Pipe-Soil Interaction Model for Lateral Buckling of Pipelines Including Berm Formation

Unrestricted

SR.16.12645

September 2016

Unrestricted **SR.16.12645**

Pipe-Soil Interaction Model for Lateral Buckling of Pipelines Including Berm Formation

By

Ralf Peek, EP Projects (SIEP-EPT-PNR)
Heedo Yun, EP Projects (SIEP-EPT-PDF)

Contributions from:

Ruben van Schalkwijk (RvS Engineering, The Netherlands)

Sponsor:	John Pelletier
Reviewed by:	Ed Phifer
Approved by:	George Rodenbusch, Jan Oude Hengel
Date of issue:	5/31/2006
Account code:	Altas No. U/000945/04-0010
ECCN number:	EAR 99

SHELL INTERNATIONAL EXPLORATION AND PRODUCTION Inc, Houston

Further copies can be obtained from the Global EP Library, BTC

SUMMARY

Global finite element modeling of pipe lateral buckling due to thermal expansion is usually performed with beam-type elements to represent the pipe, and a Coulomb frictional model to represent the interaction between the pipe and the seabed. According to such frictional models the highest strains due to lateral buckling occur when the pipe is first heated. Subsequent cycles increase the amplitude of the buckle, but also the wavelength of the buckle, so that the maximum curvature, the maximum strains, and the stress ranges decrease for subsequent cycles. However in reality soil berms tend to form when the pipe buckles laterally. Upon cooling, these soil berms remain in place following the profile of the buckled pipeline when it was first heated. As a result, upon subsequent cycles the pipe will tend to adopt a shape similar to that at the first cycle. This has a significant impact on the stress ranges for low cycle fatigue analysis. For this reason a model is developed that explicitly accounts for berm formation, and can track the formation and location of any number of berms (up to a user-specified maximum number). Both formation as well as coalescence of berms is modeled. The model is 2-dimensional in that all pipe deformations take place in the horizontal plane.

By including an initial berm, the berms model can also be used to model an initial higher break-out resistance due to embeddment of the pipe.

The berms model includes considerable flexibility in describing the evolution of berm volume with pipe displacement, and the relationship between berm resistance and berm volume. These relationships are described by linear interpolation between any number of user-specified points. Coalescence of berms is accounted for by adding the volumes of the coalescing berms.

The berms model is complemented by an anisotropic frictional model for the frictional component of the resistance. This allows different friction coefficients to be specfied in the lateral and axial directions. It also allows coupled or uncoupled behaviour to be modeled for the axial and lateral directions. For the coupled model an elliptical slip surface is constructed which is treated very much like a yield surface in plasticity theories, which the direction of slip being normal to the slip surface as for associated plastic flow in plasticity theories.

Both the berms model and the anisotropic frictional model have been adapted so that they can be used as a user-subroutine in the ABAQUS commercial general purpose finite element analysis program. An example shows good agreement between the results calculated with ABAQUS and those from the program (NPEX) for which the models were first developed. Instructions for use of these models within ABAQUS are included.

TABLE OF CONTENTS

1. INTRODUCTION

Lateral buckling of pipelines can be a cost-effective way to accommodate thermal expansion and limit the expansion at the ends of pipelines (affecting spools or jumpers), especially in subsea pipelines, where burial for protection and/or insulation is not required. However one must ensure that the bending moments and curvatures due to such lateral buckling are not excessive, so that local buckling/wrinkling of the pipe wall, and/or fracture at a girth weld is avoided. This has already lead to failures of at least 2 (non-Shell) pipelines. Therefore a realistic modelling capability for lateral buckling is essential in order to correctly quantify the risk that lateral buckles become over-stressed. Yet it has been found by the authors that the pipe-soil interaction element in ABAQUS does not properly or fully describe the pipe-soil interaction mechanism. In order to overcome this difficulty, a user subroutine is developed in this study to account for different resistances in the axial and lateral directions, as well as the interaction between these two directions.

Another important aspect of the soil resistances to lateral buckling is the formation of soil berms. Lateral movement of a pipeline lying on a seabed (as a result of lateral buckling or subsequent motion) causes the formation and growth of a soil berm on one side of a pipeline. Such berms not only increase the resistance to lateral movements, but also affect the cyclic response in a way that tends to increase the stress range due to shut-down and restart cycles. Therefore it was decided during the course of the work to add the modelling of berms to the scope. The model developed also allows an initial berm to be used to represent embeddment and initial break-out of the pipe.

When a pipline buckles laterally on a sandy or clayey seafloor, it tends to push some soil laterally, which forms a berm against the pipeline, thereby increasing the resistance to lateral movement. Upon cooling the line will tend to pull back from the berm, but at the next heating cycle there is an increase in resistance when the pipe encounters the berm. Thus on the 2nd and subsequent heating cycles, the berm tends to make the pipe into the same shape to which it has buckles at the 1st cycle. This behaviour differs from that for a purely frictional seafloor, with no berm formation, where the maximum lateral displacement at the buckle tends to grow with additional cycles, but the peak bending moment decreases.

Although berm formation could also be accounted for by using higher effective friction factors, a proper model for berm formation has the following advantages:

1. For fatigue analyses the berms model gives a much more realistic representation of the stress histories. Using a purely frictional model with an equivalent friction factor gives non-conservative results because:
    a. The decrease in peak bending moment for subsequent cycles that is predicted by the purely frictional model does not materialise because berms tend to make the pipe deform at each cycle into the same shape as for the first cycle.
    b. The equivalent fricitonal model will tend to over-estimate the soil resistance to pull back of the pipe upon cooling.
2. A number of cycles at a lower temperature, followed by a higher temperature, can cause increased peak bending moments, as the pipe wants to break through the berm built up in the previous cycles. This can be important for design, and can only be predicted by modeling the berm formation.

3. By modeling initial embeddment as initial berms, the additional break out resistance due to embeddment can be modeled. This can have a significant effect on the buckling mode (favouring a single lobe mode rather than a 3-lobe mode, as in Kerr's mode 3), and resulting in higher peak bending moments. Not modeling the berm can be non-conservative.

The proposed model explicitly accounts for the presence of berms. Depending on the displacement history, any number of berms can form. In general a change in the direction of movement of the pipe will result in a berm being left behind and a new one being started. On the other hand a berm that is pushed over by the pipe may engulf other berms, thereby increasing the volume of the berm.

In general the lateral displacement history will determine the number of berms that are formed, and the number of history parameters that need to be stored to describe the current state of the system.

The mathematical formulation of the model has been developed and documented in this report. The berm model is implemented in ABAQUS in association with the existing pipe-soil interaction element of ABASUS - PSI element. The formation of soil berms as a result of pipe-soil interaction is calculated in the user supplied material subroutine – UMAT of the PSI element. User Manual, Example Solutions, and the source code list of the user subroutine UMAT are provided in the Appendices.

The model has been adapted for use within ABABQUS. The user routine developed for this purpose is being tested against the prototype version (which works within an in-house code "NPEX").

If permission is granted by Shell Oil, it is intended to make the model developed available throughout the Shell Group (i.e. SIEP and its Affiliates), and also to the pipeline design contractors who are selected to perform lateral buckling design, with the provision that the contractors may only use the model for design of pipelines in which a Shell Affiliate has a significant equity stake.

2. MODEL DESCRIPTION

Two types of the pipe-soil interaction mechanisms are included in the model:

- Coulombic Friction Model, and
- Berm Formation Model.

2.1. Description of the Friction Model

2.1.1. Isotropic Friction Model

In many systems involving sliding interfaces the friction force (F) is proportional to the normal contact force (N).

$$F = \mu N \tag{2.1.1}$$

in which μ is the coefficient of friction. This is often termed Coulomb Friction or Dry Friction. Equation (2.1.1) represents the isotropic friction, where frictional characteristics are the same in all directions.

2.1.2. Anisotropic Friction Model

For a three dimensional problem there are two orthogonal components of the friction force – F_{Axial} and $F_{Lateral}$ for pipelines, and the corresponding coefficients of friction – μ_{Axial} and $\mu_{Lateral}$. In general pipeline friction is not isotropic. Frictional characteristics of sliding pipelines are represented in two models - uncoupled model and coupled model.

2.1.2.1. Uncoupled Model

In the uncoupled model the frictional responses in each orthogonal directions are unaffected by each other. The maximum friction force of the uncoupled model is represented as follows:

$$F_{Axial} = \mu_{Axial} N \tag{2.1.2}$$

$$F_{Lateral} = \mu_{Lateral} N \tag{2.1.3}$$

2.1.2.2. Coupled Model

For many three dimensional systems friction in one direction affects the friction in the other direction. The initiation of sliding is often described by an elliptical failure envelop on the friction force space:

$$(F_{Axial} / \mu_{Axial})^2 + (F_{Lateral} / \mu_{Lateral})^2 = N^2 \tag{2.1.4}$$

Direction of slip is normal to the slip surface of (2.1.4) (this is termed the associated flow). Therefore, slip is not always in the direction of the friction force. In the special isotropic friction case, where the two friction coefficients are equal, the slip surface becomes a circle, and slip surface of (2.1.4) becomes equivelant to (2.1.1).

2.2. Mathematical Formulation of the Berms Model

2.2.1. Description of the Berms Model

The model provides lateral resistance as a functional of the lateral displacement history only. It does not consider coupling of axial and lateral effects.

Each berm is described by its volume V, and location x. Here "berm volume" means the volume per unit length along the pipeline, which is really the cross sectional area of the berm.

Berms can be on either side of the pipe: the ones that resist increases in the lateral pipe displacement u will be referred to as "positive berms" whereas "negative berms" are on the other side of the pipe. A variable ξ is introduced, which takes the value $\xi = 1$ for a positive berm, and $\xi = -1$ for a negative berm.

For simplicity, consider first berms that are concentrated at a point. As the pipe moves it pushes a berm in front of it. The resistance q provided by the berm depends on the volume V of the berm, which in turn changes in a specified way with increasing displacement, in such a way that for large displacements it converges to an equilibrium, steady state value V_{equil}, from either above, or from below, as defined by two volume-displacement curves.

If the pipe changes the direction of motion, it leaves a berm behind. As a result, there may be any number of berms on the seafloor. If the pipe runs into a berm, it engulfs it. In the process, the volume of the current berm being pushed by the pipe is increased by the volume of the engulfed berm. This gives an upwards jump in the resistance to movement, with the new resistance being based on the sum of the two volumes. This new resistance will then again evolve acoording to one of the two (upper or lower) volume-displacement curves until the next berm is encountered, or the direction of movement changes.

The "input parameters" to this basic model are 3 functions: a berm volume-resistance relation, and upper and lower volume-displacement relations, that describe how a berm volume larger or smaller than the equilibrium value would converges to the equilibrium value as the displacement increases.

In applications it is more convenient to specify resistance-displacement relations, than volume-displacement relations, since the former can be measured directly during a test, and is more often displayed in publications or reports about such tests. Clearly the volume-displacement relation can be derived from the resistance-displacement relation, and vice-versa, using the volume-resistance relations. Thus the implementation of the model is based on specifying a volume-resistance relation, and two force-displacement relations as input to the model. These relationships will be referred to as input functions. In an implementation of the model they may be referred to as user-defined functions (e.g. by piecewise linear interpolation between values provided by the user). These input functions need to be defined from two tests: one starting with a large berm and monitoring how berm volume and resistance to movement evolves, and another starting with no berm.

2.2.2. Formulation of the Berms Model

Concentrating the berm at a point, as assumed in the previous section, leads to a jump in the resistance-displacement relation whenever a berm is engulfed. Such jumps are not only physically unrealistic, but also cause convergence difficulties in numerical computations. In

this section the jumps are removed by introducing a mobilisation displacement into the formulation.

The mobilisation displacement is assumed to be a function of berm volume, V, so that it can be written as,

$$u_{mob} = u_{mob}(V) \tag{2.2.1}$$

where

u_{mob} = mobilisation displacement to mobilise the full resistance of the berm;

$u_{mob}(.)$ = input function[i]; gives mobilisation displacement for the berm resistance as a function of berm volume.

If a berm is being pushed over by the pipe it is referred to as the sliding berm. The resistance provided by this berm is then written as,

$$q = \xi\, q(V) \tag{2.2.2}$$

where

q = resistance provided by berm (force per unit length), positive when the force which the pipe exerts on the soil is in the increasing x and u direction.

q(.) = input function, giving the magnitude of the berm resistance as a function of berm volume V.

Consider the evolution of berm volume as the pipe pushes it along. If one starts with a small berm, the berm will increase in volume until it reaches a steady-state condition at which the berm remains constant. Tests suggest that this occurs over a few pipe diameters. This means that there is an equilibrium berm volume, V_{equil}. On the other hand, if the berm is larger than the equilibrium size, then material from the berm will tend to be left behind, as the pipe is riding over the berm. In this case the volume of the berm is decreasing as the pipe continues to push the berm over. This berm volume evolution is best described by providing two force-displacement relationships that can readily be determined experimentally:

a) The first force-displacement relationship to be provided is determined by pushing over a very large berm (it must be at least as large as the largest berm that will encountered for the displacement history for which the model is to be applied).

b) The second force-displacement relationship is determined by starting at zero berm volume, and applying displacements until an equilibrium value of the resistance is reached.

These two force displacement relations can be written as

$$R = R(u) \tag{2.2.3}$$

and

$$r = r(u), \tag{2.2.4}$$

respectively, where

R = resistance for berm volume exceeding the equilibrium value (decreasing monotonically with increasing displacements u and converging to a value $R = q_{equil}$ as $u \rightarrow \infty$);

---

[i] "User-defined function" means a function defined by input parameters to the model, e.g. by piecewise linear interpolation between points points defined as part of the input parameters.

r = resistance for berm volume smaller than the equilibrium value (increasing monotonically and converging to a value $r = q_{equil}$ as $u \rightarrow \infty$);

R(.) = user-specified function, determined from a test starting with a very large berm volume, and pushing it over with the pipe until an equilibrium resistance q_{equil} is reached;

r(.) = user-specified function, determiend from a test starting with zero berm volume, and increasing the displacements of the pipe until an equilibrium resistance is reached.

u = lateral pipe displacement (During tests to determine the functions R(.) and r(.), the displacement u is the pipe displacement. However in the application of the model for other displacement histories the argument of the functions r(.) and R(.) will in general differ from the pipe displacement u. In defining r(.), r(0) should be taken to be the resistance at zero berm volume. This is the minimum resitance, and may include the sand-pipe friction, if this is not included separately. On the other hand for the function R(.), the choice of the point where u=0 does not matter: e.g. one can take the displacement to be zero at the largest expected berm volume. One might then also extrapolate the function R(.) for negative displacements, to represent larger than expected berms[ii].

The above resistance functions implicity also define the evolution of berm volume as a function of displacement. Indeed, the evolution of the volume can be written as

$V = V(R(u))$ for above equilibrium berm volumes (2.2.5)

$V = V(r(u))$ for below-equilibrium berm volumes (2.2.6)

where

V = berm volume, as before

V(.) = inverse function to the function q(.) in q = q(V) defined in Eq. 2.2; this provides the volume of the berm as a function of the resistance

Although the model accounts for berm volume reduction, as material is left behind, it does not account for the influence the material left behind may have on the pipe, if during a subsequent cycle the pipe comes back to the left-behind material. Thus in essence the model can represent the resistance forces involved in going over a berm, but once the pipe has gone over a berm, the original berm is lost.

So far only the resistance against movement provided a single berm being pushed along by the pipe has been described. In general several berms will be present. In this case the status of each berm may be classified as follows:

a) The berm is *sliding* if the pipe has reached the location of the berm (u=x), and is pushing the berm along. The sliding berm offers the full resistance q=q(V), as defined by Eq. 2.2. There can be only one sliding berm at the time, because any other berm that the pipe runs into coalesces into the sliding berm.

b) The berm is *engaged* if the pipe comes within one mobilisation displacement of it. In this case the berm starts to offer some resistance to the pipe. Over the mobilisation displacement the resistance must increase in a continuous fashion from that provided by the sliding berm just before the engagement, that that of the combined berms. Engaged berms do not change in volume V or location x, until the become engulfed by the sliding berm. As a result the energy absorbed by an

[ii] This would ensure that a solution the resistance model will not fail if larger than expected berm volumes develop, even though the function R(.) is not properly defined from experimental data in this range.)

engaged berm as the pipe moves towards it elastically recoverable if the pipe changes the direction of motion.

c) The berm is *active* if it is sliding or engaged. Otherwise it is *inactive*.

Consider several berms on the same side of the pipe. Starting with the berm closest to the pipe, the locations of the berms will be denoted by $x_1, x_2, \ldots$(with $\xi\, x_1 < \xi\, x_2 < \xi\, x_3 < \ldots$), and the volumes denoted by $V_1, V_2, \ldots$, respectively[iii].

In addition to the actual berm volumes V, it is convenient to work with hypothetical berm volumes. These represent the volume of the sliding berm if the pipe movement were to continue to and past the berm considered. There are two hypothetical berms volumes: V' represents the sliding berm volume just before the pipe reaches the berm considered, and V" just after. The guiding principle is that when two berms coalesce their volume is added. Thus,

$$V_i'' = V_i' + V_i \qquad (2.2.7)$$

in the above

V_i' = hypothtical volume that the berm being pushed by the pipe would reach just before the pipe reaches the i^{th} berm;

V_i'' = hypothetical berm volume that would develop if the pipe were pushed to just beyond the i^{th} berm location;

V_i = current volume of i^{th} berm (before the pipe reaches it).

The procedure to calculate V_{i+1}' from V_i'' is as follows:

1. Start with V_i''.
2. Calculate the resisting force from $q_i'' = q(V_i'')$.
3. Calculate the displacement on the reference curves defined by Eqs. 2.3 and 2.4 from

   $u_i'' = R^{-1}(q_i'')$ if $q_i'' > q_{equil}$

   $u_i'' = r^{-1}(q_i'')$ if $q_i'' < q_{equil}$

   $u_i'' = \infty$ if $q_i'' = q_{equil}$ (no u_i value needed subsequently in this case)

   in which $R^{-1}(.)$, and $r^{-1}(.)$ denote the inverse functions to R(.) and r(.), as defined in Eqs. 2.3 and 2.4, respectively. This represents the displacement on the reference curve when the pipe is the ith berm.
4. Calculate the displacement on the reference curve when the pipe reaches the $(i+1)^{th}$ berm from

   $u_{i+1}' = u_i'' + \xi\,(x_{i+1} - x_i)$
5. Calculate the corresponding soil resistance q_{i+1} from

   $q_{i+1}' = R(u_{i+1}')$ if $q_i'' > q_{equil}$

   $q_{i+1}' = r(u_{i+1}')$ if $q_i'' < q_{equil}$

[iii] For simplicity in notation, the same symbols are used to describe berms on both sides of the pipe. Unless otherwise noted equations or inequalities given in what follows apply for either side of the pipe. For instance the inequality $\xi\, x_1 < \xi\, x_2$, really implies two inequalities, one for each side of the pipe.

$q_{i+1}' = q_{equil}$ if $q_i'' = q_{equil}$

6. Calculate the berm volume just before reaching the $(i+1)^{th}$ berm from

$V_{i+1}' = V(q_{i+1}')$

The above procedure can also be defined by a single equation, as in

$$V_{i+1}' = V(R(R^{-1}(q(V_i'')) + \xi (x_{i+1} - x_i))) \quad \text{if } q(V_i'') > q_{equil}$$

$$V_{i+1}' = V(r(r^{-1}(q(V_i'')) + \xi (x_{i+1} - x_i))) \quad \text{if } q(V_i'') < q_{equil}$$

$$V_{i+1} = V_{equil} \quad \text{if } q(V_i'') = q_{equil} \qquad (2.2.8)$$

Once V_{i+1}' is calculated from the Eq. 2.8 or the above procedure, V_{i+1}'' may be calculated from Eq. 2.7. Thus given V" for the first berm all other V" values can be calculated, by repeated applications of Eqs. 2.8 and 2.7.

It remains to define V" for the first berm on each side of the pipe, i.e. to define V_1''. For the side towards which the pipe is moving, V_1'' is simply the volume of the sliding berm; i.e. $V_1''=V_1$.

For the other side, one considers a ficticious change in the direction of movement. Such change in direction would create a new berm at the pipe location, with zero initial volume. The parameters for this new berm are then given by,

$$x_0 = u \qquad (2.2.9)$$

$$V_0'' = V_0 = 0 \qquad (2.2.10)$$

and can be used to initiate the calculation of the ficticious volumes of all berms on the side which the pipe is moving away from.

Once all current berm parameters are defined (including the ficticious ones), the total resisting force is calculated from,

$$q = q_{sliding} + \Sigma\, q_{engaged} \qquad (2.2.11)$$

in which

$q_{sliding}$ = contribution from the sliding berm (i.e. the berm for which x=u);

$q_{engaged}$ = contribution from an engaged berm (i.e. a berms for which $\xi\, x - u_{mob} \le \xi\, u \le \xi\, x$);

Σ = denotes summation over all engaged berms

For the sliding berm, the resistance is calculated from

$$q_{sliding} = \xi\, q(V_{sliding}) \qquad (2.2.12)$$

where $V_{sliding}$ is the actual volume of the sliding berm, and for the engaged berms, it is

$$q_{engaged} = (q_i'' - q_i')\, (\xi + (u - x_i)/u_{mob}) \qquad (2.2.13)$$

in which

$$u_{mob} = u_{mob}(V_i'') \qquad (2.2.14)$$

2.2.3. Implementation for a Finite Increment Δu of the Berm Model

For implementation of the model, consider a finite displacement increment from a diplacement u_0 to a displacement u. The model is exact for arbitrarily large displacement increments, and does not require any integration. This section describes how the berm parameters are updated for an arbitrary increment,

$$\Delta u = u - u_0 \tag{2.2.15}$$

The side towards which the pipe moves is characterised by $\xi\,\Delta u > 0$. Here berms may be engulfed, but for those that are not engulfed none of the berm parameters (real or ficticious) do not change as a result of the increment. It is expedient to calculate these parameters based on conditions at the beginning of the increment. For this purpose the calculation of the ficticious berm parameters is initiated by

$$V_1'' = V_1 \qquad \text{where} \quad \xi\,\Delta u > 0 \tag{2.2.16}$$

$$x_1 = u_0 \qquad \text{where} \quad \xi\,\Delta u > 0 \tag{2.2.17}$$

in which V_1 refers to the volume of the sliding berm at the beginning of the loadstep. The calculation of the remaining ficticious berm parameters using Eqs. 2.8 and 2.7, is also based berm parameters at the beginning of the increment.

On the other side, i.e. the one the pipe is moving away from, the ficticious volumes should be based on conditions at the end of the increment. They should therefore be initiated by

$$V_0'' = 0 \qquad \text{where } \xi\,\Delta u < 0 \tag{2.2.18}$$

$$x_0 = u \qquad \text{where } \xi\,\Delta u < 0 \tag{2.2.19}$$

and propagated using Eqs. 2.8 and 2.7, and the fact that none of the real berm parameters change during the increment. (The ficticious parameters do change, however, and it it therefore important to evaluate them for the end of the increment.)

It remains only to update the parameters for the sliding berm. For this purpose consider first the case when the sliding berm does not absorb other berms during the increment. For this case, the contribution to the resistance from the sliding berm is given by

$$q_{sliding} = \xi\, R(\, u_1'' + \xi\,(u - x_1\,)) \qquad \text{if } q_1'' > q_{equil} \tag{2.2.20}$$

$$q_{sliding} = \xi\, r(\, u_1'' + \xi\,(u - x_1\,)) \qquad \text{if } q_1'' < q_{equil} \tag{2.2.21}$$

$$q_{sliding} = \xi\, q_{equil} \qquad \text{if } q_1'' = q_{equil} \tag{2.2.22}$$

in which u_1'' and x_1 are based on conditions at the start of the increment, by following the procedure for Eq. 2.8, starting with the values from Eqs. 2.18 to 2.19.

The above applies if the sliding berm does not absorb other berms during the increment, i.e. if no berms are engulfed. Only berms on the side towards which the pipe moves can become engulfed. The locations x_i of engulfed berms at the beginning of the increment satisfy,

$$\xi\, x_i \le \xi\, u \tag{2.2.23}$$

Suppose one or more berms are engulfed in during the increment, and let j denote the identification number of the last one that is engulfed, i.e. the one for which $\xi\, x_i$ is largest, but still less than $\xi\, u$. In this case the resistance at the end of the increment is given by

$$q_{sliding} = \xi R(u_j'' + \xi (u - x_j)) \quad \text{if } q_j'' > q_{equil} \tag{2.2.24}$$

$$q_{sliding} = \xi r(u_j'' + \xi (u - x_j)) \quad \text{if } q_j'' < q_{equil} \tag{2.2.25}$$

$$q_{sliding} = \xi q_{equil} \quad \text{if } q_j'' = q_{equil} \tag{2.2.26}$$

in which again u_j'' and x_j are based on conditions at the start of the increment. Once the resistance from the sliding berm is calculated its volume may be determined from

$$V_{sliding} = V(q_{sliding}) \tag{2.2.27}$$

where V(.) is the input function defining the volume-resistance relation.

Once all calculations for the increment are complete, the berms are re-numbered sequentially with i=1 for the sliding berm. No renumbering or other updates are required for the berms the pipe has moved away from.

2.2.4. Axial and Lateral Coupling of the Berm Model

As described this berm formation does not account for such coupling. If it is desired to introduce such coupling, an appropriate way do this is to introduce a purely frictional resistance component for which the axial and lateral effects are coupled, and an additional component due to berm formation, for which the lateral resistance can reasonably be assumed to be uncoupled from axial movements. In this case one would specify zero lateral resistance in the berm formation model when the volume of the berm is zero. The frictional resistance at zero berm volume can then be included in a coupled frictional model.

2.2.5. Implementation Considerations of the Berm Model

A difficulty in the implementation of the above-described berms model is the the number of berms is not known a priori. It depends on the displacement history of the pipe. This can make storage allocation for the state parameters describing the system challenging. To simplify this, a maximum number of berms should be specified a priori and storage allocated for it. If this maximum number is exceeded, the system should be programmed to "forget" the outermost berm. As long as the berm to be forgotten is not active this will not give rise to any discontinuity in the response. In the current implementation the total volume of the forgotten berms on each side is calculated. When the forgotten volume is significant, it is advisable to repeat the analysis accounting of a larger maximum possible number of berms.

3. COMPUTER CODE

Both the friction model and the berm formation model are implemented in the user subroutine UMAT of the PSI element of ABAQUS. The pipe-soil interaction must be modeled using the PSI elements and interaction characteristics (of frictional and berm formation) must be defined in the user subroutine UMAT. For more details refer to the User's Guide in Appendix 1. Input files of example prlblems are listed in Appendix 2. Fortran source code of the user subroutine UMAT is listed in Appendix 3.

4. EXAMPLE SOLUTIONS

4.1. Definition of Model

The berms formulation of the previous section has been implemented as an element in the NPEX program. This program was developed as the University of Michigan, in a modular structure, so that new types of elements to be added. This section presents the input and results from a lateral buckling calculation with NPEX, using the berms model developed.

A heavily concrete-coated, 14-inch pipe is considered, with properties and loading conditions defined in Table 4.1. The structural strength of the coating is neglected. Only its weight and buoyancy are considered in the determination of the submerged weight of the pipe. The steel is assumed to be elastoplastic according to the Von Mises yield criterion with isotropic strain hardening according to the stress-strain curve shown in Fig. 4.1.

Buckles at a uniform spacing of 800m are considered, and symmetry about the apex of the buckle, as well as the midpoint between adjacent buckles is exploited so only the pipe from the apex of the buckle (x=0) to the midpoint between buckles (x=400m) needs to be modelled.

The pipe elements used are based on the moderate-deflection beam theory, in which the axial strains at any point on the cross section of the pipe care calculated from

$$\varepsilon = u' + \tfrac{1}{2} w'^2 - w'' y \qquad (4.1)$$

in which u and w denote the displacement components in the axial and lateral directions; y denotes the distance of the point considered from the centroidal axis normal to the plane of bending (i.e. from the neutral axis under pure bending); and a prime, as in (.)' denotes differentiation with respect to the axial coordinate, x.

The NPEX pipe elements do account for the effect of internal pressure, but they are based on small strain theory. I.e. the difference between true stresses (defined as force per unit area after deformation) and nominal stress (defined as force per unit area before deformation) is neglected. (Indeed if these differences become important, it is questionable whether beam theory could still provide a good approximation, except perhaps for special cases. Thus the small strain approximation is judged to be consistent with the beam theory approximation.)

The discretisation used is as follows: the first 90m from the apex of the buckle (from x=0 to x=90m) are modelled with 200 elements of a constant length, L_e=90m/200=0.45m. This represents the region where significant lateral displacements could develop. Beyond this point, only axial displacements are expected, feeding into the buckle. This axial feed region (from x=90m to x=400m) is modelled with 100 of the same elements, but their length increases in constant steps from L_e=0.45m at x=90m, to L_e=5.7m at x=400m.

The soil resistance is modelled by adding a frictional component of the resistance, to the resistance due to the berms.

For the berms component the model described in this report is used. The input functions are defined as follows:

- The mobilisation displacement function, $u_{mob}=u_{mob}(V)$, from Eq. 2.1, is defined by taking the mobilisation displacement to be a constant u_{mob}=1cm.

- The berm resistance-volume relation, q=q(V), from Eq. 2.2, is defined by assuming that the resistance q is proportional to the berm volume V. Under this assumption it does not matter what the constant of proportionality is, so q=V is used.
- The upper resistance-displacement relation, R=R(u), in Eq. 2.3, is defined by piecewise linear interpolation from the values shown in Table 4.2. If the displacement falls outside the range of u values, linear extrapolation is used. A plot of this relation can be found in Fig. 4.2.
- The lower resistance-displacement relation, r=r(u), in Eq. 2.4, is defined by piecewise linear interpolation from the values shown in Table 4.3. If the displacement falls outside the range of u values, linear extrapolation is used. A plot of this relation can be found in Fig. 4.2.

In addition, to represent pipe embeddment, an initial berm volume corresponding to a resistance of

$$q_{init} / W = 1.12 \qquad (4.2)$$

was used, where $q_{init} = q(V_{init})$ represents the resistance based on the initial berm volume, V_{init}, and W denotes the submerged weight of the pipe, which is W=3.389kN/m.

For the frictional component, a simple uncoupled[iv] elastic-perfectly plastic model is used for the axial and lateral friction force, with friction coefficients given in Table 4.1, and mobilisation displacements of 2cm and 3cm for the axial and lateral directions, respectively.

The loading sequence, and buckling initiation is as follows:

1. The pipe is assumed to be initially straight at stress free. All boundary conditions and soil resistance elements are active, but not loads have been applied yet.
2. To simulate laying of the pipe, the external pressure is applied together with an equivalent temperature change, to ensure that the effective axial force in the pipe corresponds to the on-bottom effective lay tension, N_{lay}. This equivalent temperature increment is calculated based on the assumption of elastic behaviour of the pipe, and using thin-walled shell theory based on the diameter to the midsurface[v], to obtain

   $$\Delta T_{lay} = -[N_{lay} + (1-2\nu)\, \tfrac{1}{4}\, \pi\, (D-t)^2\, p_{ext}\,] /(EA\, \alpha\,)$$

   in which ΔT_{lay} represents a decrease in temperature to compensate for the change in length of the pipe that occurs before touchdown due to the lay tension and the external pressure; $A=\pi\,(D-t)\,t$ is the cross sectional area of the pipe; $p_{ext}=\gamma_{fw}\, G_{sw}\, h_{sw}$ is the external presssure; and the remaining symbols are defined in Table 3.1.

   (An alternative to this approach, is to apply the external pressure and lay tension before activating the

[iv] The NPEX program also includes a coupled model based on an elliptical slip condition (i.e. slip boundary on a plot of axial vs. lateral friction force is an ellipse). Therein slip is modelled in the same way as the plastic flow of metals, using a flow rule with a direction of slip that is normal to the slip surface. However for this example, probabily because of the high axial friction coefficient, it was found that the coupled model could result in the largest buckling lateral displacement occurred away from the apex of the intended buckle (x=0), where there was more axial slip. For this reason the coupled model is not used for this example.

[v] This is not the most accurate approximation, but it is the same approximation made in the FE formulation, by integration of the virtual work at the midsurface of the pipe wall, and therefore it is the approximation that will result in the correct effective axial force in the FE analysis.

soil resistances and boundary conditions for axial displacements, but the current NPEX program does not allow changes in boundary conditions and the existence of elements over time.)

3. An imperfection to initiate the lateral buckle is introduced by pushing the pipe laterally at the apex in a displacement controlled fashion, using a bumper element. This bumper element connects the node at the apex of the buckle to a bumper node with a specified lateral displacement, which during this step increases from 0 to 0.12m, and subsequently remains at 0.12m. Whenever the lateral displacemen of the pipe exceeds that of the bumper node, the bumper element is not active. However, any penetration of the bumper node into the pipe is resisted by a stiff linear spring (stiffness of 6.16MN/m).
4. The internal pressure is applied.
5. Apply the temperature change.[vi]

4.2. Results

The calculations have been performed for 5 cycles of heating to 77°C above ambient, followed by cooling back to the ambient temperature. The results are shown in Figs. 4.3 to 4.17. Therein integer cycle numbers 1,2,3,… represent the end of the heating cycle (temperature rise of 77°C above ambient), whereas cycle numbers 1.5, 2.5, 3.5, … represent the end of the cooling cycle (ambient temperature).

The following observations can be made from the plots of the results obtained by using NPEX code:

1. Plasticity occurs only at the first heating cycle; subsequent cycles are elastic (Fig. 4.17).
2. The maximum axial strains are 0.73% in compression (Fig. 4.13), and 0.52% in tension (Fig. 4.14). These occur at the end of the first heating cycle, at the apex of the buckle at diametrically opposite sides. In reality these strains could be affected by point-to-point variations in wall thickness and/or yield strength, and by the stiffening effect of the concrete coating, neither of which is included in this analysis.
3. The lateral displacement at the apex of the buckle in the hot condition increases slightly from cycle to cycle (Figs. 4.3, 4.6 and 4.10), but this is largely prevented by the build-up of berms, which provide increasing resistance at the apex of the buckle (Fig. 4.9).
4. At the end of the 2nd heating cycle, the bending moment at the apex of the buckle is 6% lower than at the first cycle. This reduction increases to 14% at the 5th cycle. (Figs. 4.4, 4.7 and 4.11.)
5. The effective axial force at the buckle increases slightly from cycle to cycles, as the pipe becomes increasingly constrained between berms. (Figs. 4.8 and 4.12.)
6. The very high axial soil frictional resistance used results in high axial forces at the midpoint between buckles (x=400m), close to the fully constrained axial force. (Fig. 4.8)

The lateral displacement results obtained by using the ABAQUS are significantly smaller than those of NPEX results. The results obtained by using ABAQUS are listed in Figs. 4.18 to 4.20 for comparison with the corresponding results obtained by using NPEX in

---

[vi] To track the solution path a special algorithm is needed, because the temperature reaches a maximum then drops and increases again. For this purpose an algorithm is used in which the increment in lateral displacement at the apex of the buckle is controlled at each increment. Initially, while the pipe is in contact with the bumper, very small lateral displacement increments are used. However once the pipe looses contact with the bumper, the algorithm is switched to a Riks-type approach in which the increment in archlength of the solution path in load-displacement space with a suitably defined norm is controlled at every timestep.

Figs. 4.3 to 4.5. The maximum values at the extreme temperature values of each cycle are listed in Table 4.5 for comparison purposes.

Description	Symbol	Value	Units
Outer Pipe Diameter	D	0.3556	m
Wall Thickness	t	0.0173	m
Coating Thickness	t_c	0.105	m
Young's Modulus	E	185207	MPa
Poisson's Ratio	ν	0.3	-
Coefficient of Thermal Expansion	α	1.24E-05	1/°C
Unit Weight of Fresh Water	γ_{fw}	9.81	kN/m^3
Specific Weight of Contents	G_p	0.1	-
Specific Weight of Steel	G_s	7.868	-
Specific Weight of Coating	G_c	2.963	-
Specific Weight of Seawater	G_{sw}	1.025	-
Water Depth	h_{sw}	140	m
Axial Friction Coefficient	μ_A	2.484	-
Lateral Friction Coefficient	μ_L	0.4?	-
Internal Pressure	p_{int}	14.4	MPa
Temperature Rise	ΔT	77	°C
Spacing of Lateral Buckles	H	800	m
Effective on Bottom Lay Tension	N_{lay}	578	kN

Table 4.1: Pipe properties and Loading conditions.

Displacement, u (m)	0	0.07	0.14	0.5	1.12	1.5
Normalised Resistance, R(u)/W	1.6	1.52	1.2	0.72	0.4	0.4

Table 4.2: Data defining the input functin for the upper resistance-displacement relation, R=R(u), by piecewise linear interpolation, or extrapolation, where needed.

Displacement, u (m)	0	0.5	1	1.5	3
Normalised Resistance, r(u)/W	0	0.2	0.32	0.4	0.4

Table 4.3: Data defining the input functin for the upper resistance-displacement relation, R=R(u), by piecewise linear interpolation, or extrapolation, where needed.

Plastic Strain	Stress MPa
0	270
0.0001	294.165
0.0002	306.9023
0.0005	324.5903
0.001	338.6449
0.0015	347.1468
0.002	353.3081
0.003	362.1781
0.004	368.6062
0.005	373.6707
0.006	377.8603
0.008	384.5667
0.01	389.8505
0.012	394.2215
0.015	399.6379
0.02	406.7308
0.03	416.942
0.04	424.3421
0.05	430.1724
0.06	434.9955
0.07	439.1155
0.08	442.716
0.09	445.9163
0.1	448.7987
0.15	460.066
0.2	468.2315
0.25	474.6648
0.3	479.9867
0.35	484.5329
0.4	488.5057
0.45	492.037
0.5	495.2176
1	516.6603
5	570.098

Table 4.4: Data defining the stress-strain relationship for the steel pipe. Piecewise linear interpolation between the data provided is used. (This is the stress strain curve used in a small strain formulation. This should be used as a true-stress strain curve if a large strain formulation is being used to solve the same problem, since it is found in [1] based on a large strain formulation that the small strain approximation for pipe bending is best when the true stress-strain relation is used as inut.)

Temperature Cycle	Temperature Change (deg C)	Maximum Lateral Displacement (m)	Maximum Bending Moment (MN-m)	Maximum Effective Tension (MN)
1st Heat-up	77	1.34 / n.pex	0.575	1.18
Cool-down	0	0.68 / n.pex	0.072	-0.35
2nd Heat-up	77	1.36	0.542	1.18
Cool-down	0	0.71	0.072	-0.38
3rd Heat-up	77	1.37	0.520	1.21
Cool-down	0	0.73	0.072	-0.39
4th Heat-up	77	1.36	0.507	1.24
Cool-down	0	0.74	0.072	-0.40
5th Heat-up	77	1.36	0.502	1.25
Cool-down	0	0.75	0.073	-0.41

Table 4.5: Comparisons of the maximum values obtained by using ABAQUS / NPEX.

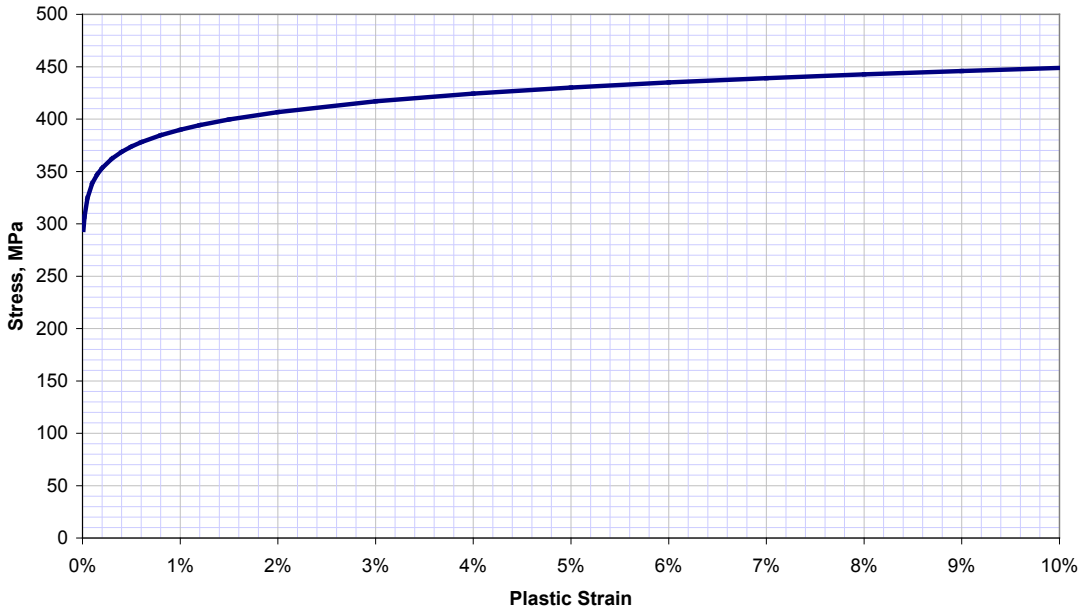

Fig. 4.1: Stress-strain curve used for the steel. (See Table 4.4 for numerical data.)

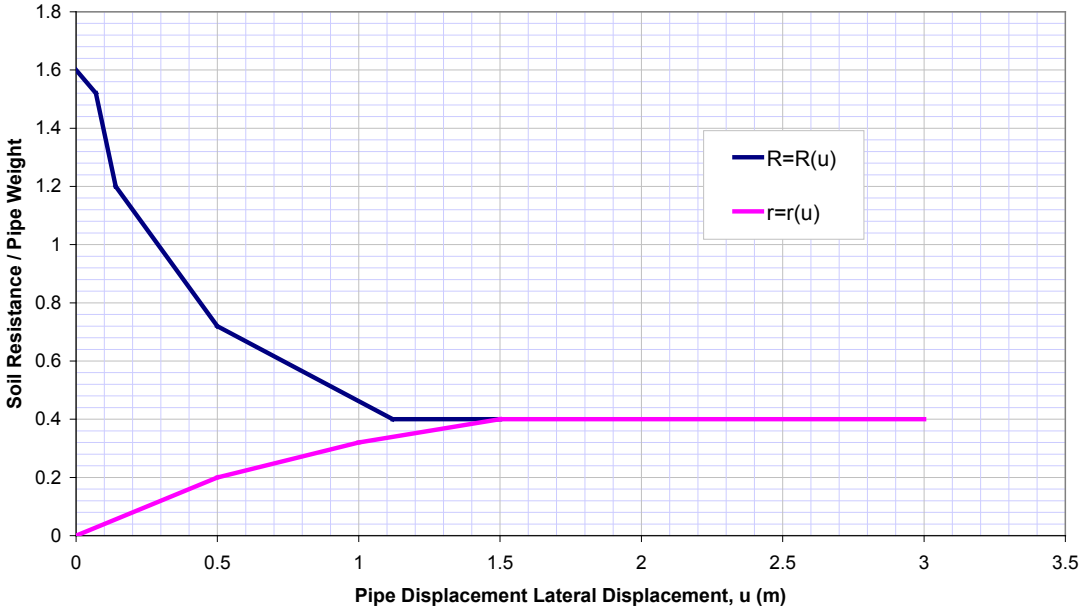

Fig. 4.2: Resistance-Displacement Relations describing covergence of the berm to the equilibrium size. A larger berm diminishes by leaving material behind, whereas a smaller berm grows until it reaches the equilibrium size.

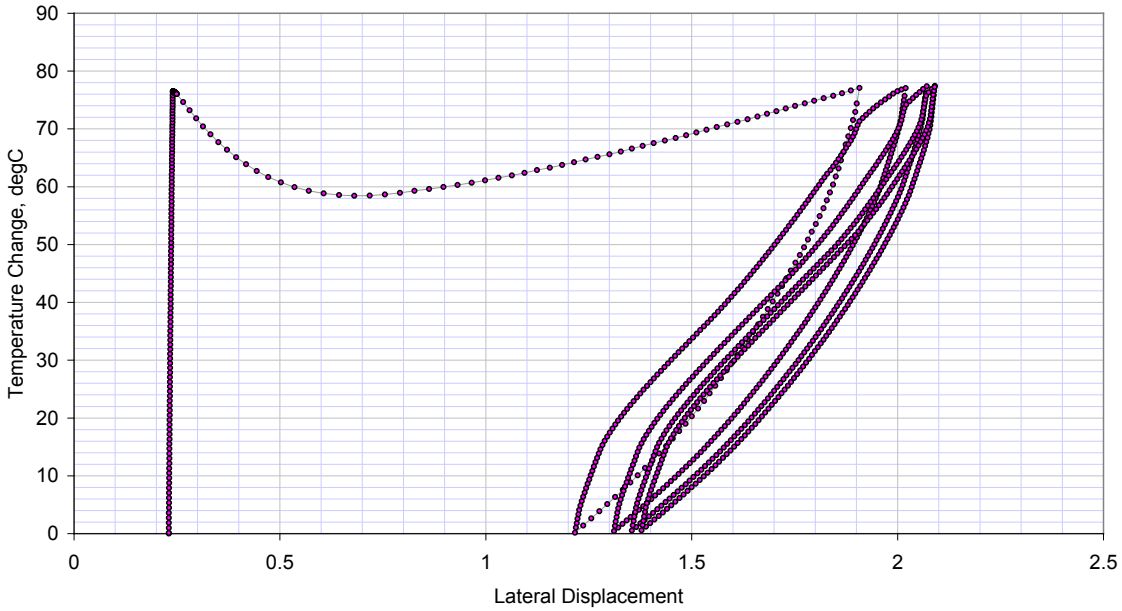

Fig. 4.3: History of temperature and lateral displacement at the apex of the buckle.

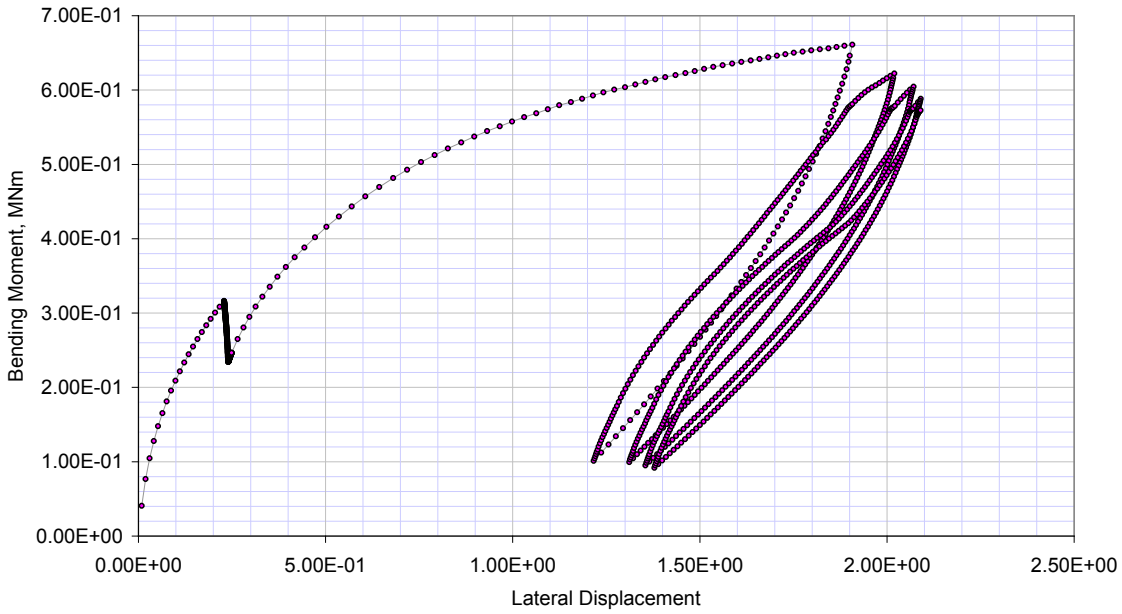

Fig. 4.4: History of bending moment and lateral displacement at the apex of the buckle.

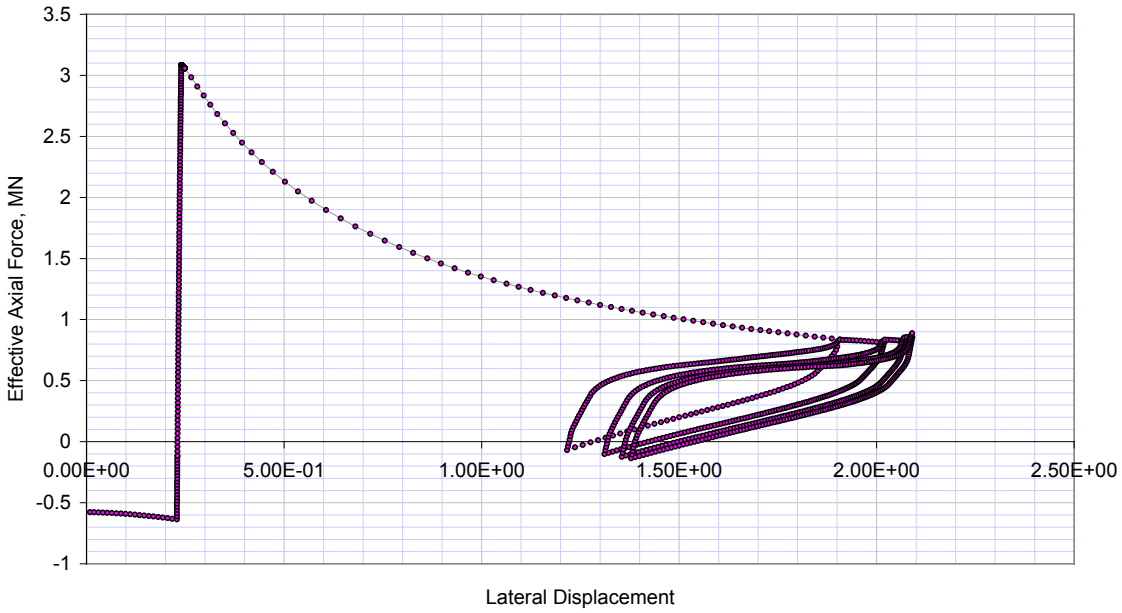

Fig. 4.5: History of effective axial force and lateral displacement at the apex of the buckle.

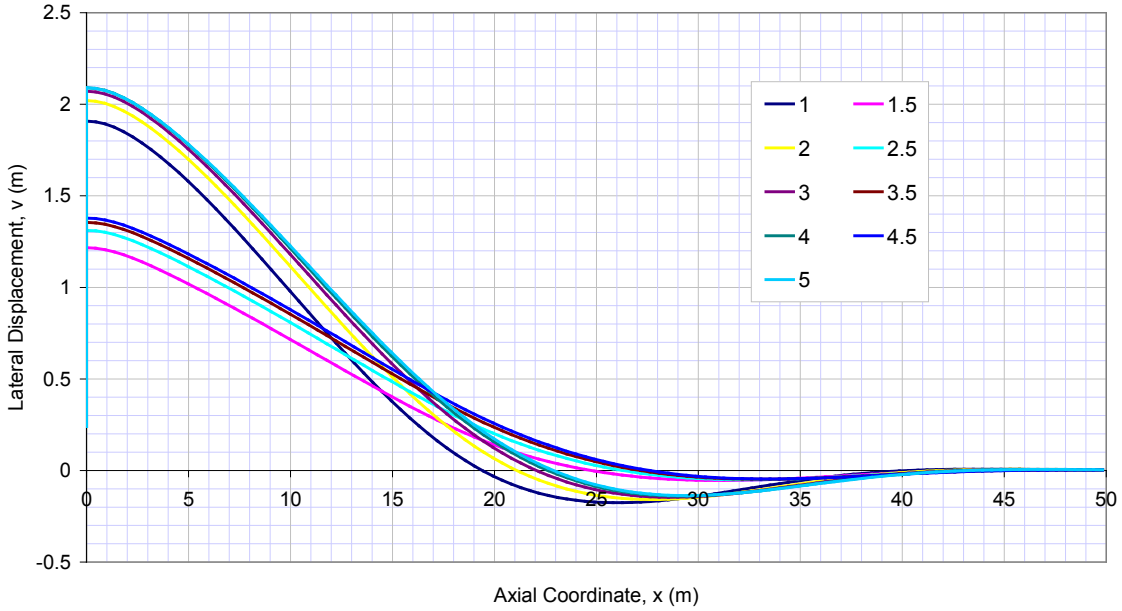

Fig. 4.6: Lateral Displacements at the end of each heating and cooling cycle. Integer cycle numbers 1,2,3,… represent the end of the heating cycle (temperature rise of 77°C above ambient), whereas cycle numbers 1.5, 2.5, 3.5, … represent the end of the cooling cycle (ambient temperature).

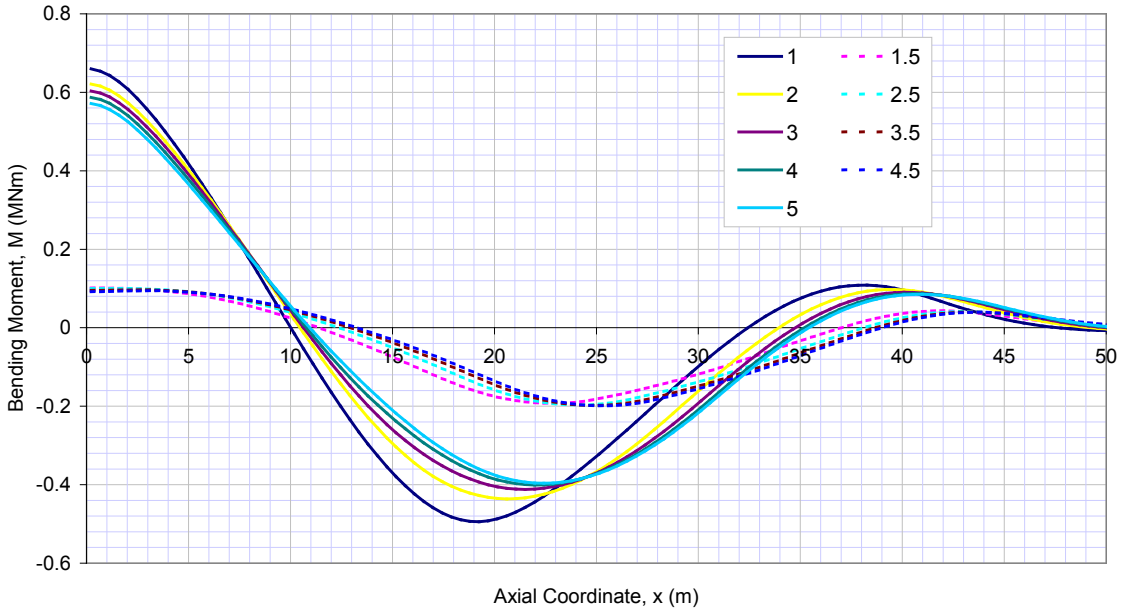

Fig. 4.7: Bending Moment at the end of each heating and cooling cycle.

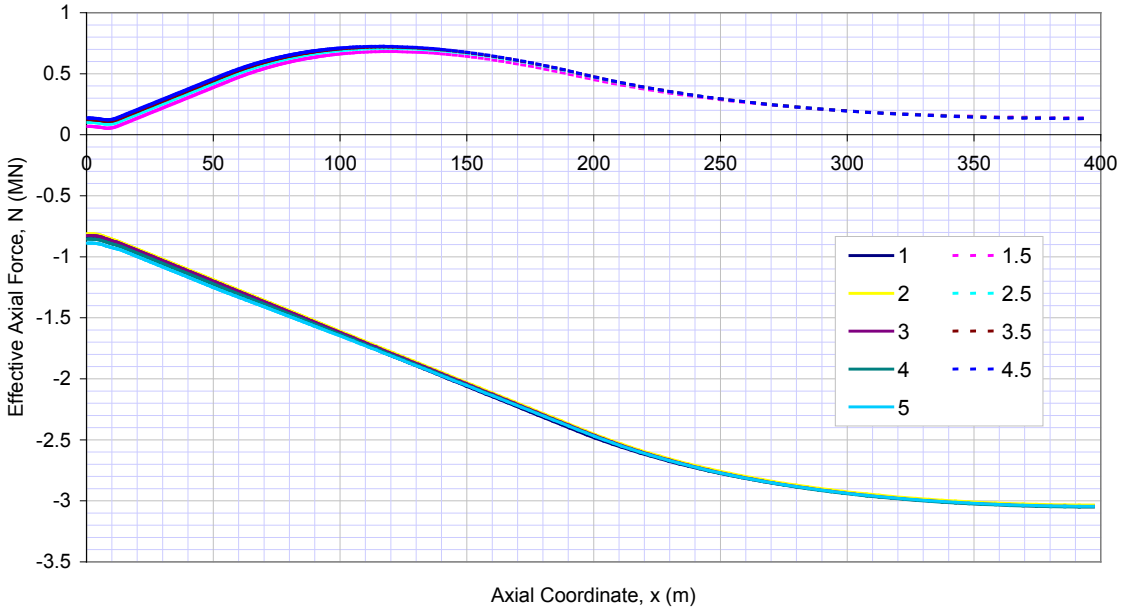

Fig. 4.8: Effective axial Force at the end of each heating and cooling cycle. Positive axial force represents tension.

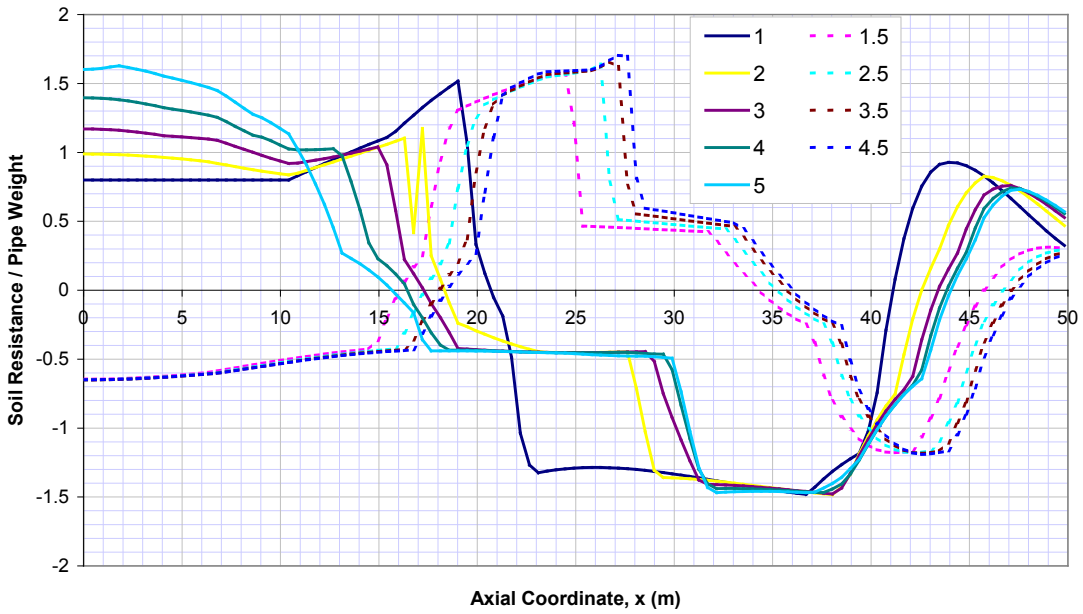

Fig. 4.9: Lateral soil resistance force normalised with respect to submerged weight of pipe at the end of each heating and cooling cycle.

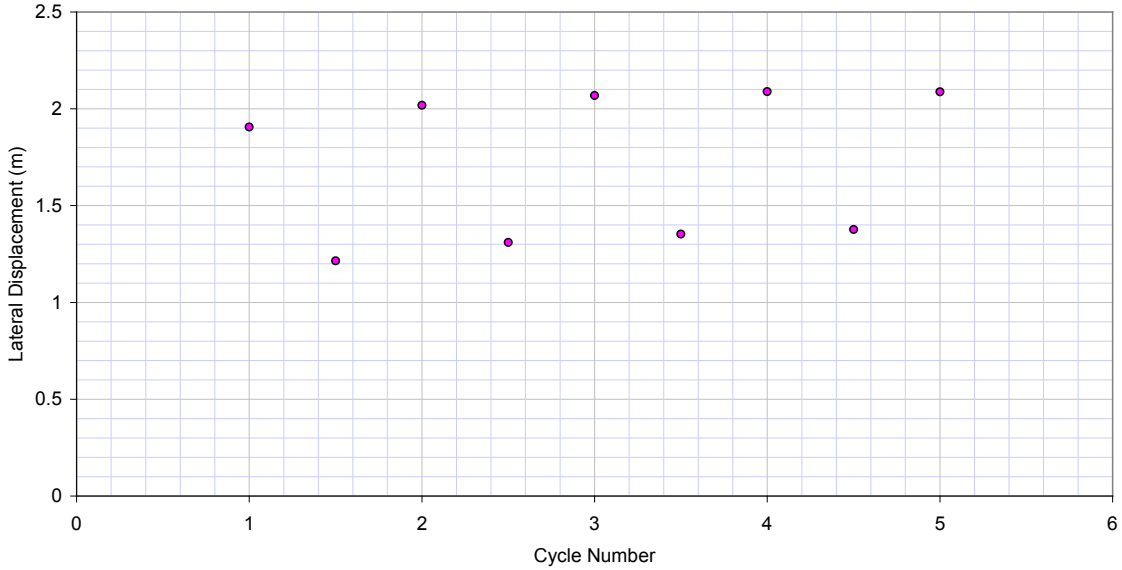

Fig. 4.10: Lateral displacement at the apex of the buckle at the end of each heating and cooling cycle.

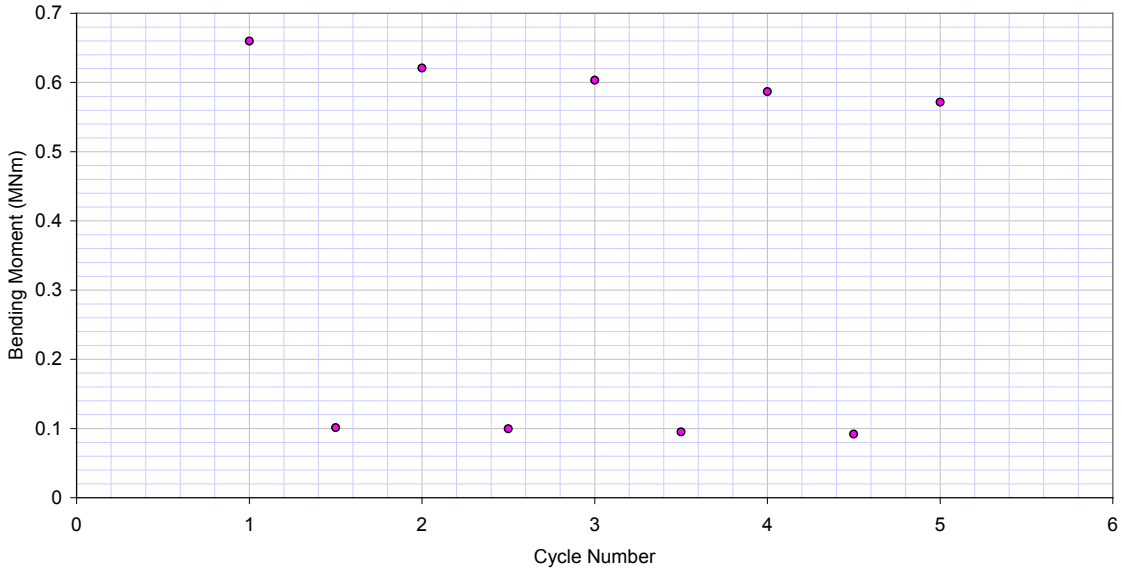

Fig. 4.11: Bending moment at the apex of the buckle at the end of each heating and cooling cycle.

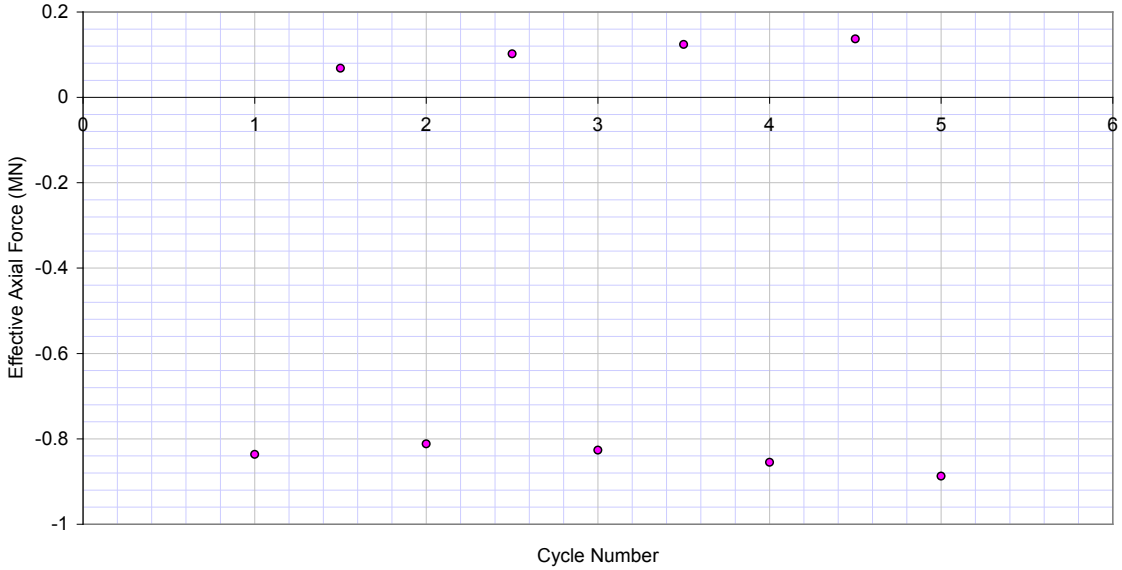

Fig. 4.12: Effective Axial force at the apex of the buckle at the end of each heating and cooling cycle. Positive axial force represents tension.

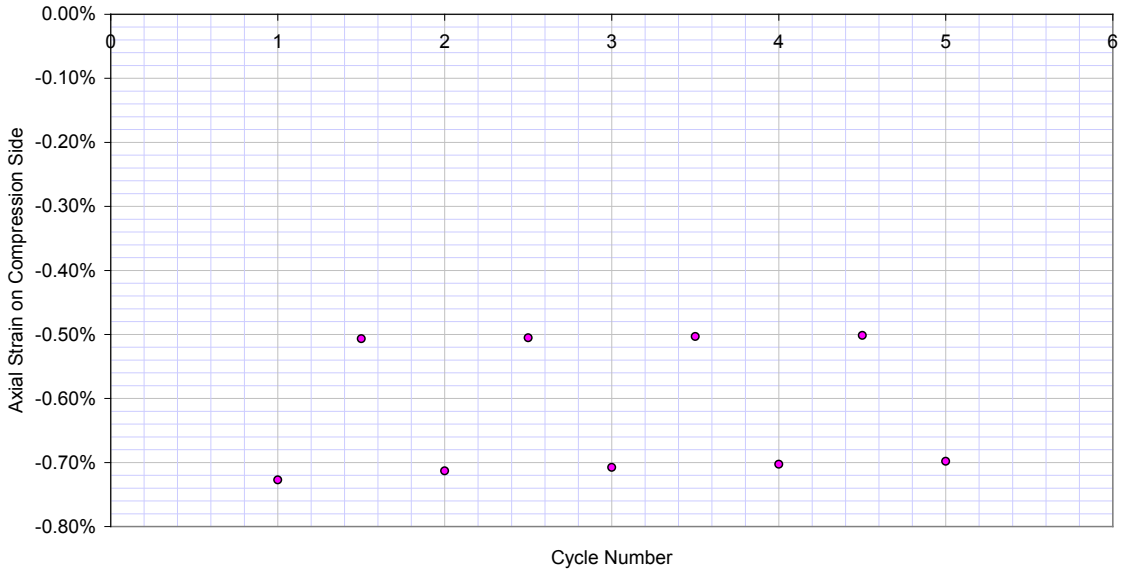

Fig. 4.13: Axial strain on the compression (concave) side at the apex of the buckle at the end of each heating and cooling cycle.

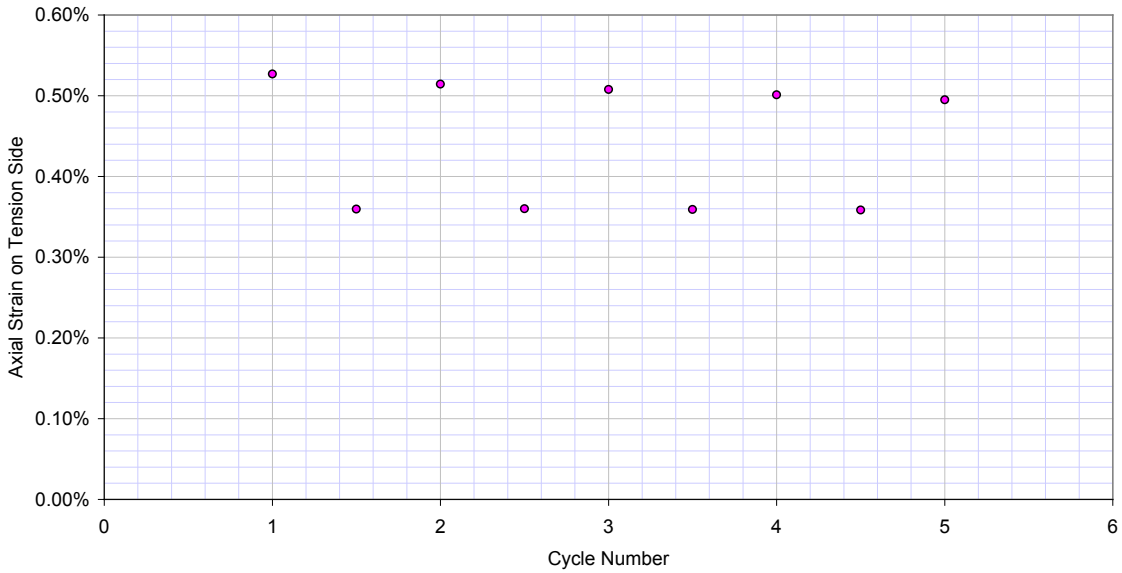

Fig. 4.14: Axial strain on the tension (convex) side at the apex of the buckle at the end of each heating and cooling cycle.

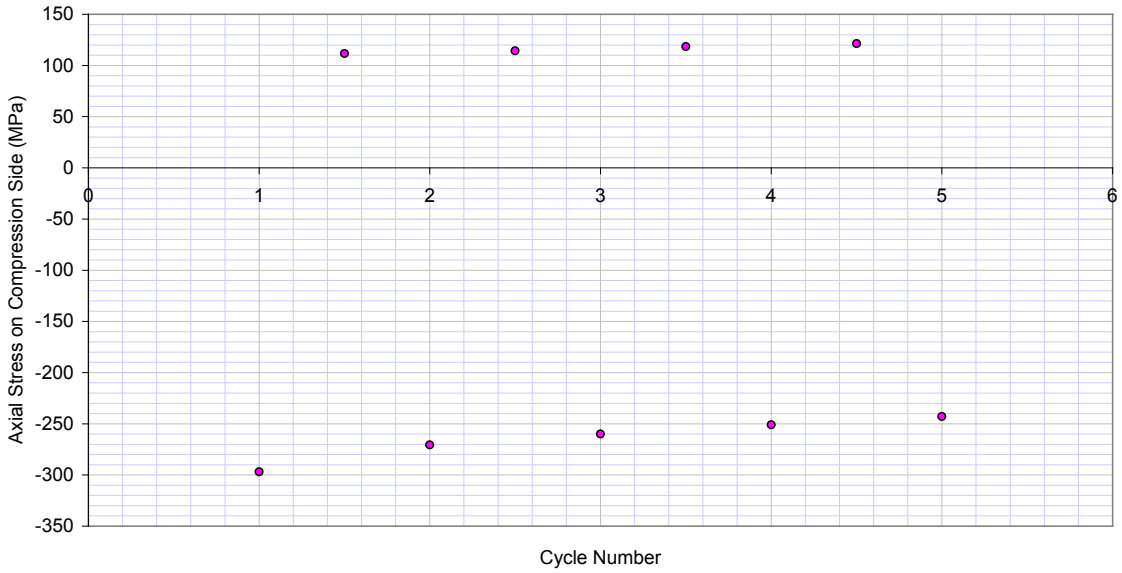

Fig. 4.15: Axial stress on the tension (convex) side at the apex of the buckle at the end of each heating and cooling cycle. Positive stress indicates tension.

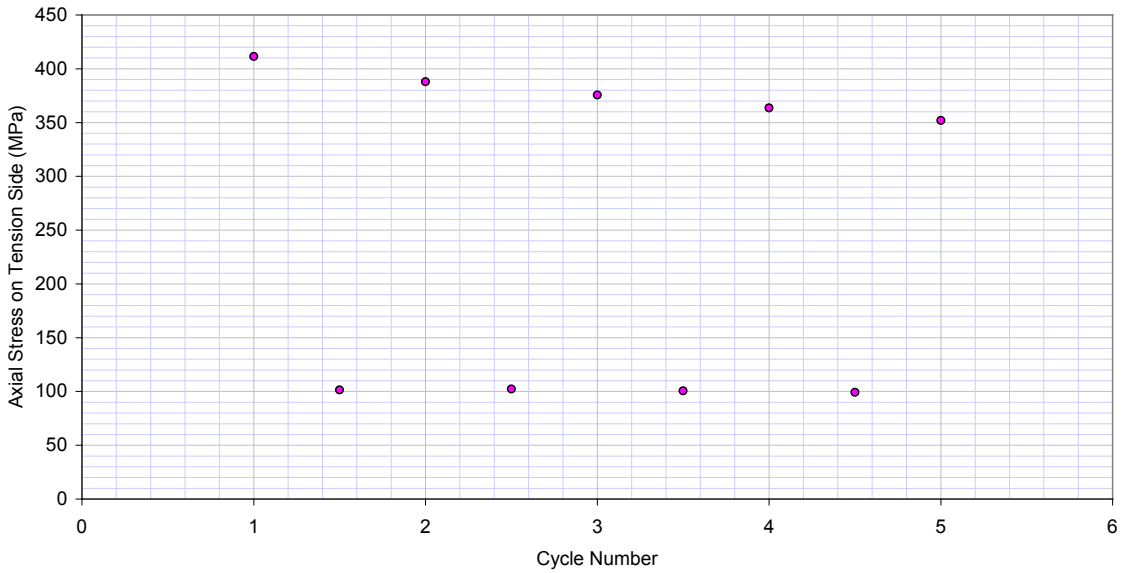

Fig. 4.16: Axial stress on the compression (concave) side at the apex of the buckle at the end of each heating and cooling cycle. Positive stress indicates tension.

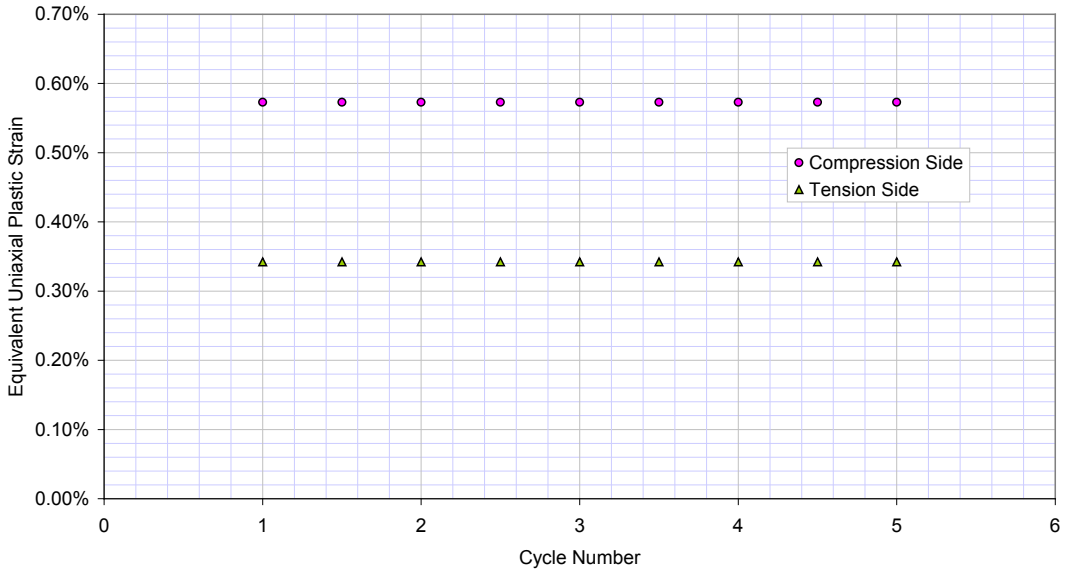

Fig. 4.17 Equivalent uniaxial platic strain at the apex of the buckle at the end of each heating and cooling cycle.

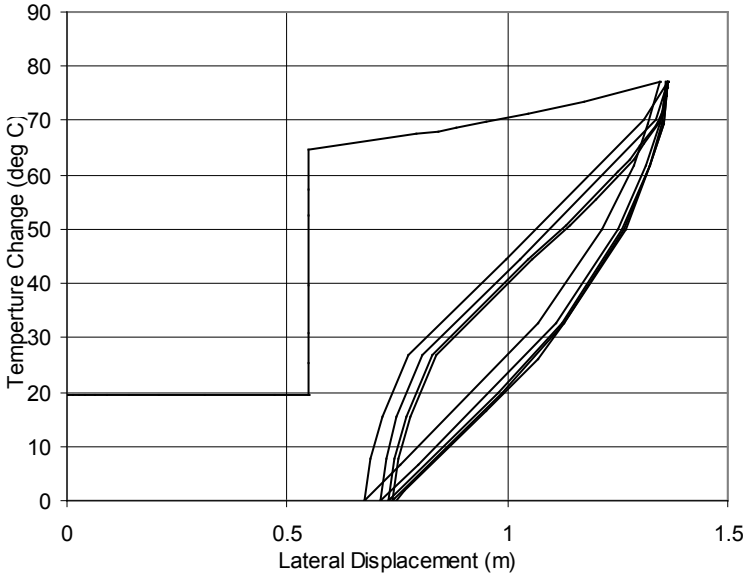

Fig. 4.18 History of temperature and lateral displacement at the apex of the buckle – results from analysis using ABAQUS.

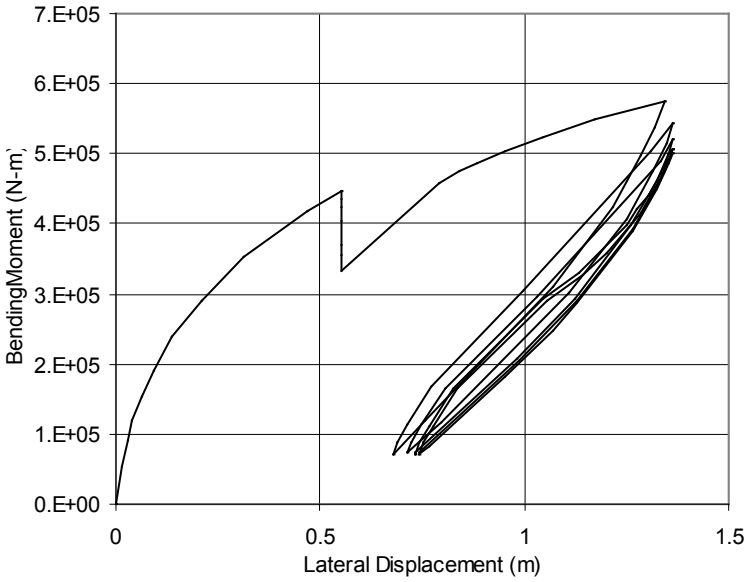

Fig. 4.19 History of bending moment and lateral displacement at the apex of the buckle – results from analysis using ABAQUS.

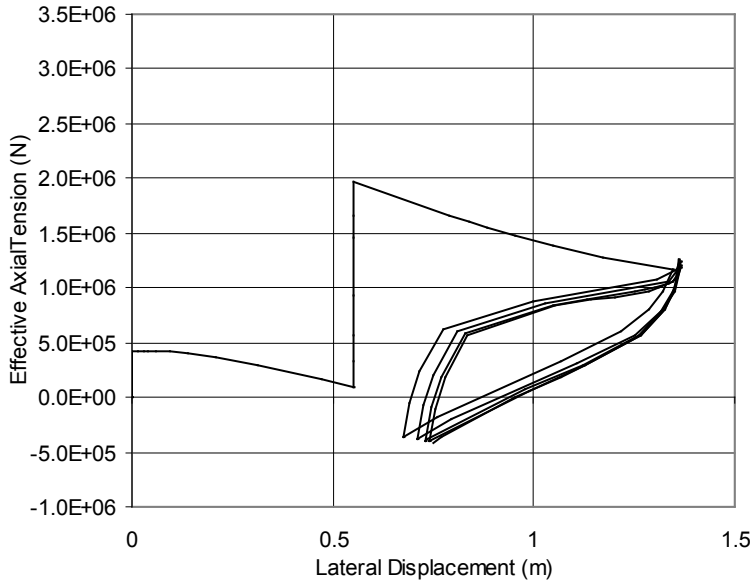

Fig. 4.20 History of effective axial force and lateral displacement at the apex of the buckle – results from analysis using ABAQUS.

5. CONCLUSIONS

A berm formation model is developed and implemented in ABAQUS. The model includes options of the coupled and uncoupled frictional interfaces.

The developed computer model is verified by solving example problems and by comparing the results with the results obtained by in-house code NPEX.

6. REFERENCES

1. Peek, R., (2001), "Wrinkling of Tubes in Bending using Large Strain Continuum Theory," International Journal of Solids and Structures, Vol. 29, pp. 709-723. (SIEP Disclosure/Ref. No. EP 2000-8171)

2. ABAQUS/Standard User's Manual, Version 6.5, HKS Inc., 2004

APPENDIX 1. USER'S GUIDE: BERM FORMATION MODEL

A1.1. General

In order to model the soil's resistance to the axial and the lateral displacements of pipelines on the seabed the pipe-soil interaction (PSI) element of ABAQUS of type "PSI24" is used [2]. The PSI24 element is a ABAQUS provided, 2-dimensional, 4-node element, and its purpose is to model the axial and the lateral pipe-soil interactions on even seabed. The two nodes on one side of this nominally rectangular-shaped element are attached on the pipe and move with the pipe, while the other two nodes on the other side are fixed in space, thus the deformation of the PSI element represents the axial and the lateral displacements of pipe on seabed. The responses of the PSI element to the pipe displacement, or the material properties of the PSI element represent the response characteristics of the axial and the lateral pipe-soil interface, and these are defined via a user subroutine UMAT. This report provides the user subroutine UMAT for ABAQUS. Formulation and the theoretical background of the material model used in UMAT is presented in Appendix 2. Source coe of the use subroutine UMAT is provided in Appendix 3. The associated input parameters must be provided in the files <jobname>.inp and <jobname>.brm. <jobname> represents the name of the ABAQUS job. Following must be done by the user in order to use the frictional and soil berm formation models in ABAQUS:

- Develop the ABAQUS model including the pipe elements, material properties, boundary conditions, and loading conditions.
- Generate PSI24 elements to model the pipe-soil interactions. Two nodes on one side of the element must be attached on the pipe, and the other two nodes must be fixed in space Reference the ABAQUS user's manual in Ref. [2] for information on the PSI element. Example is presented in Appendix 2.
- In the <jobname>.inp file define the 7 input parameters. Definition of the 7 input parameters and an example are presented in Appendix 2.
- Tabulated berm formation model data must be provided in the file <jobname>.brm, if the berm formation model is used to represent the pipe-soil interaction. Example is presented in Appendix 2.
- The user subroutine UMAT must be linked to ABAQUS. Source list of UMAT is presented in Appendix 3.

A1.2. Generation of PSI Elements in ABAQUS

Refer to the ABAQUS user's guide in Ref. [2] for the usage of the PSI24 element. The PSI elements must be generated and attached to the pipe sections which interact with seabed. For example, the following ABAQUS input lines are used to generate 300 PSI elements, and to assign the name SOIL for the PSI elements. The nodes 1 through 301 are commonly shared by both pipe and PSI elements, thus the pipe and PSI elements are attached to each other through these nodes. The nodes 1001 through 1301 are on the other side of the PSI elements and must be fixed in space later in the section where boundary conditions are defined (not shown below).

```
…
*element, type=psi24, elset=soil
```

1001, 1, 2, 1002, 1001

*elgen, elset=soil

1001, 300, 1, 1

…

A1.3. Definition of the 7 Input Parameters in <jobname>.inp file

Input parameters are defined in the <jobname>.inp file in the following format:

…

*pipe-soil interaction, elset=soil

*pipe-soil stiffness, type=user, prop=7, variables=100

<YY1>, <YY2>, <UM1>, <UM2>, <NB>, <QINIT>, <WEIGHT>

…

where the seven parameters are defined as:

- YY1 = Friction force limit per unit pipe length in the pipe axial direction.
- YY2 = Absolute value of YY2 is the friction force limit per unit pipe length in the pipe lateral direction. The axial and the lateral frictions defined by YY1 aand YY2 are coupled with the elliptical slip surface and the normal flow rule if YY2 is positive. They are uncoupled (with the rectangular slip surface) if YY2 is negative.
- UM1 = Mobilization displacement in the pipe axial direction
- UM2 = Mobilization displacement in the pipe lateral direction
- NB = Maximum number of soil berms on each side of pipe. If larger number of berms are formed during simulation the berms that are located farthest from the initial pipe location will be removed automatically. If NB = 0, the berm formation model is de-activated and only the frictional interaction is calculated. Both frictional and the berm formation model is calculated when NB is positive.
- QINIT = Initial berm resistance force per unit pipe length in both lateral directions of the pipe. This represents the initial berm size surrounding the pipe at its initial location. Embedment of pipe at its initial location causes this additional soil resistance. If this initial resistance is larger than the equilibrium value QEQUIL, the force-displacement relation for the first cycle will reach a maximum at a displacement UM(QINIT), i.e. the mobilization displacement UM corresponding to a berm resistance of QINIT.
- WEIGHT= Submerged pipe weight per unit length

Following is an example input lines in <jobname>.inp file:

…

*pipe-soil interaction, elset=soil

*pipe-soil stiffness, type=user, prop=7, variables=100

2.484, -0.4, 0.02, 0.03, 10, 1.12, 3389.

…

A1.4. Coupled Friction Model

If NB is positive the axial and the lateral frictions are coupled based on the elliptical slip condition, and the normal flow rule (similar to the yield surface of the metal plasticity). Therefore,

- If $(Y1/YY1)^2 + (Y2/YY2)^2 < 1$, then the frictional slip does not occur and the pipe sticks on the seabed.
- If $(Y1/YY1)^2 + (Y2/YY2)^2 = 1$, then the pipe slips on the seabed in the direction normal to the elliptical slip surface.

in which

Y1 = shear force between pipe and seabed in the pipe axial direction

Y2 = shear force between pipe and seabed in the pipe lateral direction

A1.5. Uncoupled Friction Model

If NB = 0 the axial and the lateral frictions are independent (un-coupled):

- If Y1 < YY1 and Y2 < YY2, then the frictional slip does not occur and the pipe sticks on the seabed.
- If Y1 = YY1 or Y2 = YY2, then the the pipe slips on the seabed in either axial or lateral direction.

The corresponding slip surface is thus rectangular. Normal directions are not defined at the four corners of the rectangle, but this does not cause any numerical problem since the shear forces are calculated for the prescribed displacements in the finite element calculation processes.

A1.6. Berm Formation Model

If NB is positive, then the berm formation model is activated, and berm resistances are calculated. The tabulated data for the berm model must be input by the user in the file <jobname>.brm. The lateral resistance of the soil berm model acts independently from the axial friction force (i.e. uncoupled). The tabulated data for the berm model must be input in the following format:

```
<nqv>
<BV,BQ_1>
<BV,BQ_2>
..
<BV,BQ_nqv>
<nqum>
<UM,BQ_1>
<UM,BQ_1>
..
<UM,BQ_nqum>
<ncru>
<U,CR_1>
<U,CR_2>
..
<U,CR_ncru>
<nsru>
<U,SR_1>
<U,SR_2>
..
<U,SR_nsru>
```

where parameters are defined as follows:

- Nqv = Number of data sets <BV, BQ>
- BV, BQ = Berm volume, and berm resistance force per unit pipe length data set. Nqv sets of these data are used to define piecewise linear relationships. BV values must be monotonically increasing.
- Nqum = Number of data sets <UM, BQ>
- UM, BQ = Berm mobilization, and berm resistance force per unit pipe length data set. Nqum sets of these data are used to define piecewise linear relationships. UM values must be monotonically increasing.
- Ncru = Number of data sets <U, CR>
- U, CR = Berm displacement, and berm resistance force per unit pipe length data set (upper response). Ncru sets of these data are used to define piecewise linear relationships. U values must be monotonically increasing. Only the differences between the U values input are important. Results are invariant to constant shift in U values. The last two values of the berm resistance must be identical and they are defined as the equilibrium resistance QEQUIL.
- Nsru = Number of data sets <U, SR>
- U, SR = Berm displacement, and berm resistance force per unit pipe length data set (lower response). Nsru sets of these data are used to define piecewise linear relationships. U values must be monotonically increasing. Only the differences between the U values input are important. Results are invariant to constant shift in U values. The last two values of the berm resistance must be identical, and they must be equal to the equilibrium resistance QEQUIL.

Following is an example <jobname>.brm file:

```
2          nqv
 0.0  0.0   BV,BQ
 1.0  1.0   BV,BQ
2          nqum
 0.01 0.0   UM,BQ
 0.01 1.0   UM,BQ
6          ncru
 0.0  1.6   U,CR
 0.07 1.52  U,CR
 0.14 1.2   U,CR
 0.5  0.72  U,CR
 1.12 0.4   U,CR
 1.5  0.4   U,CR
5          nsru
 0.0  0.0   U,SR
 0.5  0.2   U,SR
 1.0  0.32  U,SR
 1.5  0.4   U,SR
 3.0  0.4   U,SR
```

APPENDIX 2. INPUT EXAMPLES: BERM FORMATION MODEL

A2.1. Example.inp (ABAQUS model input file)

```
*heading
Shell SIEP, 2D Berms Soil Interaction Model
…
…
```

A2.2. Example.brm (Berm Model Parameters)

```
2          nqv
 0.0  0.0  BV,BQ
 1.0  1.0
2          nqum
 0.01 0.0  UM,BQ
 0.01 1.0
6          ncru
 0.0  1.6  U,CR
 0.07 1.52
 0.14 1.2
 0.5  0.72
 1.12 0.4
 1.5  0.4
5          nsru
 0.0  0.0  U,SR
 0.5  0.2
 1.0  0.32
 1.5  0.4
 3.0  0.4
```

APPENDIX 3. SOURCE CODE: BERM FORMATION MODEL

ABAQUS uses the user supplied subroutine UMAT in order to simulate the friction and the berm models of the pipe-soil interaction. ABAQUS execution procedures are described in volume 1 of ABAQUS manual [2]. The pipe-soil interface is modeled using the PSI elements (Pipe-Soil Interface element). The friction and the berm interface models are simulated through the non-linear material behaviours of the PSI elements. The subroutine UMAT is used to input the input parameters which define the interface properties, and to simulate the friction and the berm models.

```
      SUBROUTINE UMAT(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
     1 RPL,DDSDDT,DRPLDE,DRPLDT,STRAN,DSTRAN,
     2 TIM,DTIM,TEMP,DTEMP,PREDEF,DPRED,MATERL,NDI,NSHR,NTENS,
     3 NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,CELENT,
     4 DFGRD0,DFGRD1,NOEL,NPT,KSLAY,KSPT,KSTEP,KINC)
c
      INCLUDE 'ABA_PARAM.INC'
C
      CHARACTER*80 MATERL
      DIMENSION STRESS(NTENS),STATEV(NSTATV),
     1 DDSDDE(NTENS,NTENS),DDSDDT(NTENS),DRPLDE(NTENS),
     2 STRAN(NTENS),DSTRAN(NTENS),TIM(2),PREDEF(1),DPRED(1),
     3 PROPS(NPROPS),COORDS(3),DROT(3,3),
     4 DFGRD0(3,3),DFGRD1(3,3)
C
      DIMENSION ESTIFF(3),EELAS(3)
c
      character*256 filename1,filename2,jobname,outdir
c
      common/kdata17/aelprop17(1000),aelpar17(1000),elk17(2,2),elp17(2),
     &               elp_l17(2)
      common/kdata17I/ielprop17(1000),ielpar17(1000),
     &                nielprop17,naelprop17,nielpar17,naelpar17,
     &                ihave17
      common/kdata27/aelprop27(1000),aelpar27(1000),elk27(2,2),elp27(2),
     &               elp_l27(2),
     &               rup(2,100),rlow(2,100)
      common/kdata27I/ielprop27(1000),ielpar27(1000),
     &                nielprop27,naelprop27,nielpar27,naelpar27,
     &                ihave27
      common/kdata/u(2,3),ub_l(2,3),u0(2,3), x(2,3)
      common/kdataI/ien(3), lm(3), id(2,3),jread, info
C
C -------------------------------------------------------------
C     UMAT FOR SHELL SIEP Berms model
C -------------------------------------------------------------
C
      if(ndi.ne.2.or.nshr.ne.0) then
        write(0,*) '*** UMAT: incorrect dimensions'
        call xit
      end if
c
c Define element storage requirements for element 17 end 27:
c
      nen=1                 !number of element nodes used
      nninc=1               !number of nodes
      ndof=2                !number of degrees of freedom per node needed
      idof=2
      nsd1=1                !number of coordinates per node used by element
      nen=3
      ndof4bc=2
      nnodes=3
      nsd=2
      neldof=2
c
      iel=noel
      ielflag=0
c
c force scale factor
c
      WL=props(7)
c     write(0,*) 'all forces scaled with:', WL
c
c define properties (once) for element 17
c
      if(ihave17.eq.0) then
        ihave17=1
c
c get properties for element 17
c
        ielprop17(1)=nninc
        aelprop17(1)=WL*props(1)
        aelprop17(2)=WL*props(2)
        aelprop17(3)=WL*props(1)/props(3)      !axial stiffness  SKK1
        aelprop17(4)=WL*abs(props(2)/props(4)) !transverse stiffness SKK2
        write(0,*)'stiffness=',aelprop17(3),aelprop17(4)
c
```

```
c Define element storage requirements for element 17:
c
        naelpar17=2           !number of real element parameters needed
c       nielpar17=0           !number of integer element parameters needed
        nielpar17=1           !number of integer element parameters needed
        naelprop17=4          !number of element properties needed (defined above)
        nielprop17=1          !number of integer element properties needed (defined
above)
      end if
c
c n27 (= nb); n27=0 -> element 17 only
c
      n27=int(props(5))
c
c define properties (once) for element 27
c read functions (once) for element 27
c
      if(ihave27.eq.0.and.n27.gt.0) then
        ihave27=1
        call getoutdir(outdir, lenoutdir)
        call getjobname(jobname, lenjobname)
        filename1=outdir(1:lenoutdir) // '/'
     &            // jobname(1:lenjobname) // '.brm'
        write(0,*) ' '
        write(0,*) 'resistance data read from :'
        write(0,*) filename1
c
        open(91,file=filename1,status='old')
c
c read V-Q
c
        j=1
        read(91,*,end=1000,err=1000) nqv
        do i=1,nqv
          read(91,*,end=1000,err=1000) xx,yy
          aelprop27(j+i)=xx
          aelprop27(j+nqv+i)=yy*WL
        end do
        j=j+2*nqv
c
c read UM-BQ
c
        read(91,*,end=1000,err=1000) nqum
        do i=1,nqum
          read(91,*,end=1000,err=1000) xx,yy
          aelprop27(j+i)=xx
          aelprop27(j+nqum+i)=yy*WL
        end do
        j=j+2*nqum
c
c read U-CRn
c
        read(91,*,end=1000,err=1000) ncru
        do i=1,ncru
          read(91,*,end=1000,err=1000) xx,yy
          aelprop27(j+i)=xx
          aelprop27(j+ncru+i)=yy*WL
        end do
        j=j+2*ncru
c
c read U-SRn
c
        read(91,*,end=1000,err=1000) nsru
        do i=1,nsru
          read(91,*,end=1000,err=1000) xx,yy
          aelprop27(j+i)=xx
          aelprop27(j+nsru+i)=yy*WL
        end do
        j=j+2*nsru
        jread=j
c
        go to 2000
1000    continue
        write(0,*) '*** UMAT: I/O error reading functions'
        call xit
2000    continue
c
```

```
c get properties for element 27
c
        nb=int(props(5))
        qinit=props(6)*WL
c
        ielprop27(1)=nninc
        ielprop27(2)=idof
        ielprop27(3)=nb
        ielprop27(4)=nqv
        ielprop27(5)=nqum
        ielprop27(6)=ncru
        ielprop27(7)=nsru
c
        aelprop27(1)=qinit
c
c Define element storage requirements for element 27:
c
        naelpar27=4*nb+2      !number of real element parameters needed
        nielpar27=2           !number of integer element parameters needed
        naelprop27=jread      !number of element properties needed
        nielprop27=7          !number of integer element properties needed
      end if
c
c get statevarables
c
      is=0
      do i=1,naelpar17
        aelpar17(i)=statev(i)
      end do
      is=is+naelpar17
      do i=1,nielpar17
        ielpar17(i)=int(statev(is+i))
      end do
      is=is+nielpar17
      if(n27.gt.0) then
        do i=1,naelpar27
          aelpar27(i)=statev(is+i)
        end do
        is=is+naelpar27
        do i=1,nielpar27
          ielpar27(i)=int(statev(is+i))
        end do
        is=is+nielpar27
      end if
      xinit=statev(is+1)
      is=is+1
      if(info.eq.0) then
        info=1
        write(0,*) '*** UMAT: nr. of statevariables used:',is
      end if
c
c force initialization the first time
c
      if(xinit.eq.0.0) then
        ielflag=6
        xinit=1.0
      end if
c
c define dummy nodes and system matrices to be able to use the original NPEX routines
c it forced to calculate CLL=1.0 and will also work in arbitray directions
c
      ien(1)=2
      x(1,1)=coords(1)-1.0d0
      x(2,1)=coords(2)
      x(1,2)=coords(1)
      x(2,2)=coords(2)
      x(1,3)=coords(1)+1.0d0
      x(2,3)=coords(2)
      id(1,1)=1
      id(2,1)=2
      id(1,2)=1
      id(2,2)=2
      id(1,3)=1
      id(2,3)=2
c
c routine input
c
```

```
      time0=tim(2)
      time=tim(2)+dtim
      u0(1,2)=stran(1)
      u0(2,2)=stran(2)
      u(1,2)=stran(1)+dstran(1)
      u(2,2)=stran(2)+dstran(2)
      ub_l(1,2)=(u(1,2)-u0(1,2))/dtim
      ub_l(2,2)=(u(2,2)-u0(2,2))/dtim
c
c call element 17 routine
c
      call GENELKP_17(AELPAR17,AELPROP17,
     &   ELK17,ELP17,ELP_L17,
     &   ID,IELPAR17,IELPROP17,
     &   IEN,LM,
     &   U,UB_L,U0,X,
     &   IEL,IELFLAG,IGRP,ISTEP,NAELPAR17,NAELPROP17,NDOF,NDOF4BC,
     &   NELDOF,NELDOFI,
     &   NEN,NIELPAR17,NIELPROP17,NNODES,NSD,TIME,TIME0)
c
c call element 27 routine
c
      if(n27.gt.0)
     &call GENELKP_27(AELPAR27,AELPROP27,
     &   ELK27,ELP27,ELP_L27,
     &   ID,IELPAR27,IELPROP27,
     &   IEN,LM,
     &   U,UB_L,U0,X,
     &   IEL,IELFLAG,IGRP,ISTEP,NAELPAR27,NAELPROP27,NDOF,NDOF4BC,
     &   NELDOF,NELDOFI,
     &   NEN,NIELPAR27,NIELPROP27,NNODES,NSD,TIME,TIME0)
c
c return stiffness matrix
c
      DDSDDE(1,2)=0.0
      DDSDDE(2,1)=0.0
      DDSDDE(1,1)=elk17(1,1)
      DDSDDE(2,2)=elk17(2,2)+elk27(1,1)
c
c  return forces
c
      STRESS(1)=-elp17(1)
      STRESS(2)=-elp17(2)-elp27(1)
c
c store statevarables
c
      is=0
      do i=1,naelpar17
        statev(i)=aelpar17(i)
      end do
      is=is+naelpar17
      do i=1,nielpar17
        statev(is+i)=float(ielpar17(i))
      end do
      is=is+nielpar17
      if(n27.gt.0) then
        do i=1,naelpar27
          statev(is+i)=aelpar27(i)
        end do
        is=is+naelpar27
        do i=1,nielpar27
          statev(is+i)=float(ielpar27(i))
        end do
        is=is+nielpar27
      end if
      statev(is+1)=xinit
      is=is+1
C
      RETURN
      END
c-----------------------------------------------------------------------
      logical function LF_INDIC(i,j)
c Ruben's modified version for ABAQUS:
c
c allow all:
      lf_indic=.true.
c don't print
```

```
      if(j.eq.3) lf_indic=.false.
c don't initialize each time
      if(j.eq.6) lf_indic=.false.
c special: i=j=6 initilize once
      if(i.eq.j) lf_indic=.true.
      return
      end
c-----------------------------------------------------------------------
c Routine belows are unmodified SIEP NPEX routines
c-----------------------------------------------------------------------
C
      SUBROUTINE GENELKP_27(AELPAR,AELPROP,
     &   ELK,ELP,ELP_L,
     &   ID,IELPAR,IELPROP,
     &   IEN,LM,
     &   U,UB_L,U0,X,
     &   IEL,IELFLAG,IGRP,ISTEP,NAELPAR,NAELPROP,NDOF,NDOF4BC,
     &   NELDOF,NELDOFI,
     &   NEN,NIELPAR,NIELPROP,NNODES,NSD,TIME,TIME0)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AELPAR(NAELPAR),AELPROP(NAELPROP),
     & ELK(NELDOF,NELDOF),ELP(NELDOF),ELP_L(NELDOF),
     & ID(NDOF4BC,NNODES),IELPAR(NIELPAR),IELPROP(NIELPROP),
     & IEN(NEN),LM(NELDOF),
     & U(NDOF,NNODES),UB_L(NDOF,NNODES),U0(NDOF,NNODES),X(NSD,NNODES)
C
C Local declarations:
      ALLOCATABLE
     & BV(:,:),BX(:,:),BVL(:),  !berm resistances & locations
     & VQV(:),QQV(:),           !Q-V relation
     & UQUM(:),QQUM(:),         !Q-UM relation
     & UCRU(:),RCRU(:),         !CR-U relation
     & USRU(:),RSRU(:)          !SR-U relation
      LOGICAL LF_INDIC,P_TOWARDS_B
      DIMENSION NEB(2)
C
C Recover element properties from IELPROP & AELPROP, & store locally:
      MI=1
      NNINC=IELPROP(MI)
      MI=MI+1
      IDOF=IELPROP(MI)
      MI=MI+1
      NB=IELPROP(MI)
      MI=MI+1
      NQV=IELPROP(MI)
      MI=MI+1
      NQUM=IELPROP(MI)
      MI=MI+1
      NCRU=IELPROP(MI)
      MI=MI+1
      NSRU=IELPROP(MI)
      MI=MI+1
      ALLOCATE( !user-specified function data
     & VQV(NQV),QQV(NQV),         !Q-V relation
     & UQUM(NQUM),QQUM(NQUM),     !Q-UM relation
     & UCRU(NCRU),RCRU(NCRU),     !CR-U relation
     & USRU(NSRU),RSRU(NSRU))     !SR-U relation
      MA=1
      QINIT=AELPROP(MA)
      MA=MA+1
      CALL DUPA2B(AELPROP(MA),VQV,NQV)
      MA=MA+NQV
      CALL DUPA2B(AELPROP(MA),QQV,NQV)
      MA=MA+NQV
      CALL DUPA2B(AELPROP(MA),UQUM,NQUM)
      MA=MA+NQUM
      CALL DUPA2B(AELPROP(MA),QQUM,NQUM)
      MA=MA+NQUM
      CALL DUPA2B(AELPROP(MA),UCRU,NCRU)
      MA=MA+NCRU
      CALL DUPA2B(AELPROP(MA),RCRU,NCRU)
      MA=MA+NCRU
      CALL DUPA2B(AELPROP(MA),USRU,NSRU)
      MA=MA+NSRU
      CALL DUPA2B(AELPROP(MA),RSRU,NSRU)
      MA=MA+NSRU
      !Derived element properties:
```

```
      QEQUIL=RCRU(NCRU)
      UEQUIL=MAX(UCRU(NCRU),USRU(NSRU))  !as good as infinity
C
C Recover element parameters & store them locally:
      NB2=NB+NB
      NB4=NB2+NB2
      ALLOCATE(BV(NB,2),BX(NB,2),BVL(2))
      CALL DUPA2B(AELPAR(1),BV,NB2)
      CALL DUPA2B(AELPAR(NB2+1),BX,NB2)
      CALL DUPA2B(AELPAR(NB4+1),BVL,2)
      CALL DUIA2B(IELPAR(1),NEB,2)
C
C Form LM, UELB_L, and NELDOFI:
      INODE=IEN(1)
      IEQ=ID(IDOF,INODE)
      LM(1)=IEQ
      UELB_L=UB_L(IDOF,INODE)
      NELDOFI=1
C
C Displacements & Coordinates:
      UU=U(IDOF,INODE)
      UU0=U0(IDOF,INODE)
      INODE2=INODE-NNINC
      INODE3=INODE+NNINC
      CLL=0.5D0*ABS( X(1,INODE3)-X(1,INODE2))  !tributary length
      XX=X(1,INODE)
C
C Initialise element parameters, if appropriate:
      IF(LF_INDIC(IELFLAG,6)) THEN
        VINIT=AINTERP(QINIT,QQV,VQV,NQV,0)
        BV=0.D0 !all elements of array
        BX=0.D0 !all elements of array
        DO J=1,2
          BV(1,J)=VINIT
          BX(1,J)=UU0
          NEB(J)=1
        ENDDO
        CALL DUPB2A(AELPAR(1),BV,NB2)
        CALL DUPB2A(AELPAR(NB2+1),BX,NB2)
        CALL DUPB2A(AELPAR(NB4+1),BVL,2)
        CALL DUIB2A(IELPAR(1),NEB,2)
        IF(IELFLAG.EQ.2**6) RETURN
      ENDIF
C
C Step 1) Identify direction of motion. If this has changed,
C    create new berms with zero volume, & shift existing ones:
      IF(UU.GT.UU0) THEN
        JDIR=2
        GXI=1.D0
      ELSE
        JDIR=1
        GXI=-1.D0
      ENDIF
      IF(GXI*(BX(1,JDIR)-UU0).GT.0.D0) THEN
C       reversal of slip direction; create new berm
        NEB0=NEB(JDIR)
        IF(NEB0.EQ.NB) THEN
          BVL(JDIR)=BVL(JDIR)+BV(NB,JDIR)
        ELSE
          NEB(JDIR)=NEB0+1
        ENDIF
        DO I=NEB(JDIR),2,-1
          BV(I,JDIR)=BV(I-1,JDIR)
          BX(I,JDIR)=BX(I-1,JDIR)
        ENDDO
        BV(1,JDIR)=0.D0
        BX(1,JDIR)=UU0
      ENDIF
C
C Steps 2-5) Resistance, stiffness & update:
      SS=0.D0    !soil resistance per unit length of pipe
      DSDU=0.D0  !corresponding stiffness
      GXI=1.D0
      DO J=1,2   !sides
        GXI=-GXI
        QPP0=0.D0
        UPP0=0.D0
```

```
        P_TOWARDS_B=GXI*(UU-UU0).GT.0.D0 !true if pipe moves towards berm
        IF(P_TOWARDS_B) THEN
          XB0=UU0 !Berms on side of displ increment
        ELSE
          XB0=UU  !Berms on opposite side to displ increment
          XB0_U=1.D0
          UPP0_U=0.D0
        ENDIF
        IG=0
        DO I=1,NEB(J) !berms on Jth side
          XB1=BX(I,J)
c*ruben:
c*ruben: This is the only NPEX element code change:
c*ruben: due to the large displacement formulation the program can violate
c*ruben: the check below slightly. The check has been switched off.
c*ruben:
c*ruben:  CALL CHKRMIN(gxi*(xb1-xb0),0.d0,"nelmt27a",8) !temp check logic code (tclc)
          UP=UPP0+GXI*(XB1-XB0)
          IF(QPP0.GT.QEQUIL) THEN
            QP=AINTERP(UP,UCRU,RCRU,NCRU,0)
          ELSE IF(QPP0.LT.QEQUIL) THEN
            QP=AINTERP(UP,USRU,RSRU,NSRU,0)
          ELSE
            QP=QEQUIL
          ENDIF
          VP=AINTERP(QP,QQV,VQV,NQV,0)
          VPP=VP+BV(I,J)
          QPP=AINTERP(VPP,VQV,QQV,NQV,0)
          IF(QPP.GT.QEQUIL) THEN
            UPP=AINTERP(QPP,RCRU,UCRU,NCRU,0)
          ELSE IF(QPP.LT.QEQUIL) THEN
            UPP=AINTERP(QPP,RSRU,USRU,NSRU,0)
          ELSE
            UPP=UEQUIL
          ENDIF
          IF(GXI*(UU-XB1).GT.0.D0) THEN
            !Berm is engulfed
            IG=I
            QPPG=QPP
            UPPG=UPP
            XBG=XB1
          ELSE
            !Berm engaged or inactive
            UM=AINTERP(QPP,QQUM,UQUM,NQUM,0)  !mob displ
            XM=XB1-GXI*UM                     !activation location
            IF(.NOT.P_TOWARDS_B) THEN
              !Address complications in calculation of consistent tangent
              !for engaged berms the pipe is moving away from.
              !What follows is essentially a differentiation of the code
              !above to calculate the resistance force.
              UP_U=UPP0_U-GXI*XB0_U
              IF(QPP0.GT.QEQUIL) THEN
                QP_U=AINTERP(UP,UCRU,RCRU,NCRU,1)*UP_U
              ELSE IF(QPP0.LT.QEQUIL) THEN
                QP_U=AINTERP(UP,USRU,RSRU,NSRU,1)*UP_U
              ELSE
                QP_U=0.D0
              ENDIF
              VP_U=AINTERP(QP,QQV,VQV,NQV,1)*QP_U
              VPP_U=VP_U
              QPP_U=AINTERP(VPP,VQV,QQV,NQV,1)*VPP_U
              IF(QPP.GT.QEQUIL) THEN
                UPP_U=AINTERP(QPP,RCRU,UCRU,NCRU,1)*QPP_U
              ELSE IF(QPP.LT.QEQUIL) THEN
                UPP_U=AINTERP(QPP,RSRU,USRU,NSRU,1)*QPP_U
              ELSE
                UPP_U=0.D0
              ENDIF
              UPP0_U=UPP_U
              XB0_U=0.D0
            ENDIF
            IF(GXI*(UU-XM).GT.0.D0) THEN
              !Berm is engaged (i.e. active, but not engulfed)
              SKB=(QPP-QP)/UM
              SS=SS+SKB*(UU-XM)
              IF(P_TOWARDS_B) THEN
                DSDU=DSDU+SKB
```

```
              ELSE
                UM_U=AINTERP(QPP,QQUM,UQUM,NQUM,1)*QPP_U
                XM_U=-GXI*UM_U
                SKB_U=((QPP_U-QP_U)-(QPP-QP)*UM_U/UM)/UM
                DSDU=DSDU+SKB*(1.D0-XM_U)+SKB_U*(UU-XM)
              ENDIF
            ENDIF
          ENDIF
          QPP0=QPP
          UPP0=UPP
          XB0=XB1
        ENDDO
        !Update last engulfed berm:
        IF(IG.GT.0) THEN
          UPP=UPPG+GXI*(UU-XBG)
          IF(QPPG.GT.QEQUIL) THEN
            QPP=AINTERP(UPP,UCRU,RCRU,NCRU,0)
            QU1=AINTERP(UPP,UCRU,RCRU,NCRU,1)
          ELSE IF(QPPG.LT.QEQUIL) THEN
            QPP=AINTERP(UPP,USRU,RSRU,NSRU,0)
            QU1=AINTERP(UPP,USRU,RSRU,NSRU,1)
          ELSE
            QPP=QEQUIL
            QU1=0.D0
          ENDIF
          DSDU=DSDU+QU1  !stiffness contribution from sliding berm
          SS=SS+GXI*QPP  !resistance contribution from sliding berm
          VPP=AINTERP(QPP,QQV,VQV,NQV,0)
          !Let last engulfed berm become current sliding berm:
          BV(IG,J)=VPP
          BX(IG,J)=UU
          IF(IG.GT.1) THEN !Re-number berms
            NEB(J)=NEB(J)-IG+1
            DO I=1,NEB(J)
              II=IG+I-1
              BV(I,J)=BV(II,J)
              BX(I,J)=BX(II,J)
            ENDDO
          ENDIF
          !Not necessary to zero no-longer-existing berms.
        ENDIF
      ENDDO
      ELP(1)=-SS*CLL
C
C Compute ELK & ELP_L if apropriate (uses consistent tangent):
      IF(LF_INDIC(IELFLAG,1)) THEN
        ELK(1,1)=DSDU*CLL
        ELP_L(1)=-ELK(1,1)*UELB_L
      ENDIF  !LF_INDIC(IELFLAG,1))
C
C Update element parameters if apropriate:
      IF(LF_INDIC(IELFLAG,2)) THEN
        CALL DUPB2A(AELPAR(1),BV,NB2)
        CALL DUPB2A(AELPAR(NB2+1),BX,NB2)
        CALL DUPB2A(AELPAR(NB4+1),BVL,2)
        CALL DUIB2A(IELPAR(1),NEB,2)
      ENDIF  !LF_INDIC(IELFLAG,2)
C
C Print stresses in element:
      IF(LF_INDIC(IELFLAG,3)) THEN
        WRITE(6,"(' IEL=',I6,' X=',G13.4,' U=',G13.4,' S=',G13.4)")
     &             IEL,XX,UU,SS
      ENDIF   !LF_INDIC(IELFLAG,3)
      RETURN
      END
C
C-----------------------------------------------------------------
C
      SUBROUTINE GENELKP_17(AELPAR,AELPROP,
     &   ELK,ELP,ELP_L,
     &   ID,IELPAR,IELPROP,
     &   IEN,LM,
     &   U,UB_L,U0,X,
     &   IEL,IELFLAG,IGRP,ISTEP,NAELPAR,NAELPROP,NDOF,NDOF4BC,
     &   NELDOF,NELDOFI,
     &   NEN,NIELPAR,NIELPROP,NNODES,NSD,TIME,TIME0)
      IMPLICIT REAL*8 (A-H,O-Z)
```

```
      DIMENSION AELPAR(NAELPAR),AELPROP(NAELPROP),
     & ELK(NELDOF,NELDOF),ELP(NELDOF),ELP_L(NELDOF),
     & ID(NDOF4BC,NNODES),IELPAR(NIELPAR),IELPROP(NIELPROP),
     & IEN(NEN),LM(NELDOF),
     & U(NDOF,NNODES),UB_L(NDOF,NNODES),U0(NDOF,NNODES),X(NSD,NNODES)
C
C Local declarations:
      LOGICAL LF_INDIC,PLASTIC_LOADING,UNCOUPLED !26.3.2000
      DIMENSION UELB_L(2)
C
C Initialise element parameters, if appropriate:
      IF(LF_INDIC(IELFLAG,6)) THEN
        IF(NAELPAR.GT.0) CALL CLEAR(AELPAR,NAELPAR)
        IF(NIELPAR.GT.0) CALL ICLEAR(IELPAR,NIELPAR)
        IF(IELFLAG.EQ.2**6) RETURN
      ENDIF
C
C Form LM, UELB_L, and NELDOFI:
      INODE=IEN(1)
      DO J=1,2     !element dofs
        IEQ=ID(J,INODE)
        LM(J)=IEQ
        UELB_L(J)=UB_L(J,INODE)
      ENDDO
      NELDOFI=2
C
C Recover element properties:
      YY1=AELPROP(1)
      YY2=AELPROP(2)
      CKK1=AELPROP(3)
      CKK2=AELPROP(4)
      NNINC=IELPROP(1)  !node number increment for element length
      YSQ1=YY1*YY1
      YSQ2=YY2*YY2
      UNCOUPLED=YY2.LT.0.  !26.3.2000
      YY2=ABS(YY2)  !26.3.2000
C
C Recover Element Parameters:
      UPL1=AELPAR(1)
      UPL2=AELPAR(2)
C
C Displacements & Coordinates:
      UU1=U(1,INODE)      !displ in x-direction or u-displ
      UU2=U(2,INODE)      !displ in y-direction or v-displ
      INODE2=INODE-NNINC
      INODE3=INODE+NNINC
      CLL=0.5D0*ABS( X(1,INODE3)-X(1,INODE2))  !tributary length
C
C Uncoupled case:
      IF(UNCOUPLED) THEN !26.3.2000 block that follows
C
C   Calculate ELK, ELP & ELP_L: (uncoupled case)
        ELK(1,2)=0.
        ELK(2,1)=0.
        PLASTIC_LOADING=.FALSE.
C     Axial direction (1):
        SEL1=CKK1*(UU1-UPL1)
        IF(ABS(SEL1).GT.YY1) THEN
          PLASTIC_LOADING=.TRUE.
          SS1=SIGN(YY1,SEL1)
          ELP(1)=-CLL*SS1
          ELK(1,1)=0.
          ELP_L(1)=0.
          UPL1=UU1-SS1/CKK1
        ELSE
          SS1=SEL1
          ELP(1)=-CLL*SS1
          ELK(1,1)=CLL*CKK1
          ELP_L(1)=-ELK(1,1)*UELB_L(1)
        ENDIF
C     Transverse direction (2):
        SEL2=CKK2*(UU2-UPL2)
        IF(ABS(SEL2).GT.YY2) THEN
          PLASTIC_LOADING=.TRUE.
          SS2=SIGN(YY2,SEL2)
          ELP(2)=-CLL*SS2
          ELK(2,2)=0.
```

```
          ELP_L(2)=0.
          UPL2=UU2-SS2/CKK2
        ELSE
          SS2=SEL2
          ELP(2)=-CLL*SS2
          ELK(2,2)=CLL*CKK2
          ELP_L(2)=-ELK(2,2)*UELB_L(2)
        ENDIF
C
C   Update element parameters if apropriate:
        IF(LF_INDIC(IELFLAG,2)) THEN
          AELPAR(1)=UPL1
          AELPAR(2)=UPL2
        ENDIF  !LF_INDIC(IELFLAG,2)
C
C   Print stresses in element:
        IF(LF_INDIC(IELFLAG,3)) THEN
          XX1=X(1,INODE)      !initial x-coordinate
          WRITE(6,"(' IEL=',I6,' X=',G13.4,' SS1=',G13.4,
     &' SS2=',G13.4,' PL=',L1)")
     &     IEL,XX1,SS1,SS2,PLASTIC_LOADING
        ENDIF   !LF_INDIC(IELFLAG,3)
        RETURN
      ENDIF !End uncoupled case.  26.3.2000
C
C Compute ELP:
      SEL1=CKK1*(UU1-UPL1)
      SEL2=CKK2*(UU2-UPL2)
      RR1=SEL1/YSQ1
      RR2=SEL2/YSQ2
      TAU=SQRT(SEL1*RR1+SEL2*RR2)
      PLASTIC_LOADING=TAU.GT.1.
      IF(PLASTIC_LOADING) THEN
        DLAMBDA=0.D0
        TAUP=-CKK1*RR1*RR1-CKK2*RR2*RR2
        DO ITER=1,1000
          DLAMBDA=DLAMBDA-(TAU-1.)/TAUP
          QQ1=1.+CKK1*DLAMBDA/YSQ1
          QQ2=1.+CKK2*DLAMBDA/YSQ2
          SS1=SEL1/QQ1
          SS2=SEL2/QQ2
          RR1=SS1/YSQ1
          RR2=SS2/YSQ2
          TAU=SQRT(SS1*RR1+SS2*RR2)
          TAUP=-RR1*RR1*CKK1/QQ1-RR2*RR2*CKK2/QQ2
          IF(ABS(TAU-1.).LT.1.D-13) GO TO 701
        ENDDO
        WRITE(6,*) " No Convergence in nelmt17 state calculations."
        STOP
  701   CONTINUE !Iterations converged
      ELSE
        SS1=SEL1
        SS2=SEL2
      ENDIF
      ELP(1)=-CLL*SS1
      ELP(2)=-CLL*SS2
C
C Compute ELK & ELP_L if apropriate (uses consistent tangent):
      IF(LF_INDIC(IELFLAG,1)) THEN
        IF(PLASTIC_LOADING) THEN
          TST1=CKK1/QQ1
          TST2=CKK2/QQ2
          RT1=RR1*TST1
          RT2=RR2*TST2
          ELK(1,1)=CLL*(TST1+RT1*RT1/TAUP)
          ELK(1,2)=CLL*RT1*RT2/TAUP
          ELK(2,1)=ELK(1,2)
          ELK(2,2)=CLL*(TST2+RT2*RT2/TAUP)
        ELSE
          ELK(1,1)=CLL*CKK1
          ELK(1,2)=0.
          ELK(2,1)=0.
          ELK(2,2)=CLL*CKK2
        ENDIF
        DO I=1,NELDOFI
          TST1=0.
          DO J=1,NELDOFI
```

```
            TST1=TST1- ELK(I,J)*UELB_L(J)
          ENDDO
          ELP_L(I)=TST1
        ENDDO
      ENDIF  !LF_INDIC(IELFLAG,1))
C
C Update element parameters if apropriate:
      IF(LF_INDIC(IELFLAG,2)) THEN
        IF(PLASTIC_LOADING) THEN
          UPL1=UPL1+RR1*DLAMBDA
          UPL2=UPL2+RR2*DLAMBDA
          AELPAR(1)=UPL1
          AELPAR(2)=UPL2
        ENDIF
      ENDIF  !LF_INDIC(IELFLAG,2)
C
C Print stresses in element:
      IF(LF_INDIC(IELFLAG,3)) THEN
        XX1=X(1,INODE)      !initial x-coordinate
        WRITE(6,"(' IEL=',I6,' X=',G13.4,' SS1=',G13.4,
     &' SS2=',G13.4,' PL=',L1)")
     &   IEL,XX1,SS1,SS2,PLASTIC_LOADING
      ENDIF   !LF_INDIC(IELFLAG,3)
      RETURN
      END
C
C Log of Changes
C 29 Jan 2000 Implementation started, with a copy of nelmt16.
C 26 March 2000 added possibility of no coupling.
C
c-----------------------------------------------------------------------
      SUBROUTINE DUPA2B(A,B,N)
      !DUPlicate real array
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(N),B(N)
      DO 1 I=1,N
  1   B(I)=A(I)
      RETURN
      END
C
      SUBROUTINE DUPB2A(A,B,N)
      !DUPlicate real array
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(N),B(N)
      DO 1 I=1,N
  1   A(I)=B(I)
      RETURN
      END
C
      SUBROUTINE DUIA2B(IA,IB,N)
      !DUplicate Integer array
      DIMENSION IA(N),IB(N)
      DO 1 I=1,N
  1   IB(I)=IA(I)
      RETURN
      END
C
      SUBROUTINE DUIB2A(IA,IB,N)
      !DUplicate Integer array
      DIMENSION IA(N),IB(N)
      DO 1 I=1,N
  1   IA(I)=IB(I)
      RETURN
      END
C
      SUBROUTINE CHKRMIN(A,AMIN,STRING,NCHARS)
      IMPLICIT REAL*8 (A-H,O-Z)
C Check that A does not exceed AMIN.  Stop if it does.
      CHARACTER*1 STRING(NCHARS)
      IF(A.LT.AMIN) THEN
        WRITE(6,*) ' Error detected by CHKREAL:'
        WRITE(6,*) ' A=',A,' value of integer'
        WRITE(6,*) ' AMIN=',AMIN,' MINimum value'
        WRITE(6,*) STRING
        STOP
      ENDIF
      RETURN
```

```
      END
C
      subroutine clear(a,m)
      implicit double precision (a-h,o-z)
      dimension a(M)
      do 100 i=1,m
      a(i) = 0.
  100 continue
      return
      end
c
      subroutine iclear(ia,m)
      dimension ia(M)
      do i=1,m
        ia(i) = 0
      enddo
      return
      end
c-----------------------------------------------------------------------
      SUBROUTINE CHKINT(NN,NNMAX,STRING,NCHARS)
C Check integer, NN
      CHARACTER*1 STRING(NCHARS)
      IF(NN.GT.NNMAX) THEN
        WRITE(6,*) ' Error detected in CHKINT (Check integer):'
        WRITE(6,*) ' NN=',NN,' value of integer'
        WRITE(6,*) ' NNMAX=',' maximum value'
        WRITE(6,*) STRING
        STOP
      ENDIF
      RETURN
      END
C
c-----------------------------------------------------------------------
      DOUBLE PRECISION FUNCTION AINTERP(X,XX,YY,NPTS,NDIFF)
C Function for linear interpolation from a look-up table.
C Uses extrapolation if argument TT is outside range [XX(1),XX(NPTS)]
C XX values must be monotonic (increasing or decreasing)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XX(NPTS),YY(NPTS)
      N1=1
      N2=NPTS
      TT=X
      X1=XX(N1)
      X2=XX(N2)
      GXI=SIGN(1.D0,X2-X1)
      TT=GXI*TT
      X1=GXI*X1
      X2=GXI*X2
      IF(TT.GE.X2) THEN
        N1=NPTS-1
        X1=GXI*XX(N1)
      ELSE IF(TT.LE.X1) THEN
        N2=2
        X2=GXI*XX(2)
      ELSE
        DO WHILE(N2-N1.GT.1)
          NM=(N2+N1)/2
          XM=GXI*XX(NM)
          IF(TT.GT.XM) THEN
            N1=NM
            X1=XM
          ELSE
            N2=NM
            X2=XM
          ENDIF
        ENDDO
      ENDIF
      IF(NDIFF.EQ.0) THEN
        AINTERP=YY(N1)+(YY(N2)-YY(N1))*(TT-X1)/(X2-X1)
      ELSE IF(NDIFF.EQ.1) THEN
        AINTERP=GXI*(YY(N2)-YY(N1))/(X2-X1)
      ELSE
        AINTERP=0.D0
      ENDIF
      RETURN
      END
C
```

```
C History of Development
C This history entry added to file dated 1997.
C 3 July 2004: generalised so that program can handle nonotonically
C   decreasing as well as monotonically increasing values in the XX
C   array.
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
```