Sandia National Laboratories

Operated for the U.S. Department of Energy by
Sandia Corporation

Albuquerque, New Mexico 87185-0826

date: 1 March 2004

to: Distribution

from: P. R. Schunk, 9114, MS0834; E. D. Wilkes, Gram Inc.

subject: GOMA's Shell Structure Capability: User Tutorial (GT-027.1)

PREFACE

This tutorial assumes the user has gone through the beginner's training tutorial on GOMA (GT- 001.2) and the slot coating tutorial (GT-006.3) or equivalently has in depth working knowledge and experience of converging on steady states for steady capillary hydrodynamics coating problems. If you would like copies of these tutorials, contact Duane Labreche (dalabre@sandia.gov), Randy Schunk (prschun@sandia.gov), or Tom Baer (tabaer@sandia.gov). You also need to have CUBIT to complete these exercises.

Key concepts covered in this tutorial: Structural shell equations (theory of inextensible

shells) EXAMPLE PROBLEMS COVERED IN THIS MEMO:

- Lid driven cavity with bending shell lid
- Slot coater rolling bank with shell web

GOMA Input cards/BC cards and Models discussed:

BC = SH_K, BC =SH_TENS, BC =SH_FLUID_STRESS, EQ = shell_curvature, EQ = shell_tension

Key Limitations of the Algorithm Discussed here (as of 11/1/2003): serial processing only, two- dimensions only. Q2_Bar elements not yet debugged. Q1Q1 elements only.

Introduction

The structural shell equation capability in Goma builds on the shell-element capability built by Pat Notz and Ed Wilkes in FY03. Basically we are solving the following equations for the shell tension, shell curvature, and shell coordinates (all in the shell layer of elements that is either dry or wetted by a surronding fluid):

$$-D\frac{d^2}{dS}K + KT + \underset{\sim}{n}\cdot\underset{\sim}{n}\cdot\underset{\sim}{\sigma} = 0 \quad (1)$$

$$\frac{dT}{dS} - DK\frac{dK}{dS} + \underset{\sim}{t}\cdot\underset{\sim}{n}\cdot\underset{\sim}{\sigma} = 0 \quad (2)$$

$$\frac{d^2X_{sh}}{dS^2} + K\frac{dY_{sh}}{dS} = 0 \quad (3)$$

$$-\frac{d^2Y_{sh}}{dS^2} + K\frac{dX_{sh}}{dS} = 0 \quad (4)$$

$$d_x = X_{sh} - X \quad (5)$$

$$d_y = Y_{sh} - Y \quad (6)$$

These equations result from the theory of inextensible shells. That theory and relevant references are available in a paper by Carvalho ("Elastohydrodynamics of tensioned web roll coating processes", *Int. J. Numer. Meth. Fluids*, Vol 41, pp 561-576, 2003). In these equations we define the bending stiffness $D=Et^3/12(1-\nu^2)$, where E is the elastic modulus, ν Poison ratio, and t the shell thickness. T and K are the web tension and curvature at each position, and X_{sh} and Y_{sh} are the shell coordinates, defined in relation to Goma's stress-free-state or mesh coordinates and mesh displacements by equations 5 and 6. S is the arclength measured along the shell surface; n and t are the normal and tangent vectors to that surface. σ is the fluid stress loading that results when the shell is bounded a fluid region.

Initial testing of these equations was successful but performance and robustness was not acceptable. We found that the formulation was dependent on orientation of the domain and that the nodal dis- placements along an interface between a shell region and the bulk were unstable at large deforma- tions. We suspected an over-specification or some redundancy in Equations 3 and 4. We arbitrarily replaced Eqn. 4 with an "equal arc length equation" on shell regions not bounded by a bulk fluid and let the volume mesh residual replace the same equation for shells bounded by an bulk region with arbitrary pseudo-solid mesh motion. These tactics led to a stable formulation.

We can write the equal element arc length equation using the following surface determinant defini- tion:

$$\frac{dS}{d\xi} = \sqrt{x_\xi^2 + y_\xi^2}$$

and so we would solve

$$\frac{d^2}{d\xi^2} S = 0 \qquad (7)$$

$\frac{dS}{d\xi}$ is the sensitivity of the arclength to the local isoparametric coordinate. Interestingly, we had to make sure that if the initial guess was a flat shell, that the Goma Jacobian sensitivity `ddet_J_dmesh` is set to a small number so that we don't get a singular jacobian.

Implementation Nuances:

- All equations must be applied in the shell element blocks. For simplicity we substitute Equations 5 and 6 into Equations 3 and 4 (more below).
- We apply Eq 7 in place of Eq. 4 for unwetted shells, and the R_MESH2 or R_MESH_TANGENT mesh stress residual equation in place of Eq. 4 in the wetted case. This formulation seems to lead to stable nodal displacements along the shell in both regions.
- Boundary conditions are in 2 forms right now: Dirichlet: SH_K, and SH_TENS. Note that boundary conditions DX and DY are still used in the shell region to pin displacements, due to the substitution cited above.
- Collocated_surf: Two BCs applied with SH_FLUID_STRESS bc card use used to add the fluid stress terms on the shell equations 1 and 2 above. These conditions are applied in the spirit of FLUID_SOLID and NO_SLIP BCs, viz. by using volume-residual manipulations.
- Routines: `assemble_shell_structure` (in `mm_fill_shell.c`) assembles 1 and 2. `assemble_shell_coord` (in `mm_fill_shell.c`) assembles equations 3 and 4. (all debugged and free of Jacobian errors)
- Dynamic contact angle: The application of moving contact angles on a shell surface warrants a special discussion. Here we address DCA application for the steady state case. In this case we set the velocity components of the fluid to zero at the DCL, which effectively satisfies the kinematic boundary condition on the free surface and the overall mass conservation at the substrate, and relieves the infinite stress by allowing for a perfect slip velocity at that point. If you apply these conditions as in the case of a rigid substrate defined by say a PLANE condition, you get an incorrect result, viz. the mesh-displacement equations are completely displaced at the DCL by both the CA condition and the KINEMATIC condition. The shell structure equations being totally dropped fail to pin shell position appropriately. We devised a trick to get around this. If one applies a dummy boundary condition that displaces the KINEMATIC condition but actually leads to no actual contribution, then the shell-coordinate equation will be used at the DCL. We advocate using a GD_LINEAR BC along the substrate with trivial coefficients. If this bc is applied, the BC_dup.txt file indicates the following combination of conditions at the DCL:

  At global node 1 (proc node 1) (7.74597, 0)
  R_MOMENTUM1 gets U (8) from NS
  200

R_MOMENTUM2 gets V (9) from NS
200 R_MESH1 gets CA (12) from NS 200
R_MESH2 gets GD_LINEAR (14) from SS
60
R_SHELL_CURVATURE gets SH_FLUID_STRESS (21) from SS 60
R_SHELL_TENSION gets SH_FLUID_STRESS (21) from SS 60

Note that the kinematic condition is applied with the velocity specification at the DCL. The CA (contact angle condition) is applied to one mesh equation and a dummy GD_LINEAR boundary condition with zero coefficients is applied to the other mesh equation, thus add nothing in addition to the shell-position equation at that point. We will cover this more below.

- Remaining Tasks:

~~Bending Stiffness as property in input~~

~~deck Unwetted shells (unattached~~

~~shells)~~

Q2 Bar elements

Validation of fluid stress

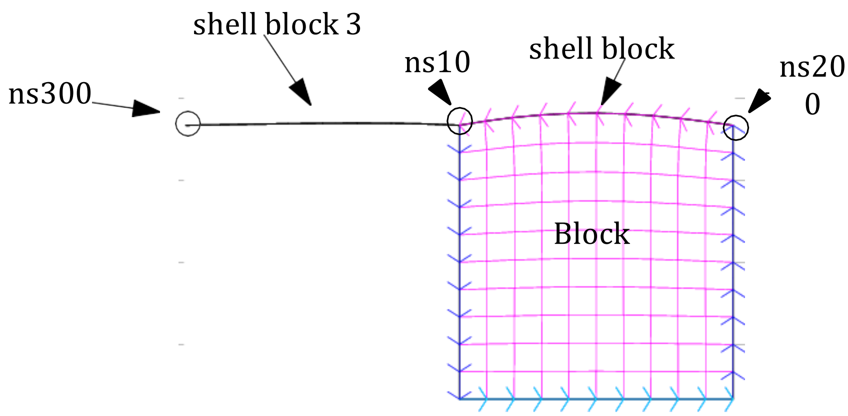
terms Jacobian errors final

check

Contact angle specification at DCL is suspect still. Needs further testing. Seems to work better with KINEMATIC_PETROV for free surface for linear elements.

User tips on geometry setup and other sundry notes

The basic test problems are set up according to the following picture:



Note: the separate mesh blocks for wetted vs. unwetted shell. The key elements in the Cubit journal file required to generate this mesh are highlighted below:

```
## Cubit Version 8.1
## Cubit Build 26
## Revised 11/24/2003 14:05:36 MST
## Running 12/05/2003 10:05:47 AM
## Command Options:
## -warning = On
## -information = On
brick width 1 depth 1 height 2
list volume
brick width 1 depth 1 height 2
body 2 move x -1
merge all
surface all size 0.1
mesh surface 1
curve 14 interval 8
mesh curve 14
...
block 1 surface 1
block 2 curve 2
block 3 curve 14
block 1 element type quad4
block 2 element type bar2
block 3 element type bar2
export genesis "tmp.exoII"
```

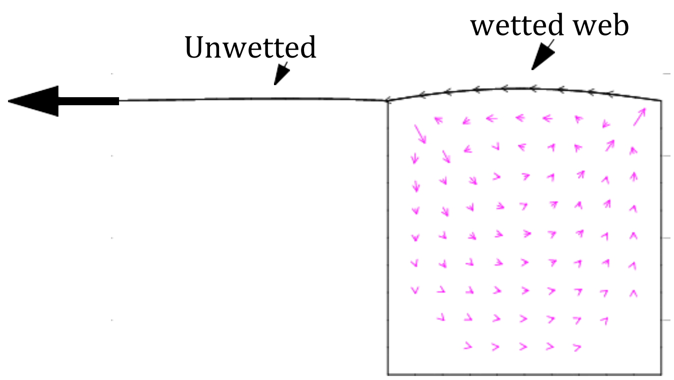
Note that we mesh surface 1 (which corresponds to the bulk cavity mesh block 1 in the picture) and curve 14 (which corresponds to the unattached shell surface in the picture). Curve 2 is the top sur- face of block 1. Note the declaration of 3 element blocks for this merged body. You need separate shell blocks for wetted shells and unwetted shells for several reasons that are beyond the scope of this tutorial.

We list here some trouble-shooting tips:

CANNOT put SH_TENS boundary conditions on both endpoints of a connected shell or GOMA NaNs. This is putting 2 BCs on a first order equation (cf. Eq. 2). Specifically, in the figure above, you can only tension the shell at either NS 200 or NS 300.

Problem 1: Lid-driven cavity

The files required to run this problem are in **shell_element.tst/two_block_tst**. In that directory you will find the Goma input files **input_dry**, **shell.mat**, and **fluid.mat**. Some of the key nodesets and side sets, and the CUBIT mesh file are discussed earlier in this memo. A sample solution is given in the figure below. Note that the web is moving from right to left and it drives flow in the cavity. The flow pressure in the cavity deforms the shell surface. Note that the mesh here is very coarse, which explains the wiggles in the velocity field in the corners. We recommend that you run this problems to get a feel for the shell element behavior before proceeding to the slot-meniscus problem below. This problem is easy to run. Specifically we suggest changing the shell tension at nodeset 300 (increasing it to get rid of the bowing), change the shell end point position at nodeset 300, and change the web speed and end point curvature at various bending stiffnesses and tensions. The boundary conditions which can be altered to make these changes are highlighted below. To get a complete description of the problem setup, viz. the necessary problem description equations, etc., please see the discussion under problem 2.



Boundary Condition Specifications

Number of BC = -1

BC = DX NS 1 0. <<<<<<<<<<<<<Box surface no
slip BC = DY NS 1 0.

BC = DX NS 100 0. <<<<<<<<<<<<pinned shell at left side of box (see figure in previous
section) BC = DY NS 100 0.

BC = DX NS 300 -0.<<<<<<<<<<<<pinned shell position at left hand side of unwetted web (vary this for
effect) BC = DY NS 300 0.

$BC = SH_K NS 100 0.
BC = SH_K NS 200 0.
BC = SH_K NS 300 0.
BC = SH_TENS NS 300 10000. <<<<<<<<<Applied shell tension (only applied to one end of web).

BC = VELO_NORMAL SS 2 0.
BC = VELO_TANGENT SS 2 0 5.e1 0. 0. <<<<<<<<<<<<No slip and impenetrability on web/fluid surface

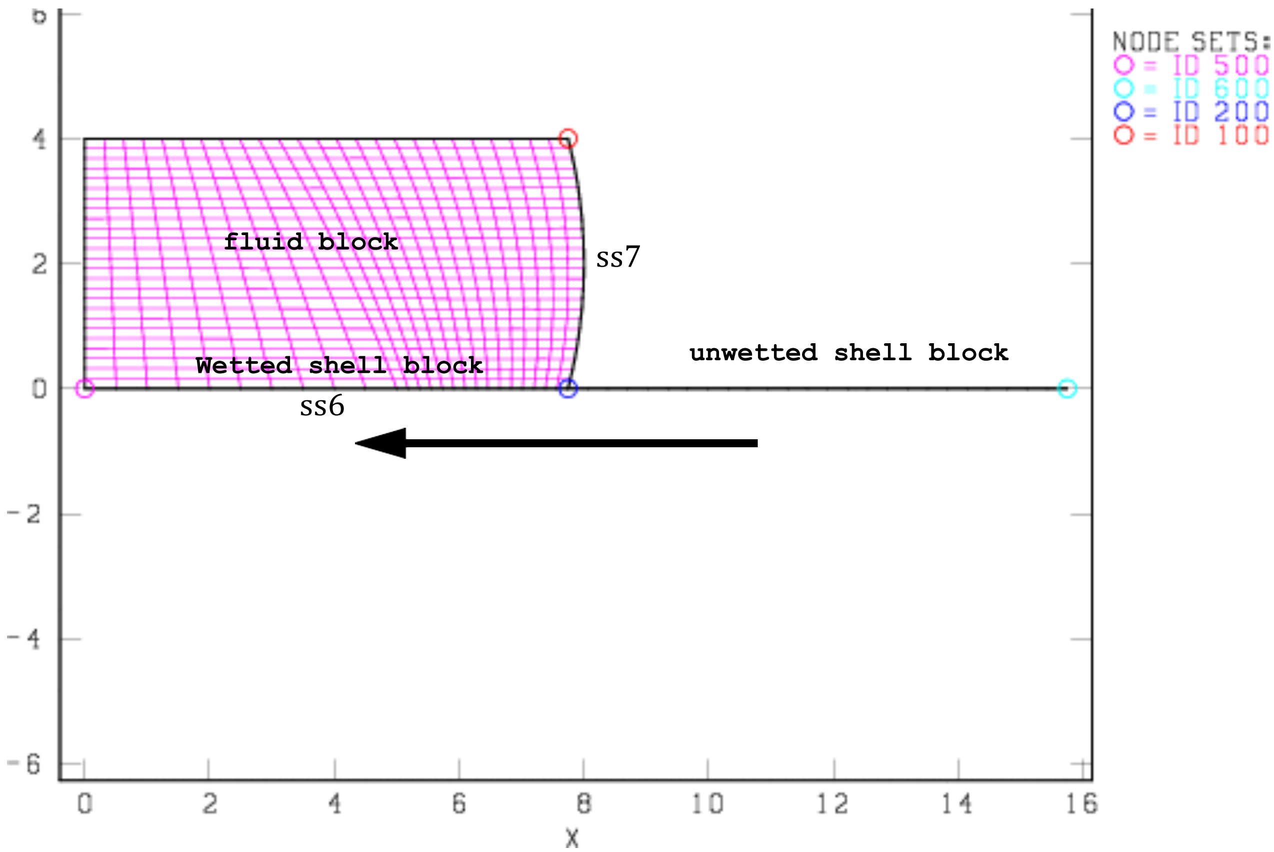
BC = SH_FLUID_STRESS SS 2 1.0 <<<<<<<<<<<<balance of web stress and fluid stress

BC = P NS 3 10000. <<<<<<<<Used to set arbitrary pressure level in box.

BC = U NS 1 0.0
BC = V NS 1 0.0

########
END OF
BC
#########

Problem 2: Upstream slot coating meniscus (rolling meniscus) on free shell web The main applications driving the development of the structural shell capability in Goma are thin oxide-layer containment flow of molten aluminum and tension-web coating flows. Below we show a prototypical geometry of the latter:



The files required to run this problem can be found in **shell_element.tst/slot_meniscus**. Here we discuss the relevant features of each one. We start with the problem setup file, called **free.input**. The first section:

```
{Include(geom)}
         FEM Problem Specifications
---------------------------------------------------------------------
FEM file= twoblock.exoII
Output EXODUS II file= free_out.exoII
```

The problem parameters are specified in the file **geom**, which is included in this file and in the material files. The mesh file is called **twoblock.exoII** and is generated from the file **twoblock.jou** using CUBIT.

In the problem description section of **free.input** we have specifications for three blocks:

```
--------------------------------------------------------------------------------
                              Problem Description
--------------------------------------------------------------------------------
Number of Materials= 3

MAT                    = fluid 1
Coordinate System= CARTESIAN
Element Mapping   =
isoparametric Mesh Motion  =
ARBITRARY Number of bulk
species= 0

Number of EQ = 5
EQ = momentum1  Q1 U1 Q1  0.0  1.0  1.0  1.0  1.0  0.0
EQ = momentum2  Q1 U2 Q1  0.0  1.0  1.0  1.0  1.0  0.0
EQ = continuityQ1   P       Q1 1.0             0.0
EQ = mesh1     Q1   D1      Q1  0.0  0.0  0.0  1.0  0.0  0.0
EQ = mesh2     Q1   D2      Q1  0.0  0.0  0.0  1.0  0.0  0.0
                          div mas adv bnd dif src porous
MAT = shell 2

Coordinate System    = CARTESIAN
Element Mapping      =
isoparametric Mesh Motion  =
ARBITRARY Number of bulk
species = 0

Number of EQ = 4
EQ = shell_curvature Q1  K    Q1       1.
EQ = shell_tension   Q1  TENS  Q1
                    1. EQ = mesh1          Q1
D1 Q1 0 0 1 1 0
EQ = mesh2     Q1 D2 Q1  0 0 1 1 0
                           dif src por
MAT = shell 3

Coordinate System    = CARTESIAN
Element Mapping      =
isoparametric Mesh Motion  =
ARBITRARY Number of bulk
species = 0

Number of EQ = 4
EQ = shell_curvature Q1  K    Q1       1.
EQ = shell_tension   Q1  TENS  Q1
                    1. EQ = mesh1          Q1
D1 Q1 0 0 1 1 0
EQ = mesh2     Q1 D2 Q1  0 0 1 1 0
                           dif src por
```

Note that the mesh motion equations are defined in all three blocks, but the nature of these equations is different on the shell-block than what we are accustomed to from the bulk fluid blocks. Recall in the bulk fluid blocks the fluid momentum equations and the moving mesh equations are couched in an Arbitrary Lagrangian Eulerian formulation, with boundary kinematic equations connecting the mesh motion to the fluid interface motion. The fluid in this case has two free surfaces: the tensioned web surface and the capillary free surface. The mesh equations on the shell blocks 2 and 3 are used as distinguishing conditions on the mesh equations in the bulk fluid block, viz. they force the bulk mesh motion to follow the shell structure motion through Equation 3. On the capillary free surface, the **KINEMATIC** boundary condition serves the same mesh-distinguishing function. The stresses in the shell structure are governed by the shell tension equation and shell curvature equation, also applied on the shell blocks. The kinematic constraint on the shell-fluid surface is enforced with the **VELO_NORMAL** condition, which is mathematically equivalent to the **KINEMATIC** equation but applied to a different differential equation. As mentioned above, notice how we separate the wetted shell block and the unwetted shell block for book-keeping purposes, even though both have the same properties file **shell.mat**.

The boundary conditions that are most relevant to this problem are as follows:

```
Number of BC = -1

## Geometry first
##
BC = PLANE SS 10 0.0 1.0 0.0 -4.0
BC = PLANE  SS 50  1.0 0.0 0.0 0.0


## No slip surfaces
##
BC =  U  NS 10 0.0
BC =  V  NS 10 0.0

## Web
##
 BC = VELO_TANGENT SS 60 0 {webspeed} 0.0 0.0
 BC = VELO_NORMAL SS 60 0.0

## Contact lines
##
## Upstream static contact line
##
BC =  U  NS 100 0.0
BC =  V  NS 100 0.0

## DCL
##
BC =  U  NS 200 0.
BC =  V  NS 200 0.0

## Now for menisci and DCL disting conditions
##
## First upstream
## Do you want the kinematic condition at the upstream meniscus?
BC = VELO_NORMAL SS 70 0.0
$BC = KINEMATIC_PETROV SS 70 0.0
$BC = CAPILLARY SS 70 {surface_tension} {vacuum} 0.0
$BC = CA  NS 200  {ca_200}  0.0  .0  0.0
$BC = CA  NS 100  {ca_100}  0.0  .0  0.0
###This next bc is used to fool goma's fancy conflict resolution
###algorithm to drop the kinematic equation. Note that this adds
###nothing. But you need to comment it out for the fixed
grid case. BC = GD_LINEAR SS 60 R_MESH2 0
MESH_POSITION1 0 -0 0.
```

```
## This is for the outflow boundary
##
BC = VELO_TANGENT SS 50 0 0.0 0.0 0.0 0.0

## Upstream idler
BC = DX NS 600
0.0
BC = DY NS 600 0.0 1.0

## Downstream
idler BC = DX NS
500 0.0
BC = DY NS 500 0.0
#BC = SH_K NS 500 -0.0

BC = SH_TENS NS 600 1000000.0

$BC = PLANE  SS 60  0.0 1.0 0.0 0.0
$BC = DX NS 60 0.
$BC = DY NS 60 0.
BC = SH_FLUID_STRESS SS 60 1.0

# Note: these are only to disable web deformation at the

start. END OF BC
```

We begin with the **VELO_NORMAL** and **VELO_TANGENT** conditions on sideset 60 (the wetted web). Note that these are used to enforce impenetrability and no-slip, with the line velocity set with the **aprepro** parameter {**webspeed**}. The distinguishing condition for the mesh motion on this boundary is automatically applied through the shell-mesh equations, as noted above.

Next we discuss the mesh-related and stress-related boundary conditions on the shell web at nodesets 500 and 600 (the downstream and upstream idlers, respectively), and along the web denoted by SS
60.Note that at NS 600 and 500 we pin the web with zero displacements. After one obtains a solution you can change **DX** or **DY** position at the upstream end (dry web) that effectively changes the position of the idler. We will demonstrate this in the results below. We also impart a tension to the web at the upstream end, which is a boundary condition on Eq. 2 above. This tension is critical. The higher it is, the less likely external forces (hydrodynamic in this case) will deform the web. Note that we do not apply this condition at the other end of the web, as in that case the equations are degenerate (the tension equation is first order, and one does not have the freedom to specify the tension at both ends). In fact, if you try to set the tension at both nset 500 and 600, Goma stops with a underflow error.

Note also that we can impart a curvature at either end (as demonstrated below: note the **SH_K** bc on NS 500 that is commented out). If the tension of the web is high and a curvature is set, the whole web bows. In all of these cases take it easy on the size of the parameter steps, as these tensioned web problems are very sensitive and nonlinear.

Also notice the boundary condition **SH_FLUID_STRESS** on SS 60 (the wetted web). This boundary condition results in the fluid-stress terms of Eq. 1 and 2 to be added in. No such boundary condition is required on the dry portion of the web.

The final set of noteworthy boundary conditions are

BC = VELO_NORMAL SS 70 0.0
$BC = KINEMATIC_PETROV SS 70 0.0
$BC = CAPILLARY SS 70 {surface_tension} {vacuum} 0.0
$BC = CA NS 200 {ca_200} 0.0 .0 0.0
$BC = CA NS 100 {ca_100} 0.0 .0 0.0
###This next bc is used to fool goma's fancy conflict resolution
###algorithm to drop the kinematic equation. Note that this adds
###nothing. But you need to comment it out for the fixed
grid case. BC = GD_LINEAR SS 60 R_MESH2 0
MESH_POSITION1 0 -0 0.

Here, these are set up for a fixed grid solution. When you release the capillary free surface (viz. sideset 70), you will comment out the **`VELO_NORMAL`** condition and uncomment the following four conditions. This procedure is outlined in just about every coating flow template memo that has been published, including GT-001.4 (knife coating), GT-002.1 (slot coating), and GT-006.3 (roll coating). The most important condition here is the **`GD_LINEAR`** card, which actually does nothing but force the kinematic condition to be displaced by the mesh-motion residual along the wetted web, which is the shell position equation (Eq. 3 above). This trick is used to avoid major BC conflict resolution coding and seems to work well.

As for the material files **`fluid.mat`** and **`shell.mat`**, we only need to mention an additional property record that is required in the latter.

```
Lame MU                        = CONSTANT 1.0
Lame LAMBDA                    = CONSTANT     1.0
Shell bending stiffness        = CONSTANT     1.6e+8
```

The Lame coefficient cards are not used in the shell material as the shell bending stiffness is the only physical property that results from the inextensible shell theory. As mentioned above it is defined as $D=Et^3/12(1-\nu^2)$.

In the figure below we show several converged solutions from this model. The base conditions are set in the **`geom`** file as:

```
# tmp.geom
#
# slot_width: {slot_width=0.2}
# y7: {y7=2.03}
# curve_radius: {curve_radius=8.0}
# y1: {y1=0.0}
# x8: {x8=-5.1}
# gd_c0: {gd_c0=270.0}
# gd_c1: {gd_c1=0.0}
# gd_c2: {gd_c2=-27000.0}
# beta: {beta=0.001}
# webspeed: {webspeed=-100.0}
# ca_100: {ca_100=1.5775}
# ca_200: {ca_200=1.9707}
# density: {density=0.93e-04}
# viscosity: {viscosity=1.0}
# surface_tension: {surface_tension=100.0}
# vacuum: {vacuum=-330}
```

The first step to achieve a converged solution is to start with a high tension and high bending stiffness and achieve a solution with the mensicus fixed, viz. with the 4 BCs below the VELO_NORMAL BC commented out. This can be achieved with full newton iteration in about 4 steps. Make sure you start with a zero initial guess. At this point one can post-process the solution and plot the pressure, noting the pressure level at the fixed but slippery capillary surface (ss70) If one applies a back pressure on the capillary boundary condition of about that level, the free surface solution will be close enough to the fixed grid geometry. This pressure in this case is around -300, which we set in **`geom`**.

Then we proceed with continuation, turning on first the initial guess option to **`read`** and uncommenting the capillary free surface BCs and commenting out the **`VELO_NORMAL`** bc. This procedure is discussed in other tutorials (as mentioned above). Obtaining a free surface solution on this step requires ten Newton iterations at a relaxation of 0.1 and then six more at full newton (i.e. relaxation at 1.0). Now you have a base solution from which you can explore the parameter space. Shown in the figure below are two such solutions. In the both cases the surface tension was lowered to 100, and the upstream idler position (viz. nodeset 600 DY bc) was displaced upward by 1.0 units, viz. the BC on 600 is

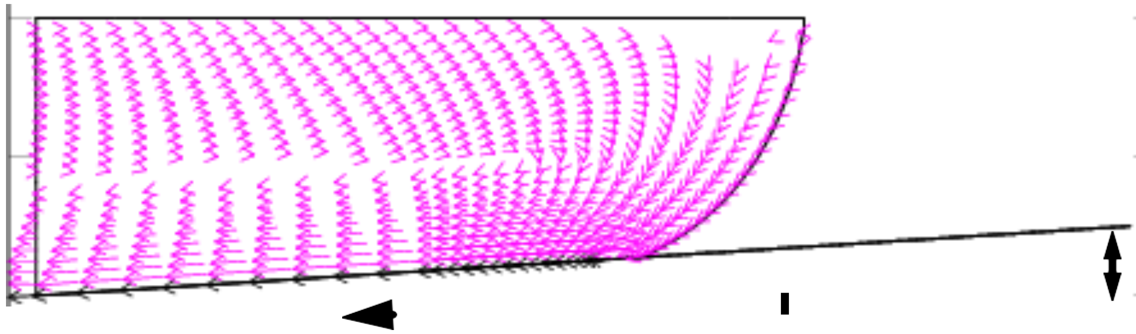
```
## Upstream idler
BC = DX NS 600
0.0
BC = DY NS 600 1.0 1.0
```

Please note that each of these changes was made gradually, using relaxation and continuation strategies that analysts doing this class of problems should be accustomed to. The other solution shown in the figure corresponds to an additional change, i.e., the boundary condition

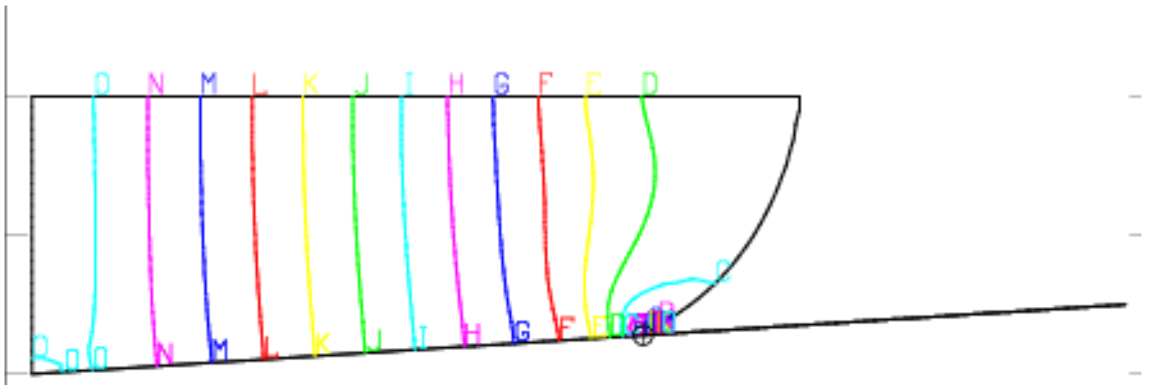
```
BC = SH_K NS 500 -0.01
```

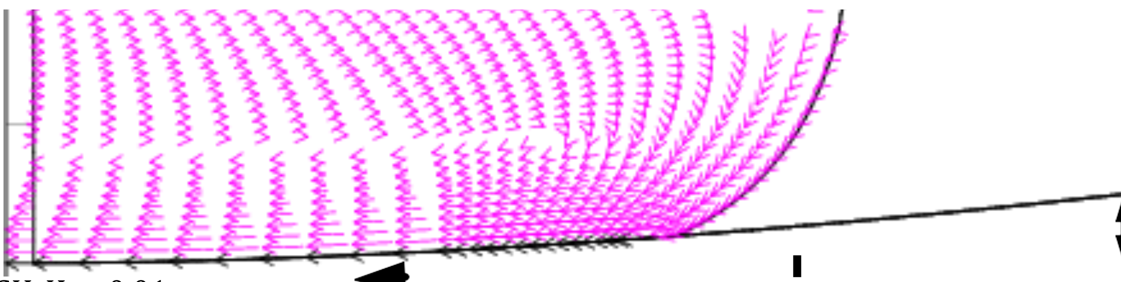
was added. Here we specify the curvature of the web at the downstream pinned point to be -0.01. Note how this leads to a curved web that spans between the endpoints, due to the high bending stiffness and tension.

- Sample results.

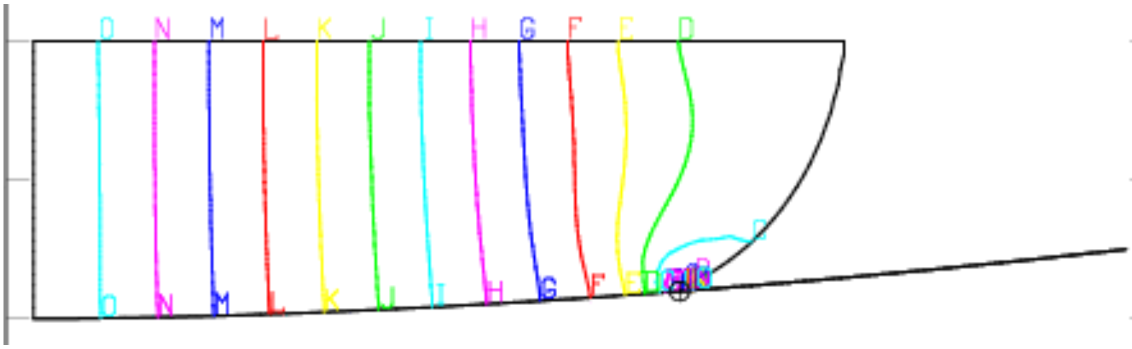


Pressure Field





SH_K = -0.01



Algorithm details in Goma

Base Algorithm:
Initialize Unwetted shells as separate
block Initialize wetted shells as separate
block Bulk mesh blocks as usual

In problem description section define mesh1 and mesh2 equations on all blocks

Shell-Y equation (4) applied on all shell blocks to the R_MESH2 equation (or experimenting with R_MESH_NORMAL equation for wetted shells) iff R_MESH is defined and iff Equations 1 and 2 are defined. viz. this equation (4) is never applied from the bulk.

Shell-X equation (3) is subbed out for the arc-length equation and applied to R_MESH1 for unwetted shells. For wetted shells the pseudo-solid mesh equations are used. Here we need to look-across from the wetted Shells to the companion abutting bulk elements to see if they are in fact wetted, so that the arch length equation can be left out.

Shell hydrodynamic stresses to Eqns (1) and (2) are applied as boundary conditions with the BC = SH_FLUID_STRESS SS 60 1.0 cas