Sandia National Laboratories

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

Albuquerque, New Mexico 87185- 0835

date: August 3, 1999

to: Distribution

from: R. R. Lober, 9113, MS0835 and P. R. Schunk, 9111, MS0826

subject: Slot and Roll coating with remeshing templates and tutorial for GOMA and CUBIT/ MAPVAR (GT-006.4)

keywords: CUBIT, TALE, CUBIT with FASTQ, remeshing, remapping, MAPVAR, slot coating, roll coating, TALE verification

input records: Initial Guess, FEM file, Mesh Motion=TALE, Mesh Motion=ARBITRARY, Mesh Motion=LAGRANGIAN

Introduction

This tutorial is similar to the original slot coating and roll coating tutorials (GT-02.1 and GT-003.1), but has been extended to include the use of the **`cubit`** mesh generator in place of **`fastq`**. The **`cubit`** mesher will produce the mesh with the same input files as **`fastq`**, and will also perform deformed geometry reconstruction and remeshing from the subsequent output files. An introduction to **`cubit`**, sufficient ONLY to produce two- dimensional **`exodusII`** mesh files with this **`fastq`**interface will be presented here. A complete user manual (November 1998 PDF) on **`cubit`** can be found at

http://endo.sandia.gov/SEACAS/Documentation/Cubit.pdf

in addition to other **`cubit`** related information. *It should be noted that our trial run here will not address 3D meshing issues with* ***`cubit`****, and for the time being we will not support such usage on your behalf. The user may of course after reading the manual decide to pursue 3D geometries on his/her own.*

This tutorial assumes the user has gone through the beginner's training tutorial on GOMA and SEAMS. Some of the more advanced examples require thorough understanding of fluid-structural interactions, as deformation gradient tensors and material reference states will be discussed. Specifically, it assumes the user has a strong familiarity with GOMA itself and the tools **`fastq, ex1ex2v2, aprepro`** and **`blot`**. If not, please go through that tutorial if you have it, or contact Duane Labreche (dalabre@sandia.gov) or Randy Schunk (prschun@sandia.gov) to get it.

This tutorial will demonstrate remeshing/remapping with 3 example problems: (1) a single layer slot coater, (2) a flooded nip forward deformable roll coating flow using a computational Lagrangian mesh motion in the solid, and (3) a forward deformable

roll coating flow using the TALE mesh motion in the solid. The first example is a single phase problem, with arbitrary mesh motion, and requires no special considerations to preserve the solution fields across the remap. The second and third examples represent a significant recent extension to the remapping capability, as it allows for remapping with preserved stresses in deformable solid materials, where the reference state is extremely important. The key to these problems, as in all problems with free surfaces, is to obtain a good initial guess and to understand the details of the problem parameterization. This tutorial will try to cover all phases.

Cubit Usage Issues

The following instructions will help to provide a recipe-like guide to using **cubit** as a **fastq** replacement and as a remeshing tool for deformed grids. In general, **cubit** has many more modeling and meshing features than **fastq**, and it is more complex because of this. However, **cubit** can be easily used as a **fastq** replacement, and allows access to significantly enhanced and more robust version of the paving algorithm than **fastq** does.

First we will give some basic instructions and caveats before you begin:

- When you use **APREPRO**, like so often in the CRMPC templates, make sure that no line begins with a "{", i.e., you must have a comment like delimiter on **APREPRO** lines. Here is an example:

```
{Gap_new = 0.01} {Gap_old = 0.01}
```

  must appear in the **cubit** (or **fastq**) input deck as

```
$$ {Gap_new = 0.01} {Gap_old = 0.01}
```

  with any sort of comment line delimiter at the beginning.
- As will be mentioned again below, it is a good idea to keep your chosen length unit so that coordinate values significantly greater than 1.e-6. **Cubit** is solid-model based, and the underlying solid modeler has difficulty with numbers less than 1.e-6. So scale appropriately.
- Some basic commands that are helpful, but are better and more completely explained in the manual:

  Cubit> display *(redisplays the current geometry/mesh)*

  Cubit> mesh off *(turns off mesh so that you can see the surface and curve numbers)*

  A *surface* in **cubit** is equivalent to a *region* in **fastq**.

Cubit as a Fastq Replacement

Cubit can read in your **fastq** deck and create a two-dimensional solid model via the following two means: first as a command line option

```
cubit -fastq block.fsq
```

and second, using the import command from within an interactive **cubit** session

```
CUBIT> import fastq "block.fsq"
```

Once **cubit** has read the **fastq** deck, most of the effort required to generate a mesh in **cubit** is eliminated. Since the **fastq** deck specifies the mesh density with interval settings and specifies the surface meshing scheme on the REGION card, all you need to do is mesh the surfaces, and export an **exodusII** file (**cubit** only creates **exodusII** files, so you do not need to run mesh files created via **cubit** through the **ex1ex2v2** translator). The following commands illustrate these actions from within a **cubit** interactive session:

```
% cubit
CUBIT> import fastq "block.fsq"
CUBIT> mesh surface all
CUBIT> export genesis "block.gen"
CUBIT> exit
```

There are but a few things to avoid when using **cubit** as a **fastq** replacement, and none of them should cause any fatal errors. **Cubit** does not recognize the BODY card from **fastq**, and you can omit it from your **fastq** deck. If you have a BODY card present, **cubit** will issue an error, but will continue to build the model and mesh as if it were not there. Also, try not to use the symbol "\" even in a comment within a **fastq** file, as the cubit parser thinks it indicates a continuation line, and will also flag an error, but continue to build the model. Next, you will notice that Line entities in fastq of the type **CIRC**, which puts a circular arc between two points about a center point, is interpreted in the opposite sense of direction in **Cubit**. That is, if you have a circular arc going from point 1 to point 2 about a center point 3, then in **fastq** if the arc is scribed between 1 and 2 in a clockwise fashion, it will occur in a counter clockwise fashion in **Cubit**. You need then to reverse the order of the end points on lines of type "**CIRC**". Finally, the **RENUM** command has not been implemented in **cubit**, so the **RENUM** card will have no effect when it is included in a **fastq** deck being read by **cubit**.

There are many other options for displaying and creating mesh, geometry, and boundary conditions within cubit, which are documented in the Users Guide mentioned at the beginning of this memo. Cubit also contain online syntactical help, accessed via "help", "?", or the beginning of command followed by "?", as in "import?".

Cubit as a Remeshing Tool

Cubit also can provide deformed geometry creation and meshing, which will be demonstrated in the following typical commands. Note the presence of the two

associativity commands which allow for deformed geometry creation in subsequent **cubit** sessions.

```
% cubit
CUBIT> nodeset associativity on
CUBIT> set associativity complete on
CUBIT> import fastq "block.fsq"
CUBIT> mesh surface all
CUBIT> export genesis "block.gen"
CUBIT> exit
```

Now **goma** can run with **block.gen**, and produce a deformed mesh **exodusII** file which we will refer to as "**out.exoII**". **Cubit** will then read in the deformed file and apply the deformations from the time plane specified by the user (via step number, actual time value, or the keyword "last").

```
% cubit
CUBIT> nodeset associativity on
CUBIT> set associativity complete on
CUBIT> import free mesh "out.exoII" last
CUBIT> display
```

At this point, the new geometry is present, but unlike in the **fastq** deck, no settings currently exist to indicate to **cubit** what mesh density or schemes should be used for the new mesh. These must be set by the user and are documented in the manual. Some typical examples are demonstrated below.

```
CUBIT> label surface on
CUBIT> display
CUBIT> surface 1 scheme pave
CUBIT> surface 1 size 0.008
CUBIT> surface 2 scheme submap
CUBIT> surface 2 interval 14
CUBIT> mesh surface 2, 1
CUBIT> export genesis "block_rem.gen"
CUBIT> exit
```

In the above example, the mesh schemes for the surfaces created were set interactively, as well as the mesh density. To determine what the surface and curve numerical ids are, the labels can be set to on as indicated in the example (for more options, execute "help label" inside **cubit**). For more syntactical help on what mesh scheme and density setting options exist, execute the help command in **cubit** on "scheme", "size", and "interval". For convenient interactive graphics control inside a **cubit** session, execute the "mouse" command and then enter "h" while your mouse pointer is over the **cubit** graphics window. This will cause a number of mouse-based graphical options to print out to the command window.

It is important to note that "surfaces" as referred by cubit are the same as "regions" in the fastq deck, or regions over which a certain meshing scheme is applied. The associated ID numbers in cubit are NOT necessarily the same, so you must display them as described above before prescribing the mesh scheme. The recommended approach for your own problem is to start with the slot coating example files provided,

and explained below. Follow this procedure (but only after you have been through the example below):

1. Copy the **slot_initial.jou** file over to your new directory and rename it, using the *.jou suffix. E.G. **roll_initial.jou**.
2. Edit that file and change the name of the imported **fastq** file to your already made, ready for meshing **fastq** file, e.g. **roll.fas**. Also change the name of the output genesis file (like from **slot_initial.g** to **roll_initial.g**). These are just **exodusII** initial files.
3. Run cubit as described above, and below in the example. This is now your base starting mesh that contains exodusII entities that permit remeshing.
4. Run Goma with this initial mesh and continue through parameter space until you get sufficient distortion to warrant remeshing.
5. Clone the **slot_remesh.jou** file to your new directory. Comment out all of the surface, curve and map commands below, as they will not work on your mesh, but retain everything else, making appropriate changes to the imported **out.exoII** name, if needed, and the exported remeshed file, if required. Also comment out the "**exit**" command in the ***.jou** file so that when you "play back" the journal file with **cubit**, you can see what the fitted geometry looks like and can read the surface numbers from it without exiting cubit.
6. Now choose your meshing schemes for each surface. You may be able to pave most of them, and the example should make that clear how you do this. Some regions (or "surfaces") you will want to map, like long thin regions, and you can control the mesh size with the commands provided, or you can dig into the actually curve commands that allow you to grade the mesh, etc.
7. Add these lines to the **slot_remesh.jou** file and run cubit with that file. Now you are ready to remap the solution from the old mesh to the new mesh. We will use a tool called **MAPVAR** to do this.

MAPVAR Remeshing Tool: The Basics

MAPVAR (for MAP VARiables) is actually a part of your **SEAMS** distribution, but you must have an up-to-date version of **SEAMS** (at least April 1999 or later). **MAPVAR** maps a solution from one grid to another, and manipulate the displacement fields to accommodate arbitrary grids, Lagrangian grids, or combinations thereof, the last two cases requiring preservation of stress (or reconstruction of the reference state) across the remap. The grids do not necessarily have to overlap perfectly, but you hope in your remesh/remap case that they do. **MAPVAR** is simple to run, and can be run in separate input modes (command line, input file, or with prefix/postfix conventions). Typing "**mapvar**" at the prompt with the "**-help**" switch will give you a usage summary. By default all nodal quantities are mapped to the new grid, but quantities with a lower-order interpolation are ignored (e.g. a pressure interpolated with a lower order than the

mesh, viz. biquadratic velocities but linear pressures from a mixed interpolation scheme). We find that you don't need to map all the variables to converge to the new solution. Remember, all your are doing is generating an initial guess to the new solution, or a restart file for a transient solution.

Comprehensive up-to-date documentation for MAPVAR can be obtained through Sandia's technical reports office and the citation is:

> *MAPVAR--A computer program to transfer solution data between finite element meshes. G. W. Wellman. Sandia Report SAND99-0466. Unlimited release 1999.*

The easiest way to run MAPVAR is to choose a common prefix name, and simply create the necessary files with the appropriate postfix. For instance, if I want to map the solution currently in **`out.exoII`** onto a new mesh generated by **`Cubit`** called **`new_mesh.gen`**, I follow this procedure:

```
cp out.exoII tmp.e
cp newmesh.gen tmp.g
mapvar tmp
CMD> def 2
CMD> run
```

The first two commands use the common file prefix "**`tmp`**". Third command basically invokes **`mapvar`** with the appropriate donor and recipient meshes. The commands at the **`mapvar`** prompt are the interactive part of **`mapvar`**. Many options exist and they can be viewed by typing "**`help`**". The result from this run is a file called **`tmp.int`**, which is the new mesh, as in **`newmesh.gen`**, with the solution mapped onto it.

There are several real important options that warrant attention here.

- The "deformed" option which is abbreviated "**`def`**" in **`mapvar`**, enables the user to select what is to be done to displacements in a deformed geometry remap. There are current 3 options available.

  def 0: all mapping in mapvar is done with original, nondisplaced coordinates. Donor mesh displacements are not considered. The recipient mesh boundaries should coincide with the boundaries of the original finite element mesh that was used to produce the donor mesh.

  def 1: This is the default option. In this option, MAPVAR processing is performed in current, deformed coordinates. The boundaries of the recipient mesh are consistent with the deformed shape of the donor mesh, by virtue of CUBIT deformed geometry remeshing. During processing the displacements from the donor mesh are read and ADDED onto the coordinates of the donor mesh to produce current coordinates. Just prior to writing the interpolated mesh, the transferred displacements are subtracted from the recipient mesh coordinates. This produces an interpolated mesh that, at the time step of interest, has current coordinates identical to the recipient mesh. That is, when plotted with a displacement magnification of one, the interpolated mesh is identical to the recipient mesh. This

technique allows the displacement field to remain meaningful across a rezoned restart, and is essential for Lagrangian mesh motion regions.

def 2: This option is called mesh "annealing". It was supplied in support of GOMA. In mesh annealing, the data transfer is performed in the deformed, current coordinates. Thus, during processing, the donor mesh displacements are added to the donor mesh coordinates to produce the required current coordinates. However, prior to writing the interpolated solution, the displacements are zeroed. This produces an interpolated mesh, that while consistent with the deformed shape of the donor mesh, has a zero displacement field.

The next **MAPVAR** option of relevance is the "**searchbox**" option. The "search" command allows the tolerances on extrapolation and interpolation to be loosened. Its default value is 0.01, but sometimes for high curvature regions, it has to be raised to 0.1 or higher, not exceeding 1.0. We have found this to be useful in getting smoother real-solid displacement fields along fluid-structure boundaries after a remap. REMEMBER to look carefully at those remapped fields after running **mapvar** with **blot** or something the equivalent of **blot**, and make sure they are smooth.

The next **MAPVAR** option we discuss here is important for fluid-solid conjugate problems. It is the "**map**" option and is demonstrated examples below. It is used to specify that individual blocks of material are to be mapped only to the corresponding block on the recipient mesh. This is essential for conjugate problems where you don't want the value of a variable that is not defined on a neighboring block region to be interpolated at the interface. It is also important to map nodal variables associated with the solid regions before liquid regions, when using TALE.

Single Layer Slot Coater

The key files in this directory, called "**1lyr.cubit**" under the "**slot**" subdirectory in the **example_problems** section of your master distribution, are **slot_input.initial** and **geometry.fsq**. The **geometry.fsq** file describes the parameterization of the problem. The top of the file looks like:

```
$                inflow
$ -------------5        6---------------
$ \          |       |            /
$  \         |       |           /
$   \        |       |          /
$    \       |       |         /
$     \      |       3----------4       14
$      1 --------2
$
$
$  7-----------8--web-----9----------10 ------ 8
$
$Geometrical paramterization:
$
$Define: x is parallel with the web, y is perpendicular to it.
$Gap [=] y1-y7, PERIOD! All rotation is done about point 1
```

```
$Lip 1: Specify y position of 2, relative to 1
$Lip 3: Specify y position of 3 and 4, relative to 1
$Slot [=] x6 - x5 == x3 - x2
$
$
$Also, x and y are just generic orthogonal directions here. Actually
$the model can be rotated with an angle alpha with the web, about point
$1, and so the distances specified above remain the same, because
$they will be relative to an imaginary line that goes through point 1
$and theta=0 means parallel to the web.
$
$
$Operating Conditions
$$
$ Volumetric inflow rate/unit width m2/s {volumetric_inflow = .0002}
$ Vacuum backpressure    [=] Pa          {vacuum = -3300.}
$ Substrate (web) speed  [=] m/s         {webspeed = 1.}
$ Viscosity (Newtonian)  [=] Pa-s        {viscosity = 0.1}
$ Density   (Newtonian)  [=] m^3/s       {density = 1000.}
$ Surface Tension        [=] N/m         {surface_tension = 0.04}
$
```

The first header comment attempts to explain some of the key points and other items, but the postscript file in the directory entitled **`slot.fm.ps`** has a more complete description. The operating conditions are fairly standard, as can be seen. When making operating condition changes, use this file. All units are again in MKS. *We should caution the user against choosing MKS units due to the* **`ACIS`** *solid model software, which does not deal with size tolerances less than 1.e-6, which in MKS is the micron level. So if your mesh model has features of that order, then you should convert to CGS or a larger length unit.* After the operating condition section you will notice the Geometry description section. This section is the most important to understand.

```
$$
$Set Geometrical paramters (all parameters in MKS)
$$
$$ To make geometrical changes after having obtained a solution (btw
$$ see accompanying README file for details on how to obtain a solution)
$$ Please change the *_new parameter and leave the original geometry
$$ parameter the same because GOMA needs to know the difference and magnitude
$$ of the change. If you want to start with a different mesh, change both
$$ parameters and run cubit.
$$
$$
$  Original mesh         You have changed
$  was built with:      it to:
$----------------------------------------
$  {Gap = 0.0002 }      {Gap_new = 0.00020}
$ {h1 = 0.00006} {h1_new = 0.00006}
$  {h2  = 0.00004}      {h2_new  = 0.00004}
$  {h3  = 0.00001}      {h3_new  = 0.00001}
$  {alpha = 0.}     {alpha_new = 0.1}
$
$  {S2  = 0.0002}  {S2_new = 0.0002}
$  {S1  = 0.0005}  {S1_new = 0.0005}
$  {L2  = 0.0004}  {L2_new = 0.0004}
$  {L1  = 0.0004}  {L1_new = 0.0004}
$
```

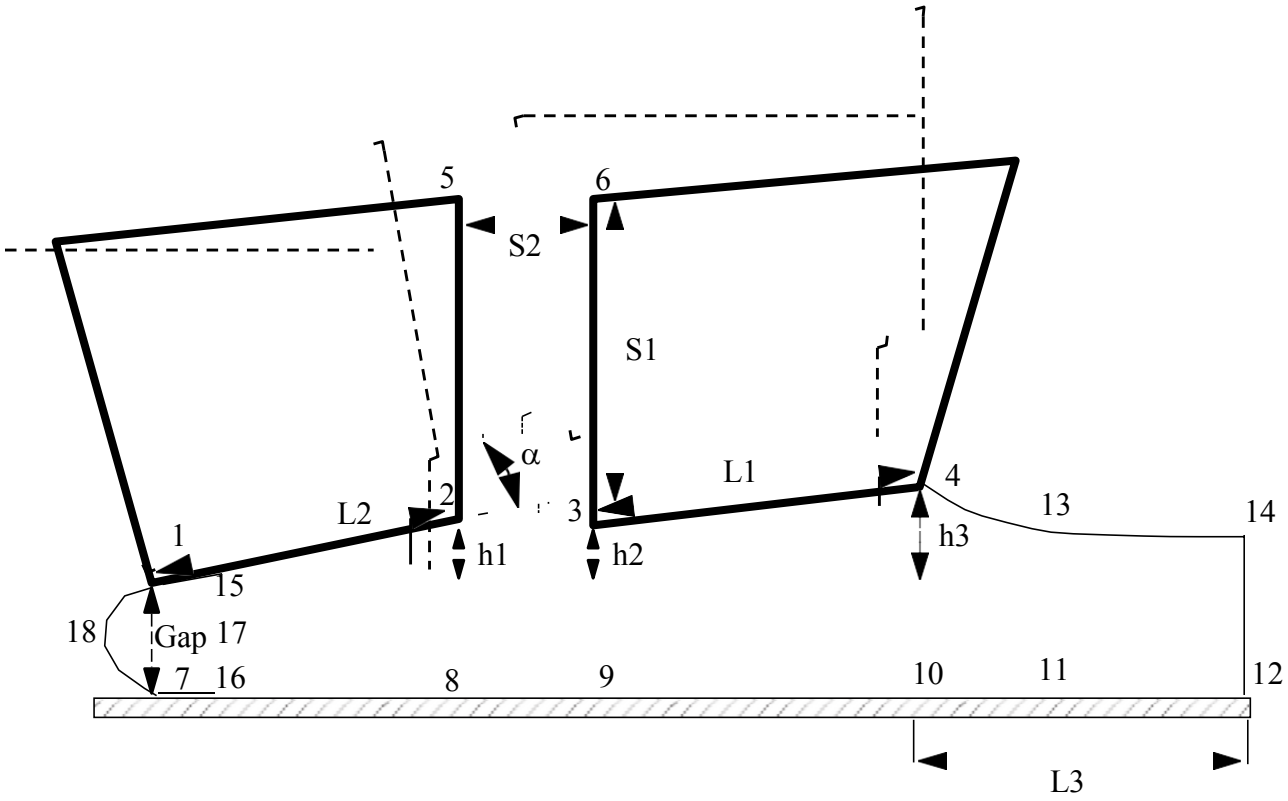
```
$ {L3 = 0.0012} {L3_new = 0.0012}
$
$ {slip_length = 0.00003}
$
$
$$$
$Do a simple calculation
$$$
$ {inflow_speed = volumetric_inflow/S2_new}
$
$
$$Set mesh density
${no_elem_L2 = 12}
${no_elem_S1 = 6}
${no_elem_S2 = 6}
${no_elem_L1 = 10}
${no_elem_L3 = 15}
${no_elem_film = 4}
${no_elem_along_slip_length = 3}
```

The key geometrical quantities, like **`Gap, h1, h2, h3, alpha, S2, S1, L1, L3`**, are specified for the original mesh and also the updated or new conditions, during operation. As explained in the roll coating memo, GOMA is based on *displacements*. The **`*_new`** quantities are the geometrical changes that can be made in the file during a case study, i.e, after an initial solution has been obtained. It is important to understand here that GOMA solves for displacements of the nodes and not the actual coordinates as they evolve, and so coordinate DIFFERENCES must be tracked instead of absolute values. Hence, the boundary condition changes for Gap and die/substrate angle **`alpha`** alterations must have both the initial value and the desired value, the initial value being that at which the mesh was originally generated. Anyway, the geometrical parameterization looks like the figure to follow below.

You will notice that the die itself is parameterized such that it can be moved in and out towards the web and tilted about its heal for different angles of attack. If you scan down the geometry file you will see that all **`fastq`** geometry points have original and changed states defined. Then, with that as the case, then the coefficients a,b,c, and d for the PLANE command conditions **`ax+by+cz+d = 0`** can be computed with Kramer's rule at the bottom of the file. In other words, when you make a geometrical change, the coefficients which parameterize the positions of the faces of the die are automatically computed. For fun, you should try changing some of the "original" geometrical parameters and the changed parameters, keeping both the same, to different values and regenerating a mesh to get used to the parameterization. To begin this example, generate the initial grid by executing the following:

```
cubit -nojournal -input slot_initial.jou
```

which will produce an **`exodusII`** mesh file. The **`cubit`** journal file **`slot_initial.jou`** imports the fastq geometry file **`geometry.fsq`** and generates a mesh producing the **`slot.exoII`** output file. The "-nojournal" option above prevents **`cubit`** from saving a journal file of the session to the current directory, which is the default behavior. When you start to use **`cubit`** for more complicated models, you will want to have the journal



Parameters:

Gap: Distance between substrate and leading edge of upstream land

h1: Vertical distance (could be negative) between trailing edge of upstream land, and a Gap height above substrate, at zero angle α of attack.

h2: Vertical distance (could be negative) between leading edge of downstream land, and a Gap height above substrate, at zero angle α of attack.

h3: Vertical distance (could be negative) between trailing edge of downstream land, and a Gap height above substrate, at zero angle α of attack.

α: Angle of attack of die and substrate, with zero angle defining h1, h2, and h3

L1: Upstream land length

L2: Downstream land length

S1: slot length

S2: slot width

files for convenient session records and rerunning previous meshing sessions, to allow the journal files to be saved, just omit the "`-nojournal`" argument when executing `cubit`.

Slot coating initial guesses are considerably easier to obtain than roll coating initial guesses. The procedure here is quite simple. (1) Get a fixed grid solution using

VELO_NORMAL conditions on the upstream and downstream menisci, (2) release the downstream meniscus by substituting **KINEMATIC, CAPILLARY**, and **SURFTANG** cards where appropriate, (3) postprocessing the results and picking off the pressure at the upstream meniscus, (4) setting the vacuum pressure to equal that pressure, and releasing the upstream meniscus. Both "release" stages do require some relaxation, as is diagrammed below. When picking the pressure off the fixed grid solution, or the single downstream meniscus solution is done simply by running "**blot**" on the **out.exoII** file, zooming in on the upstream meniscus, and contouring the pressure. Increase the number of contours with the **spectrum** command, or adjust to range with the **crange** command to get an accurate reading.

The following steps will effectively enact this procedure following the generation of the initial mesh via **cubit** as above. First obtain the fixed grid solution.

```
goma -a -i slot_input.initial
```

Now copy the solution data from this run into the continuation data file for the next

```
cp soln.dat contin.dat
```

and then execute the first free surface run with relaxation and five Newton iterations, releasing the downstream meniscus and achieving plug flow downstream

```
goma -a -i slot_input.1free -r 0.1 -n 5
```

Copy the solution data once more

```
cp soln.dat contin.dat
```

and repeat the last run, now with no relaxation or Newton limitations

```
goma -a -i slot_input.1free
```

Copy the solution data again

```
cp soln.dat contin.dat
```

and now run the second free surface run with relaxation and five Newton iterations, releasing the upstream meniscus

```
goma -a -i slot_input.2free -r 0.1 -n 5
```

Copy the solution data again

```
cp soln.dat contin.dat
```

and again repeat the last run, now with no relaxation or Newton limitations

```
goma -a -i slot_input.2free
```

Now we will increase the backpressure vacuum (modifying **geometry.fsq**) by changing the value from -4750. to -5000. and again copying the solution data over from the last run

Inside **geometry.fsq**

```
$Operating Conditions
$$ (Initial conditions here for the mesh settings below should be
$$ inflow_speed = 12.0 ; webspeed 42.3; vacuum -4750 )
$$
$ Average Velocity at inflow cm/s          {inflow_speed = 12.0}
$ Vacuum backpressure    [=] cgs           {vacuum = -5000.}
$ Substrate (web) speed  [=] cm/s          {webspeed = 42.3}
$ Viscosity (Newtonian)  [=] cgs           {viscosity = 0.29}
$ Density   (Newtonian)  [=] cm^3/s        {density = 1.2}
$ Surface Tension        [=] dyn/cm        {surface_tension = 61}
```

then copy solution data over

```
cp soln.dat contin.dat
```

and again repeat the last run

```
goma -a -i slot_input.2free
```

Now increase the backpressure vacuum again (modifying **`geometry.fsq`**) by changing the value from -5000. to -5500. and again copying the solution data over from the last run

```
cp soln.dat contin.dat
goma -a -i slot_input.2free
```

Now we can regrid the deformed mesh to mitigate the element quality problems.

```
cubit -nojournal -input slot_remesh.jou
```

This journal file will use a regular grid for two of the meshed regions, and an irregular grid for two other which have mesh density transitions in them (they will be paved). These selections can be changed by modifying the journal file **`slot_remesh.jou`** or running **`cubit`** interactively. At this point the new mesh file, **`slot_rem.g`**, has been created from **`cubit`** and we need to remap the solution data from the last **`goma`** run onto the new mesh.

```
cp out.exoII tmp.e
cp slot_rem.g tmp.g
mapvar
CMD> def 2
CMD> run
cp tmp.int restart.exoII
```

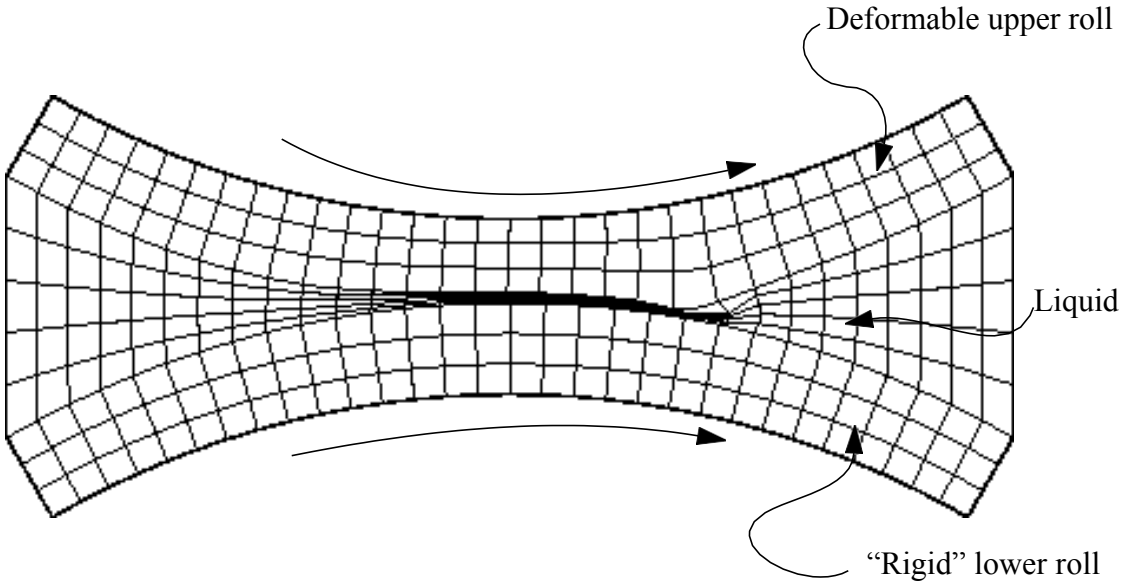
Now we can continue the goma simulation using the new grid

```
goma -a -i slot_input.rem_2free
```

It is useful to do some exercises with the model to see how far you can push the parameters with the current setting and the current mesh. WARNING: Don't take too big of step in geometrical parameters unless using significant relaxation. For instance, changing the angle-of-attack from **`alpha=0`** to **`alpha=0.1`**, or **`0.1`** radians, and even with that you have to do 10 iterations at **`-r = 0.1`**, plus another 5 iterations at **`-r=0.5`**. Then I had to raise the backpressure to keep the upstream meniscus from being sucked in the nip. Almost all parameters have been tested, but please provide feedback if you have it.

Fluid/Lagrangian Solid Forward Roll Problem

Please consult the updated TALE tutorial memo (GT-005.3) to get the details of this test problem. In this section we only discuss the remeshing aspects of it. Basically here we are immersing the rolls in liquid and predicting the interaction of the liquid and rollers, including the roller deformation. This is the simplest of all roll coating models. You should have a directory in "**remesh**" which is called "**def_roll_nip**". We have updated this a bit from the original distribution so contact Duane Labreche (dalabre@sandia.gov) or Randy Schunk (prschun@sandia.gov) for an updated copy, if yours is older than July of 1999. Unlike the roll-coating tutorial memo (GT-003.1), we will use CUBIT to generate the initial mesh much like we did in the slot coating example above. You will notice in this tutorial directory you will find a CUBIT journal file "**initial_mesh.jou**" which reads in the **fastq** geometry in **roll.fas**. The following figure shows the basic geometry:



As a part of this tutorial, go ahead and generate an initial mesh with **CUBIT**, viz.

```
cubit initial_mesh.jou
```

It is important to note that the **exodusII** file that is exported from this **CUBIT** step is named "**roll.exoII**". This is your initial mesh. Go ahead and run the roll-coating problem with the initial input file:

```
goma -a -i roll_input
```

To make matters interesting, push the lower roll up by 0.01, i.e., change **{Gap_new = 0.01}** to {Gap_new = 0.0} and re-run (making sure that the **Initial Guess** card is set to "**read**"):

```
cp soln.dat contin.dat
goma -a -i roll_input.
```

Having a look at the mesh with **blot**, you will notice some distortion in the grid in the coating nip. To try to alleviate this we will remesh and remap, with the intent on preserving the entire solution. Before we do that, it is pertinent to discuss the mechanics that must be preserved.

During a remesh (or rezone) step, the initial stress free state which is encoded into the mesh point coordinates in the mesh file, is obviously altered. That is to say, the nodal coordinates which serve as the material coordinates, denoted here by X_i at each node i, are changed to a new set of coordinates X_i^{new}, thereby marking a new set of material "parcels". Clearly something must be done to recover the stress-free-state in the Lagrangian solid regions for these new deformed material points. Luckily, the change in the stress-free state is encoded in the map of the material displacement field d_i, for which we can recover the local functional form on each element through **mapvar**. So the new stress free state is simply:

$$X_i = X_i^{new} - d_i$$

for each node i. The d_i are of course the mapped values of the displacement field. The figure below illustrates some of these concepts for both Lagrangian solid regions and TALE solid regions.

Basically, the mapped material displacement field is subtracted off of the new mesh point coordinates. This is the default mode in which **MAPVAR** works. Upon a restart the stresses in the Lagrangian portions of your domain are preserved, to within the accuracy of the mapping. To make things convenient, these steps are made options in **mapvar**, through the **Deform** command, as shown below. It is highly recommended that you review the **mapvar** manual for all the mapping and deformation options.

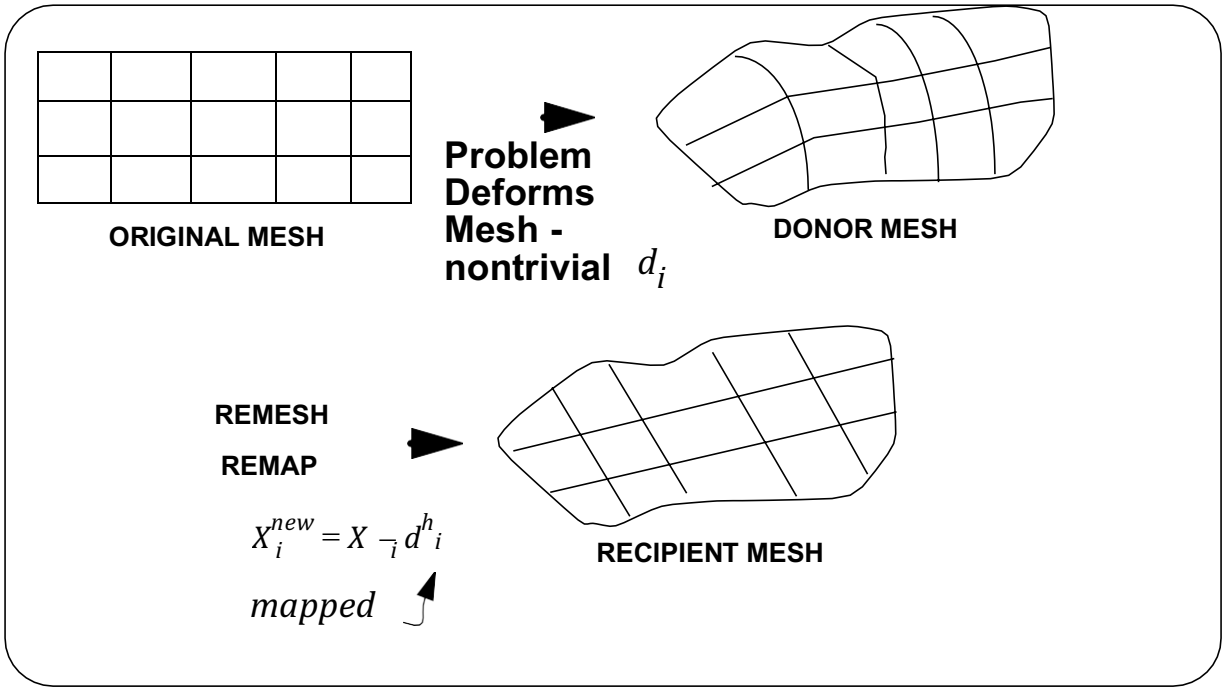
Now remesh your deformed domain,

```
cubit remesh.jou
```

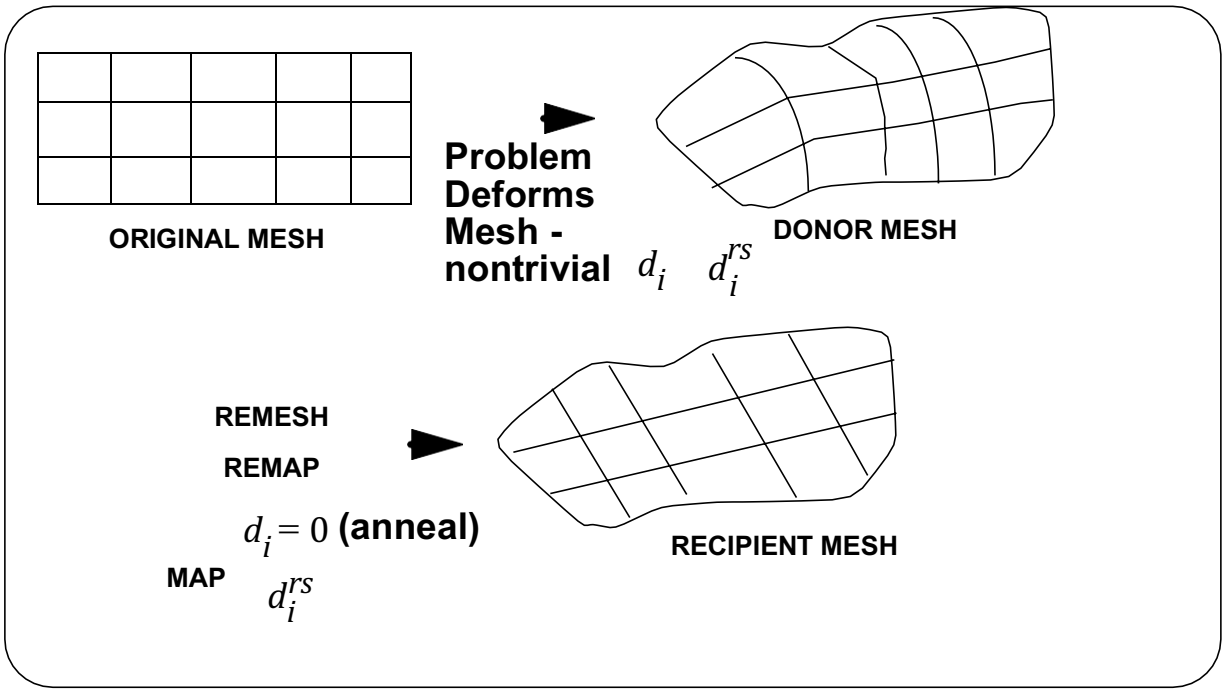
Note that the exported exodusII file is **rem.gen**. As in the slot coating example, do the following steps:

```
cp rem.gen tmp.g
cp out.exoII tmp.e
mapvar tmp
```

but add on the additional interactive steps in mapvar



Lagrangian solid remeshing/remapping



TALE solid remeshing/remapping

```
mapvar> time 4
mapvar> deform 1 (which is default)
```

One thing you will notice is that we did not need to specify the mapping of regions in **mapvar**, as must be done when TALE mesh motion is employed (discussed below). The results of the remap are in the **tmp.int**. Before your restart **GOMA**, it is always a good idea to inspect the new mesh for flaws and poorly mapped quantities. With **blot**, or any other compatible post processing program, display the results and zoom in on critical regions, or regions which have been highly deformed and possibly with large aspect ratios. On this problem it is imperative that you inspect the mesh and mapped quantities (velocity and displacement field, mainly) in the nip where the rolls come closest to contact. Contour displacements and velocity components, to make sure the are smooth. Occasionally, the tolerance option of **mapvar** (see manual) has to be adjusted to get a smooth field. The final step in this process is to recover your solution, that you had before your remesh and remap, on the new mesh. First, make the following changes to your **GOMA** input deck:

```
FEM file = roll.exoII
```

to

```
FEM file = tmp.int
```

and

```
Initial Guess = read
```

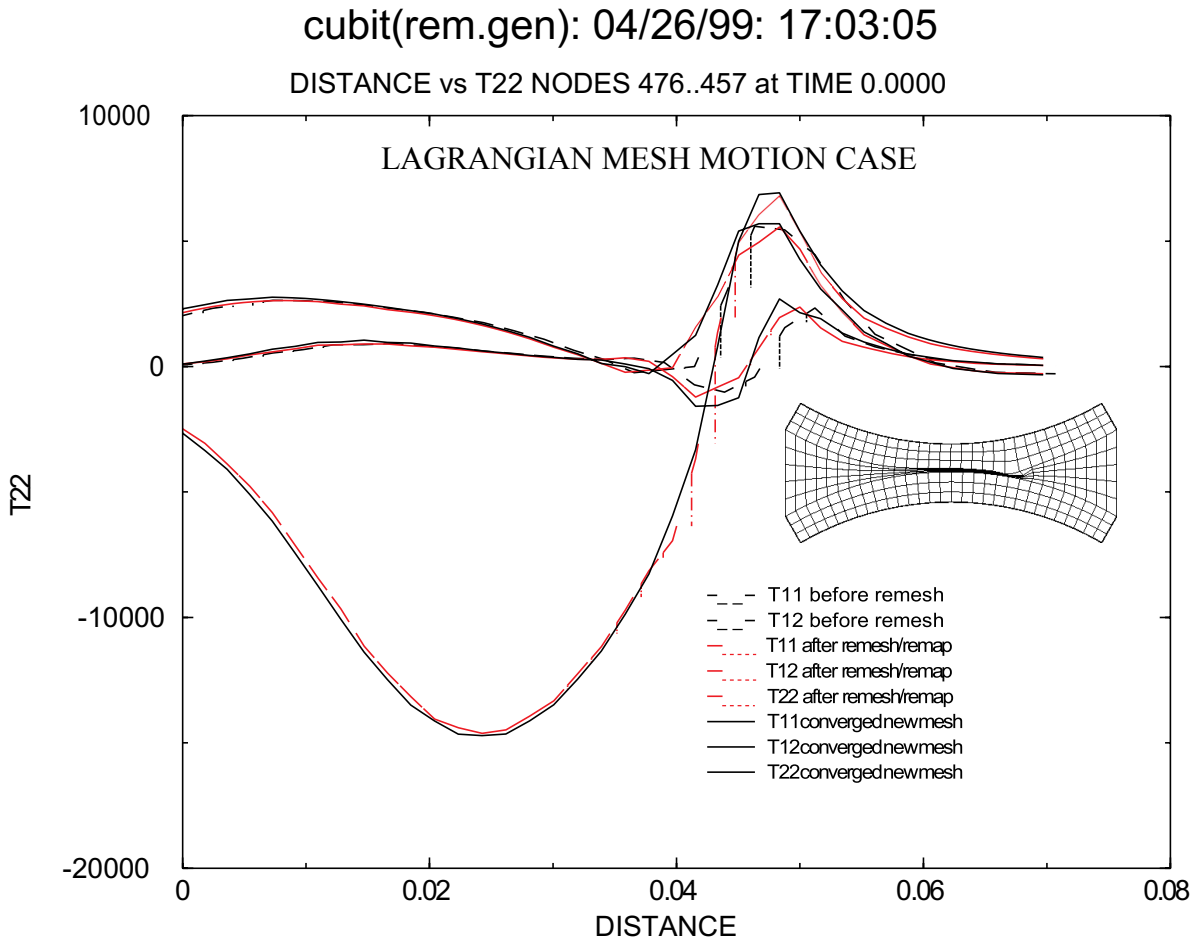
to

```
Initial Guess = read_exoII_file tmp.exoII
```

At first you might be tempted to just use your **rem.gen**, the file created from the **CUBIT** remesh, as your input FE database on the "**FEM file**" card. Recall, however, that **mapvar** actually altered the mesh point coordinates of **rem.gen** when processing the map. Also, **GOMA** gets all of it base finite element entities from the **FEM file**, and not the file on the "**Initial Guess**" card. When regaining the solution on the new mesh, we have noticed that sometimes you need to relax the Newton iteration some, and sometimes you can use full-Newton iteration. It all just depends on the state of the mesh, the problem type, and the parameter range you are dealing with. In this case full-Newton works, viz.

```
goma -a -i roll_input
```

In the figure we compare the stresses on the roll/liquid surface just before (i.e., the solution to be mapped on the donor mesh) and just after the **mapvar** remap (i.e., mapped onto the new mesh), and then after GOMA has been used to converge on the solution on the new grid. Notice that the continuity of the stress profiles along the roll surface is not perfect, but captures the essential features of the solution at least well enough to allow convergence upon restart.



Fluid/TALE Solid Flooded Forward Roll Problem

We also repeated this exercise with a TALE formulation in the solid rolls. You can find this example in the directory **def_roll_nip_TALE**. The details of running this example are given in the updated TALE tutorial memo (GT-005.2). However, the remeshing/ remapping steps deserve some additional attention over and above what is provided in that memo. The remap step for a deformed TALE mesh, in conjunction with an arbitrary fluid mesh, needs to consider the additional material displacement field in the solid, together with the mesh displacement field. You begin as normal here, with a **CUBIT** journal file ready for the remeshing step of a deformed geometry, viz.

```
cubit remesh.jou
```

for the journal file "**remesh.jou**". In **remesh.jou** file you will see:

```
## /usr/local/bin/cubit_Sep1698, Cubit Version 2.0.3-16, Run on 09/16/98
at 03:24:51 PM
## Command Options:
## -warning = On
```

```
## -information = On
nodeset associativity on
set associativity complete on
import free mesh "out.exoII" last
label surface on
display
curve 1 interval 30
curve 1 scheme bias 1.11
curve 3 interval 30
curve 3 scheme bias 1.11
curve 4 2 7 interval 4
curve 4 7 scheme bias 0.5
curve 2 scheme bias {1/0.5}
curve 11 interval 30
curve 11 scheme bias 1.11
curve 8 interval 40
curve 8 scheme bias 0.95
curve 6 interval 40
curve 6 scheme bias 0.95
curve 14 interval 40
curve 14 scheme bias 0.95
curve 15 interval 3
curve 10 interval 3
curve 12 interval 3

surface 1 scheme  map
surface 1 size 0.0021
mesh surface 1
surface 2 scheme  map
surface 2 size 0.0021
surface 3 smooth Scheme Laplacian
mesh surface 2
surface 3 scheme map
surface 3 size 0.0011
mesh surface 3
surface 4 scheme map
surface 4 size 0.0011
mesh surface 4
export genesis "rem.gen"
```

Notice how this journal file instructs **CUBIT** to take the deformed mesh solution in "**out.exoII**" and remesh the deformed geometry with the "**associativity**" commands. The remainder of the commands redefine the mesh density and mesh grading in all of the roll and liquid regions. These are basic **CUBIT** commands. I find it advantageous to just set basic surface and mesh commands, run **CUBIT**, and then display the curve number with "**label curve on**". With those numbers and your new mesh, you can assess how you might want to improve mesh quality be altering the intervals and mesh biasing.

The results of pushing this example to the limit of roll compression, combatting the distortion with remeshing and remapping, and the overall continuation strategy are

covered elsewhere (GT-005.3). Here we will just remesh/remap one step, point out the concepts peculiar to a TALE mesh, and show some results.

With the new mesh in **rem.gen**, as indicated in the "**remesh.jou**" file above, we can remap with **mapvar** as usual:

```
cp rem.gen tmp.g
cp out.exoII tmp.e
mapvar tmp < mapvar.in
```

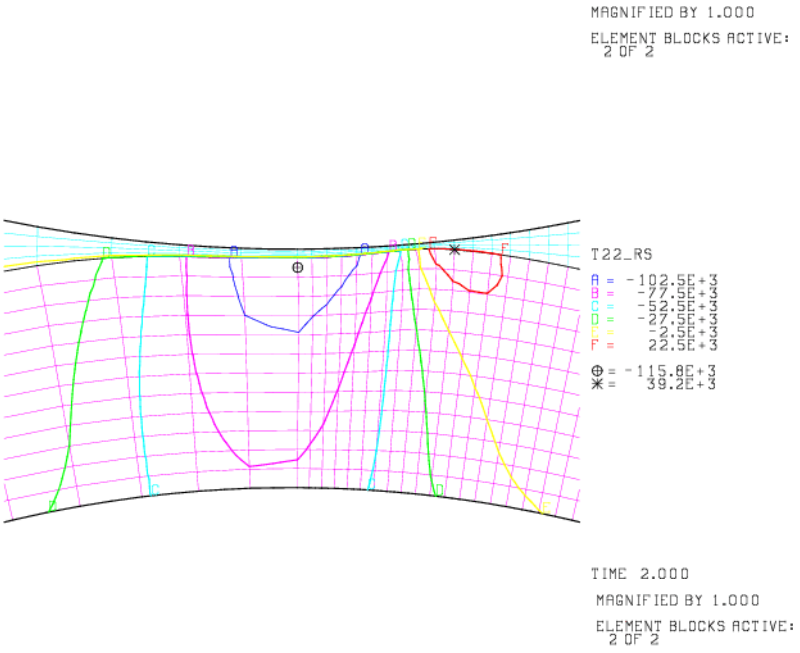
where the contents of **mapvar.in** are the following:

```
map 1 to 1
map 2 to 2
def 2
time 10
search .1
exit
```

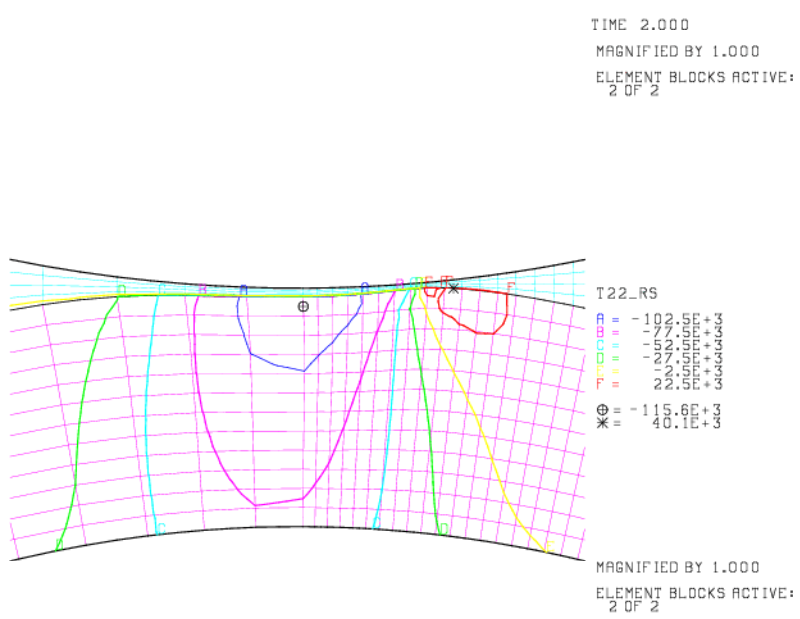
As we pointed out earlier, it is essential to map most conjugate problems one material region at a time, using the **mapvar** "**map**" command. Here we map region 1 first, and then region 2. Region 1 is the solid roll region. We want to map it first so that we obtain a smooth boundary material displacement field. This smoothness is required to preserve the fluid-solid stress balance across the remap and restart. If we map the liquid first, **mapvar** actually places an intermediate value of the material displacement (denoted by d_m or *d_rs* in related publications, not to be confused with the mesh displacement *d*, which is independent), on the boundary. The reason for this is that ExodusII demands a variable to be defined globally, and so if **GOMA** does not define it in a material, it sets it to zero for exodusII output. So **mapvar** then thinks it is zero in the liquid everywhere except at the boundary, and hence puts an interpolated value at the fluid/solid boundary. If the solid is mapped first, then the correct value is put there as the material displacements d_m is defined throughout the material.

Notice how also we use the **def 2** option. This option as discussed above actually anneals the mesh, and maps all other variables. An annealed mesh still preserves the deformation gradient tensor, as is demonstrated below, but requires the user to adjust the Dirichlet boundary values set on the mesh displacements upon restart. This is detailed in the TALE tutorial memo GT-005.3. Finally, we demonstrate the use of the "**searchbox**" option. In this example, we sometimes needed to bump this tolerance up to get a smooth real-solid displacement field upon remap (i.e., **DX_RS** and **DY_RS**).

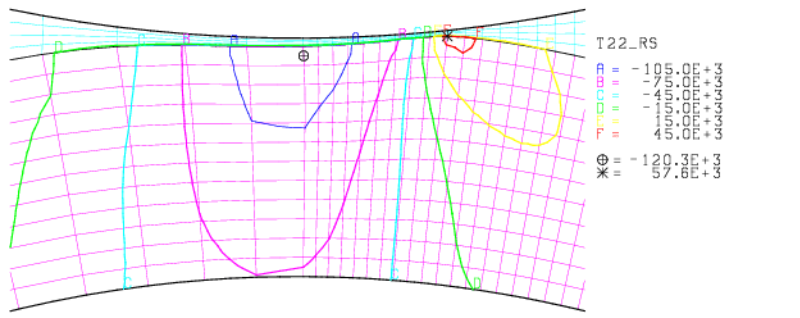
The following figure shows the vertical, or y-component, of the normal stress of the real solid, denoted by T22_RS, just before and just after a remesh/remap step. You will notice that a reasonable job is done to preserve the stress profiles across the remesh/ remap. Slight variations can be seen along the surface and in the minimum/maximum values. In the next figure we compare two cases: one in which we attempted a remesh and remap from a badly deformed grid (top of figure), and one from a relatively undistorted grid (bottom of figure). Clearly the mapping is not very smooth in the former case. Recall that for this process to be successful, we must be able to restart the



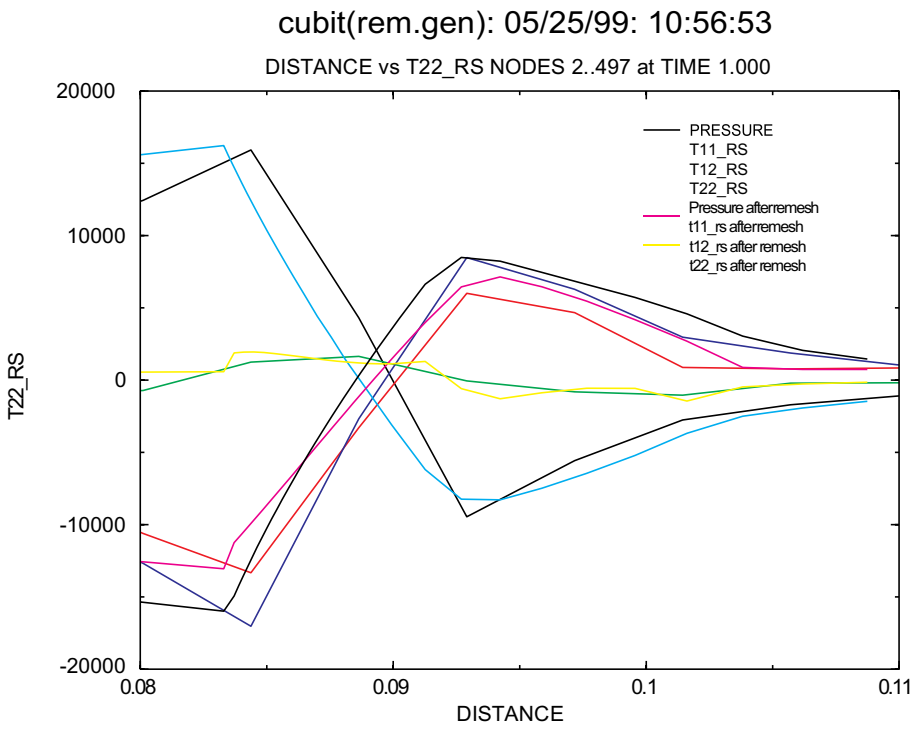
T22_rs
Just before remesh



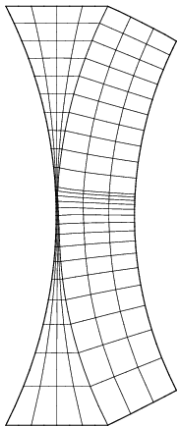
T22_rs
New mesh, remapped

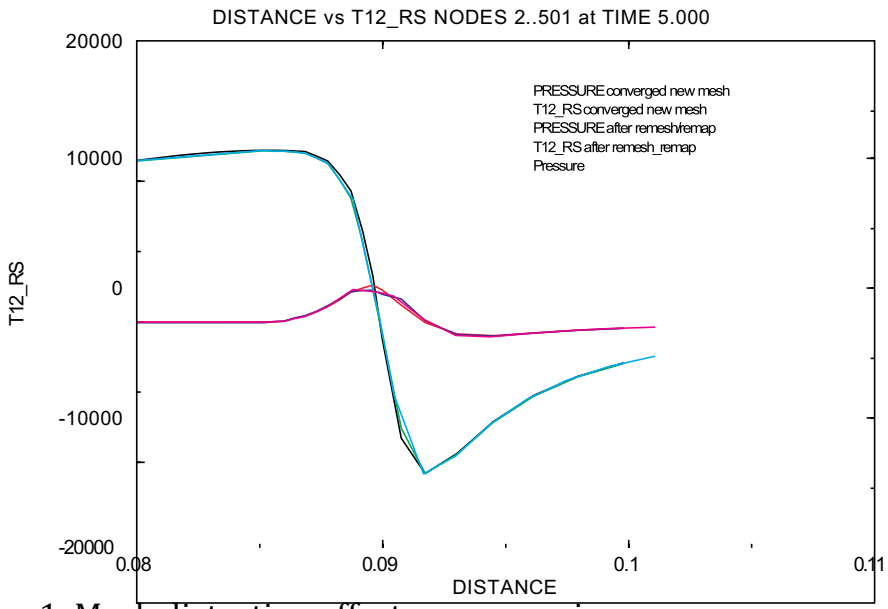


T22_rs
New mesh, remapped,
converged



Mapping from a relatively distorted donor grid to a refined new mesh.





Mapping from a relatively undistorted donor grid to a refined new mesh.

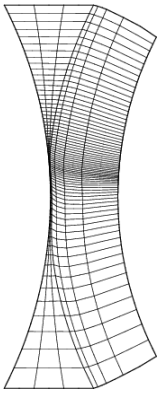
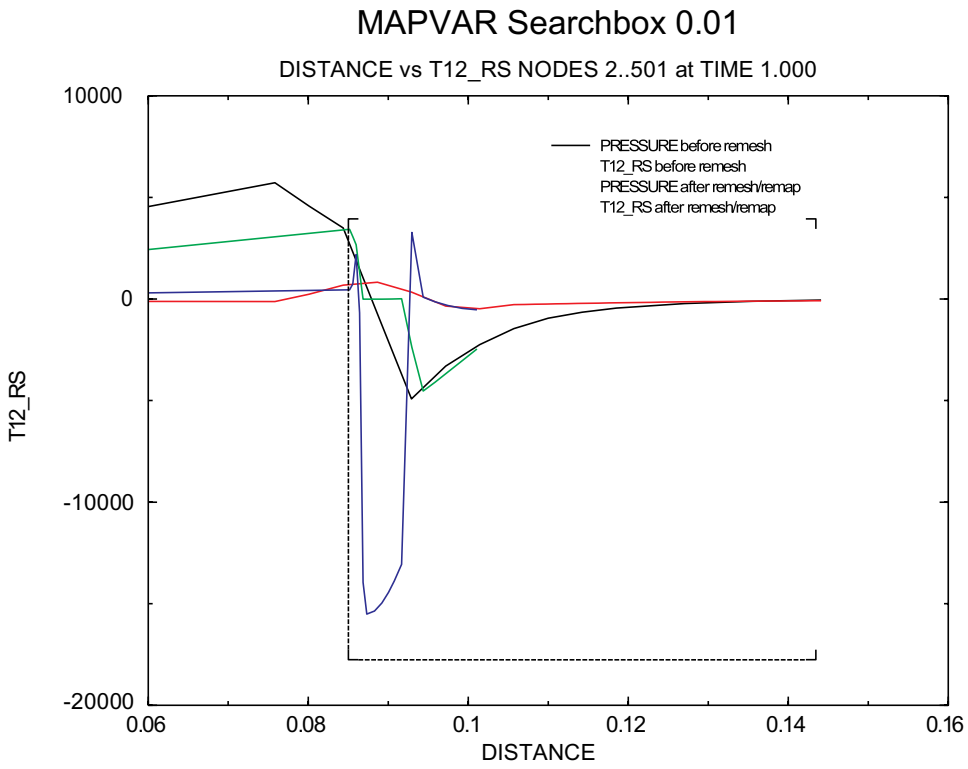


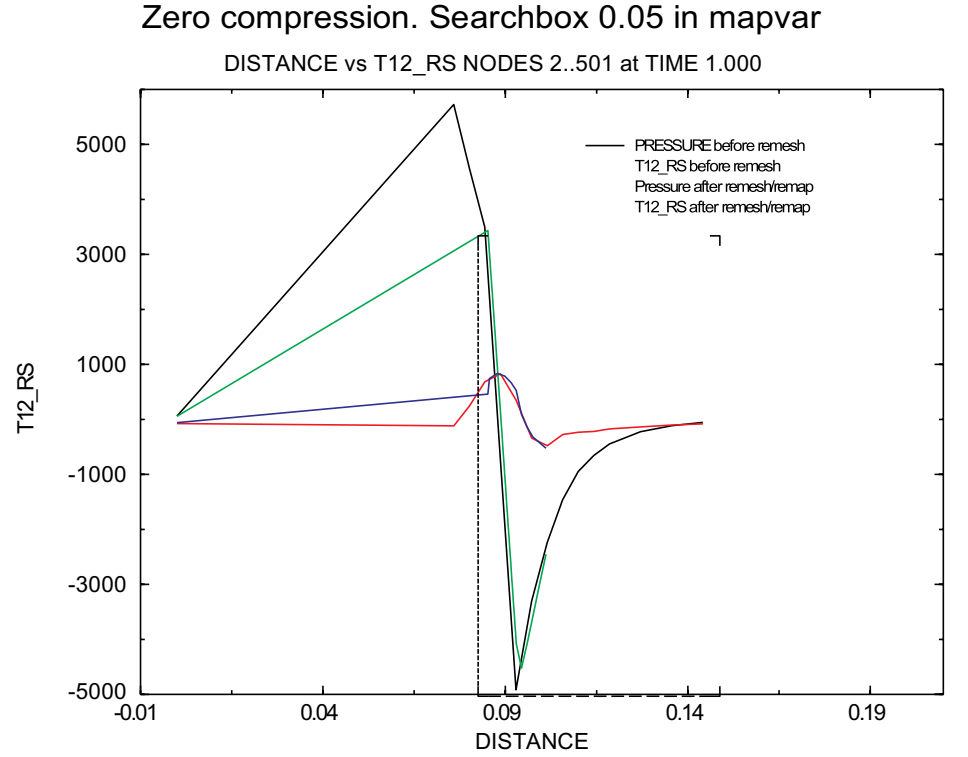
Figure 1. Mesh distortion effect on remapping.

solution process with the new mesh and mapped solution. Large errors like those revealed in these profiles can be "showstoppers".

The final comparison we make regards the effect of the **`searchbox`** option in **`mapvar`**. In the next figure we compare for a particular case the roll surface stress profiles before and after remesh/remap for two different choices, 0.01 and 0.1. The only relevant portion of these plots are shown in the dashed boxes in the figure. The rest of the figures are a result of poor reference point plotting in **`blot`**. Anyway, you can see the effect of this remapping parameter can be significant. We do find, however, that this is not always the case. In this example the gap was extremely thin and the distortion extremely high. Loosening the tolerances with the "searchbox" mapvar command helped preserve the smoothness of the remap significantly.

To demonstrate the power of remeshing, you should consult the related TALE memo GT-005.3, where the same problem is addressed from a GOMA-tutorial perspective.


MAPVAR Searchbox 0.01
DISTANCE vs T12_RS NODES 2..501 at TIME 1.000
PRESSURE before remesh
T12_RS before remesh
PRESSURE after remesh/remap
T12_RS after remesh/remap
10000
0
-10000
-20000
T12_RS
0.06
0.08
0.1
0.12
0.14
0.16
DISTANCE


Zero compression. Searchbox 0.05 in mapvar
DISTANCE vs T12_RS NODES 2..501 at TIME 1.000
PRESSURE before remesh
T12_RS before remesh
Pressure after remesh/remap
T12_RS after remesh/remap
5000
3000
1000
-1000
-3000
-5000
T12_RS
-0.01
0.04
0.09
0.14
0.19
DISTANCE