Formulation of a User MPC to Simulate Beam-Type Transport Problems

Ted Diehl
Senior Development Engineer
Eastman Kodak company
Kodak Park, Bldg. 23, Floor 8
Rochester, NY 14652-4391

MFSP Internal Report
Department of Mechanical Engineering
University of Rochester, NY
June 16, 1993

Abstract

Analyzing the transport of flexible sheet media such as paper or film has traditionally been extremely difficult due to the complexities of large displacements, large rotations, intermittent contact, and friction. Practical solutions to problems of this type are possible with the nonlinear finite element capabilities available in ABAQUS/Standard.[1] Efficient and robust solutions are obtained through the development of a user-defined rigid-beam MPC subroutine that applies and removes the required boundary conditions that a typical drive nip exerts on a transported sheet. This subroutine is based on a standard 2-D nonlinear rigid-beam constraint. The user-defined MPC subroutine contains additional algorithms which control the application or removal of the rigid constraints as needed by the analysis.

1.0 Introduction

In the design of equipment that transports thin, very flexible sheets such as paper or film, a sheet is often pushed by a set of nip drive rollers through channels with complicated contours (see Figure 1.1). Because the sheet is very thin, it deforms primarily due to bending and experiences only small strains. The difficulty in modelling this type of problem is that the sheet undergoes large rotations and large displacements due to bending and that the boundary conditions on the sheet are continuously changing. Both the drive rollers and the channels cause the boundary conditions to change. Because of these nontrivial boundary conditions and the finite motions of the sheet, numerical methods are required for solution. The traditional method of attacking this problem employs finite

1. ABAQUS/Standard is a commercially available nonlinear finite element code written by Hibbitt, Karlsson & Sorensen, Inc. For this report, ABAQUS/Standard will often be referred to as ABAQUS.

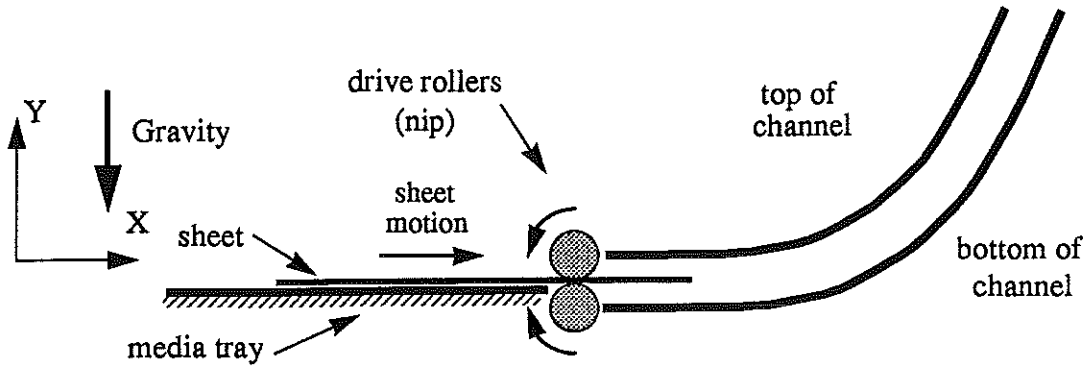


Figure 1.1: Schematic of sheet being transported into a channel.

difference techniques to solve the equations of the Elastica (a thin inextensible beam that experiences large rotations, large displacements, and small strains due to bending). The solution of these equations include additional algorithms to simulate the ever-changing boundary conditions. References [2, 3, and 7] provide discussion of the Elastica formulation. Solutions to these problems can also be obtained with the nonlinear finite element methods available in ABAQUS. The generic nature of a commercial finite element code is extremely advantageous when developing solutions to complicated industrial sheet-transport problems. This report describes the formulation of a user-defined MPC[1] subroutine that can be used in conjunction with *flexible* beam elements and *rigid surfaces* to simulate many sheet transport problems such as the problem depicted in Figure 1.1. A listing of the MPC subroutine is found in Appendix B

Proper & efficient modeling of the ever-changing boundary conditions is very important for developing solutions to sheet transport problems. For finite element solutions via ABAQUS, the changing boundary conditions caused by the sheet contacting the channel can be modeled with rigid elements. Diehl [5] provides an in-depth discussion of the use of rigid elements for this problem along with many other aspects regarding the simulation of sheet-transport via nonlinear finite element methods. The changing boundary conditions produced by the nip rollers are more difficult to handle. This report will concentrate only on the details of using rigid MPC relationships to simulate the drive roller nip boundary condition.

1. MPC stands for Multi-Point Constraint.

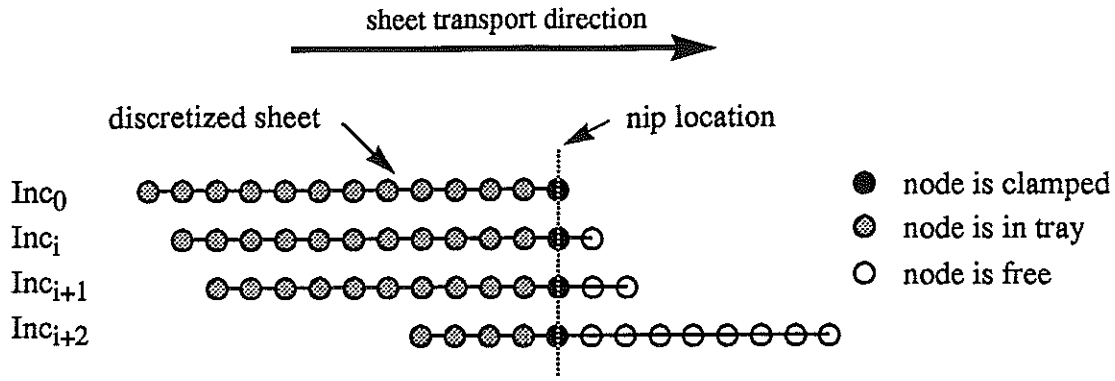


Figure 2.1: Schematic of a discretized sheet transporting through the nip.

2.0 Simulation of Boundary Conditions Produced by Drive Rollers

Referring to Figure 1.1, it is seen that the drive rollers produce a nip that generates a clamped boundary condition (zero Y-displacement, zero rotation about the Z-axis, and a specified motion in the X-direction) on the sheet. As the sheet transports through the nip, the location of the clamped boundary condition continuously moves with respect to the sheet. Figure 2.1 depicts a discretized sheet transporting through the nip at different times in a simulation. The time is denoted by the various increments (Inc_0, Inc_i, etc.). As the sheet transports through the nip, nodes become *free* of the clamped condition and new nodes become clamped by the nip. Because the clamped boundary condition totally constrains all degrees of freedom (DOF) at the nip location on the sheet, it essentially decouples the sheet into two regions: nodes that are in the tray and nodes that are out of the tray (free). This problem could be simulated by applying a clamped boundary condition to a different node at each time step while advancing the sheet forward. This method, known as the *BOUNDARY method, is discussed in [5] and is found to be very robust but inefficient and lengthy to code into an input deck.

The easiest boundary condition to apply to the sheet is one that does not change during the solution. Hence, it would be easy to apply a clamped boundary condition at the far left edge of the sheet and simply specify that the sheet should be displaced forward in the direction of sheet transport. Unfortunately, this does not properly model the location of the clamped boundary condition on the sheet (which is generally somewhere in the middle of the sheet) and would produce very incorrect answers for most simulations. However, if this far left node was clamped and *connected* by a rigid MPC to all the other nodes deemed to be inside the tray, then this would have the effect of having the clamped

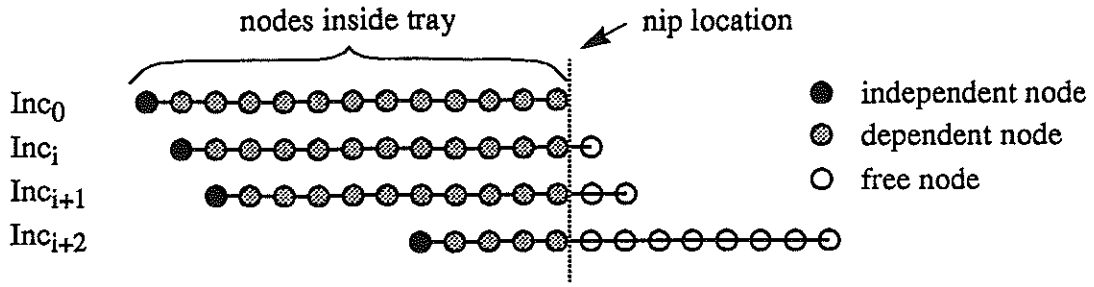


Figure 2.2: MPC method of modeling nip boundary condition during transport.

boundary condition being correctly applied to the node at the nip location. This MPC method is depicted in Figure 2.2. In this method, each node inside the tray (except the far left node) is a dependent node. Each dependent node is connected to the far left node, which is defined as independent, by a rigid MPC relationship. The method could be directly implemented with ABAQUS except that there is no method of *turning off* the MPC relationship between the independent node and any dependent node that has cleared the nip and should be deemed free. This problem can solved with the user-defined MPC subroutine capability of ABAQUS. With a user-defined MPC subroutine, additional logic can be coded that will allow any set of MPC relationships to be applied and removed throughout the solution based on nodal locations and other solution parameters.

2.1 General Concept of User-Defined MPC Subroutine

To define a rigid MPC with on/off logic capability, a user-defined subroutine must be written that provides sufficient information such that ABAQUS can apply the correct constraint relationships at the proper time in the solution. The constraint equations used in this report are the same as those developed in the ABAQUS/Standard Users's Manual [1]. The subroutine developed is coded for MPC MODE = NODE which means that the constraint equations for all dependent DOF's of a given dependent node are imposed at the same time for a given call to the subroutine. To apply general on/off logic capabilities, the user supplies additional dummy nodes on the *MPC card that are used to define the drive nip location. The user-defined MPC subroutine listed in the ABAQUS/Standard Users's Manual requires modification because of the additional dummy nodes and on/off algorithms. Additional derivatives of the constraint functions with respect to the dummy nodes must be computed as well as additional matrix DOF identifiers for the additional nodes.

For the nodal version of the user-defined MPC subroutine (MODE = NODE), a general constraint equation is of the form:

$$f_i(\underset{\sim}{u}^1, \underset{\sim}{u}^2, \underset{\sim}{u}^3, ..., \underset{\sim}{u}^N, \text{geom., temp., field vars.}) = 0 \qquad i = 1, 2, ..., \text{NDEP} \qquad (2.1)$$

where NDEP is the number of dependent DOFs involved in the constraint, N is the total number of nodes involved in the constraint, $\underset{\sim}{u}$ represent the DOFs for each node, and the bold superscript on each DOF vector identifies the node[1]. The maximum number of DOFs used in the MPC is defined as MDOF. The value of NDEP should be between 1 and MDOF. The first nodal DOF vector, $\underset{\sim}{u}^1$, is dependent and will eliminated. This vector must contain NDEP DOFs. All other nodal DOF vectors are independent and can contain between 1 and MDOF degrees of freedom. For the general case of a 3-D structural model, the DOF for any node would usually be defined as

$$\underset{\sim}{u}^T = \left[u_1\ u_2\ u_3\ u_4\ u_5\ u_6\right] = \left[u_x\ u_y\ u_z\ \phi_x\ \phi_y\ \phi_z\right] \qquad (2.2)$$

where $\underset{\sim}{u}^T$ represents the transpose of $\underset{\sim}{u}$. The first three components represent displacement DOFs and the last three components represent rotational DOFs. The MPC subroutine used in this report is for 2-D (X-Y plane) analysis only. Hence, we have

$$\underset{\sim}{u}^T = \left[u_1\ u_2\ u_6\right] = \left[u_x\ u_y\ \phi_z\right] \qquad (2.3)$$

Expansion of the MPC equations for 3-D analysis requires special care with respect to rotational DOFs and is described in [1].

For ABAQUS to process a user defined MPC, the user supplies three components:

1. A matrix of DOF identifiers, JDOF(MDOF,N). This will be described shortly.
2. Matrices representing derivatives of the constraint function with respect to the nodal DOFs. These are of the form:

$$A^1_{ij} = \frac{\partial f_i}{\partial u^1_j}, \qquad A^2_{ij} = \frac{\partial f_i}{\partial u^2_j}, \qquad ... \qquad (2.4)$$

where the subscripts i and j are counters for the degrees of freedom. For our 2-D analysis, both i and j take values from $1-3$ where:

$i = 1$	$\rightarrow$ X-displacement	$j = 1$	$\rightarrow$ X-displacement
$i = 2$	$\rightarrow$ Y-displacement	$j = 2$	$\rightarrow$ Y-displacement
$i = 3$	$\rightarrow$ Z-rotation	$j = 3$	$\rightarrow$ Z-rotation

1. The bold superscript generally does not denote the actual node number. It is a counter for the nodes in the constraint.

Based on Equation 2.3, this appears to be inconsistent. For Equation 2.4, a subscript value of 3 denotes Z-rotation while in Equation 2.3, the Z-rotation is denoted by a subscript value of 6. This is why the matrix DOF identifier, JDOF, is required. For our 2-D analysis, most nodes will have three DOFs (u_x, u_y, ϕ_z). Thus, for any node **k**, the JDOF vector would be defined as:

$$\text{JDOF}(1{:}3,\mathbf{k}) = \begin{bmatrix} 1 \\ 2 \\ 6 \end{bmatrix}$$

3. The values of the dependent DOFs based on the independent DOF values. This step is optional. ABAQUS will compute the dependent DOF values based on the linearized constraint function if the dependent DOFs are not directly computed by the subroutine. In the later case, the constraint equations are not likely to be satisfied exactly.

More detailed descriptions of the general capabilities of user-defined MPC subroutines are found in [1].

2.2 Equations for Rigid MPC

In this section, the basic equations used to define the rigid MPC are defined. The detailed derivation of these equations is found in Appendix A.

Figure 2.3 depicts two nodes (**a** - dependent, **b** - independent) connected by a rigid beam of length L in the X-Y plane. In the figure, t denotes time.[1] For all derivations, we will need to keep track of the original locations ($t = 0$) of the nodes, denoted by (X, Y), and the current ($t > 0$) locations of the nodes, denoted by (x, y). A superscript, such as $X^{\mathbf{a}}$, denotes that the variable X is associated with node **a**. The X-displacement and Y-displacement of any node **c** are defined as

$$u_x^{\mathbf{c}} \equiv x^{\mathbf{c}} - X^{\mathbf{c}} \tag{2.5}$$

$$u_y^{\mathbf{c}} \equiv y^{\mathbf{c}} - Y^{\mathbf{c}} \tag{2.6}$$

The rotational DOF for any node **c** about the Z-axis will be denoted as $\phi_z^{\mathbf{c}}$. For derivation purposes the angle that the rigid beam, between nodes **a** and **b** at time $t = 0$, makes with the X-axis is denoted as Φ_z. Since the beam between nodes **a** and **b** is rigid, the length L is a constant with respect to time. At time $t = 0$, we have

1. The time used in these derivation can be psuedotime for quasi-static analysis or *real* time for dynamic (inertial affects included) analysis.

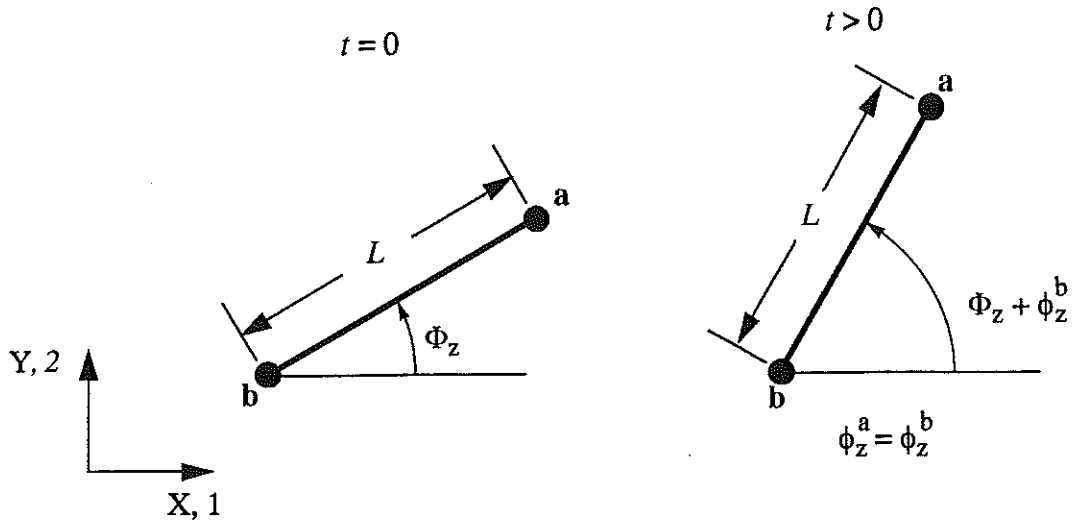


Figure 2.3: Schematic for rigid MPC in X-Y plane

$$L = \sqrt{L_X^2 + L_Y^2} \tag{2.7}$$

where

$$L_X = X^a - X^b = L\cos\Phi_z \tag{2.8}$$

$$L_Y = Y^a - Y^b = L\sin\Phi_z \tag{2.9}$$

The three constraint equations written in terms of nodal displacements and rotations are:

$$f_1(\underset{\sim}{u}^a, \underset{\sim}{u}^b) = (u_x^a - u_x^b) + L_X(1 - \cos\phi_z^b) + L_Y\sin\phi_z^b = 0 \tag{2.10}$$

$$f_2(\underset{\sim}{u}^a, \underset{\sim}{u}^b) = (u_y^a - u_y^b) + L_Y(1 - \cos\phi_z^b) - L_X\sin\phi_z^b = 0 \tag{2.11}$$

$$f_3(\underset{\sim}{u}^a, \underset{\sim}{u}^b) = \phi_z^a - \phi_z^b = 0 \tag{2.12}$$

where for any node **c**

$$\left[\underset{\sim}{u}^c\right]^T = \left[u_x^c\ u_y^c\ \phi_z^c\right] = \left[u_1^c\ u_2^c\ \phi_6^c\right] \tag{2.13}$$

The solution of Equations 2.10 - 2.12 in terms of the dependent DOFs yields

$$u_x^{\mathbf{a}} = u_x^{\mathbf{b}} - L_X(1-\cos\phi_z^{\mathbf{b}}) - L_Y\sin\phi_z^{\mathbf{b}} \tag{2.14}$$

$$u_y^{\mathbf{a}} = u_y^{\mathbf{b}} - L_Y(1-\cos\phi_z^{\mathbf{b}}) + L_X\sin\phi_z^{\mathbf{b}} \tag{2.15}$$

$$\phi_z^{\mathbf{a}} = \phi_z^{\mathbf{b}} \tag{2.16}$$

Equations 2.14 - 2.16 are used to compute the updated values of the dependent DOFs. In the FORTRAN coded subroutine, the updated dependent DOFs are stored in the vector UE(NDEP).

Since ABAQUS uses a nonlinear equation solver, derivatives of the constraint functions with respect to nodal DOFs are required. Following the form of Equation 2.4, we compute the partial derivatives of the constraint functions (Equations 2.10 - 2.12) with respect to the nodal DOFs. For the dependent node, **a**, the following matrix results:

$$\left[A^{\mathbf{a}}\right] = \left[A^{\mathbf{1}}\right] = \begin{bmatrix} \frac{\partial f_1}{\partial u_1^{\mathbf{a}}} & \frac{\partial f_1}{\partial u_2^{\mathbf{a}}} & \frac{\partial f_1}{\partial u_3^{\mathbf{a}}} \\ \frac{\partial f_2}{\partial u_1^{\mathbf{a}}} & \frac{\partial f_2}{\partial u_2^{\mathbf{a}}} & \frac{\partial f_2}{\partial u_3^{\mathbf{a}}} \\ \frac{\partial f_3}{\partial u_1^{\mathbf{a}}} & \frac{\partial f_3}{\partial u_2^{\mathbf{a}}} & \frac{\partial f_3}{\partial u_3^{\mathbf{a}}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.17}$$

This matrix is coded in the FORTRAN subroutine and stored as A(1:3,1:3,**1**) where **1** is the node identifier. The matrix DOF identifier for the dependent node, **a**, is defined as

$$\text{JDOF}(1{:}3,\mathbf{1}) = \begin{bmatrix} 1 \\ 2 \\ 6 \end{bmatrix} \tag{2.18}$$

Calculation of the partial derivative of the constraint functions with respect to the independent node, **b**, yields

$$\left[A^{\mathbf{b}}\right] = \left[A^{\mathbf{2}}\right] = \begin{bmatrix} \frac{\partial f_1}{\partial u_1^{\mathbf{b}}} & \frac{\partial f_1}{\partial u_2^{\mathbf{b}}} & \frac{\partial f_1}{\partial u_3^{\mathbf{b}}} \\ \frac{\partial f_2}{\partial u_1^{\mathbf{b}}} & \frac{\partial f_2}{\partial u_2^{\mathbf{b}}} & \frac{\partial f_2}{\partial u_3^{\mathbf{b}}} \\ \frac{\partial f_3}{\partial u_1^{\mathbf{b}}} & \frac{\partial f_3}{\partial u_2^{\mathbf{b}}} & \frac{\partial f_3}{\partial u_3^{\mathbf{b}}} \end{bmatrix} = \begin{bmatrix} -1 & 0 & L_x \sin\phi_z^{\mathbf{b}} + L_y \cos\phi_z^{\mathbf{b}} \\ 0 & -1 & L_y \sin\phi_z^{\mathbf{b}} - L_x \cos\phi_z^{\mathbf{b}} \\ 0 & 0 & -1 \end{bmatrix} \tag{2.19}$$

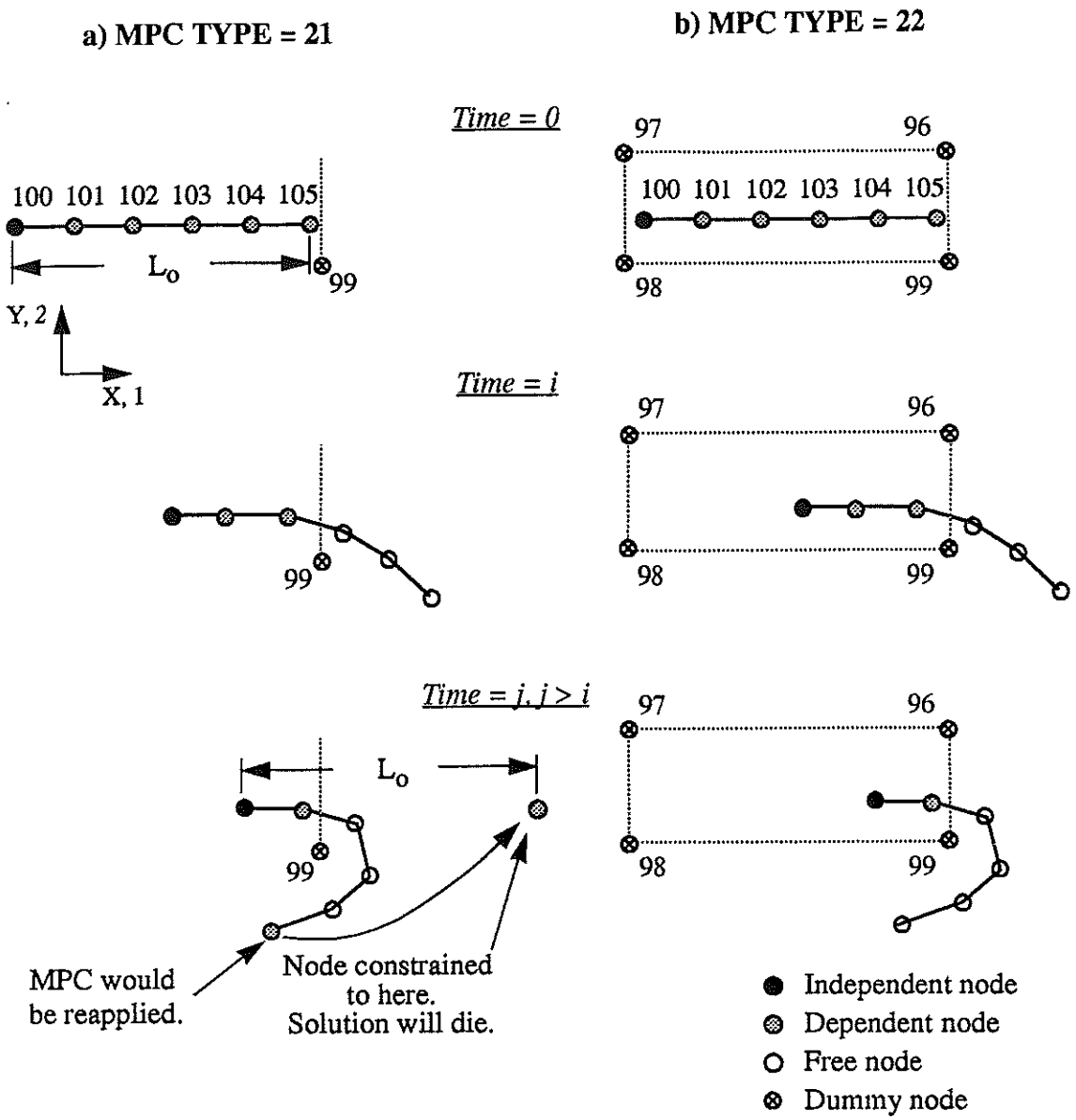
This matrix is coded in the FORTRAN subroutine and stored as A(1:3,1:3,**2**). The matrix DOF identifier for the dependent node, **b**, is defined as

$$\text{JDOF}(1{:}3,\mathbf{2}) = \begin{bmatrix} 1 \\ 2 \\ 6 \end{bmatrix} \tag{2.20}$$

At this point, we have satisfied the basic requirements for coding a user-defined MPC: compute matrix DOF identifiers (Equations 2.18 and 2.20), compute the partial derivatives of the constraint function with respect to the dependent and independent DOFs (Equations 2.17 and 2.19), and compute the updated values of the dependent DOFs (Equations 2.14 – 2.16). Next, the on/off logic must be defined such that the MPC can be imposed or released as required.

2.3 On/Off Logic for MPC

Figure 2.4 shows the two types of on/off logic that are coded into the user subroutine. The FORTAN code for this MPC subroutine is found in Appendix B. For TYPE = 21, the MPC subroutine only checks to see if the X coordinate of a given dependent node is past a single dummy node (depicted as node 99 in Figure 2.4). For most cases, this check is sufficient. However, if the loading is such that any node would attempt to deflect such that its X location is less than that of the dummy node ($t = j$ in Figure 2.4), the solution will die. This is because the MPC will be reapplied to the dependent node which in turn will cause this node to be constrained to its original length from the independent node (node 100 in the figure). The second type of on/off logic, TYPE = 22, uses a more complex checking algorithm to determine if a given dependent node should be constrained or not. This second MPC type checks whether or not a given node is inside a four-sided region defined by four dummy nodes. This quadrilateral is divided into two triangles and the checking algorithm uses the shape functions of a triangle to determine a given node's status. If the node is inside the quadrilateral, the MPC is applied; otherwise it is removed.



Nodes 96, 97, 98, 99 are dummy nodes to define where MPC is on/off

Input Deck Cards

```
*NSET, NSET=DEPEND, GENERATE
101, 105, 1
*MPC, USER, MODE=NODE
21, DEPEND, 100, 99
```

```
*NSET, NSET=DEPEND, GENERATE
101, 105, 1
*MPC, USER, MODE=NODE
22, DEPEND, 100, 96, 97, 98, 99
```

Figure 2.4: Schematic of two versions of user MPC subroutine: (a) simple check of X coordinates of nodes and (b) check of whether nodes are inside or outside of the quadrilateral defining the drive mechanism space.

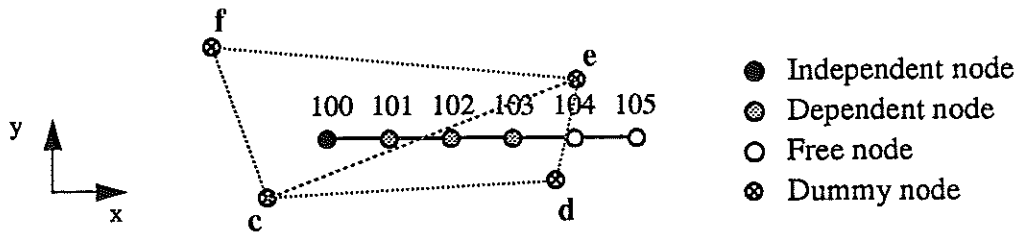


Figure 2.5: Schematic of quadrilateral region subdivided into two triangles for use with MPC TYPE = 22 on/off algorithm

This additional checking adds little to the computational costs of the solution. Both MPC types, TYPE = 21 and TYPE = 22, are capable of simulating problems that involve *pushing* sheets out of a nip or *pulling* sheets into a nip. The algorithms coded are not limited to motion only in the X-direction. The user-defined MPC subroutine, as coded, can handle any generic drive exit orientation in the X-Y plane. Diehl [5] demonstrates many of the capabilities and robustness of the user-defined MPC subroutine on several benchmark problems.

2.3.1 Equations required to implement on/off algorithm for MPC TYPE=22

Figure 2.5 depicts a general quadrilateral region defined by four dummy nodes (**c**, **d**, **e**, and **f**). The region is artificially divided, by the algorithms coded in the user-defined MPC subroutine, into two triangles: **cde** and **cef.** To determine if a given node should have the MPC enforced, it must be inside either **cde** or **cef.** For each potential dependent node, the natural area coordinates (ζ_1, ζ_2, ζ_3) are computed for each triangle based on shape function formulae derived for a straight sided triangle (See Cook [4]). These natural area coordinates for a general triangle are depicted in Figure 2.6. For a given triangle, if all the values of ζ_1, ζ_2, and ζ_3 for any given node are between 0 and 1, then that node would be inside the triangle and the MPC would be enforced. Referring to Figure 2.6, the area coordinates (ζ_1, ζ_2, ζ_3) are related to the physical coordinates (x, y) by

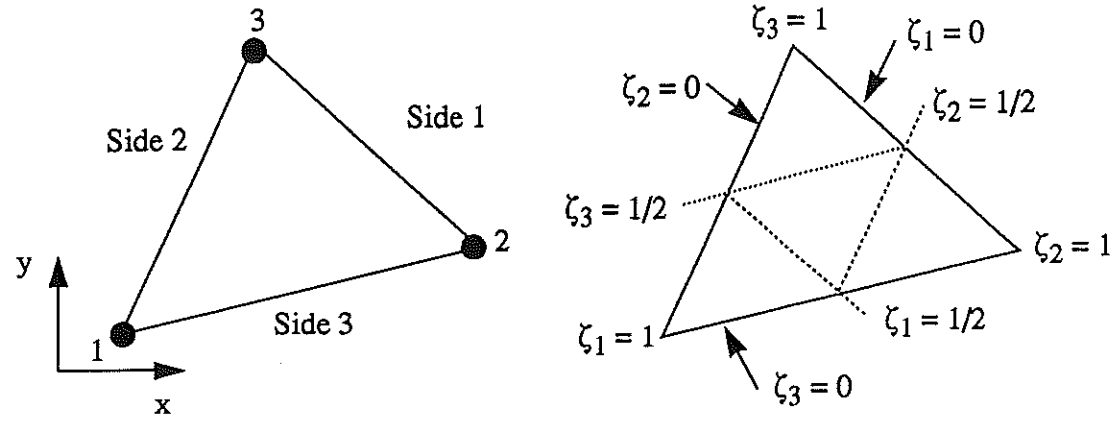


Figure 2.6: Natural area coordinates for a triangle

$$\begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \mathbf{[B]} \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix} = \mathbf{[B]}^{-1} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} \tag{2.21}$$

where

$$\mathbf{[B]} = \begin{bmatrix} 1 & 1 & 1 \\ x^1 & x^2 & x^3 \\ y^1 & y^2 & y^3 \end{bmatrix} \quad \text{and} \quad \mathbf{[B]}^{-1} = \frac{1}{2B} \begin{bmatrix} x^2y^3 - x^3y^2 & y^{23} & x^{32} \\ x^3y^1 - x^1y^3 & y^{31} & x^{13} \\ x^1y^2 - x^2y^1 & y^{12} & x^{21} \end{bmatrix} \tag{2.22}$$

$$2B = \det \mathbf{[B]} = x^{21}y^{31} - x^{31}y^{21} \tag{2.23}$$

and

$$x^{ij} = x^i - x^j \quad \text{and} \quad y^{ij} = y^i - y^j \tag{2.24}$$

The equations above use the same notation as before: a bold superscript identifies a node. For Equations 2.22 – 2.24, the superscripts range from 1 to 3 for the three nodes of the triangle depicted in Figure 2.6. These equations assume that the triangle is numbered counter-clockwise. This is consistent with the numbering scheme required by ABAQUS for 2-D analysis. For the on/off algorithm, the three area coordinates of any given dependent node are calculated as

$$\begin{bmatrix}\zeta_1\\ \zeta_2\\ \zeta_3\end{bmatrix} = \frac{1}{x^{21}y^{31} - x^{31}y^{21}} \begin{bmatrix} x^2y^3 - x^3y^2 & y^{23} & x^{32} \\ x^3y^1 - x^1y^3 & y^{31} & x^{13} \\ x^1y^2 - x^2y^1 & y^{12} & x^{21} \end{bmatrix} \begin{bmatrix} 1 \\ x^a \\ y^a \end{bmatrix} \tag{2.25}$$

Equation 2.25 must be evaluated for the triangles defined by dummy nodes (**c, d, e**) and (**c, e, f**) of Figure 2.5. For triangle **cde**, the superscripts (**1, 2, 3**) in Equation 2.25 are replaced by superscripts (**c, d, e**). A similar exchange is performed for triangle **cef**. A dependent node is deemed to be inside a given triangle if

$$0 \le \zeta_1 \le 1 \quad \text{and} \quad 0 \le \zeta_2 \le 1 \quad \text{and} \quad 0 \le \zeta_3 \le 1 \tag{2.26}$$

Since additional dummy nodes are required to be entered on the *MPC card, ABAQUS will expect to see the partial derivatives of the constraint function with respect to these dummy nodes (see Equations 2.4, 2.10 – 2.12). These nodes do not enter into the constraint equations, Equations 2.10 – 2.12, and the derivatives are all zero. For MPC TYPE = 22, the following must be defined for the four dummy nodes.

$$\text{A(1:3,1:3,k)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \text{JDOF(1:3,k)} = \begin{bmatrix} 1 \\ 2 \\ 6 \end{bmatrix} \qquad k = 3, 4, 5, 6 \tag{2.27}$$

For MPC TYPE = 21, only one dummy node is required and k = 3 only.

Care must be taken when coding the MPC subroutine with logic that turns the constraint equations on and off throughout the analysis. For each MPC defined, the MPC subroutine is called three times during each analysis iteration. The first call is to compute the transformation matrix required for elimination of dependent DOFs from the system matrix. After the system matrix is solved, the MPC subroutine is called two more times: once again to compute the transformation matrix[1] and once to recover the dependent DOF values. It is important that the MPC state (on/off) does not change during an iteration to avoid a change in the transformation matrix definition. The checking logic of the subroutine must use the ABAQUS supplied variable UINIT (displacements at the beginning of the iteration) to determine on/off logic. The ABAQUS supplied variable U (current values of displacements) must be used for all other constraint-type calculations. This method does present the possibility that the MPC constraint logic may lag the solution by one iteration. In practice, as long as two or more iterations are performed in a

1. This is recomputed because the MPC subroutines do not store it the first time.

given increment, this will not occur. ABAQUS can be forced to almost always take at least two increments by entering the following cards in the first *STEP section of the input deck:

```
*CONTROLS, PARAMETER=FIELD
5.0E-30, , , ,0.005
*CONTROLS, PARAMETER=TIME
, ,1
```

The first two lines make the initial convergence criterion for any given increment unreasonably tight and almost impossible to pass. The command sets the alternative convergence criterion to the default initial convergence (0.5%). The last two lines instruct ABAQUS to use the alternative convergence criterion after one unsuccessful iteration.

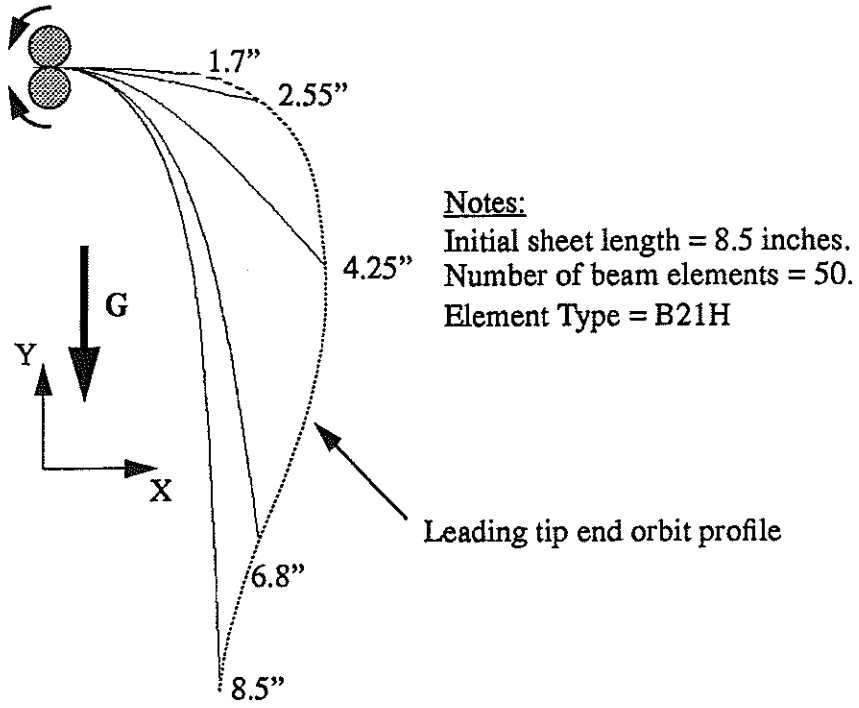
2.4 Example Test Case

As a simple test case, a quasi-static simulation of a sheet being fed into a gravity field is solved. Results of this simulation are presented in Figure 2.7. For the solutions presented, electrostatic, aerodynamic, and inertial dynamic affects are ignored. The material parameters for the sheet are: elastic modulus equals 2.38×10^5 psi, shear modulus equals 1.19×10^5 psi, and mass density equals 6.92×10^{-5} lbf sec^2 in^{-4}. The sheet has a width (Z direction) of 11.0 inches and a thickness of 0.004 inch.

To verify the accuracy of the ABAQUS solutions, a finite difference solution based on equations of the Elastica is used as a benchmark (Stack [6]). In Figure 2.7, three solutions are shown: (1) the Elastica solution of Stack, (2) an ABAQUS solution where the nip boundary condition is simulated with multiple steps using numerous *BOUNDARY definitions (denoted *BOUNDARY method)[1], and (3) an ABAQUS solution where the nip boundary condition is simulated by a user-defined MPC subroutine (MPC TYPE = 22) in one step with an automatic time-step algorithm. Both ABAQUS solutions use fifty B21H beam elements to model the sheet. Since only elastic deformations are considered, the *BEAM GENERAL SECTION option is used instead of the *BEAM SECTION option for defining the beam element properties for the two ABAQUS solutions. This can save in some cases up to 35% CPU time since numerical integration through the thickness of the beam is avoided at each iteration. The ABAQUS solutions were computed with ABAQUS/Standard V5.2 on a Sun SPARCstation 10-30. The user MPC solution required 35 CPU seconds for the entire solution. The input deck for the user MPC solution is found in Appendix C. The *BOUNDARY method input deck is substantially longer and not listed.

1. This method is briefly described in Section 2.0. A more complete description of this method is described by Diehl [5].

a) Deformed Shapes of Media predicted by ABAQUS



b) End Orbit Profile

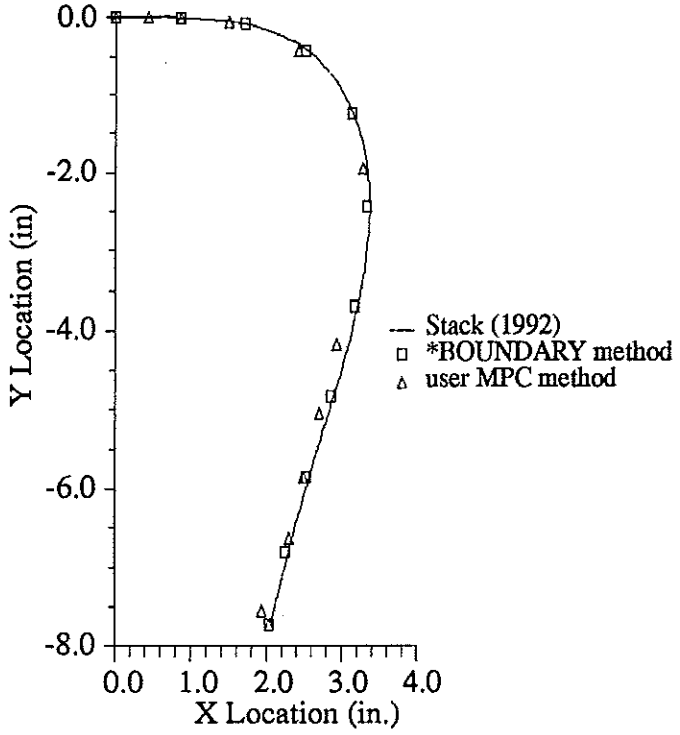


Figure 2.7: Example Test Case

The end orbit profiles plotted in Figure 2.7b compare the two ABAQUS solutions with the benchmark. The *BOUNDARY method solution produces almost the exact same orbit as the benchmark. In the *BOUNDARY method solution, the multiple boundary conditions that were applied in numerous steps ensured that the nip-generated clamped boundary conditions on the sheet were applied to the sheet in the correct (exact) locations.[1] A close inspection of the user MPC method shows that this solution has good agreement with the benchmark (and the *BOUNDARY method as well) but does appear to differ slightly from the benchmark. The user MPC solution's end orbit is always equal to or slightly inside of the end orbit predicted by benchmark. Since both ABAQUS solutions used the same element formulation, the only cause for this difference is the approximate nature of the user MPC technique.

The user MPC technique is approximate in that the effective location of the clamped boundary condition on the sheet may differ from the exact (or desired) location by one element length. This potential error is depicted in Figure 2.8. In general this error is small and can be neglected. Furthermore, the error only occurs when using the automatic time-step algorithm (with a minimal number of steps; such as one) because there is no effort made by the solution algorithm to ensure that a discrete node of the sheet falls exactly at the desired nip location during any given increment.[2] If this error is objectional, three approaches can be used to minimize or eliminate the error: (1) refine the model with smaller elements, (2) increase the number of steps, ensuring that each step ends at a coordinate such that a node on the sheet will be located at the correct nip location, or (3) use the DIRECT option (avoiding automatic time-stepping) on the *STATIC or *DYNAMIC cards and select an increment size such that a node on the sheet will be located at the correct nip location. For most practical problems analyzed with reasonable element sizes, the author has found that these *fixes* are not required to obtain valid answers.

In terms of solving most problems efficiently, it is suggested that the automatic time-step algorithm should always be employed. This allows for very efficient solutions because this method is capable of feeding multiple nodes out of the nip at any given increment with excellent convergence rates. If for complicated problems, the solution requires smaller increments, then the automatic time-step algorithm will adjust accordingly.

1. This confirms that the nonlinear finite element beam formulation for the B21H element is performing correctly.

2. In ABAQUS, loading steps are subdivided into increments.

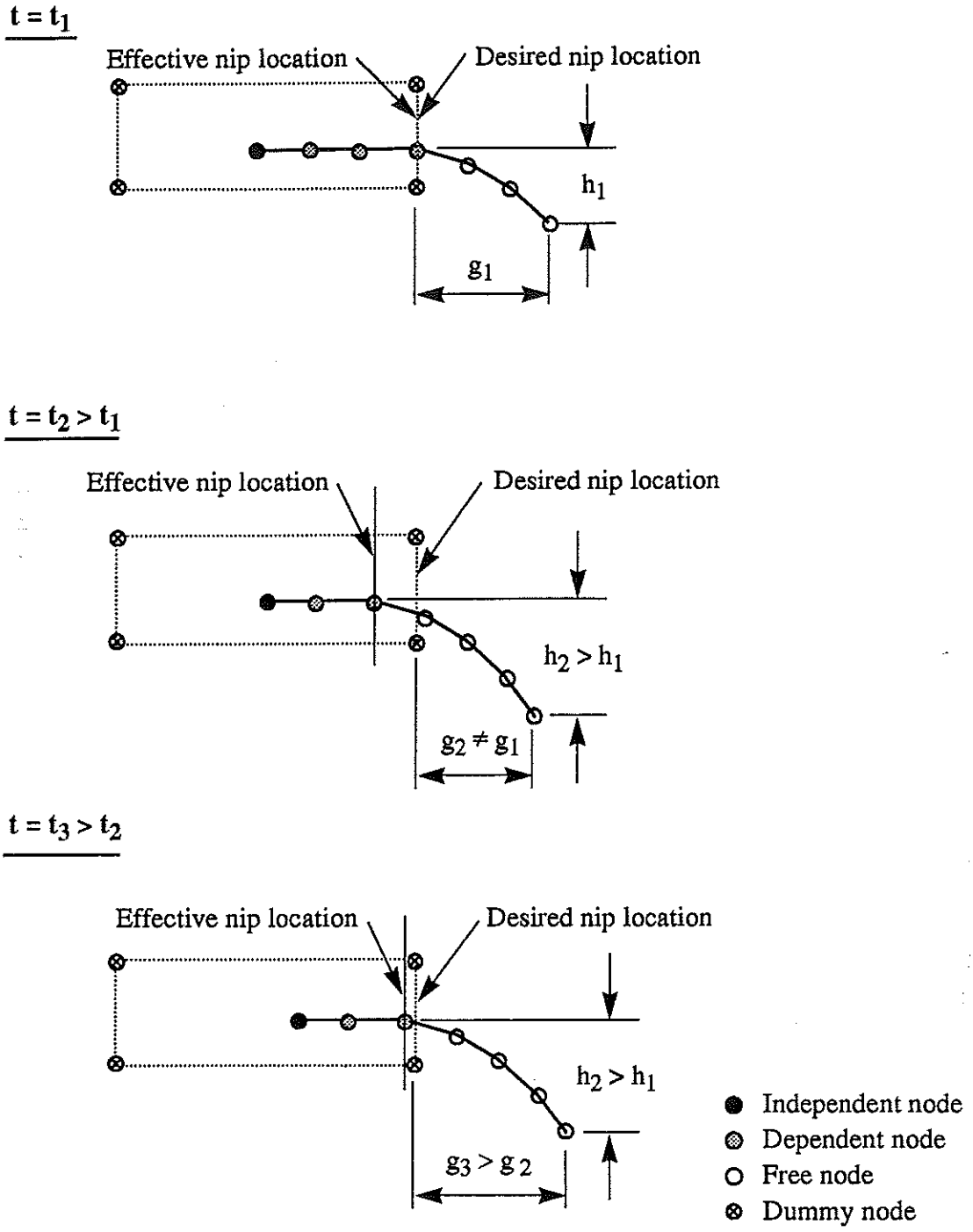


Figure 2.8: Explanation of Potential Error Induced by User MPC with Automatic Incrementation.

3.0 Summary

Analyzing the transport of flexible sheet media can efficiently be performed in ABAQUS/Standard by incorporating a user-defined rigid-beam MPC subroutine with on/off logic. A 2-D user-defined MPC subroutine has been developed with two forms of on/off algorithms: a simple X-coordinate location check and a more robust quadrilateral region check. The second check divides the quadrilateral region into two triangles and uses shape function formulae for straight sided triangles to determine when the constraints need to be applied. Both quasi-static and dynamic analyses are possible.

Acknowledgments

Thanks to Jean Claude Ramirez and Ian Stevenson from HKS for their detailed explanation of the MPC routines in ABAQUS. Thanks to Ken Stack for supplying the benchmark.

References

[1] ABAQUS/Standard Users's Manual, V 5.2, 1992, section 5.2.11

[2] Benson, R., C., "The Deformation of a Thin, Incomplete, Elastic Ring in a Frictional Channel," *Journal of Applied Mechanics,* Vol. 48, No. 4, Dec. 1981, pp. 895-891.

[3] Benson, R. C., "The Feeding of a Thin, Inextensible Beam onto a Rigid Surface," MFSP Internal Report, Department of Mechanical Engineering, University of Rochester, NY, 1983.

[4] Cook, R. D., Malkus, D. S., Plesha, M. E., *Concepts and Applications of Finite Element Analysis*, Third Edition, John Wiley & Sons, New York, 1989, pp. 149-150.

[5] Diehl, T., "Simulating the Transport of Very Flexible Sheets: Beam-Type Solutions," ABAQUS User's Conference Proceedings, Aachen Germany, June 23–25, 1993.

[6] Stack, K. D., Private communication, 1992.

[7] Timoshenko, S. P., Gere, J. M., *The Theory of Elastic Stability*, McGraw-Hill, New York, 1961.

Appendix A: Complete Derivation of MPC Equations

This appendix provides a detailed derivation of the constraint equations for the rigid MPC constraint. For reading ease, some of the equations listed in previous sections will be repeated here and numbered consistently with the other equations in this appendix. The reader should refer to Figure 2.3 for a schematic of the rigid MPC.

The X-displacement and Y-displacement of any general node **c** are defined as

$$u_x^{\mathbf{c}} \equiv x^{\mathbf{c}} - X^{\mathbf{c}} \tag{A.1}$$

$$u_y^{\mathbf{c}} \equiv y^{\mathbf{c}} - Y^{\mathbf{c}} \tag{A.2}$$

where the original location ($t = 0$) of the nodes are denoted by uppercase variables. Lowercase variables denote the current location ($t > 0$) of the nodes. Since the beam between nodes **a** and **b** is rigid, the length L is a constant with respect to time. At time $t = 0$, we have

$$L = \sqrt{L_X^2 + L_Y^2} \tag{A.3}$$

where

$$L_X = X^{\mathbf{a}} - X^{\mathbf{b}} = L\cos\Phi_z \tag{A.4}$$

$$L_Y = Y^{\mathbf{a}} - Y^{\mathbf{b}} = L\sin\Phi_z \tag{A.5}$$

From Figure 2.3, we have at time $t > 0$

$$x^{\mathbf{a}} = x^{\mathbf{b}} + L\cos(\Phi_z + \phi_z^{\mathbf{b}}) \tag{A.6}$$

$$y^{\mathbf{a}} = y^{\mathbf{b}} + L\sin(\Phi_z + \phi_z^{\mathbf{b}}) \tag{A.7}$$

$$\phi_z^{\mathbf{a}} = \phi_z^{\mathbf{b}} \tag{A.8}$$

From Equations A.1 and A.2, we have for any node **c**

$$x^{\mathbf{c}} = X^{\mathbf{c}} + u_x^{\mathbf{c}} \tag{A.9}$$

$$y^{\mathbf{c}} = Y^{\mathbf{c}} + u_y^{\mathbf{c}} \tag{A.10}$$

Using the general form of Equation A.9, we can write Equation A.6 as

$$X^a + u_x^a = X^b + u_x^b + L\cos(\Phi_z + \phi_z^b)$$

which can be rearranged to yield

$$(u_x^a - u_x^b) + (X^a - X^b) - L\cos(\Phi_z + \phi_z^b) = 0 \qquad \text{(A.11)}$$

Substituting Equation A.4 into the above equation yields

$$(u_x^a - u_x^b) + L_X - L\cos(\Phi_z + \phi_z^b) = 0 \qquad \text{(A.12)}$$

Similar manipulation of Equation A.7 gives

$$(u_y^a - u_y^b) + L_Y - L\sin(\Phi_z + \phi_z^b) = 0 \qquad \text{(A.13)}$$

From trigonometry we have the following identities

$$\cos(\theta + \alpha) = \cos\theta\cos\alpha - \sin\theta\sin\alpha \qquad \text{(A.14)}$$

$$\sin(\theta + \alpha) = \sin\theta\cos\alpha + \cos\theta\sin\alpha \qquad \text{(A.15)}$$

Using these identities we have

$$L\cos(\Phi_z + \phi_z^b) = L\cos\Phi_z\cos\phi_z^b - L\sin\Phi_z\sin\phi_z^b \qquad \text{(A.16)}$$

Substituting Equations A.4 and A.16 yields

$$L\cos(\Phi_z + \phi_z^b) = L_X\cos\phi_z^b - L_Y\sin\phi_z^b \qquad \text{(A.17)}$$

Similarly we find

$$L\sin(\Phi_z + \phi_z^b) = L_Y\cos\phi_z^b + L_X\sin\phi_z^b \qquad \text{(A.18)}$$

Substitution of Equation A.17 into Equation A.12 yields

$$(u_x^a - u_x^b) + L_X - L_X\cos\phi_z^b + L_Y\sin\phi_z^b = 0$$

which simplifies to

$$(u_x^a - u_x^b) + L_X(1 - \cos\phi_z^b) + L_Y \sin\phi_z^b = 0 \tag{A.19}$$

Similar manipulation of Equation A.13 leads to

$$(u_y^a - u_y^b) + L_Y(1 - \cos\phi_z^b) - L_X \sin\phi_z^b = 0 \tag{A.20}$$

We now have the three constraint equations required to impose the rigid MPC between nodes **a** and **b**. The three constraint equations written in terms of nodal displacements and rotations are:

$$f_1(\underset{\sim}{u}^a, \underset{\sim}{u}^b) = (u_x^a - u_x^b) + L_X(1 - \cos\phi_z^b) + L_Y \sin\phi_z^b = 0 \tag{A.21}$$

$$f_2(\underset{\sim}{u}^a, \underset{\sim}{u}^b) = (u_y^a - u_y^b) + L_Y(1 - \cos\phi_z^b) - L_X \sin\phi_z^b = 0 \tag{A.22}$$

$$f_3(\underset{\sim}{u}^a, \underset{\sim}{u}^b) = \phi_z^a - \phi_z^b = 0 \tag{A.23}$$

where for any node **c**

$$\left[\underset{\sim}{u}^c\right]^T = \left[u_x^c \; u_y^c \; \phi_z^c\right] = \left[u_1^c \; u_2^c \; \phi_6^c\right] \tag{A.24}$$

The partial derivatives of the constraint equations (Equations A.21 – A.23) are given in Section 2.2.

Appendix B: Subroutine MPC

The subroutine listed in this appendix is named `feeder_2d.f`. It is located in ~diehl/ab/mpc.

```
      SUBROUTINE MPC (UE,A,JDOF,MDOF,N,JTYPE,X,U,UINIT,MAXDOF,LMPC,
     +                KSTEP,KINC,TIME,NT,NF,TEMP,FIELD)
C
C     NONLINEAR RIGID BAR MPC WITH CAPABILITY TO BE TURNED ON & OFF AT ANY TIME
C     SIMULATES A 2-D FEEDER FOR PAPER SIMULATIONS
C     LOCATED IN:     /diehl/ab/mpc/feeder_2d.f
C     WRITTEN BY:     TED DIEHL
C     DATE:           11/17/92
C     MODIFIED:       6/16/93 (documention only)
C
C     This rigid-beam mpc is between two nodes (A - dependent, B - independent).
C     Additional "dummy" nodes (C, D, E, F) are used to determine whether or not the
C     mpc is on or off.  Two forms are allowed (JTYPE 21 - a simple check against dummy
C     node C (nodes D, E, F should not be defined); JTYPE 22 - a more complex check
C     against the quadrilateral defined by dummy nodes (C,D,E,F). More details on this
C     subroutine may be obtained from  Diehl, T., "Formulation of a User MPC to
C     Simulate Beam-Type Transport Problems", MFSP Internal Report, Department of
C     Mechanical Engineering, Univeristy of Rochester, NY.  This will also be published
C     in the Kodak library.
C
C     To insure good results when using this routine, include the following cards in
C     the *STEP section of your input deck. This will ensure that the on/off logic will
C     work correctly.
C
C     *CONTROLS, PARAMETER=FIELD
C     5.0E-30,,,,0.005
C     *CONTROLS, PARAMETER=TIME
C     ,,1
C
C     Defintion of variables:
C     -----------------------
C     The variables used in the subroutine call are defined in the ABAQUS/Standard User's
C     Manual, Vol II, pp. 5.2.11-7 through 5.2.11-11.
C
C     Additional variables defined: (Only the basic "form" of the variables is listed)
C
C     CURXA - current coordinate in the X-direction of node A.
C     XDC   - X coordinate of dummy node D minus the X coordinate of dummy node C
C     CDE   - twice the area of triangle CDE.
C     CEF   - twice the area of triangle CEF
C     Z1CDE - first natural area coordinate of triangle CDE
C     IOFF  - logic check for turning MPC on or off.
C     R**   - implicitly declared Real variable
C     RLX   - Variable Lx defined in report "Formulation ..." listed above.
C
C
C
C     BEGINNING OF SUBROUTINE
C
      INCLUDE 'ABA_PARAM.INC'
C
      DIMENSION UE(MDOF), A(MDOF,MDOF,N), JDOF(MDOF,N), X(6,N),
     +          U(MAXDOF,N), UINIT(MAXDOF,N), TIME(2), TEMP(NT,N),
     +          FIELD(NF,NT,N)
      PARAMETER( ZERO = 0.D0, ONE = 1.D0, TWO = 2.D0, THREE = 3.D0)
C
C
C
C************** LOGIC TO DETERMINE IF CONSTRAINT SHOULD BE TURNED OFF **************
C
C
C MPC jtype = 21 =>  Simple check if node "A" X location is greater than node "C" X location.
      IF(JTYPE .EQ. 21) THEN
          CURXA = X(1,1) + UINIT(1,1)
          CURXC = X(1,3) + UINIT(1,3)
        IF(CURXA .GT. CURXC) THEN
```

```
            LMPC = 0      ! turn off mpc
            GOTO 1000
          ENDIF
C
C
C MPC jtype = 22 => Check if node "A" is outside the box defined by nodes "C,D,E,F"
         ELSEIF(JTYPE .EQ. 22) THEN
C
          IOFF = 0      ! assume mpc is off, check below to see if it should be on.
C
          CURXA = X(1,1) + UINIT(1,1)
          CURYA = X(2,1) + UINIT(2,1)
          XDC = X(1,4) - X(1,3)
          XEC = X(1,5) - X(1,3)
          XFC = X(1,6) - X(1,3)
          XDE = X(1,4) - X(1,5)
          XEF = X(1,5) - X(1,6)
          YDC = X(2,4) - X(2,3)
          YEC = X(2,5) - X(2,3)
          YFC = X(2,6) - X(2,3)
          YDE = X(2,4) - X(2,5)
          YEF = X(2,5) - X(2,6)
C
C
C  Compute twice the area of subtriangle CDE and check if node "A" is inside.
          CDE = XDC*YEC - XEC*YDC
          IF (CDE .LE. ZERO) THEN
             GOTO 105
          ELSE         ! compute natural area coordinate of dependent node A.
            Z1CDE = ((X(1,4)*X(2,5)-X(1,5)*X(2,4)) +
     +                YDE*CURXA - XDE*CURYA)/CDE
            Z2CDE = ((X(1,5)*X(2,3)-X(1,3)*X(2,5)) +
     +                YEC*CURXA - XEC*CURYA)/CDE
            Z3CDE = ((X(1,3)*X(2,4)-X(1,4)*X(2,3)) -
     +                YDC*CURXA + XDC*CURYA)/CDE
          ENDIF
          IF ((Z1CDE .GE. ZERO) .AND. (Z1CDE .LE. ONE) .AND.
     +        (Z2CDE .GE. ZERO) .AND. (Z2CDE .LE. ONE) .AND.
     +        (Z3CDE .GE. ZERO) .AND. (Z3CDE .LE. ONE)) THEN
C
            IOFF = 1    ! signal to keep mpc on.
C
          ENDIF
C
C
105     CONTINUE
C  Compute twice the area of subtriangle CEF and check if node "A" is inside.
          IF (IOFF .EQ. 0) THEN
          CEF = XEC*YFC - XFC*YEC
            IF (CEF .LE. ZERO) THEN
              GOTO 110
            ELSE            ! compute natural area coordinate of dependent node A.
              Z1CEF = ((X(1,5)*X(2,6)-X(1,6)*X(2,5)) +
     +                  YEF*CURXA - XEF*CURYA)/CEF
              Z2CEF = ((X(1,6)*X(2,3)-X(1,3)*X(2,6)) +
     +                  YFC*CURXA - XFC*CURYA)/CEF
              Z3CEF = ((X(1,3)*X(2,5)-X(1,5)*X(2,3)) -
     +                  YEC*CURXA + XEC*CURYA)/CEF
            ENDIF
            IF ((Z1CEF .GE. ZERO) .AND. (Z1CEF .LE. ONE) .AND.
     +        (Z2CEF .GE. ZERO) .AND. (Z2CEF .LE. ONE) .AND.
     +        (Z3CEF .GE. ZERO) .AND. (Z3CEF .LE. ONE)) THEN
C
              IOFF = 1   ! signal to keep mpc on.
C
            ENDIF
```

```
          ENDIF

C
C
110       IF(IOFF .EQ. 0) THEN
            LMPC = 0
            GOTO 1000
          ENDIF
C
C
C MPC type is not jtype 21 or 22
        ELSE
          GOTO 1000
        ENDIF
C
C ***************** APPLY CONSTRAINT EQUATIONS *****************
C
C
C Compute the following values
          COSU6B = COS(U(6,2))
          SINU6B = SIN(U(6,2))
          RLX = X(1,1) - X(1,2)
          RLY = X(2,1) - X(2,2)
C
C Compute the total displacements of node A (the Dependent node)
          UE(1) = U(1,2) - RLX*(ONE - COSU6B) - RLY*SINU6B
          UE(2) = U(2,2) - RLY*(ONE - COSU6B) + RLX*SINU6B
          UE(3) = U(6,2)
C
C Compute derivatives of the constraint function
C
C   Derivatives w.r.t. node "A" (denoted by "1" in the third array location in A)
          A(1,1,1) = ONE
          A(2,2,1) = ONE
          A(3,3,1) = ONE
          A(1,2,1) = ZERO
          A(1,3,1) = ZERO
          A(2,1,1) = ZERO
          A(2,3,1) = ZERO
          A(3,1,1) = ZERO
          A(3,2,1) = ZERO
C
C   Derivatives w.r.t. node "B" (denoted by "2" in the third array location in A)
          A(1,1,2) = -ONE
          A(2,2,2) = -ONE
          A(3,3,2) = -ONE
          A(1,2,2) = ZERO
          A(2,1,2) = ZERO
          A(3,1,2) = ZERO
          A(3,2,2) = ZERO
          A(1,3,2) = RLX*SINU6B + RLY*COSU6B
          A(2,3,2) = RLY*SINU6B - RLX*COSU6B
C
C   Derivatives w.r.t. node "C" (denoted by "3" in the third array location in A)
          DO I=1,3
            DO J=1,3
              A(I,J,3) = ZERO
            END DO
          END DO
C
C   Derivatives w.r.t. nodes "D,E,F" (denoted by "4,5,6" in the third array location in A)
        IF (JTYPE .EQ. 22) THEN
          DO I=1,3
            DO J=1,3
              A(I,J,4) = ZERO
              A(I,J,5) = ZERO
```

```
              A(I,J,6) = ZERO
            END DO
          END DO
        ENDIF
C
C Define Matrix DOF Identifiers
C
        DO I=1,N
          JDOF(1,I) = 1
          JDOF(2,I) = 2
          JDOF(3,I) = 6
        END DO
C
C
1000    RETURN
        END
```

Appendix C: ABAQUS Input Deck for Test Case

The input deck listed here (named `grav_b21h_user.inp`) is for the user-defined MPC solution depicted in Figure 2.7. To execute this file use ABAQUS/Standard V5.2, type the following command at the prompt in a unix shell.

```
abaqus j=grav_b21h_user user=feeder_2d
```

For this deck to run, the MPC subroutine `feeder_2d.f` (from Appendix B) must be in the same directory as the input deck.

The following code is the input deck `grav_b21h_user.inp`.

```
*HEADING
grav_b21h_user.inp
*RESTART, WRITE, FREQ=1
**
*NODE
 100, -8.5, 0.0
 150, 0.0, 0.0
*NSET,NSET=PUSH
100
*NODE,NSET=BOX
9996, 0.0, -0.3
9997, 0.0, 0.3
9998, -9.0, 0.3
9999, -9.0, -0.3
*NGEN, NSET=PAPER
 100,150,1
*NSET,NSET=DEPEND, GENERATE
101,150,1
**
**
*ELEMENT,TYPE=B21H, ELSET=PAPER
 1,100,101
*ELGEN,ELSET=PAPER
 1,50,1,1
**
**
*BEAM GENERAL SECTION, ELSET=PAPER,SECTION=RECT,DENSITY=6.923E-5
11.0,0.004
0.0,0.0,-1.0
2.3775E5, 1.18875E5
**
**
*MPC, USER, MODE=NODE
22, DEPEND, 100, 9996, 9997, 9998, 9999
**
*******************************
**
**
*AMPLITUDE, NAME=D1
0.0,1.0,1.0,1.0
*AMPLITUDE, NAME=A1
0.0,1.0,1.0,1.0
**
```

```
**************************************
**
** STEP 1
**
*STEP,NLGEOM,INC=100, MONOTONIC=YES
PUSH PAPER OUT
*STATIC
0.05,1.0,1.0E-6
** The following parameters are to ensure correct
** results from the User MPC routine
*CONTROLS, PARAMETER=FIELD
5.0E-30,,,,0.005
*CONTROLS, PARAMETER=TIME
,,1
**
**
*DLOAD, AMPLITUDE=D1
PAPER, GRAV, 386.0886, 0.0, -1.0
*BOUNDARY, TYPE=DISPLACEMENT
PUSH, 2,, 0.0
PUSH, 6,, 0.0
BOX, 1,2, 0.0
BOX, 6,, 0.0
**
*BOUNDARY, TYPE=DISPLACEMENT
PUSH, 1,, 8.5
**
*NODE PRINT,FREQ=1
U, COORD
RF
*EL PRINT,FREQ=99
SF
*NODE FILE
U,RF,COORD
*EL FILE,FREQ=1
SF
*END STEP
```