Sandia National Laboratories

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

Albuquerque, New Mexico 87185-

date: June 13, 2005

to: Distribution

from: Thomas Baer, 1514, MS0834

subject: Tutorial on solution of 3D microfilling problem using level set method implemented in GOMA (GT-32.0)

Introduction

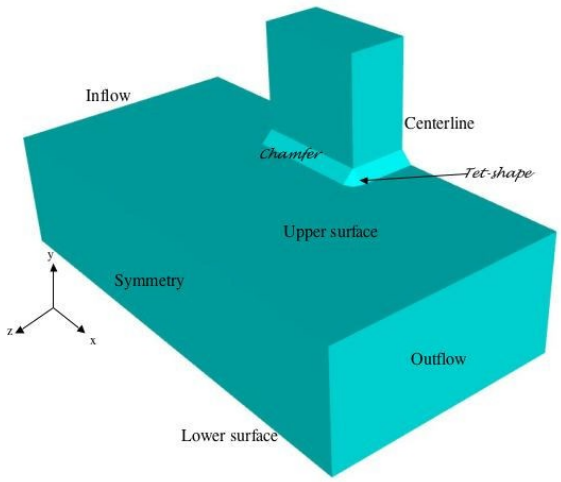
Extension of the level set method to problems that are three dimensional is an important aspect of the method's development. This tutorial concerns the application of the level set method to a three dimensional analog of the 2D microfilling example documented elsewhere. Briefly this problem is pressure driven flow between parallel plates but the upper plate has a rectangular box project transverse to the flow direction. Depending upon the fluid flow parameters the fluid may or may not fill this feature.

In this tutorial we will discuss briefly the meshing of this geometry with CUBIT. Then we shall describe the salient cards in the input deck and the material file. Since iterative solution methods are key to 3D problems, we shall discuss the choices and compromises necessary that lead to convergence. As with the 2D tutorial we shall describe how function space enrichment, integration of surface tension source terms, and models for introducing wetting line motion are applied for 3D problems.

At the outset we must also not that the problem we are presenting in this tutorial was chosen because it was the easiest to solve in three dimensions. It poses a problem where the viscosity is quite high with respect to the surface tension forces. . The capillary number for this problem is rougly 10.0. As a result, we can dispense with a number of the special features developed to cope with high surface tension values We have found that three dimensional problems at the current time are hard enough to solve without introducing this complications.

Geometry and Meshing

A more detailed rendering of the geometry at its proportions is depicted in the following:



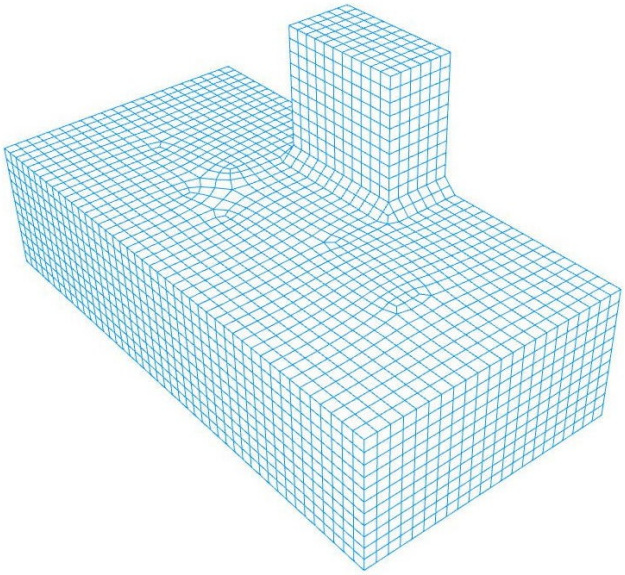
The boundaries labeled “Symmetry” and “Centerline” have no normal velocity component, but natural “shear-free” boundary conditions applied to the other momentum directions. The fluid is introduced at the inflow boundary using “plug flow” boundary conditions while the outflow boundary is assumed to have fully developed boundary conditions: zero velocities on the “in plane” components and natural boundary conditions on the normal momentum component. Lastly, the upper and lower boundaries are in general no slip except in the vicinity of the wetting line where specialized boundary conditions are applied to introduce wetting line motion.

Creation of this geometry in CUBIT is no more complicated, initially, than stacking one box on another. However, as in the 2D example, the need for chamfered transitions from the big box to the little box is required for better behavior of the wetting line. As can be seen, we are using 45° chamfers with small tet-shaped pieces transitioning between the chamfers. Creation of these smaller features is not difficult and the CUBIT journal file, 3DFM.jou, has been included to illustrate this.

The sequence of meshing commands is also included in the file, 3DFM-mesh.jou. In general, anyone familiar with meshing in CUBIT should be able to create a mesh on this geometry without difficult save for the tet-shape features. In this case, a specialized meshing routine is employed. This is invoked by the CUBIT commands:

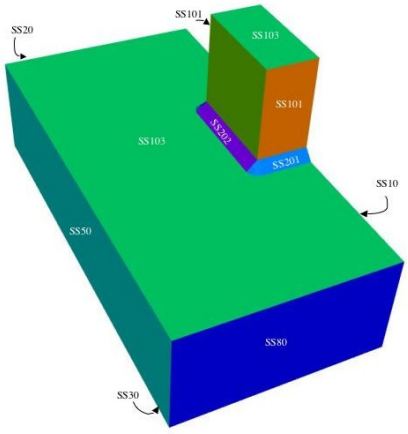
```
volume 4  scheme TetPrimitive
volume 6  scheme TetPrimitive
```

Volumes 4 and 6, of course, are the two tet-shapes and these commands permit meshing of tet-shaped volumes with hexahedron elements. The final result is shown below:



Sidesets

Interpretation of the input deck in three dimensional problems requires so idea of the location of the sidesets (nodesets have exactly the same numbers) in the problem. They are shown in the following figure.



Briefly then, sideset 20 is the inflow plane, sideset 80 is the outflow plane, sideset 10 is the centerline and is matched on the opposite side by the symmetry plane sideset 50. Sideset 30 is the underside of the geometry. On the top of the geometry, sideset 103 combines the horizontal surfaces of the flow channel wall with the upper surface of the feature. Similary sideset 101 combines the vertical surfaces of the feature that are normal to the flow while sideset 102 is the vertical surface of the feature perpendicular to the flow as shown. The surfaces of the chamfer feature are composed of sideset 202, the back side of which is not shown in the picture, and sideset 201 as shown. Finally, sideset 100 is not shown at all but is the union of all the sidesets on the upper surface, 103, 101, 201 and 202.

Input Deck

Next we shall review the input deck and the material file and describe the cards and features that are relevant to solving this problem. The first line in the input deck is the typical aprepro include command which draws the parameter values contained in microfill.def into the input deck and makes them available for use later on.

Most of the cards in this input deck are found in most input decks and they won't be discussed here. We assume the reader is a comfortable Goma user. Nonetheless, there are some that are specific to the level set method. The first two of this occur in the Time Integration section:

```
Fill Weight Function              = Explicit
Courant Number Limit              = 0.1
```

The first card sets the method for integration of level set evolution equation to fully-explicit. That is, the velocity field used is taken from the previous time step and NO coupling is assumed between level set function and velocity during the non-linear iteration process. In truth, this is exactly the formulation employed when the `DECOUPLED_FILL` compiler flag was employed. However, in the current case no separate subcycling of the level set field is done. The level set field is updated at the same time as the other degrees of freedom in the problem. Evolving the level field according to this formulation was determined to possess much superior robustness properties and use of it is suggest for most cases.

Introduction of this integration method, however, introduced a new problem. Explicit integration methods tend to impose much greater restrictions on the permissible time step size. That is the subject of the second of these two new lines. This card imposes a upper limit on the time step size, irrespective of the variable time integrator already in place. . This upper limit is computed from the following relation:

$$dt_{\min} = C \min_{e} \left| \frac{h_e}{\left\| \hat{U} \right\|_e} \right|$$

where on the element, e , h_e is the averaged size of the element and

$$\left\| \hat{U} \right\|_e = \frac{\int_e \delta_\alpha(\phi) v \cdot n d\Omega}{\int_e \delta_\alpha(\phi) d\Omega}$$

The user should be aware that this time step limit is often quite restrictive. However, it is also the case that the actual physics of high surface tension problems with wetting lines imposes time step restrictions themselves. That is, in practice this is not as bad as it sounds.

Level Set Cards

The next set of cards we shall discuss are those of the level set section.

```
Level Set Interface Tracking      = on
Level Set Length Scale          = {elem_size}
```

```
Level Set Initialization Method  = Surfaces 1
                          SURF = PLANE 1 0 0 -0.15

#Level Set Initialization Method  = Exodus
Level Set Renormalization Method = Huygens_Constrained
#Force Initial Level Set Renormalization = yes
Level Set Renormalization Tolerance = 0.21
Level Set Subgrid Integration Depth = 1
```

The first of these cards activates level set tracking. The second card sets a length scale parameter that has widespread use throughout the algorithm. Primarily, this parameter is the width of the distributed interfacial region. The third card of this set specifies the initialization method for the level set function. This is, of course, a crucial step. In this case, we are making use our PLANE primitive geometric shape. The syntax of this option has been discussed elsewhere. Suffice it to say, that this will initially locate a planar interface perpendicular to the flow direction somewhat upstream of the feature. The fourth card has been commented out at the moment, but allows the user to change the initialization method so that the intial level set field is obtained from the initial guess found in the input deck. This card would be uncommented (and the original initialization card commented out) to restart the problem from some intermediate time.

The fifth card specifies the renormalization method. This is the periodic process in which the level set function is manually restored to being a distance function from the interface. The method chosen here (Huygens_Constrained) identifies a large number of discrete points that lie on the interfacial surface. Renormalization of the nodal level set degree of freedoms is done with respect to this point cloud. In addition, a Lagrange multiplier constraint is applied during this process to ensure (as closely as possible) that the negative (or positive) level set volume is not affect by renormalization.

The next card instructs Goma to initiate a renormalization process prior to the first time step of any computation. Right now it has been commented out. We have found that in a large number of situations in which the level set computations have slowed are stopped and require a restart using this card often will restore the robustness of the computation.

Controlling the frequency of the renormalization procedure is the function of the Level Set Renormalization Tolerance card. At each time step the average gradient magnitude of the level set function is computed in the region surrounding the interface. Ideally this value should be exactly unity. How far it is from unity is a measure of how far the level set function has strayed from being a distance function. The renormalization tolerance specifies the range over which this average gradient is allowed to stray before a renormalization procedure is trigger. In the case we present here this range is 0.79 to 1.21.

The last card in this section relates to the higher order integration methods that are needed in the region close to the interface. This card invokes the subgrid integration method which divides elements that overlap the interface elements into smaller integration grids through use of an oct-tree algorithm (quad-tree for two dimensions). The integer parameter on this card specifies the depth to which this division is conducted. Of course, the larger this parameter is the more accurate the integration of the interfacial region. At the same time, the more computationally intensive it will be. In this case, we have chosen a very modest value of 1.

This is because the for the problem we are going to be solving the surface tension forces, which are most affected by accuracy of this integration, are small with respect to the viscous forces. In other words the capillary number is high. For lower capillary number problems, one would increase this parameter for enhanced accuracy. However, for three dimensional problems this rapidly slows the computation down and so one is effectively limit to subgrid integration depths of no more than 3.

Iterative Solver Parameters

Solution of three dimensional problems inevitably requires application of the iterative solvers attached to Goma. This is unfortunate because iterative solver technology has not been extensively applied to problems involving level set indexing of viscosity and density. Add to this the difficulties that arise when parallel processing is included, which it inevitably must be. This leads to a number of compromises being made in order to get a set of equations that can be solved reasonably robustly.

The set of solver parameters in the input deck have been arrived at through a process of trial and error and they seem to give acceptably results. We shall discuss them briefly and also describe the chief set of "knobs" that seem to have the most affect on the convergence behavior of the solver.

First, we are using the gmres solver. This really the only choice as the other methods such as cgs do not work at all. We are using the ilut Matrix subdomain solver . Again this seems to be the best choice as the other choices produce decided inferior results. We are NOT using rcm matrix reordering. Sometimes using this option works wonders other times it is disastrous. It seems to be related to the structure of the mesh. In this case it does not work at all.

The Matrix ILUT fill factor is one point where the behavior of the solver can be affected in a more controlled manner. This parameter can take on the values of 1.0, 2.0, 3.0, …etc. The larger it is, in general, the more rapid is the rate of convergence of the iterative solver. Simultaneously, however, the speed of the solver drops precipitously as this parameter is increased. Also it is sometimes the case that for fill factors other than 1.0, the iterative solver will, for lack of a better term, "choke" and not converge for even one iteration.

The following three solver cards :

```
Matrix Relative Threshold      = 1.0
Matrix Absolute Threshold      = 1.e-5
Matrix drop tolerance          = 1.e-8
```

do not seem to offer much in the way of improving or degrading the solver performance.

The size of the Krylov subspace and the number of linear solve iterations are set at fairly large values. This is a reflection of the fact that the problem size itself is relatively large (in comparison to other problems we have solved). Note that the size of the Krylov subspace and the number linear solver iterations are exactly the same. This should always be the case.

Note that the tolerance level of the linear solver and the nonlinear solver have been set to larger values (1.e-5) than is commonly seen in Goma input decks. This is a cheap way to increase the speed computation, which is viable only if it doesn't simultaneously lead to instability. In this case, it doesn't.

We have turned on pressure stabilization with a parameter value of 10.0 with the last two cards of the solver section. It is almost always true that convergence of the linear solver can be guaranteed provided that the size of the pressure stabilization parameter is large enough. The difficulty is that the larger the size of this parameter the less that the continuity equation is satisfied exactly. Which is another way of saying that mass is conserved less and less the bigger the pressure stabilization. So the procedure is to make the pressure stabilization parameter as small as possible by alter the other solver parameters and increasing it only as a last resort.

Finally, there is an additional parameter that has a significant impact on the convergence rate of our level set problem that doesn't appear in this section at all. This is the value of the light phase viscosity. Now ideally we would like to use a value for this that is close to the viscosity of air for an simple filling problem. Even for two dimensional problem this is unrealistic and we typically have to use values that are significant bigger. Our hope is that if the value of the light phase viscosity is quite a bit smaller than the heavy phase viscosity the actual behavior of the model system is not that far from the actual system, given the other modeling uncertainties.

For three dimensional systems, we must increase the light phase viscosity yet further to guarantee robust convergence of the linear solver. In the tutorial for the two dimensional microfilling case the light phase viscosity was chosen to be 0.01 P. In the current case, we found we had to increase this value by a factor of 30 to 0.3 P to smooth out the convergence rates of the iterative solver. However, both should be compared to the heavy phase viscosity of 100P.

Boundary Condition Cards

For three dimensional problems a general rule in specifying boundary conditions is to take advantage as much as possible of alignments of boundaries with coordinate axes to eliminate the need for application of the VELO_NORMAL boundary condition and other rotated boundary condition types. This makes filling out the Rotation section later on a lot easier.

The first few cards in this section deal with velocity boundary conditions on the vertical sides in the problem. This boundary condtion

```
BC = W NS 10 0.0
```

establishes the symmetry condition on the centerline boundary in the same way this boundary condition,

```
BC = W NS 50 0.0
```

Establishes the vertical boundary on the opposite side of the geometry as a symmetry boundary (see the sideset figure for help).

The inflow and outflow boundary requirements are specified by the following:

```
BC = U NS 20 {Uinlet}
BC = V NS 20 0.0
BC = W NS 20 0.0
```

and

```
BC = V NS 80 0.0
BC = W NS 80 0.0
```

Note that the inlet is a plug flow approach to setting the inlet boundary condition and will lead to a short rearrangement region in the vicinity of the inlet plane.

The remaining boundaries in the problem all have the potential for having a wetting line. Therefore, they must be handled in a special manner. The first requirement is a condition that establishes no penetration of the fluid through the boundary. Since most of these boundaries are parallel to one coordinate axis or another we can do a good portion of this using the following Dirichlet conditions:

```
BC = V NS 30 0.0
```

and

```
BC = U NS  101 0.0
BC = W NS 102 0.0
BC = V NS  103 0.0
```

The only boundaries that are not parallel to coordinate axes are sidesets 201 and 202. Hence, we need to apply VELO_NORMAL conditions to them

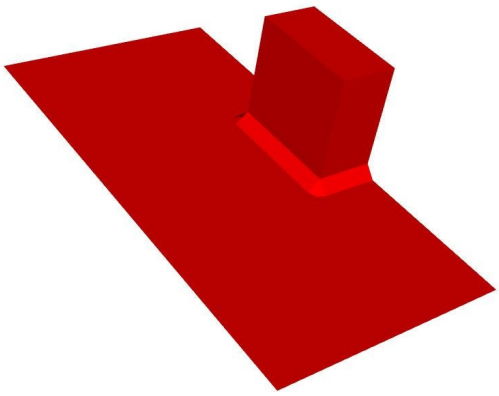
```
BC = VELO_NORMAL SS 201 0.0
BC = VELO_NORMAL SS 202 0.0
```

The two remaining requirements on the “wetted” boundaries are the fluid slips in the vicinity of the interfacial curve and there is a force applied in that region that will induce motion of the wetting line.

The first of these is set using the VELO_SLIP_LS boundary conditions:

```
BC = VELO_SLIP_LS SS 30 {2.0*elem_size} 1.0 0. 0. 0. 1.e-6
BC = VELO_SLIP_LS SS 100 {2.0*elem_size} 1.0 0. 0. 0. 1.e-6
```

Referring back to the sideset figure, sideset 30 is the underside of the geometry. Sideset 101 again is the union of all the sidesets on the top of the geometry including those making up the feature as shown below



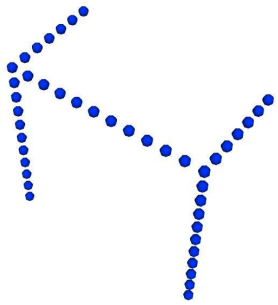
The wetting line stresses that move the contact line are applied with these boundary conditions also on sidesets 30 and 100,

```
BC = WETTING_SPEED_LINEAR SS 30 45 0.1 0. 0.1 0. 0. 0.
BC = WETTING_SPEED_LINEAR SS 100 45 0.1 0. 0.1 0. 0. 0.
```

The first parameter on these boundary conditions is the static contact line angle. The reader is referred to the two dimensional microfilling tutorial for an explanation of the other various parameters on both the VELO_SLIP_LS and WETTING_SPEED_LINEAR boundary conditions.

Now one might ask reasonably where the parameters on these cards come from. They were derived from a succession of trial and error experiments on the two dimensional analog of this problem. The desired effect in those experiments was that the contact line kept reasonable pace with the capillary free surface; neither dashing ahead or lagging behind.

We note in passing the Dirichlet conditions applied to nodeset 301. We haven't described this nodeset before but it is the set of nodes that make up the edges of the feature. It is depicted in the following:



The boundary condition enforced on this nodeset is that the vertical velocity component be zero. The reason was that the convergence of the wetting line boundary conditions above on boundaries that were 90° apart didn't properly constrained the velocity component tangent to these boundaries. The result was small threads of fluid "climbing" these corners at an

unreasonable speed. By setting the vertical velocity component to zero just on these edges we were able halt this behavior.

Finally, when solving level set problems for capillary hydrodynamics, the surface tension forces must be introduced as an embedded boundary condition. This is done with this card

```
BC = LS_CAPILLARY LS 0
```

This card refers to the material file for the value of surface tension parameter and then applies the appropriate stress tensor to the fluid distributed about the interfacial region to an extent controlled by the level set length scale set previously. The string "LS" identifies this boundary condition as being embedded and therefore not associated with any nodeset or sideset. The integer parameter on this card is set to zero because the level set length scale is non-zero and we are not using subelement integration (Reminder: subelement integration has not been implemented for three dimensional problems.)

Rotation Conditions

Now we come to the dreaded Rotation Condition section. Fortunately, by prudent choice of sidesets we have minimized its size and complexity. We have only two sidesets with rotated boundary conditions. These are 201 and 202 and their VELO_NORMAL conditions. The associated rotation conditions are:

```
ROT = MOM SURFACE 201 VELO_NORMAL 201 T1 0 T2 0 SEED 0 1 0
ROT = MOM SURFACE 202 T1 0 T2 0 VELO_NORMAL 202 SEED 1 0 0
```

Rotation is the process of projecting the momentum residual equations into a coordinate from that consists of a normal to the surface and two orthogonal tangent directions. The rotation card is used to associate these three rotated components with the unrotated directions, x, y, and z. It is most often the case that the normal component is replaced by the rotated boundary condition, VELO_NORMAL in this case. The general rule is that the boundary condition should be associated with the direction its residual value is most sensitive to.

Hence, the first ROT card associates the VELO_NORMAL condition on sideset 201 with the x direction, because this boundary condition is sensitive to variations in the x velocity. It is also sensitive to y velocity variations to roughly the same degree since it is angled at 45°, so one must use trial and error to determine which direction is the best. Similarly, since sideset 202's normal has identical non-zero y and z components, the VELO_NORMAL condition has equal sensitivity to both directions. Choice again between y and z association is determined by trial and error. However, it is clear that associating this boundary condition with the x direction on this sideset is incorrect.

Since both of these boundaries are or very nearly are planes, choice of a constant seed vector is the appropriate choice. In effect, the user is defining one of the tangent directions and the second tangent direction is found through the right hand rule. In the case of sideset 201, the seed vector is (0,1,0) and in the case of 202 it is (1,0,0). The important point in this choice is that the seed vector should never be close to being normal to any portion of the sideset.

A more detailed descriptions of the Rotation Conditions can be found in Goma tutorial, GT-18.1

Equation Section

The equation section requires few clarifying remarks because it is all typical Goma. There are five degrees of freedom: three momentum equations, one continuity equation, and one level set equation. The interpolation and weight functions are all Q1, that is, trilinear hexahedral functions. Note that we are NOT using XV interpolation on the pressure interpolation. As alluded to previously, the capillary number is high enough that this step is not required for stable computations and inclusion of it has a significant effect on overall iteration times and linear solver convergence rates.

This section does reference the material file, newt.mat, which we shall discuss next.

Material File

The material file is also pretty much vanilla Goma, but it does use the Second Level Set syntax which probably needs explanation.

Consider the cards associated with the Density property

```
Density              = CONSTANT     {rho_liq}
Second Level Set Density = CONSTANT {rho_gas} POSITIVE
```

The first card is the standard means of specifying a constant value for density. However, when the second card is include, the value of density is automatically made a function of level set function. This second card gives a second value for density, {rho_gas} and because of the prescence of the string “POSITIVE” this density value is associated with positive values of the level set function. By elimination, the negative side of the interface receives density values that were specified on the first density card, {rho_liq}. The transition between the two values is done with smoothed Heaviside function as is commonplace in our level set implementation. The width of this transition is the level set length scale specified earlier.

An exactly similar method for specifying a level set-dependent viscosity is done through these cards:

```
Viscosity            = CONSTANT {mu_liq}
Second Level Set Viscosity      = CONSTANT {mu_gas} POSITIVE
```

Parallel Operation

Since this problem is three dimensional, it presents a large number of nodes and a large number of degrees of freedom. As a consequence, operation on parallel processors is highly advantageous for obtaining results in a reasonable period of time. We shall present the steps needed to launch a four processor parallel job from an exodusII file named contin.exoII. The user may modify this procedure for different numbers of processors and differently named exodusII files.

First, the exodusII file must be divided into four pieces, one for each processor. This is done using a simple utility called brk which takes as input the file brk_ulf.in. This file is included with the file package. This latter file is specific for three dimensional problems with five degree of freedom fields, velocity components, pressure and level set. It is applicable only to 8-node hexes on which all fields are interpolated with Q1 polynomials. Division of contin.exoII is done with the following command:

```
% brk –n 4 brk_ulf.in contin.exoII
```

This produces four pieces named in accordance with Sackinger's Borg convention : contin_1of4.exoII, contin_2of4, contin_3of4.exoII, contin_4of4.exoII. Goma will look for these names when the parallel job is launched. No modifications are necessary to the input deck or material file to run a parallel job as opposed to a serial job.

Launching the parallel job is done using the mpirun script, vis,

```
% mpirun –machinefile machinelist –np 4 goma –i microfill3D.inp –a –se serr –so sout &
```

Here the file machinelist is just an ASCII file with the hostnames of the four processors each on a separate line. Note the regular Goma command line following the executable name. We are using redirection of the stderr and stdout streams to the files serr and sout, respectively.

The result of this computation is a set of output exodusII files with the obvious names on the corresponding domains: out_1of4.exoII, out_2of4.exoII, out_3of4.exoII, out_4of4.exoII. These so-called daughter files must be reformed into the so-called monolith file, out.exoII. This is done by invoking the opposite utility to brk, which is call fix . This utility puts these files back together.

```
% fix –n 4 out
```

is the command that starts this utility and the result is a monolithic file, out.exoII.

Startup of the Computation

Getting a computation of this sort started often involves a few steps. This problem is no exception. It would be tempting to launch immediately into the filling computation, but performing an additional step helps startup of this computation immenously. You see the initial shape of the interface is a flat plane perpendicular to the x-axis. But our boundary conditions impose a 45° static contact angle with a small amount of surface tension at the capillary surface. The static shape of the interface therefore should be a cylindrical section that connects 45° intercepts of the upper and lower boundaries. This is the configuration that the planar interface will want to assume immediately after startup. This is a very short time scale and therefore introduces stiffness into the overall filling problem. We should acknowledge this process and solve for it first and then use the output of it as an initial guess for the filling problem.

We first modify the input deck by changing the sign on the time step size

```
delta_t                                 = -5.e-6
```

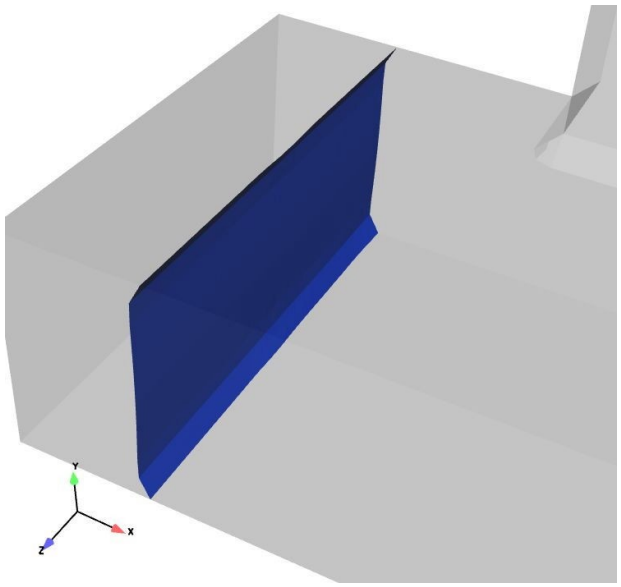
This has the effect of fixing the time step size at 5 microseconds and doesn't allow the variable time step function to change it. Depending upon the progress of this startup computation, we might stop the computation, increase this number, and restart just to keep things moving along.

The next modification to the problem is to stop the filling. This is done most easily by changing the inlet flow velocity card to the following:

```
BC = U NS 20 {0.0*Uinlet}
```

Thus, there is no externally applied flow. The only flow that is in the problem arises from rearrangement of the starting interface configuration.

The procedure is to start the computation in parallel according to the above procedure. One then periodically fixes the exodusII files and examines them. Initially, the velocities near the contact line are quite high as the wetting stress are also quite high because of the large deviation of the apparent to static contact angle. But over a period of time (which I have quite forgotten sorry) the apparent contact angle shifts obviously towards 45° and the wetting velocities slow. It may be the case, that the analyst will have to halt the computation, do the fix/brk dance, restart with a larger time step as this happens. We did this and reached the following configuration. We were becoming bored at this point and decided to throw caution to the wind and start the filling computation for real.



Filling Computation Procedure

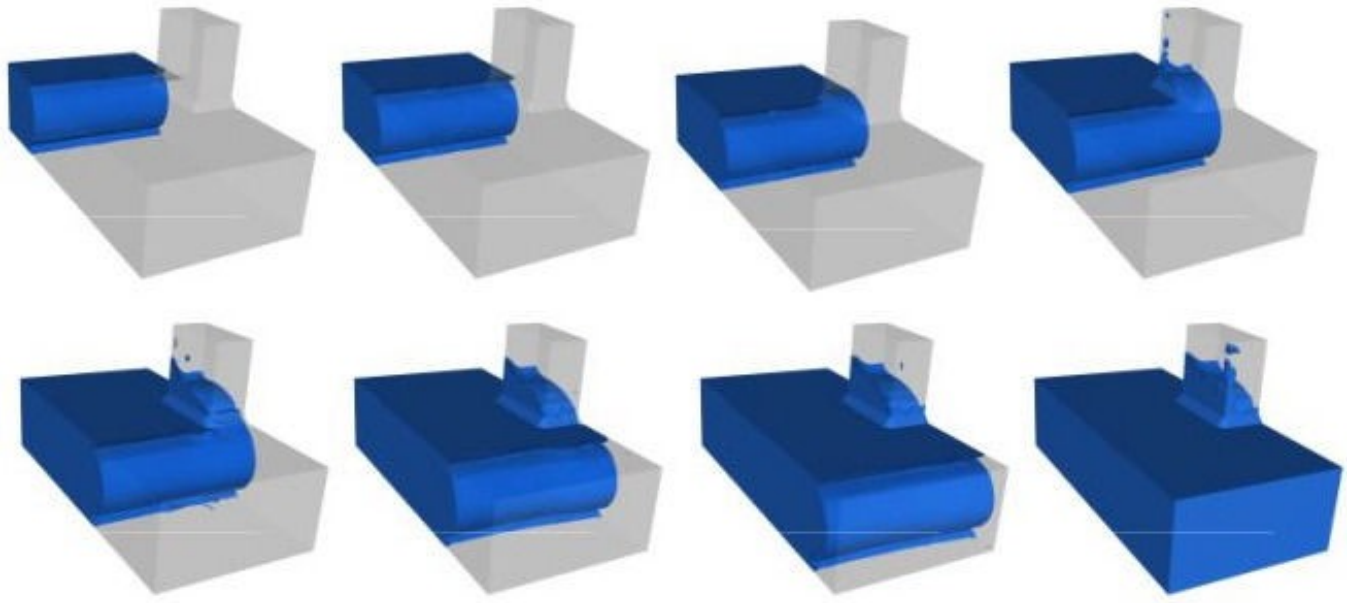
Now we reverse the steps applied just now to the input deck. The time step sign was set back to positive and we remove the zero factor on the inlet velocity card. The final output exodusII file from the previous procedure is broken into four pieces. A short prayer might be offered to the gods of computations and then we let fly with mpirun :

```
% mpirun -machinefile machinelist -np 4 goma -i microfill3D.inp -a -se serr -so sout &
```

The computational procedure from here on out involves little more than monitoring the progress of the solution. This particular computation took a little over two weeks on four Intel processors. Because we are using explicit integration of the level set equation, the time steps must be kept necessarily small to preserve accuracy. This required approximately 20,000 time steps from start to finish. This means many time consuming non-linear iterations of Goma. We cannot even place our hopes in increasing the number of processors because it is also true that these problems do not scale well with number of processors. Typically, the number of linear iterations increase right along with processors which, when combined with communications overhead, eventually yields an overall *slowdown*. This is the challenge we face solving 3D level set problems with *Goma*.

Note also that the robustness of the computation is not one hundred percent. Having to restart is required every so often. In this computation, restarting was necessary six times; albeit, sometimes due to things other than recalcitrant computations (coworkers' complaints primarily).

Nonetheless, we were able to arrive at a final set of time planes which proceed from startup to exit of the filling front from the domain. Eight of these planes are illustrated below.



We note the convex shape of the free surface in the middle of the channel. This is typical of a high capillary number filling problem. Note also that our wetting line boundary conditions are doing a good job at keeping the dynamic contact angle near to 45° and also in keeping the wetting line from falling behind the rest of the flow. To be sure, transition from free surface to contact line looks a bit strange, but this is due to the refinement of the mesh and the fact that we are interpolating the level set function with trilinear functions.

As expected, the feature did not fill at this high value for capillary number. One thing observed, however, in the feature was the prescence of stray bits of fluid climbing up the corners. We introduce an additional boundary condition (on nodeset 301) to mitigate the

formation of these bits, but they persist to a lesser degree. We conjecture that their formation is related to the wetting line stresses overlapping at the corners and producing larger values for the wetting line velocities.