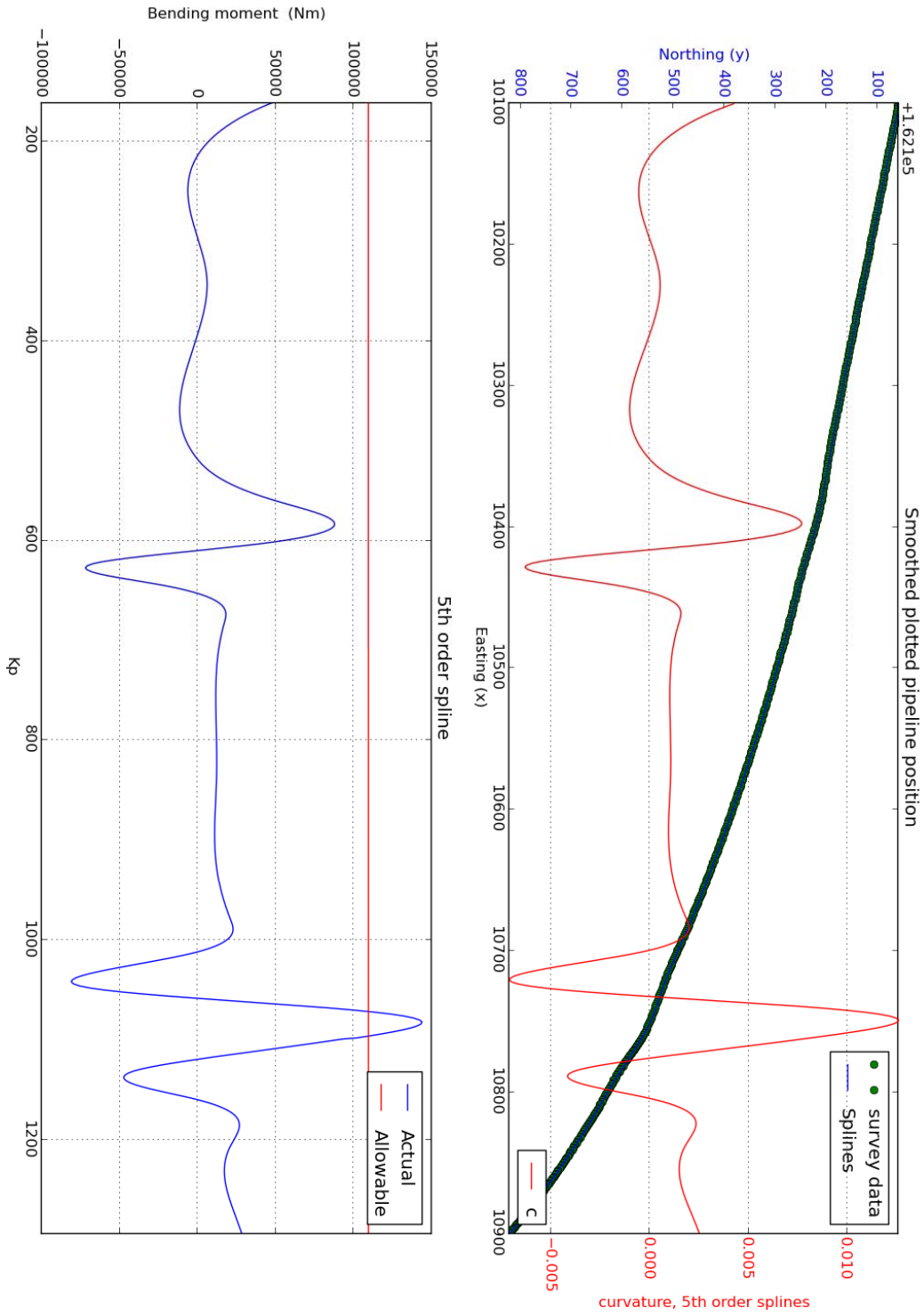

Figure D - The result from LBSS, test 1.1

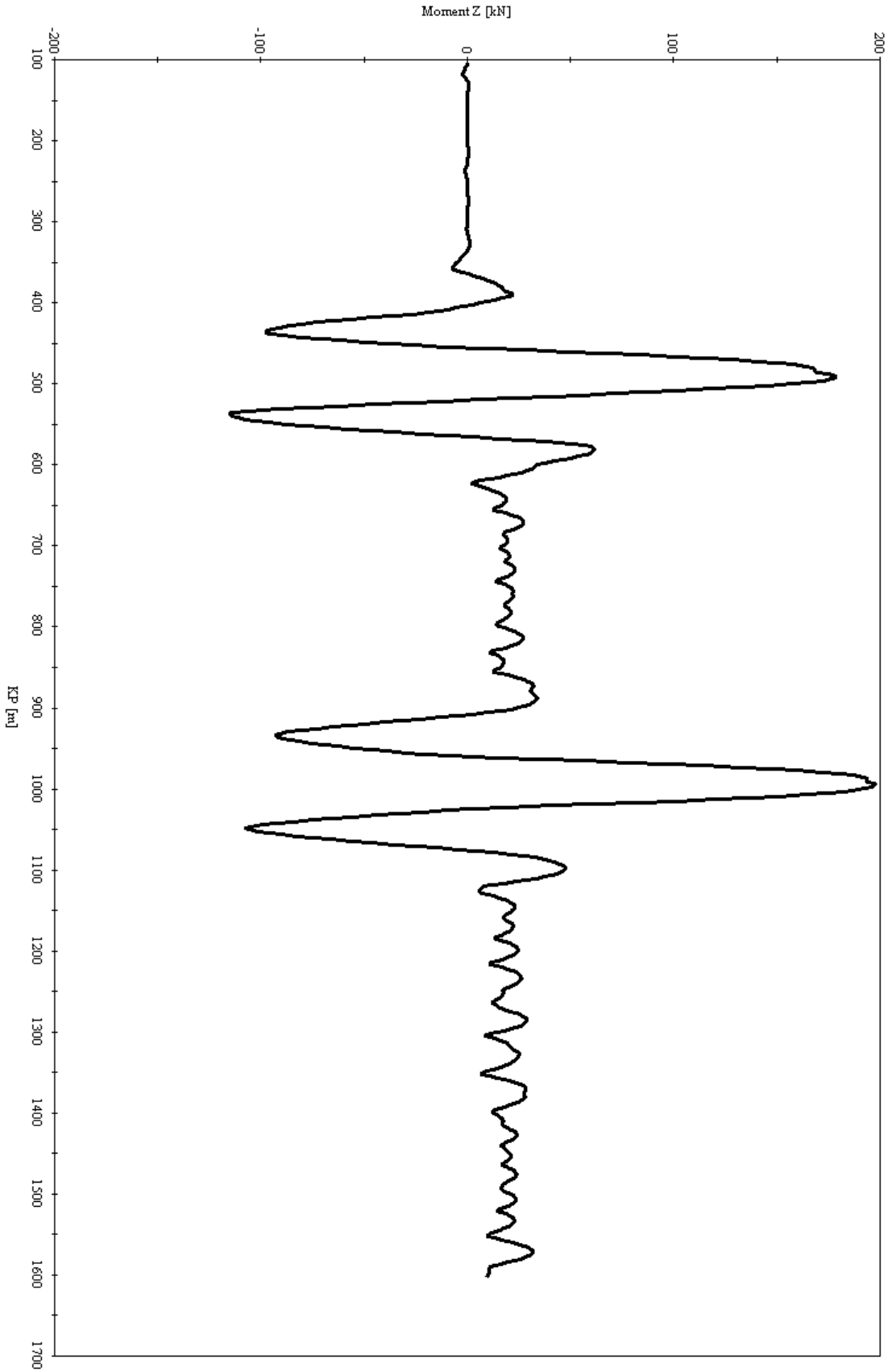

Figure E - The result from SIMLA, test 1.1

Test 1.2. SMR1300 – time step 307

SMR1300 is a self-made route constructed in SIMVIS on a pre-developed uneven seabed. It is laid with 1300m in radius. After the pipeline is installed the temperature and internal pressure is increased to trigger buckling of the pipeline. These test results are from time step 307 (of 503) in the load history. The purpose of this test is to compare the results given from an earlier time step with the results from test 1.1 (same input data, the only difference is the magnitude of the temperature and pressure, which mean less curvature). When visualized in XPOST, two sections of the pipeline are exposed to moderate buckling; both can be seen in the two result *figures F and G*.

SIMLA input:

ID: 0.254m
TH: 0.0175m
Material type: X60, elastoplastic
E: 8.26e10 N/m^2

LBSS input:

ID: 0.254m
TH: 0.0175m
Material type: x60, linear
E: 8.26e10 N/m^2

Comments/ results: Internal pipe test parameters at last time step; 93.5 Celsius and 100 bar. But this is the 307^{th} time step out of 503 time steps. So the internal pipe test parameters are around 60-70% of the final test parameters (own assumption). This test displays two possible areas for lateral buckling.

Table B - Results from test 1.2

Results	SIMLA [kNm]	SST [kNm]	Differential (SST/SIMLA) [%]
KP 500-600	≈ 101	≈ 56	≈ 55
KP 1000-1100	≈ 115	≈ 93	≈ 81

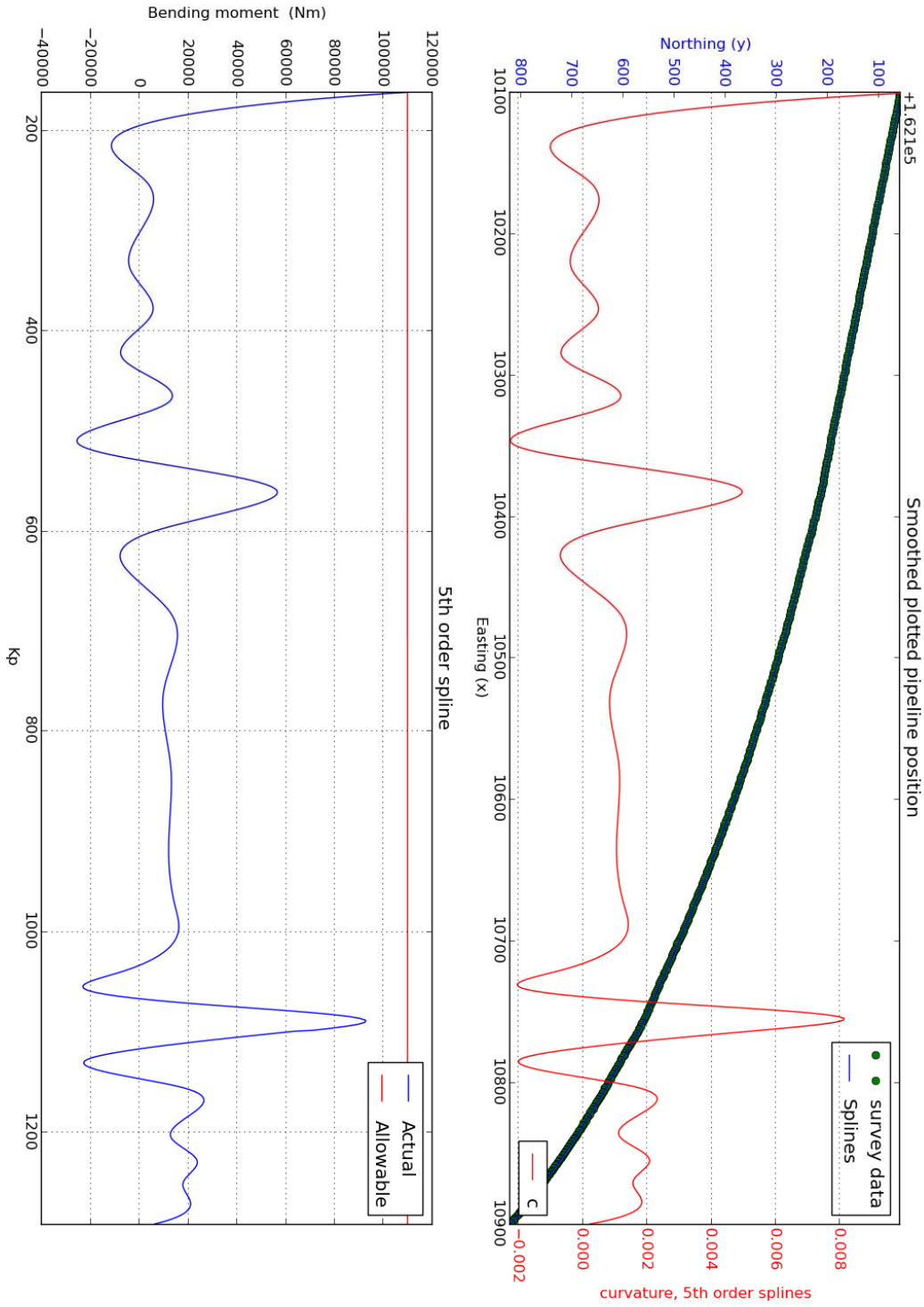

Figure F - The result from LBSS, test 1.2

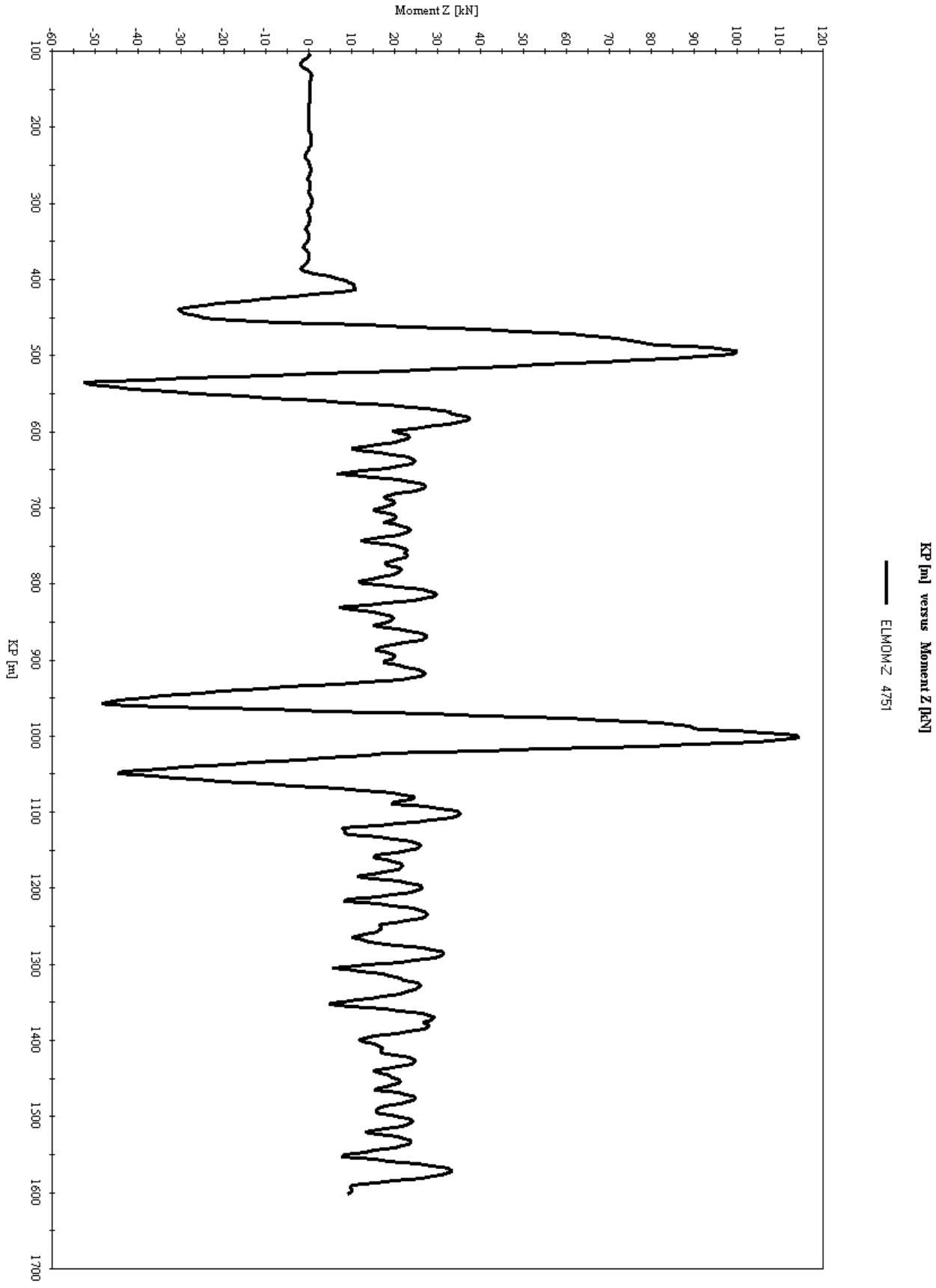

Figure G - The result from SIMLA, test 1.2

Test 2.1. SMR1 – last time step (time step 503)

SMR1 is a self-made route constructed in SIMVIS (see Appendix C) on a pre-developed uneven seabed. After the pipeline is installed the temperature and internal pressure is increased to trigger buckling of the pipeline. These test results are from the last time step in the load history. When visualized in XPOST (see Appendix C), one section of the pipeline is exposed to significant buckling; it can be seen in the result *figures H and I*.

SIMLA input:

ID: 0.254m
TH: 0.0175m
Material type: X60, elastoplastic
E: 8.26e10 N/m^2

LBSS input:

ID: 0.254m
TH: 0.0175m
Material type: x60, linear
E: 8.26e10 N/m^2

Comments/ results: Internal pipe test parameters; 93.5 Celsius and 100 bar. This test displays one possible area for lateral buckling.

Table C - Results from test 2.1

Results	SIMLA [kNm]	SST [kNm]	Differential (SST/SIMLA) [%]
KP 1200-1300	≈ 233	≈ 173	≈ 74

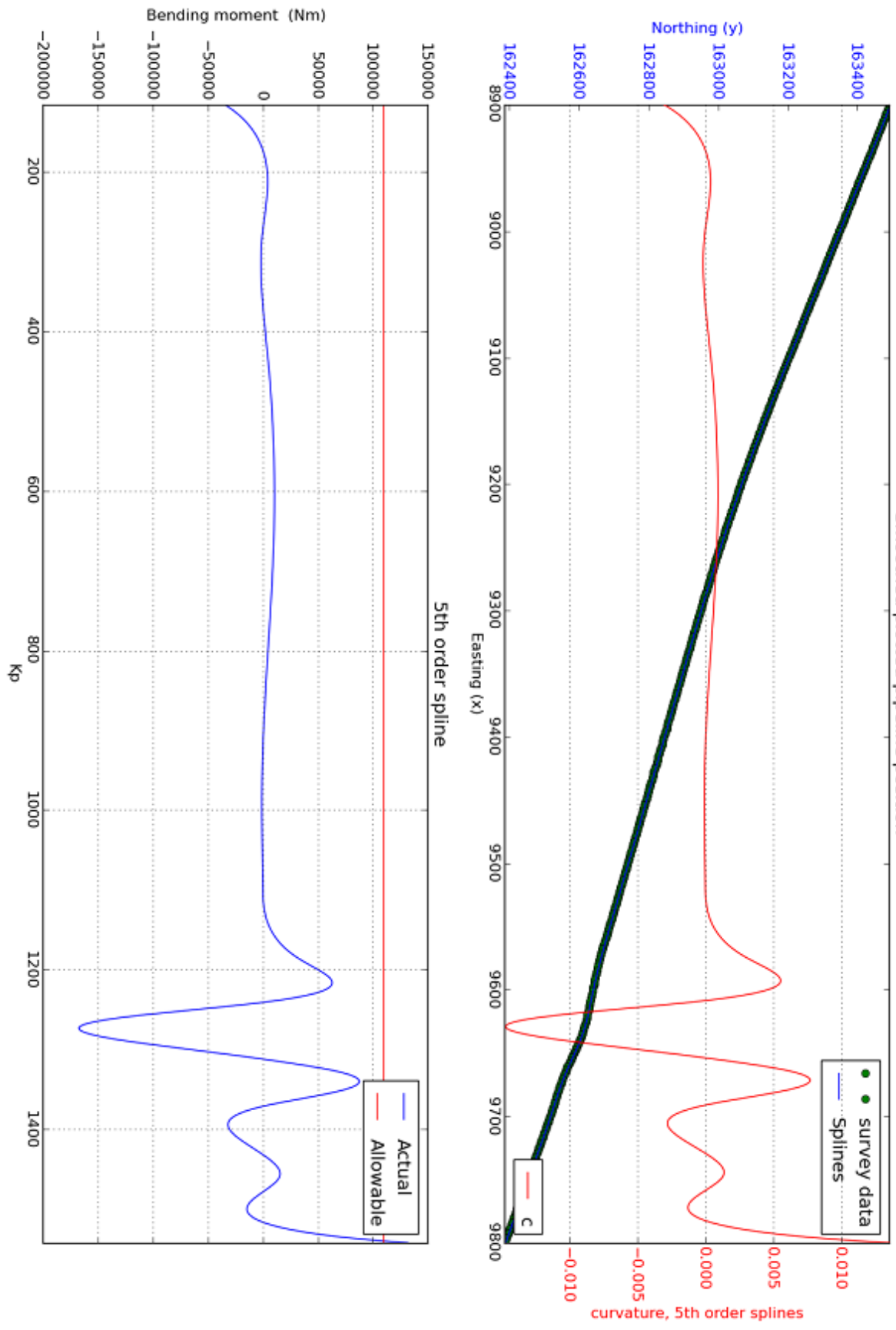

Figure H - The result from LBSS, test 2.1

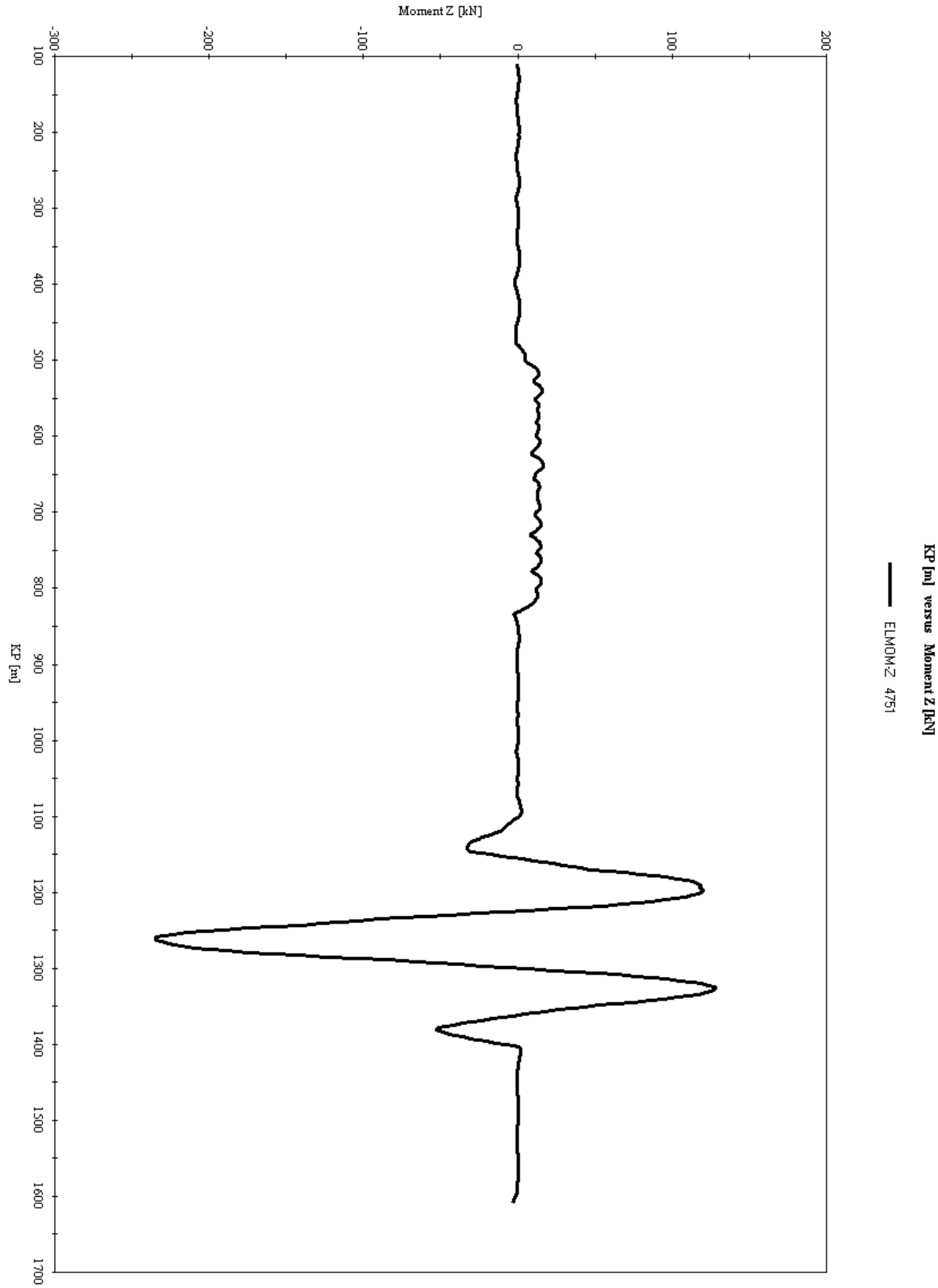

Figure I - The result from SIMLA, test 2.1

Test 2.2. SMR1 – time step 258

SMR1 is a self-made route constructed in SIMVIS on a pre-developed uneven seabed. After the pipeline is installed the temperature and internal pressure is increased to trigger buckling of the pipeline. These test results are from time step 258 (of 503) in the load history. The purpose of this test is to compare the results given from an earlier time step with the results from test 2.1 (same input data, the only difference is the magnitude of the temperature and pressure, which mean less curvature). When visualized in XPOST, one section of the pipeline is exposed to moderate buckling; it can be seen in the two result *figures J and K*.

SIMLA input:

ID: 0.254m
TH: 0.0175m
Material type: X60, elastoplastic
E: 8.26e10 N/m^2

LBSS input:

ID: 0.254m
TH: 0.0175m
Material type: x60, linear
E: 8.26e10 N/m^2

Comments/ results: Internal pipe test parameters at last time step; 93.5 Celsius and 100 bar. But this is the 258^{th} time step out of 503 time steps. So the internal pipe test parameters are around 50-60% of the final test parameters (own assumption). This test displays one possible area for lateral buckling.

Table D - Results from test 2.2

Results	SIMLA [kNm]	SST [kNm]	Differential (SST/SIMLA) [%]
KP 1200-1300	≈ 97	≈ 60	≈ 62

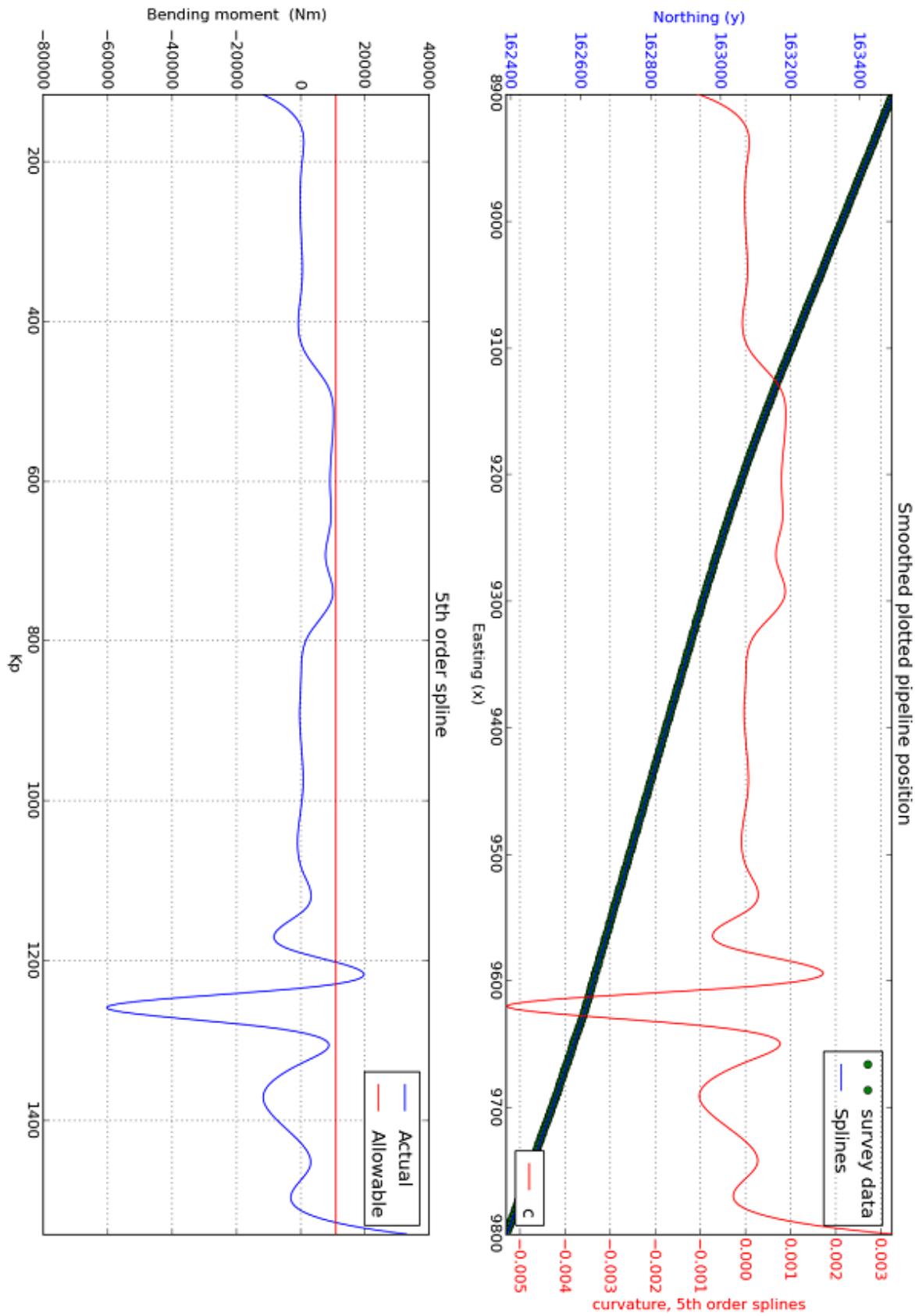

Figure J - The result from LBSS, test 2.2

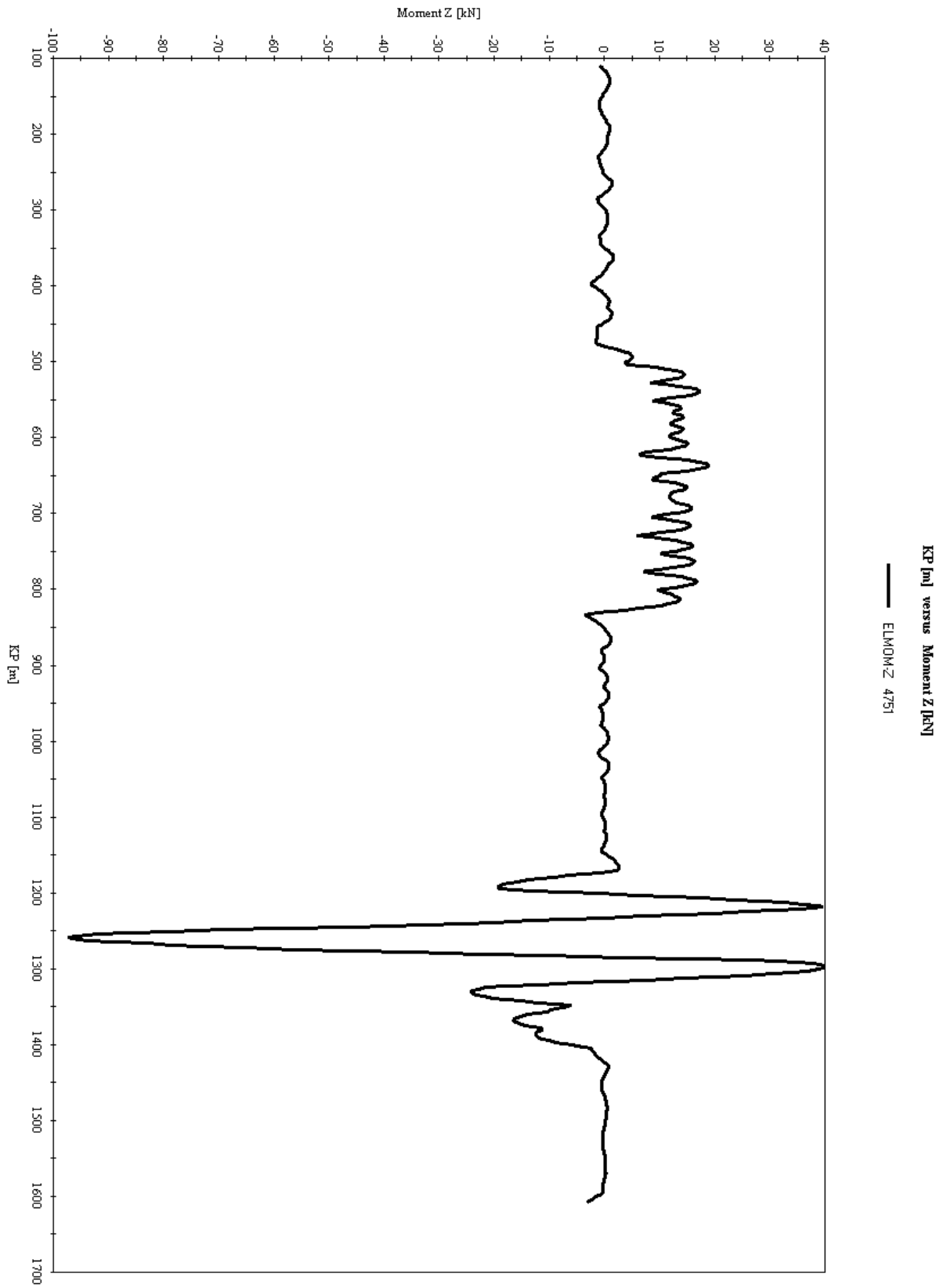

Figure K - The result from SIMLA, test 2.2

Test 3.1. SMR2 – last time step (time step 356)

SMR2 is the same self-made route from test 2.1, but with new pipe dimensions. A smaller pipe with a thinner wall thickness is experimented with. After the pipeline is installed the temperature and internal pressure is increased to trigger buckling of the pipeline. These test results are from the last time step in the load history. When visualized in XPOST (see Appendix C), two sections of the pipeline is exposed to significant buckling; it can be seen in the result *figures L and M*.

SIMLA input:

ID: 0.212m
TH: 0.014m
Material type: X60, elastoplastic
E: 8.26e10 N/m^2

LBSS input:

ID: 0.212m
TH: 0.014m
Material type: x60, linear
E: 8.26e10 N/m^2

Comments/ results: Internal pipe test parameters; 93.5 Celsius and 60 bar. This test displays two possible areas for lateral buckling compared to the one displayed in test 2.1 with the same pipeline route. Smaller pipe dimensions together with less internal pressure have caused lateral buckling in two areas instead of one area, and a better SST result compared to the SIMLA result.

Table E - Results from test 3.1

Results	SIMLA [kNm]	SST [kNm]	Differential (SST/SIMLA) [%]
KP 600-700	≈ 97	≈ 82	≈ 85
KP 1300-1400	≈ 110	≈ 92	≈ 84

Note! In this test the last time step is 356 instead of 503. This is because SIMLA needs less time to increase the pressure to 60 bar instead of 100 bar.

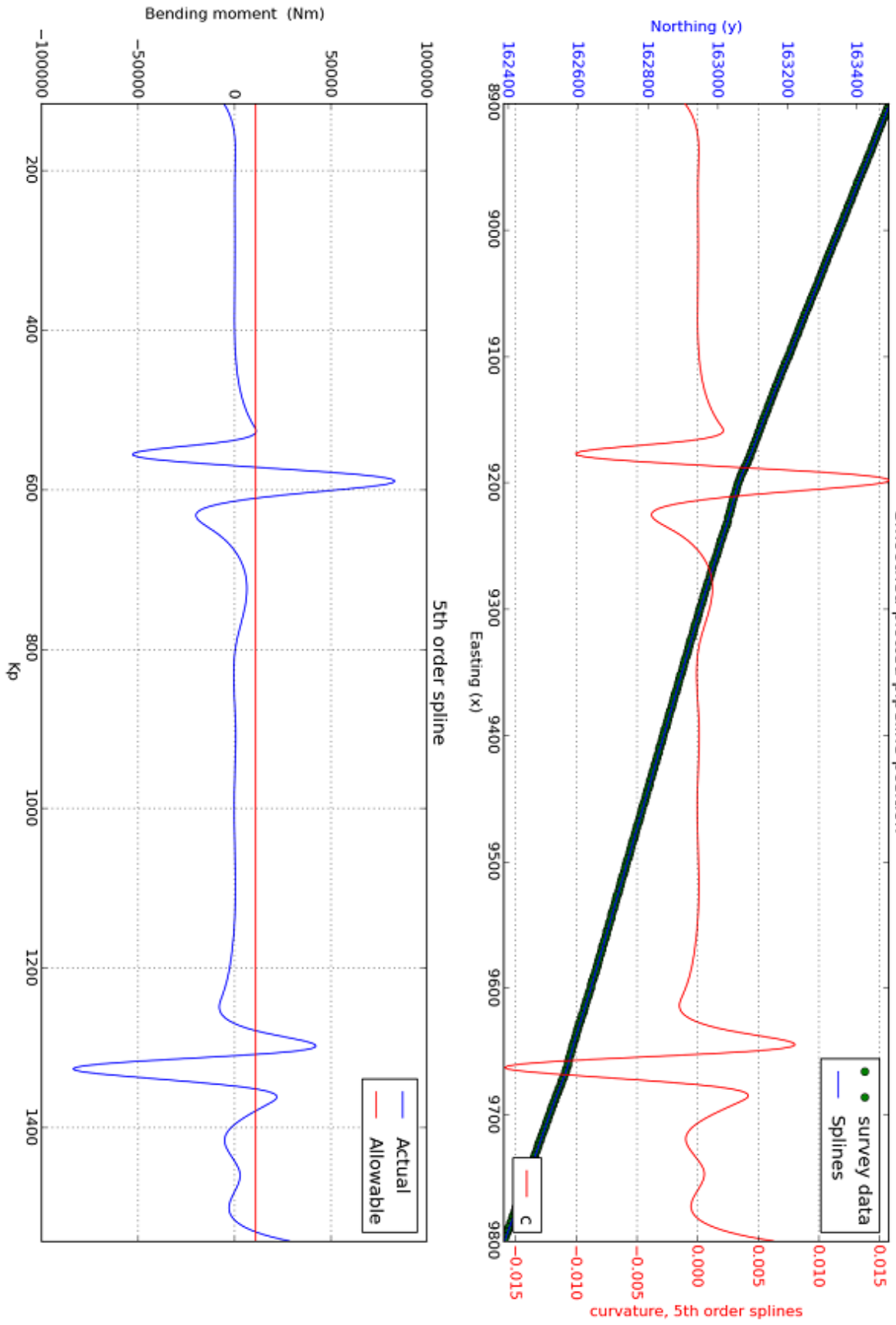

Figure L- The result from LBSS, test 3.1

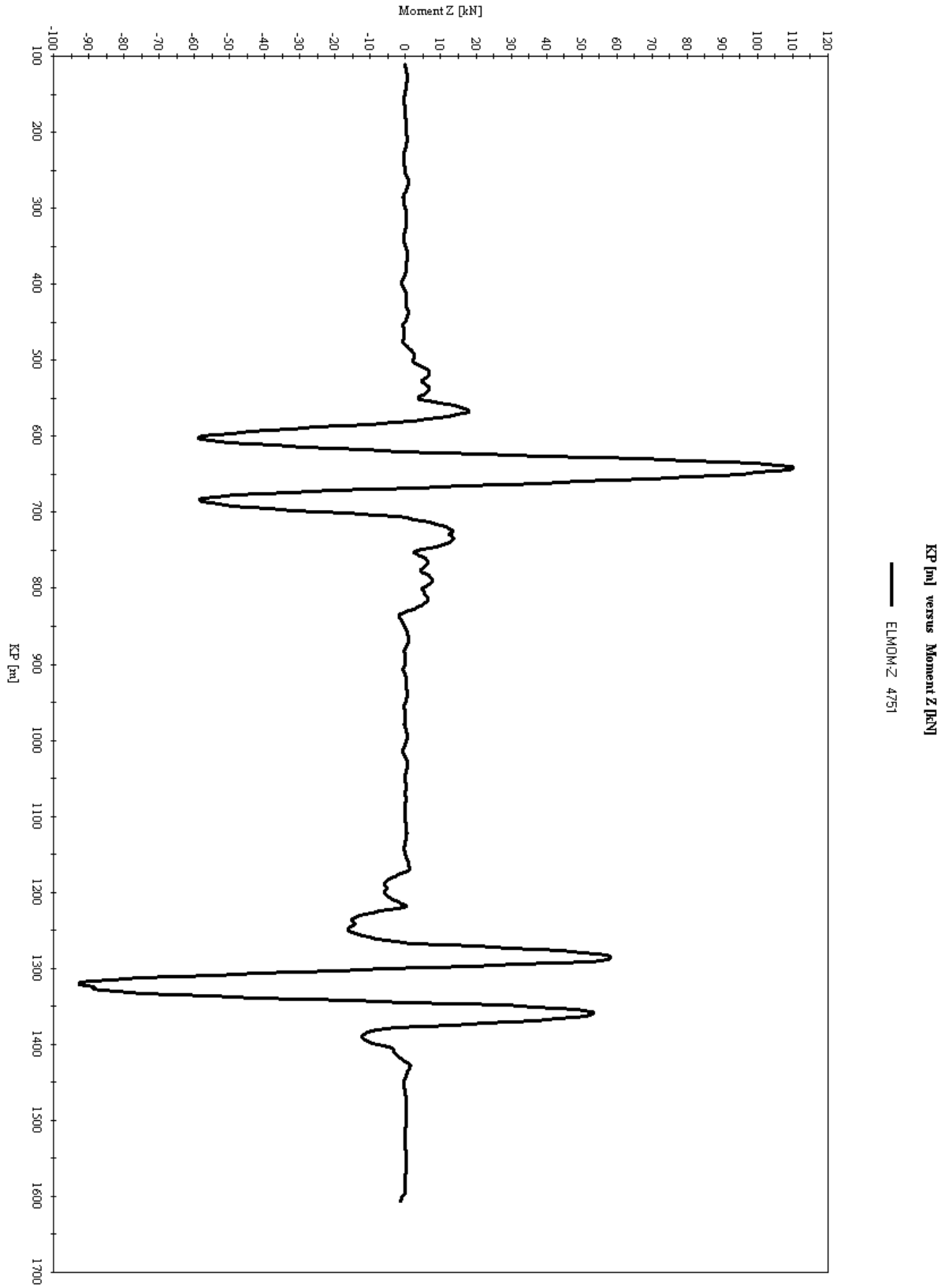

Figure M- The result from SIMLA, test 3.1

Appendix F

Script of python program

```
import string

import pylab

import numpy as np

import matplotlib.pyplot as plt

from pylab import linspace

from scipy import interpolate

from copy import deepcopy

import math


x1 = []

y1 = []

x2 = []

y2 = []

x3 = []

y3 = []

SurveyKp = []

allowable_BM = input('What is the allowable bending moment? ')

# What files to be imported


print '1 - Survey data'

print '2 - Design route and survey data'

print '3 - As-laid data and survey data'

print '4 - Design route, as-laid data and survey data'

print '\n'


#While loop, so the numbers 1-4 have to be chosen
```

```
def ask_alt(prompt = 'Which of the four alternatives above do you want to look at? In other words, what files do you have? ', complaint ='1 to 4, please!'):
    while True:
        good = [ '1', '2', '3', '4']
        alt = raw_input(prompt)
        if alt in good:
            print "Okey, alternative "+alt+" it is."
            return alt
        else:
            print complaint


def test(prompt = 'is this a test? ', complaint ='y for yes, please!'): # A test function
    while True:
        good = ['y']
        testrun = raw_input(prompt)
        if testrun in good:
            print "Okey, test it is."
            return testrun
        else:
            return "no"



def main():
    alt = ask_alt()



    if alt == '1': # Survey data
################################################################################
```

```
    print '\n'
    testrun = test()
    if testrun == 'y':
        filelocation1 = 'c:\\Sivert\\ola.txt'
        g1 = 2
        h1 = 3
        surveyKp = 1
        f1 = open(str(filelocation1))
    else:
        print '\n'
        print "To open your text file(s) the location of the file(s) is needed. The file(s) needs to be in txt format.\n"

        filelocation1 = raw_input('Where is the "survey" data file located? Example: C:\Program Files\example.txt    ')
        print '\n'
        g1 = input("Which column from the left contains the Easting(x) coordinates? ")
        print '\n'
        h1 = input("Which column from the left contains the Northing(y) coordinates? ")
        print '\n'
        surveyKp = input("Which column from the left contains the KP number? ")
        print '\n'
        f1 = open(str(filelocation1))
    tmp1 = f1.readlines()

    for i in range(len(tmp1)):
        try:
            a1 = []
```

```
        line = tmp1[i]
        a1 = string.split(line)
        try:
           if float(a1[g1-1])>1:    # This number (1), might need to be changed
                x1.append(float(a1[g1-1]))
           else:
                pass

           if float(a1[h1-1])>2100:        # This number (2100), might need to be changed
                y1.append(float(a1[h1-1]))
           else:
               pass

           if float(a1[surveyKp-1])<12000:   # This number (12000), might need to be changed
                SurveyKp.append(float(a1[surveyKp-1]))
           else:
               pass
        except ValueError:
            pass
     except IndexError:
        pass

f1.close()

#try:
#    for i in range(len(x1)):
#       if x1.count(x1[i])>1:
```

```
#           print x1[i], x1.count(x1[i])   # Duplicate check, if needed
#       else:
#           pass
#except IndexError:
#    pass



print "Find the interval you want to check through your easting coordinates."


pylab.plot(x1,y1, 'r')
pylab.ylabel('Northing (y)')
pylab.xlabel('Easting (x)')
pylab.grid(True)
pylab.title('Plotted pipeline position')  #finn tittel
pylab.legend( ('Survey') , loc = 'upper right')
#pylab.savefig('save')   # finn navn for fil


pylab.show()


print '\n'


xx1=[]
for i in range(len(x1)):
   xx1.append(float(int(x1[i])))


figo=1
start1 = input('What is the start easting coordinate of your check? ')
```

```
while figo==1:
    if start1 in xx1:
        figo=2
        print '\n'
        print str(start1)+" is the start of your interval"
        print '\n'
    else:
        start1=start1+1


lara=1
end1 = input('What is the last easting coordinate of your check? ')
while lara==1:
    if end1 in xx1:
        lara=2
        print '\n'
        print str(end1)+" is the end of your interval"
    else:
        end1=end1+1


slp1 = xx1.index(start1)
slp2 = xx1.index(end1)


if slp1 < slp2:
    XnewStart = start1
    XnewEnd = end1
    print '\n'
    print 'The interval being checked: KP '+str(SurveyKp[slp1])+' - KP '+str(SurveyKp[slp2])
```

```
print '\n'

# EI, Bending stiffness calculation for given pipe interval

E = input("Enter the Young's modulus (N/m^2): ")

Do= input("Enter the outer diameter of the pipeline (m): ")

ts= input("Enter the wall-thickness of the pipeline( m): ")

I= ((math.pi)/64)*(Do**4-(Do-2*ts)**4)

EI=E*I # (Nm^2)

print "Pipeline bending stiffness: "+str(EI)+" Nm^2"

print '\n'

KpStart = SurveyKp[slp1]

KpEnd = SurveyKp[slp2]

smoothingfactor = input("Enter a smoothing factor: ")

tck = interpolate.splrep(x1[slp1:slp2],y1[slp1:slp2], k=5, s= smoothingfactor)

xnew = linspace(XnewStart, XnewEnd, abs(1000*(slp2-slp1)+1))

ynew = interpolate.splev(xnew,tck,der=0)

yder = interpolate.splev(xnew[::1000],tck,der=1)

y2der =interpolate.splev(xnew[::1000],tck,der=2)

curvature = (y2der)/((1+yder**2)**(3/2))

abs_curvature = abs((y2der)/((1+yder**2)**(3/2)))

BM = [x*EI for x in curvature]  # EI*curvature = BM


fig = plt.figure()

ax1 = fig.add_subplot(211)

ax1.plot(x1[slp1:slp2],y1[slp1:slp2], 'go', xnew,ynew , 'b')

pylab.legend( ('survey data', 'Splines') , loc = 'upper right')

ax1.set_xlabel('Easting (x)')

ax1.set_ylabel('Northing (y)', color='b')
```

```
for tl in ax1.get_yticklabels():
    tl.set_color('b')
ax2 = ax1.twinx()
ax2.plot(xnew[::1000],curvature,'r')
ax2.set_ylabel('curvature, 5th order splines', color='r')
for tl in ax2.get_yticklabels():
    tl.set_color('r')
ax1.set_ylim(y1[slp2],y1[slp1])
ax2.set_ylim(min(curvature),max(curvature))
pylab.grid(True)
pylab.title('Smoothed plotted pipeline position')
pylab.legend( ('curvature') , loc = 'lower right')
#pylab.savefig('save')

aBM = [allowable_BM]
a_BM = aBM*len(SurveyKp[slp1:slp2])

ax3 = fig.add_subplot(212)
ax3.plot(SurveyKp[slp1:slp2],BM[:slp2-slp1], 'b', SurveyKp[slp1:slp2], a_BM, 'r' )
ax3.set_ylabel('Bending moment  (Nm)')
ax3.set_xlabel('Kp')
pylab.xlim(min(SurveyKp[slp1:slp2]),max(SurveyKp[slp1:slp2]))
pylab.grid(True)
pylab.title('5th order spline')
pylab.legend( ('Actual','Allowable') , loc = 'upper right')
#pylab.savefig('save')
```

```
pylab.show()


q = raw_input("Are you satisfied? (yes or no) ")
kiko=1
while kiko ==1:
   if q=="yes":
      kiko =kiko+1
   else:
      smoothingfactor = input("Enter a new smoothing factor: ")
      tck = interpolate.splrep(x1[slp1:slp2],y1[slp1:slp2], k=5, s= smoothingfactor)
      xnew = linspace(XnewStart, XnewEnd, abs(1000*(slp2-slp1)+1))
      ynew = interpolate.splev(xnew,tck,der=0)
      yder = interpolate.splev(xnew[::1000],tck,der=1)
      y2der =interpolate.splev(xnew[::1000],tck,der=2)
      curvature = (y2der)/((1+yder**2)**(3/2))
      abs_curvature = abs((y2der)/((1+yder**2)**(3/2)))
      BM = [x*EI for x in curvature]  # EI*curvature = BM


      fig = plt.figure()
      ax1 = fig.add_subplot(211)
      ax1.plot(x1[slp1:slp2],y1[slp1:slp2], 'go', xnew,ynew , 'b')
      pylab.legend( ('survey data', 'Splines') , loc = 'upper right')
      ax1.set_xlabel('Easting (x)')
      ax1.set_ylabel('Northing (y)', color='b')
      for tl in ax1.get_yticklabels():
         tl.set_color('b')
      ax2 = ax1.twinx()
```

```
ax2.plot(xnew[::1000],curvature,'r')
ax2.set_ylabel('curvature, 5th order splines', color='r')
for tl in ax2.get_yticklabels():
   tl.set_color('r')
ax1.set_ylim(y1[slp2],y1[slp1])
ax2.set_ylim(min(curvature),max(curvature))
pylab.grid(True)
pylab.title('Smoothed plotted pipeline position')
pylab.legend( ('curvature') , loc = 'lower right')
#pylab.savefig('save')

aBM = [allowable_BM]
a_BM = aBM*len(SurveyKp[slp1:slp2])

ax3 = fig.add_subplot(212)
ax3.plot(SurveyKp[slp1:slp2],BM[:slp2-slp1], 'b', SurveyKp[slp1:slp2], a_BM, 'r' )
ax3.set_ylabel('Bending moment  (Nm)')
ax3.set_xlabel('Kp')
pylab.xlim(min(SurveyKp[slp1:slp2]),max(SurveyKp[slp1:slp2]))
pylab.grid(True)
pylab.title('5th order spline')
pylab.legend( ('Actual','Allowable') , loc = 'upper right')
#pylab.savefig('save')

pylab.show()

q = raw_input("Are you satisfied? ")
```

```
else:
    x1.reverse()
    y1.reverse()
    SurveyKp.reverse()
    xx1.reverse()
    slp1 = xx1.index(start1)
    slp2 = xx1.index(end1)
    print '\n'
    print 'The interval being checked: KP '+str(SurveyKp[slp2])+' - KP '+str(SurveyKp[slp1])
    print '\n'
    # EI, Bending stiffness calculation for given pipe interval
    E = input("Enter the Young's modulus (N/m^2): ")
    Do= input("Enter the outer diameter of the pipeline (m): ")
    ts= input("Enter the wall-thickness of the pipeline( m): ")
    I= ((math.pi)/64)*(Do**4-(Do-2*ts)**4)
    EI=E*I # (Nm^2)
    print "Pipeline bending stiffness: "+str(EI)+" Nm^2"
    print '\n'
    KpStart = SurveyKp[slp2]
    KpEnd = SurveyKp[slp1]
    XnewStart = start1 # x1[slp1]
    XnewEnd = end1 #x1[slp2]

    smoothingfactor = input("Enter a smoothing factor: ")

    tck = interpolate.splrep(x1[slp1:slp2],y1[slp1:slp2], k=5, s=smoothingfactor)
    xnew = linspace(XnewStart, XnewEnd, 1000*(slp2-slp1)+1)
```

```
ynew = interpolate.splev(xnew,tck,der=0)

yder = interpolate.splev(xnew[::1000],tck,der=1)

y2der =interpolate.splev(xnew[::1000],tck,der=2)

curvature = (y2der)/((1+yder**2)**(3/2))

abs_curvature = abs((y2der)/((1+yder**2)**(3/2)))

BM = [x*EI for x in curvature]  # EI*curvature = BM


fig = plt.figure()

ax1 = fig.add_subplot(211)

ax1.plot(x1[slp1:slp2],y1[slp1:slp2], 'go', xnew,ynew , 'b')

pylab.legend( ('survey data', 'Splines') , loc = 'upper right')

ax1.set_xlabel('Easting (x)')

ax1.set_ylabel('Northing (y)', color='b')

pylab.grid(True)

for tl in ax1.get_yticklabels():

   tl.set_color('b')

ax2 = ax1.twinx()

ax2.plot(xnew[::1000],curvature,'r')

ax2.set_ylabel('curvature, 5th order splines', color='r')

for tl in ax2.get_yticklabels():

   tl.set_color('r')

ax1.set_ylim(y1[slp1],y1[slp2])

ax2.set_ylim(min(curvature),max(curvature))

pylab.grid(True)

pylab.title('Smoothed plotted pipeline position')

pylab.legend( ('curvature') , loc = 'lower right')

#pylab.savefig('save')
```

```
aBM = [allowable_BM]

a_BM = aBM*len(SurveyKp[slp1:slp2])


ax3 = fig.add_subplot(212)

ax3.plot(SurveyKp[slp1:slp2],BM[:slp2-slp1], 'b', SurveyKp[slp1:slp2], a_BM, 'r' )

ax3.set_ylabel('Bending moment  (Nm)')

ax3.set_xlabel('Kp')

pylab.xlim(max(SurveyKp[slp1:slp2]),min(SurveyKp[slp1:slp2]))

pylab.grid(True)

pylab.title('5th order spline')

pylab.legend( ('Actual','Allowable') , loc = 'upper right')

#pylab.savefig('save')


pylab.show()


q = raw_input("Are you satisfied? ")

kiko=1

while kiko==1:

   if q== "yes":

      kiko = kiko+1

   else:

      smoothingfactor = input("Enter a new smoothing factor: ")

      tck = interpolate.splrep(x1[slp1:slp2],y1[slp1:slp2], k=5, s=smoothingfactor)

      xnew = linspace(XnewStart, XnewEnd, 1000*(slp2-slp1)+1)

      ynew = interpolate.splev(xnew,tck,der=0)

      yder = interpolate.splev(xnew[::1000],tck,der=1)
```

```
y2der =interpolate.splev(xnew[::1000],tck,der=2)

curvature = (y2der)/((1+yder**2)**(3/2))

abs_curvature = abs((y2der)/((1+yder**2)**(3/2)))

BM = [x*EI for x in curvature]  # EI*curvature = BM


fig = plt.figure()

ax1 = fig.add_subplot(211)

ax1.plot(x1[slp1:slp2],y1[slp1:slp2], 'go', xnew,ynew , 'b')

pylab.legend( ('survey data', 'Splines') , loc = 'upper right')

ax1.set_xlabel('Easting (x)')

ax1.set_ylabel('Northing (y)', color='b')

pylab.grid(True)

for tl in ax1.get_yticklabels():

   tl.set_color('b')

ax2 = ax1.twinx()

ax2.plot(xnew[::1000],curvature,'r')

ax2.set_ylabel('curvature, 5th order splines', color='r')

for tl in ax2.get_yticklabels():

   tl.set_color('r')

ax1.set_ylim(y1[slp1],y1[slp2])

ax2.set_ylim(min(curvature),max(curvature))

pylab.grid(True)

pylab.title('Smoothed plotted pipeline position')

pylab.legend( ('curvature') , loc = 'lower right')

#pylab.savefig('save')


aBM = [allowable_BM]
```

```
            a_BM = aBM*len(SurveyKp[slp1:slp2])


            ax3 = fig.add_subplot(212)
            ax3.plot(SurveyKp[slp1:slp2],BM[:slp2-slp1], 'b', SurveyKp[slp1:slp2], a_BM, 'r' )
            ax3.set_ylabel('Bending moment  (Nm)')
            ax3.set_xlabel('Kp')
            pylab.xlim(max(SurveyKp[slp1:slp2]),min(SurveyKp[slp1:slp2]))
            pylab.grid(True)
            pylab.title('5th order spline')  #finn tittel
            pylab.legend( ('Actual','Allowable') , loc = 'upper right')
            #pylab.savefig('save')


            pylab.show()


            q = raw_input("Are you satisfied? ")



   elif alt == '2':   # Design route and survey data
##############################################################################
      print '\n'
      testrun = test()
      if testrun == 'y':
         filelocation1 = 'c:\Sivert\DesignR.txt'
         g1 = 3
         h1 = 4
         f1 = open(str(filelocation1))


         filelocation3 = 'c:\\Sivert\\new7pt.txt'
```

```
        g3 = 2

        h3 = 3

        surveyKp = 1

        f3 = open(str(filelocation3))

    else:

        print '\n'

        print "To open your text file(s) the location of the file(s) is needed. The file(s) needs to be in txt format.\n"


        filelocation1 = raw_input('Where is the "as-designed" data file located? Example: C:\Program Files\example.txt    ')

        print '\n'

        g1 = input("Which column from the left contains the Easting(x) coordinates? ")

        print '\n'

        h1 = input("Which column from the left contains the Northing(y) coordinates? ")

        print '\n'

        f1 = open(str(filelocation1))


        filelocation3 = raw_input('Where is the "survey data" file located? Example: C:\Program Files\example.txt    ')

        print '\n'

        g3 = input("Which column from the left contains the Easting(x) coordinates? ")

        print '\n'

        h3 = input("Which column from the left contains the Northing(y) coordinates? ")

        print '\n'

        surveyKp = input("Which column from the left contains the Kp? ")

        print '\n'

        f3 = open(str(filelocation3))
```

```
# For loop, adding elements to the as-designed array
tmp1 = f1.readlines()

for i in range(len(tmp1)):
   try:
      a1 = []
      line = tmp1[i]
      a1 = string.split(line)
      try:
         if float(a1[g1-1])>3000:    # This number (3000), might need to be changed
              x1.append(float(a1[g1-1]))
         else:
              pass

         if float(a1[h1-1])>3000:
              y1.append(float(a1[h1-1]))
         else:
             pass
      except ValueError:
          pass
   except IndexError:
      pass

f1.close()

# For loop, adding elements to the survey data array
tmp3 = f3.readlines()
```

```
for i in range(len(tmp3)):
   try:
      a3 = []
      line = tmp3[i]
      a3 = string.split(line)
      try:
         if float(a3[g3-1])>3000:    # This number (3000), might need to be changed
              x3.append(float(a3[g3-1]))
         else:
              pass

         if float(a3[h3-1])>3000:
               y3.append(float(a3[h3-1]))
         else:
              pass

         if float(a3[surveyKp-1])<49:   # This number (49), might need to be changed
               SurveyKp.append(float(a3[surveyKp-1]))
         else:
              pass
      except ValueError:
          pass
   except IndexError:
      pass

f3.close()
```

```
print "Find the interval you want to check through your easting coordinates."


pylab.plot(x1,y1, 'r', x3,y3, 'b')

pylab.ylabel('Northing (y)')

pylab.xlabel('Easting (x)')

pylab.grid(True)

pylab.title('Plotted pipeline position')

pylab.legend( ('As-designed', 'Survey') , loc = 'upper right')

#pylab.savefig('save')


pylab.show()


print '\n'


xx3=[]

for i in range(len(x3)):

   xx3.append(float(int(x3[i])))



figo=1

start1 = input('What is the start easting coordinate of your check? ')

while figo==1:

   if start1 in xx3:

      figo=2

      print '\n'
```

```
            print str(start1)+" is the start of your interval"
            print '\n'
        else:
            start1=start1+1  # this formula should be changed, adding .01 instead of 1. (But no luck so
far)
    lara=1
    end1 = input('What is the last easting coordinate of your check? ')
    while lara==1:
        if end1 in xx3:
            lara=2
            print '\n'
            print str(end1)+" is the end of your interval"
        else:
            end1=end1+1

    slp1 = xx3.index(start1)
    slp2 = xx3.index(end1)

    if slp1 < slp2:
        XnewStart = start1
        XnewEnd = end1
        print '\n'
        print 'The interval being checked: KP '+str(SurveyKp[slp1])+' - KP '+str(SurveyKp[slp2])
        print '\n'
        # EI, Bending stiffness calculation for given pipe interval
        E = input("Enter the Young's modulus (N/m^2): ")
        Do= input("Enter the outer diameter of the pipeline (m): ")
        ts= input("Enter the wall-thickness of the pipeline( m): ")
```

```
I= ((math.pi)/64)*(Do**4-(Do-2*ts)**4)

EI=E*I # (Nm^2)

print "Pipeline bending stiffness: "+str(EI)+" Nm^2"

print '\n'

KpStart = SurveyKp[slp1]

KpEnd = SurveyKp[slp2]

smoothingfactor = input("Enter a smoothing factor: ")

tck = interpolate.splrep(x3[slp1:slp2],y3[slp1:slp2], k=5, s=smoothingfactor)

xnew = linspace(XnewStart, XnewEnd, 1000*(slp2-slp1)+1)

ynew = interpolate.splev(xnew,tck,der=0)

yder = interpolate.splev(xnew[::1000],tck,der=1)

y2der =interpolate.splev(xnew[::1000],tck,der=2)

curvature = (y2der)/((1+yder**2)**(3/2))

abs_curvature = abs(y2der)/((1+yder**2)**(3/2))

BM = [x*EI for x in curvature]  # EI*curvature = BM


fig = plt.figure()

ax1 = fig.add_subplot(211)

ax1.plot(x3[slp1:slp2],y3[slp1:slp2], 'go', xnew,ynew , 'b')

pylab.legend( ('survey data', 'Splines') , loc = 'upper right')

ax1.set_xlabel('Easting (x)')

ax1.set_ylabel('Northing (y)', color='b')

for tl in ax1.get_yticklabels():

   tl.set_color('b')

ax2 = ax1.twinx()

ax2.plot(xnew[::1000],curvature,'r')

ax2.set_ylabel('curvature, 5th order splines', color='r')
```

```
for tl in ax2.get_yticklabels():
    tl.set_color('r')
ax1.set_ylim(y3[slp2],y3[slp1])
ax2.set_ylim(min(curvature),max(curvature))
pylab.grid(True)
pylab.title('Smoothed plotted pipeline position')
pylab.legend( ('curvature') , loc = 'lower right')
#pylab.savefig('save')

aBM = [allowable_BM]
a_BM = aBM*len(SurveyKp[slp1:slp2])

ax3 = fig.add_subplot(212)
ax3.plot(SurveyKp[slp1:slp2],BM[:slp2-slp1], 'b', SurveyKp[slp1:slp2], a_BM, 'r' )
ax3.set_ylabel('Bending moment  (Nm)')
ax3.set_xlabel('Kp')
pylab.xlim(min(SurveyKp[slp1:slp2]),max(SurveyKp[slp1:slp2]))
pylab.grid(True)
pylab.title('5th order spline')
pylab.legend( ('Actual','Allowable') , loc = 'upper right')
#pylab.savefig('save')

pylab.show()

kiko=1
q = raw_input("Are you satisfied? (yes or no) ")
while kiko==1:
```

```
if q== "yes":
    kiko=kiko+1
else:
    smoothingfactor = input("Enter a new smoothing factor: ")
    tck = interpolate.splrep(x3[slp1:slp2],y3[slp1:slp2], k=5, s=smoothingfactor)
    xnew = linspace(XnewStart, XnewEnd, 1000*(slp2-slp1)+1)
    ynew = interpolate.splev(xnew,tck,der=0)
    yder = interpolate.splev(xnew[::1000],tck,der=1)
    y2der =interpolate.splev(xnew[::1000],tck,der=2)
    curvature = (y2der)/((1+yder**2)**(3/2))
    abs_curvature = abs(y2der)/((1+yder**2)**(3/2))
    BM = [x*EI for x in curvature]  # EI*curvature = BM

    fig = plt.figure()
    ax1 = fig.add_subplot(211)
    ax1.plot(x3[slp1:slp2],y3[slp1:slp2], 'go', xnew,ynew , 'b')
    pylab.legend( ('survey data', 'Splines') , loc = 'upper right')
    ax1.set_xlabel('Easting (x)')
    ax1.set_ylabel('Northing (y)', color='b')
    for tl in ax1.get_yticklabels():
        tl.set_color('b')
    ax2 = ax1.twinx()
    ax2.plot(xnew[::1000],curvature,'r')
    ax2.set_ylabel('curvature, 5th order splines', color='r')
    for tl in ax2.get_yticklabels():
        tl.set_color('r')
    ax1.set_ylim(y3[slp2],y3[slp1])
```

```
        ax2.set_ylim(min(curvature),max(curvature))
        pylab.grid(True)
        pylab.title('Smoothed plotted pipeline position')
        pylab.legend( ('curvature') , loc = 'lower right')
        #pylab.savefig('save')

        aBM = [allowable_BM]
        a_BM = aBM*len(SurveyKp[slp1:slp2])

        ax3 = fig.add_subplot(212)
        ax3.plot(SurveyKp[slp1:slp2],BM[:slp2-slp1], 'b', SurveyKp[slp1:slp2], a_BM, 'r' )
        ax3.set_ylabel('Bending moment  (Nm)')
        ax3.set_xlabel('Kp')
        pylab.xlim(min(SurveyKp[slp1:slp2]),max(SurveyKp[slp1:slp2]))
        pylab.grid(True)
        pylab.title('5th order spline')  #finn tittel
        pylab.legend( ('Actual','Allowable') , loc = 'upper right')
        #pylab.savefig('save')

        pylab.show()

        q = raw_input("Are you satisfied? ")


else:
   x3.reverse()
   y3.reverse()
```

```
SurveyKp.reverse()

slp1 = x3.index(start1)

slp2 = x3.index(end1)

XnewStart = start1

XnewEnd = end1

print '\n'

print 'The interval being checked: KP '+str(SurveyKp[slp2])+' - KP '+str(SurveyKp[slp1])

print '\n'

# EI, Bending stiffness calculation for given pipe interval

E = input("Enter the Young's modulus (N/m^2): ")

Do= input("Enter the outer diameter of the pipeline (m): ")

ts= input("Enter the wall-thickness of the pipeline( m): ")

I= ((math.pi)/64)*(Do**4-(Do-2*ts)**4)

EI=E*I # (Nm^2)

print "Pipeline bending stiffness: "+str(EI)+" Nm^2"

print '\n'


smoothingfactor = input("Enter a smoothing factor: ")


tck = interpolate.splrep(x3[slp1:slp2],y3[slp1:slp2], k=5, s=smoothingfactor)

xnew = linspace(XnewStart, XnewEnd, 1000*(slp2-slp1)+1)

ynew = interpolate.splev(xnew,tck,der=0)

yder = interpolate.splev(xnew[::1000],tck,der=1)

y2der =interpolate.splev(xnew[::1000],tck,der=2)

curvature = (y2der)/((1+yder**2)**(3/2))

abs_curvature = abs(y2der)/((1+yder**2)**(3/2))

BM = [x*EI for x in curvature]  # EI*curvature = BM
```

```
fig = plt.figure()
ax1 = fig.add_subplot(211)
ax1.plot(x3[slp1:slp2],y3[slp1:slp2], 'go', xnew,ynew , 'b')
pylab.legend( ('survey data', 'Splines') , loc = 'upper right')
ax1.set_xlabel('Easting (x)')
ax1.set_ylabel('Northing (y)', color='b')
for tl in ax1.get_yticklabels():
   tl.set_color('b')
ax2 = ax1.twinx()
ax2.plot(xnew[::1000],curvature,'r')
ax2.set_ylabel('curvature, 5th order splines', color='r')
for tl in ax2.get_yticklabels():
   tl.set_color('r')
ax1.set_ylim(y3[slp1],y3[slp2])
ax2.set_ylim(min(curvature),max(curvature))
pylab.grid(True)
pylab.title('Smoothed plotted pipeline position')
pylab.legend( ('curvature') , loc = 'lower right')
#pylab.savefig('save')
aBM = [allowable_BM]
a_BM = aBM*len(SurveyKp[slp1:slp2])

ax3 = fig.add_subplot(212)
ax3.plot(SurveyKp[slp1:slp2],BM[:slp2-slp1], 'b', SurveyKp[slp1:slp2], a_BM, 'r' )
ax3.set_ylabel('Bending moment  (Nm)')
ax3.set_xlabel('Kp')
```

```
pylab.xlim(max(SurveyKp[slp1:slp2]),min(SurveyKp[slp1:slp2]))
pylab.grid(True)
pylab.title('5th order spline')  #finn tittel
pylab.legend( ('Actual','Allowable') , loc = 'upper right')
#pylab.savefig('save')
pylab.show()

kiko=1
q = raw_input("Are you satisfied? ")
while kiko==1:
   if q=="yes":
      kiko=kiko+1
   else:
      smoothingfactor = input("Enter a new smoothing factor: ")
      tck = interpolate.splrep(x3[slp1:slp2],y3[slp1:slp2], k=5, s=smoothingfactor)
      xnew = linspace(XnewStart, XnewEnd, 1000*(slp2-slp1)+1)
      ynew = interpolate.splev(xnew,tck,der=0)
      yder = interpolate.splev(xnew[::1000],tck,der=1)
      y2der =interpolate.splev(xnew[::1000],tck,der=2)
      curvature = (y2der)/((1+yder**2)**(3/2))
      abs_curvature = abs(y2der)/((1+yder**2)**(3/2))
      BM = [x*EI for x in curvature]  # EI*curvature = BM

      fig = plt.figure()
      ax1 = fig.add_subplot(211)
      ax1.plot(x3[slp1:slp2],y3[slp1:slp2], 'go', xnew,ynew , 'b')
      pylab.legend( ('survey data', 'Splines') , loc = 'upper right')
```

```
ax1.set_xlabel('Easting (x)')
ax1.set_ylabel('Northing (y)', color='b')
for tl in ax1.get_yticklabels():
   tl.set_color('b')
ax2 = ax1.twinx()
ax2.plot(xnew[::1000],curvature,'r')
ax2.set_ylabel('curvature, 5th order splines', color='r')
for tl in ax2.get_yticklabels():
   tl.set_color('r')
ax1.set_ylim(y3[slp1],y3[slp2])
ax2.set_ylim(min(curvature),max(curvature))
pylab.grid(True)
pylab.title('Smoothed plotted pipeline position')
pylab.legend( ('curvature') , loc = 'lower right')
#pylab.savefig('save')

aBM = [allowable_BM]
a_BM = aBM*len(SurveyKp[slp1:slp2])

ax3 = fig.add_subplot(212)
ax3.plot(SurveyKp[slp1:slp2],BM[:slp2-slp1], 'b', SurveyKp[slp1:slp2], a_BM, 'r' )
ax3.set_ylabel('Bending moment  (Nm)')
ax3.set_xlabel('Kp')
pylab.xlim(max(SurveyKp[slp1:slp2]),min(SurveyKp[slp1:slp2]))
pylab.grid(True)
pylab.title('5th order spline')  #finn tittel
pylab.legend( ('Actual','Allowable') , loc = 'upper right')
```

```
            #pylab.savefig('save')


            pylab.show()


            q = raw_input("Are you satisfied? ")



   elif alt == '3':    # as-laid data and survey data
################################################################################
      print '\n'
      testrun = test()
      if testrun == 'y':
         filelocation2 = 'c:\Sivert\R.txt'
         g2 = 1
         h2 = 2
         f2 = open(str(filelocation2))
         filelocation3 = 'c:\\Sivert\\new7pt.txt'
         g3 = 2
         h3 = 3
         surveyKp = 1
         f3 = open(str(filelocation3))
      else:
         print '\n'
         print "To open your text file(s) the location of the file(s) is needed. The file(s) needs to be in
txt format.\n"
```

```
        filelocation2 = raw_input('Where is the "as-laid" data file located? Example: C:\Program
Files\example.txt    ')

        print '\n'

        g2 = input("Which column from the left contains the Easting(x) coordinates? ")

        print '\n'

        h2 = input("Which column from the left contains the Northing(y) coordinates? ")

        print '\n'

        f2 = open(str(filelocation2))


        filelocation3 = raw_input('Where is the "survey data" file located? Example: C:\Program
Files\example.txt    ')

        print '\n'

        g3 = input("Which column from the left contains the Easting(x) coordinates? ")

        print '\n'

        h3 = input("Which column from the left contains the Northing(y) coordinates? ")

        print '\n'

        surveyKp = input("Which column from the left contains the Kp? ")

        print '\n'

        f3 = open(str(filelocation3))



    # For loop, for å fylle as-laid vektorene

    tmp2 = f2.readlines()


    for i in range(len(tmp2)):

        try:

            a2 = []

            line = tmp2[i]
```

```
        a2 = string.split(line)
        try:
            if float(a2[g2-1])>3000:    # This number (3000), might need to be changed
                x2.append(float(a2[g2-1]))
            else:
                pass

            if float(a2[h2-1])>3000:  # This number (3000), might need to be changed
                y2.append(float(a2[h2-1]))
            else:
                pass
        except ValueError:
            pass
    except IndexError:
        pass

f2.close()

tmp3 = f3.readlines()

for i in range(len(tmp3)):
    try:
        a3 = []
        line = tmp3[i]
        a3 = string.split(line)
        try:
            if float(a3[g3-1])>3000:     # This number (3000), might need to be changed
```

```
            x3.append(float(a3[g3-1]))
        else:
            pass

        if float(a3[h3-1])>3000:  # This number (3000), might need to be changed
            y3.append(float(a3[h3-1]))
        else:
            pass

        if float(a3[surveyKp-1])<49:   # This number (49), might need to be changed
            SurveyKp.append(float(a3[surveyKp-1]))
        else:
            pass
      except ValueError:
        pass
   except IndexError:
      pass

f3.close()



print "Find the interval you want to check through your easting coordinates."

pylab.plot(x2,y2, 'g', x3,y3, 'b')
pylab.ylabel('Northing (y)')
pylab.xlabel('Easting (x)')
pylab.grid(True)
```

```
pylab.title('Plotted pipeline position')

pylab.legend( ('As-laid', 'Survey') , loc = 'upper right')

#pylab.savefig('save')


pylab.show()


print '\n'


xx3=[]

for i in range(len(x3)):

   xx3.append(float(int(x3[i])))




figo=1

start1 = input('What is the start easting coordinate of your check? ')

while figo==1:

   if start1 in xx3:

      figo=2

      print '\n'

      print str(start1)+" is the start of your interval"

      print '\n'

   else:

      start1=start1+1

lara=1

end1 = input('What is the last easting coordinate of your check? ')

while lara==1:

   if end1 in xx3:
```

```
        lara=2
        print '\n'
        print str(end1)+" is the end of your interval"
    else:
        end1=end1+1

slp1 = xx3.index(start1)
slp2 = xx3.index(end1)

if slp1 < slp2:
    XnewStart = start1
    XnewEnd = end1
    print '\n'
    print 'The interval being checked: KP '+str(SurveyKp[slp1])+' - KP '+str(SurveyKp[slp2])
    print '\n'
    # EI, Bending stiffness calculation for given pipe interval
    E = input("Enter the Young's modulus (N/m^2): ")
    Do= input("Enter the outer diameter of the pipeline (m): ")
    ts= input("Enter the wall-thickness of the pipeline( m): ")
    I= ((math.pi)/64)*(Do**4-(Do-2*ts)**4)
    EI=E*I # (Nm^2)
    print "Pipeline bending stiffness: "+str(EI)+" Nm^2"
    print '\n'
    KpStart = SurveyKp[slp1]
    KpEnd = SurveyKp[slp2]
    smoothingfactor = input("Enter a smoothing factor: ")
    tck = interpolate.splrep(x3[slp1:slp2],y3[slp1:slp2], k=5, s=smoothingfactor)
```

```
xnew = linspace(XnewStart, XnewEnd, 1000*(slp2-slp1)+1)

ynew = interpolate.splev(xnew,tck,der=0)

yder = interpolate.splev(xnew[::1000],tck,der=1)

y2der =interpolate.splev(xnew[::1000],tck,der=2)

curvature = (y2der)/((1+yder**2)**(3/2))

abs_curvature = abs(y2der)/((1+yder**2)**(3/2))

BM = [x*EI for x in curvature]  # EI*curvature = BM


fig = plt.figure()

ax1 = fig.add_subplot(211)

ax1.plot(x3[slp1:slp2],y3[slp1:slp2], 'go', xnew,ynew , 'b')

pylab.legend( ('survey data', 'Splines') , loc = 'upper right')

ax1.set_xlabel('Easting (x)')

ax1.set_ylabel('Northing (y)', color='b')

for tl in ax1.get_yticklabels():

   tl.set_color('b')

ax2 = ax1.twinx()

ax2.plot(xnew[::1000],curvature,'r')

ax2.set_ylabel('curvature, 5th order splines', color='r')

for tl in ax2.get_yticklabels():

   tl.set_color('r')

ax1.set_ylim(y3[slp2],y3[slp1])

ax2.set_ylim(min(curvature),max(curvature))

pylab.grid(True)

pylab.title('Smoothed plotted pipeline position')

pylab.legend( ('curvature') , loc = 'lower right')

#pylab.savefig('save')
```

```
aBM = [allowable_BM]
a_BM = aBM*len(SurveyKp[slp1:slp2])

ax3 = fig.add_subplot(212)
ax3.plot(SurveyKp[slp1:slp2],BM[:slp2-slp1], 'b', SurveyKp[slp1:slp2], a_BM, 'r' )
ax3.set_ylabel('Bending moment  (Nm)')
ax3.set_xlabel('Kp')
pylab.xlim(min(SurveyKp[slp1:slp2]),max(SurveyKp[slp1:slp2]))
pylab.grid(True)
pylab.title('5th order spline')  #finn tittel
pylab.legend( ('Actual','Allowable') , loc = 'upper right')
#pylab.savefig('save')

pylab.show()

kiko=1
q = raw_input("Are you satisfied? (yes or no) ")
while kiko==1:
   if q== "yes":
      kiko=kiko+1
   else:
      smoothingfactor = input("Enter a new smoothing factor: ")
      tck = interpolate.splrep(x3[slp1:slp2],y3[slp1:slp2], k=5, s=smoothingfactor)
      xnew = linspace(XnewStart, XnewEnd, 1000*(slp2-slp1)+1)
      ynew = interpolate.splev(xnew,tck,der=0)
      yder = interpolate.splev(xnew[::1000],tck,der=1)
```

```
y2der =interpolate.splev(xnew[::1000],tck,der=2)

curvature = (y2der)/((1+yder**2)**(3/2))

abs_curvature = abs(y2der)/((1+yder**2)**(3/2))

BM = [x*EI for x in curvature]  # EI*curvature = BM


fig = plt.figure()

ax1 = fig.add_subplot(211)

ax1.plot(x3[slp1:slp2],y3[slp1:slp2], 'go', xnew,ynew , 'b')

pylab.legend( ('survey data', 'Splines') , loc = 'upper right')

ax1.set_xlabel('Easting (x)')

ax1.set_ylabel('Northing (y)', color='b')

for tl in ax1.get_yticklabels():

   tl.set_color('b')

ax2 = ax1.twinx()

ax2.plot(xnew[::1000],curvature,'r')

ax2.set_ylabel('curvature, 5th order splines', color='r')

for tl in ax2.get_yticklabels():

   tl.set_color('r')

ax1.set_ylim(y3[slp2],y3[slp1])

ax2.set_ylim(min(curvature),max(curvature))

pylab.grid(True)

pylab.title('Smoothed plotted pipeline position')

pylab.legend( ('curvature') , loc = 'lower right')

#pylab.savefig('save')


aBM = [allowable_BM]

a_BM = aBM*len(SurveyKp[slp1:slp2])
```

```
        ax3 = fig.add_subplot(212)
        ax3.plot(SurveyKp[slp1:slp2],BM[:slp2-slp1], 'b', SurveyKp[slp1:slp2], a_BM, 'r' )
        ax3.set_ylabel('Bending moment  (Nm)')
        ax3.set_xlabel('Kp')
        pylab.xlim(min(SurveyKp[slp1:slp2]),max(SurveyKp[slp1:slp2]))
        pylab.grid(True)
        pylab.title('5th order spline')
        pylab.legend( ('Actual','Allowable') , loc = 'upper right')
        #pylab.savefig('save')

        pylab.show()

        q = raw_input("Are you satisfied? ")


else:
    x3.reverse()
    y3.reverse()
    SurveyKp.reverse()
    slp1 = x3.index(start1)
    slp2 = x3.index(end1)
    XnewStart = start1
    XnewEnd = end1
    print '\n'
    print 'The interval being checked: KP '+str(SurveyKp[slp2])+' - KP '+str(SurveyKp[slp1])
    print '\n'
```

```
# EI, Bending stiffness calculation for given pipe interval

E = input("Enter the Young's modulus (N/m^2): ")

Do= input("Enter the outer diameter of the pipeline (m): ")

ts= input("Enter the wall-thickness of the pipeline( m): ")

I= ((math.pi)/64)*(Do**4-(Do-2*ts)**4)

EI=E*I # (Nm^2)

print "Pipeline bending stiffness: "+str(EI)+" Nm^2"

print '\n'


smoothingfactor = input("Enter a smoothing factor: ")


tck = interpolate.splrep(x3[slp1:slp2],y3[slp1:slp2], k=5, s=smoothingfactor)

xnew = linspace(XnewStart, XnewEnd, 1000*(slp2-slp1)+1)

ynew = interpolate.splev(xnew,tck,der=0)

yder = interpolate.splev(xnew[::1000],tck,der=1)

y2der =interpolate.splev(xnew[::1000],tck,der=2)

curvature = (y2der)/((1+yder**2)**(3/2))

abs_curvature = abs(y2der)/((1+yder**2)**(3/2))

BM = [x*EI for x in curvature]  # EI*curvature = BM


fig = plt.figure()

ax1 = fig.add_subplot(211)

ax1.plot(x3[slp1:slp2],y3[slp1:slp2], 'go', xnew,ynew , 'b')

pylab.legend( ('survey data', 'Splines') , loc = 'upper right')

ax1.set_xlabel('Easting (x)')

ax1.set_ylabel('Northing (y)', color='b')

for tl in ax1.get_yticklabels():
```

```
    tl.set_color('b')
ax2 = ax1.twinx()
ax2.plot(xnew[::1000],curvature,'r')
ax2.set_ylabel('curvature, 5th order splines', color='r')
for tl in ax2.get_yticklabels():
    tl.set_color('r')
ax1.set_ylim(y3[slp1],y3[slp2])
ax2.set_ylim(min(curvature),max(curvature))
pylab.grid(True)
pylab.title('Smoothed plotted pipeline position')
pylab.legend( ('curvature') , loc = 'lower right')
#pylab.savefig('save')

aBM = [allowable_BM]
a_BM = aBM*len(SurveyKp[slp1:slp2])

ax3 = fig.add_subplot(212)
ax3.plot(SurveyKp[slp1:slp2],BM[:slp2-slp1], 'b', SurveyKp[slp1:slp2], a_BM, 'r' )
ax3.set_ylabel('Bending moment  (Nm)')
ax3.set_xlabel('Kp')
pylab.xlim(max(SurveyKp[slp1:slp2]),min(SurveyKp[slp1:slp2]))
pylab.grid(True)
pylab.title('5th order spline')  #finn tittel
pylab.legend( ('Actual','Allowable') , loc = 'upper right')
#pylab.savefig('save')   # finn navn for fil

pylab.show()
```

```
kiko=1
q = raw_input("Are you satisfied? ")
while kiko==1:
   if q=="yes":
      kiko=kiko+1
   else:
      smoothingfactor = input("Enter a new smoothing factor: ")
      tck = interpolate.splrep(x3[slp1:slp2],y3[slp1:slp2], k=5, s=smoothingfactor)
      xnew = linspace(XnewStart, XnewEnd, 1000*(slp2-slp1)+1)
      ynew = interpolate.splev(xnew,tck,der=0)
      yder = interpolate.splev(xnew[::1000],tck,der=1)
      y2der =interpolate.splev(xnew[::1000],tck,der=2)
      curvature = (y2der)/((1+yder**2)**(3/2))
      abs_curvature = abs(y2der)/((1+yder**2)**(3/2))
      BM = [x*EI for x in curvature]  # EI*curvature = BM

      fig = plt.figure()
      ax1 = fig.add_subplot(211)
      ax1.plot(x3[slp1:slp2],y3[slp1:slp2], 'go', xnew,ynew , 'b')
      pylab.legend( ('survey data', 'Splines') , loc = 'upper right')
      ax1.set_xlabel('Easting (x)')
      ax1.set_ylabel('Northing (y)', color='b')
      for tl in ax1.get_yticklabels():
         tl.set_color('b')
      ax2 = ax1.twinx()
      ax2.plot(xnew[::1000],curvature,'r')
```

```
ax2.set_ylabel('curvature, 5th order splines', color='r')
for tl in ax2.get_yticklabels():
   tl.set_color('r')
ax1.set_ylim(y3[slp1],y3[slp2])
ax2.set_ylim(min(curvature),max(curvature))
pylab.grid(True)
pylab.title('Smoothed plotted pipeline position')
pylab.legend( ('curvature') , loc = 'lower right')
#pylab.savefig('save')   # finn navn for fil

aBM = [allowable_BM]
a_BM = aBM*len(SurveyKp[slp1:slp2])

ax3 = fig.add_subplot(212)
ax3.plot(SurveyKp[slp1:slp2],BM[:slp2-slp1], 'b', SurveyKp[slp1:slp2], a_BM, 'r' )
ax3.set_ylabel('Bending moment  (Nm)')
ax3.set_xlabel('Kp')
pylab.xlim(max(SurveyKp[slp1:slp2]),min(SurveyKp[slp1:slp2]))
pylab.grid(True)
pylab.title('5th order spline')  #
pylab.legend( ('Actual','Allowable') , loc = 'upper right')
#pylab.savefig('save')

pylab.show()

q = raw_input("Are you satisfied? ")
```

```
   elif alt == '4':    # Design route, as-laid data and survey data
#########################################################################

      print '\n'


      testrun = test()  # dette er kun for å gjøre det lettere å teste programmet

      if testrun == 'y':

         filelocation1 = 'c:\Sivert\DesignR.txt'

         g1 = 3

         h1 = 4

         f1 = open(str(filelocation1))

         filelocation2 = 'c:\Sivert\R.txt'

         g2 = 1

         h2 = 2

         f2 = open(str(filelocation2))

         filelocation3 = 'c:\\Sivert\\new7pt.txt'

         g3 = 2

         h3 = 3

         surveyKp = 1

         f3 = open(str(filelocation3))

      else:

         print '\n'

         print "To open your text file(s) the location of the file(s) is needed. The file(s) needs to be in
txt format.\n"


         filelocation1 = raw_input('Where is the "as-designed" data file located? Example: C:\Program
Files\example.txt    ')

         print '\n'

         g1 = input("Which column from the left contains the Easting(x) coordinates? ")
```

```
        print '\n'
        h1 = input("Which column from the left contains the Northing(y) coordinates? ")
        print '\n'
        f1 = open(str(filelocation1))

        filelocation2 = raw_input('Where is the "as-laid" data file located? Example: C:\Program Files\example.txt    ')
        print '\n'
        g2 = input("Which column from the left contains the Easting(x) coordinates? ")
        print '\n'
        h2 = input("Which column from the left contains the Northing(y) coordinates? ")
        print '\n'
        f2 = open(str(filelocation2))

        filelocation3 = raw_input('Where is the "survey data" file located? Example: C:\Program Files\example.txt    ')
        print '\n'
        g3 = input("Which column from the left contains the Easting(x) coordinates? ")
        print '\n'
        h3 = input("Which column from the left contains the Northing(y) coordinates? ")
        print '\n'
        surveyKp = input("Which column from the left contains the Kp? ")
        print '\n'
        f3 = open(str(filelocation3))

    tmp1 = f1.readlines()

    for i in range(len(tmp1)):
```

```
    try:
        a1 = []
        line = tmp1[i]
        a1 = string.split(line)
        try:
            if float(a1[g1-1])>3000:    # This number (3000), might need to be changed
                x1.append(float(a1[g1-1]))
            else:
                pass

            if float(a1[h1-1])>3000:   # This number (3000), might need to be changed
                y1.append(float(a1[h1-1]))
            else:
                pass
        except ValueError:
            pass
    except IndexError:
        pass

f1.close()

tmp2 = f2.readlines()

for i in range(len(tmp2)):
    try:
        a2 = []
        line = tmp2[i]
```

```
        a2 = string.split(line)
        try:
            if float(a2[g2-1])>3000:    # This number (3000), might need to be changed
                x2.append(float(a2[g2-1]))
            else:
                pass

            if float(a2[h2-1])>3000:        # This number (3000), might need to be changed
                y2.append(float(a2[h2-1]))
            else:
                pass
        except ValueError:
            pass
    except IndexError:
        pass

f2.close()

tmp3 = f3.readlines()

for i in range(len(tmp3)):
    try:
        a3 = []
        line = tmp3[i]
        a3 = string.split(line)
        try:
            if float(a3[g3-1])>3000:    # This number (3000), might need to be changed
```

```
            x3.append(float(a3[g3-1]))
        else:
            pass

        if float(a3[h3-1])>3000:      # This number (3000), might need to be changed
            y3.append(float(a3[h3-1]))
        else:
            pass

        if float(a3[surveyKp-1])<49:   # This number (49), might need to be changed
            SurveyKp.append(float(a3[surveyKp-1]))
        else:
            pass
      except ValueError:
        pass
   except IndexError:
      pass

f3.close()



print "Find the interval you want to check through your easting coordinates."

pylab.plot(x1,y1, 'r', x2,y2, 'g', x3,y3, 'b')
pylab.ylabel('Northing (y)')
pylab.xlabel('Easting (x)')
pylab.grid(True)
```

```
pylab.title('Plotted pipeline position')  #finn tittel
pylab.legend( ('As-designed', 'As-laid', 'Survey') , loc = 'upper right')
#pylab.savefig('save')


pylab.show()


print '\n'


xx3=[]
for i in range(len(x3)):
   xx3.append(float(int(x3[i])))


figo=1
start1 = input('What is the start easting coordinate of your check? ')
while figo==1:
   if start1 in xx3:
      figo=2
      print '\n'
      print str(start1)+" is the start of your interval"
      print '\n'
   else:
      start1=start1+1


lara=1
end1 = input('What is the last easting coordinate of your check? ')
while lara==1:
   if end1 in xx3:
```

```
        lara=2
        print '\n'
        print str(end1)+" is the end of your interval"
    else:
        end1=end1+1

slp1 = xx3.index(start1)
slp2 = xx3.index(end1)

if slp1 < slp2:
    XnewStart = start1
    XnewEnd = end1
    print '\n'
    print 'The interval being checked: KP '+str(SurveyKp[slp1])+' - KP '+str(SurveyKp[slp2])
    print '\n'
    # EI, Bending stiffness calculation for given pipe interval
    E = input("Enter the Young's modulus (N/m^2): ")
    Do= input("Enter the outer diameter of the pipeline (m): ")
    ts= input("Enter the wall-thickness of the pipeline( m): ")
    I= ((math.pi)/64)*(Do**4-(Do-2*ts)**4)
    EI=E*I # (Nm^2)
    print "Pipeline bending stiffness: "+str(EI)+" Nm^2"
    print '\n'
    KpStart = SurveyKp[slp1]
    KpEnd = SurveyKp[slp2]
    smoothingfactor = input("Enter a smoothing factor: ")
    tck = interpolate.splrep(x3[slp1:slp2],y3[slp1:slp2], k=5, s=smoothingfactor)
```

```
xnew = linspace(XnewStart, XnewEnd, 1000*(slp2-slp1)+1)

ynew = interpolate.splev(xnew,tck,der=0)

yder = interpolate.splev(xnew[::1000],tck,der=1)

y2der =interpolate.splev(xnew[::1000],tck,der=2)

curvature = (y2der)/((1+yder**2)**(3/2))

abs_curvature = abs(y2der)/((1+yder**2)**(3/2))

BM = [x*EI for x in curvature]  # EI*curvature = BM


fig = plt.figure()

ax1 = fig.add_subplot(211)

ax1.plot(x3[slp1:slp2],y3[slp1:slp2], 'go', xnew,ynew , 'b')

pylab.legend( ('survey data', 'Splines') , loc = 'upper right')

ax1.set_xlabel('Easting (x)')

ax1.set_ylabel('Northing (y)', color='b')

for tl in ax1.get_yticklabels():

   tl.set_color('b')

ax2 = ax1.twinx()

ax2.plot(xnew[::1000],curvature,'r')

ax2.set_ylabel('curvature, 5th order splines', color='r')

for tl in ax2.get_yticklabels():

   tl.set_color('r')

ax1.set_ylim(y3[slp2],y3[slp1])

ax2.set_ylim(min(curvature),max(curvature))

pylab.grid(True)

pylab.title('Smoothed plotted pipeline position')

pylab.legend( ('curvature') , loc = 'lower right')

#pylab.savefig('save')
```

```
aBM = [allowable_BM]
a_BM = aBM*len(SurveyKp[slp1:slp2])

ax3 = fig.add_subplot(212)
ax3.plot(SurveyKp[slp1:slp2],BM[:slp2-slp1], 'b', SurveyKp[slp1:slp2], a_BM, 'r' )
ax3.set_ylabel('Bending moment  (Nm)')
ax3.set_xlabel('Kp')
pylab.xlim(min(SurveyKp[slp1:slp2]),max(SurveyKp[slp1:slp2]))
pylab.grid(True)
pylab.title('5th order spline')  #finn tittel
pylab.legend( ('Actual','Allowable') , loc = 'upper right')
#pylab.savefig('save')

pylab.show()

kiko=1
q = raw_input("Are you satisfied? (yes or no) ")
while kiko==1:
   if q== "yes":
      kiko=kiko+1
   else:
      smoothingfactor = input("Enter a new smoothing factor: ")
      tck = interpolate.splrep(x3[slp1:slp2],y3[slp1:slp2], k=5, s=smoothingfactor)
      xnew = linspace(XnewStart, XnewEnd, 1000*(slp2-slp1)+1)
      ynew = interpolate.splev(xnew,tck,der=0)
      yder = interpolate.splev(xnew[::1000],tck,der=1)
```

```
y2der =interpolate.splev(xnew[::1000],tck,der=2)

curvature = (y2der)/((1+yder**2)**(3/2))

abs_curvature = abs(y2der)/((1+yder**2)**(3/2))

BM = [x*EI for x in curvature]  # EI*curvature = BM


fig = plt.figure()

ax1 = fig.add_subplot(211)

ax1.plot(x3[slp1:slp2],y3[slp1:slp2], 'go', xnew,ynew , 'b')

pylab.legend( ('survey data', 'Splines') , loc = 'upper right')

ax1.set_xlabel('Easting (x)')

ax1.set_ylabel('Northing (y)', color='b')

for tl in ax1.get_yticklabels():

   tl.set_color('b')

ax2 = ax1.twinx()

ax2.plot(xnew[::1000],curvature,'r')

ax2.set_ylabel('curvature, 5th order splines', color='r')

for tl in ax2.get_yticklabels():

   tl.set_color('r')

ax1.set_ylim(y3[slp2],y3[slp1])

ax2.set_ylim(min(curvature),max(curvature))

pylab.grid(True)

pylab.title('Smoothed plotted pipeline position')

pylab.legend( ('curvature') , loc = 'lower right')

#pylab.savefig('save')


aBM = [allowable_BM]

a_BM = aBM*len(SurveyKp[slp1:slp2])
```

```
        ax3 = fig.add_subplot(212)
        ax3.plot(SurveyKp[slp1:slp2],BM[:slp2-slp1], 'b', SurveyKp[slp1:slp2], a_BM, 'r' )
        ax3.set_ylabel('Bending moment  (Nm)')
        ax3.set_xlabel('Kp')
        pylab.xlim(min(SurveyKp[slp1:slp2]),max(SurveyKp[slp1:slp2]))
        pylab.grid(True)
        pylab.title('5th order spline')
        pylab.legend( ('Actual','Allowable') , loc = 'upper right')
        #pylab.savefig('save')

        pylab.show()

        q = raw_input("Are you satisfied? ")


else:
    x3.reverse()
    y3.reverse()
    SurveyKp.reverse()
    slp1 = x3.index(start1)
    slp2 = x3.index(end1)
    XnewStart = start1
    XnewEnd = end1
    print '\n'
    print 'The interval being checked: KP '+str(SurveyKp[slp2])+' - KP '+str(SurveyKp[slp1])
    print '\n'
```

```
# EI, Bending stiffness calculation for given pipe interval

E = input("Enter the Young's modulus (N/m^2): ")

Do= input("Enter the outer diameter of the pipeline (m): ")

ts= input("Enter the wall-thickness of the pipeline( m): ")

I= ((math.pi)/64)*(Do**4-(Do-2*ts)**4)

EI=E*I # (Nm^2)

print "Pipeline bending stiffness: "+str(EI)+" Nm^2"

print '\n'


smoothingfactor = input("Enter a smoothing factor: ")


tck = interpolate.splrep(x3[slp1:slp2],y3[slp1:slp2], k=5, s=smoothingfactor)

xnew = linspace(XnewStart, XnewEnd, 1000*(slp2-slp1)+1)

ynew = interpolate.splev(xnew,tck,der=0)

yder = interpolate.splev(xnew[::1000],tck,der=1)

y2der =interpolate.splev(xnew[::1000],tck,der=2)

curvature = (y2der)/((1+yder**2)**(3/2))

abs_curvature = abs(y2der)/((1+yder**2)**(3/2))

BM = [x*EI for x in curvature]  # EI*curvature = BM


fig = plt.figure()

ax1 = fig.add_subplot(211)

ax1.plot(x3[slp1:slp2],y3[slp1:slp2], 'go', xnew,ynew , 'b')

pylab.legend( ('survey data', 'Splines') , loc = 'upper right')

ax1.set_xlabel('Easting (x)')

ax1.set_ylabel('Northing (y)', color='b')

for tl in ax1.get_yticklabels():
```

```
    tl.set_color('b')
ax2 = ax1.twinx()
ax2.plot(xnew[::1000],curvature,'r')
ax2.set_ylabel('curvature, 5th order splines', color='r')
for tl in ax2.get_yticklabels():
    tl.set_color('r')
ax1.set_ylim(y3[slp1],y3[slp2])
ax2.set_ylim(min(curvature),max(curvature))
pylab.grid(True)
pylab.title('Smoothed plotted pipeline position')
pylab.legend( ('curvature') , loc = 'lower right')
#pylab.savefig('save')

aBM = [allowable_BM]
a_BM = aBM*len(SurveyKp[slp1:slp2])

ax3 = fig.add_subplot(212)
ax3.plot(SurveyKp[slp1:slp2],BM[:slp2-slp1], 'b', SurveyKp[slp1:slp2], a_BM, 'r' )
ax3.set_ylabel('Bending moment  (Nm)')
ax3.set_xlabel('Kp')
pylab.xlim(max(SurveyKp[slp1:slp2]),min(SurveyKp[slp1:slp2]))
pylab.grid(True)
pylab.title('5th order spline')
pylab.legend( ('Actual','Allowable') , loc = 'upper right')
#pylab.savefig('save')

pylab.show()
```

```
kiko=1
q = raw_input("Are you satisfied? ")
while kiko==1:
   if q=="yes":
      kiko=kiko+1
   else:
      smoothingfactor = input("Enter a new smoothing factor: ")
      tck = interpolate.splrep(x3[slp1:slp2],y3[slp1:slp2], k=5, s=smoothingfactor)
      xnew = linspace(XnewStart, XnewEnd, 1000*(slp2-slp1)+1)
      ynew = interpolate.splev(xnew,tck,der=0)
      yder = interpolate.splev(xnew[::1000],tck,der=1)
      y2der =interpolate.splev(xnew[::1000],tck,der=2)
      curvature = (y2der)/((1+yder**2)**(3/2))
      abs_curvature = abs(y2der)/((1+yder**2)**(3/2))
      BM = [x*EI for x in curvature]  # EI*curvature = BM

      fig = plt.figure()
      ax1 = fig.add_subplot(211)
      ax1.plot(x3[slp1:slp2],y3[slp1:slp2], 'go', xnew,ynew , 'b')
      pylab.legend( ('survey data', 'Splines') , loc = 'upper right')
      ax1.set_xlabel('Easting (x)')
      ax1.set_ylabel('Northing (y)', color='b')
      for tl in ax1.get_yticklabels():
         tl.set_color('b')
      ax2 = ax1.twinx()
      ax2.plot(xnew[::1000],curvature,'r')
```

```
        ax2.set_ylabel('curvature, 5th order splines', color='r')
        for tl in ax2.get_yticklabels():
            tl.set_color('r')
        ax1.set_ylim(y3[slp1],y3[slp2])
        ax2.set_ylim(min(curvature),max(curvature))
        pylab.grid(True)
        pylab.title('Smoothed plotted pipeline position')
        pylab.legend( ('curvature') , loc = 'lower right')
        #pylab.savefig('save')

        aBM = [allowable_BM]
        a_BM = aBM*len(SurveyKp[slp1:slp2])

        ax3 = fig.add_subplot(212)
        ax3.plot(SurveyKp[slp1:slp2],BM[:slp2-slp1], 'b', SurveyKp[slp1:slp2], a_BM, 'r' )
        ax3.set_ylabel('Bending moment  (Nm)')
        ax3.set_xlabel('Kp')
        pylab.xlim(max(SurveyKp[slp1:slp2]),min(SurveyKp[slp1:slp2]))
        pylab.grid(True)
        pylab.title('5th order spline')
        pylab.legend( ('Actual','Allowable') , loc = 'upper right')
        #pylab.savefig('save')

        pylab.show()

        q = raw_input("Are you satisfied? ")
else:
```

```
        print "ERROR"
if __name__ == '__main__':
    main()
```