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, 9114, MS0834

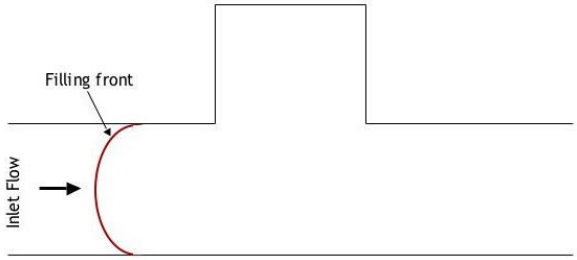
subject: Tutorial on solving microfilling problem using level set method implemented in GOMA (GT-31.0)

Introduction

Recent advances in modeling interfacial motion with the level set method in GOMA are beginning to permit solutions at surface tension regimes that were formerly inaccessible. An example problem that demonstrates this is filling of feature or groove connected to a microchannel. The material and flowrate parameters are chosen such that surface tension forces are very significant. This tutorial will describe the geometric parameters of the mesh along with features of it that facilitate solution of the problem. The input deck and material file will also be described with primary attention to the special features required for solution in this flow regime. In particular, function space enrichment, integration of surface tension source terms, and models for introducing wetting line motion.

Geometry, Material and Flow Parameters

The geometry for this problem is deceptively simple: a horizontal planar channel is interrupted on its upper surface by rectangular inclusion. Flow enters from the left side and exits at the right. A filling front is introduced to the left of the inclusion and the gist of the problem is to advect this front through the remainder of the geometry. Below is a sketch.

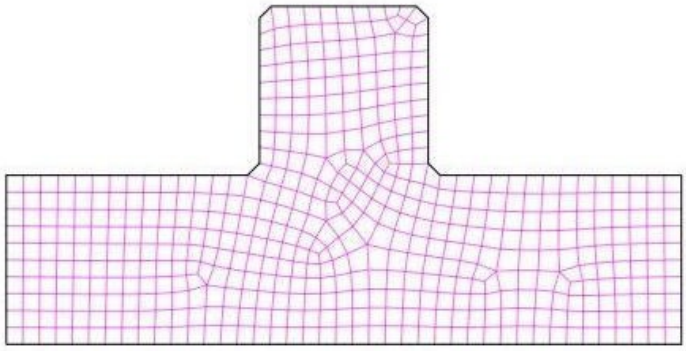


What makes this problem interesting is the presence of wetting line motion past the four right angle changes in direction and the potential for fluid-solid contact of the filling front at the

downstream corner of the inclusion. Add to this significant surface tension source terms at the filling front and the result is a challenging application for the level set method.

To be more specific, the height of the channel is fixed at 0.1 cm (1mm). Likewise, the transverse depth of the inclusion is also 0.1 cm with an aspect ratio of 1.0. The overall length of the domain from inlet to outlet is 0.4 cm. We choose an average inlet velocity of 0.1 cm/s. The viscosity of the filling fluid is fixed at 0.1 P with a density of 1.0 g/cm^3. The viscosity and density of the downstream "gas" phase fluid are in general chosen as low as possible that permit robust numerical solution. In this case, values of 0.002 P and 0.001 g/cm^3 seemed to meet this criteria. Computation of an capillary number based upon average inlet velocity and filling front properties yields 0.01. This is an indication that surface tension forces and wetting are likely to play a very important role.

Putting a mesh on this geometry is not particularly difficult. One consideration, however, is that the right angles in the geometry often present obstacles to the motion of the contact line. Smoothing these features is a means to facilitate solution later on. This can be done by replacing them with a circular arc of small radius of curvature, however, in this work we chose to replace the right angles by short chamfers angled at 45° to the original walls. These chamfers should be evident in the following picture of the mesh used.



For future reference, this mesh has 501 elements, 2125 nodes and the mean element size is approximately 0.01 cm. The `cubit` commands used to create this mesh is included with this tutorial in the file `microfill.jou`. This tutorial is not for instruction in the use of `cubit` so the details of these commands will not be commented on further.

Also included with this tutorial document, are two versions of Goma input files for solution of this problem: `microfill_subgrid.inp` and `microfill_subelem.inp`. The distinction between them, of course, relates to the method used to integrate the surface tension source terms, subgrid or subelement. This tutorial will describe in detail the first file and then consider the second file in the context of how application of subelement integration differs.

Before we do that, however, let us consider the aprepro definition file, `microfill.def`; also accompanying this tutorial. There is nothing particularly new here. Aprepro variables are defined for the liquid and "gas" phase parameters are specified as well as the average inlet flow speed and average element size. The latter is used often in level set problems. We mention here only to emphasize the use of aprepro definition files where potentially

changeable parameter values ensures consistency and is considered good practice. Note that this files are included at the top of both input deck and material files.

Moving on now to the first input deck, microfill_subgrid.inp, we note that the first few lines are boilerplate and should be familiar to anyone who has used Goma previously. The first two lines that need to be noted in the current context are:

```
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 specify parameters and methods specific to the level set function.

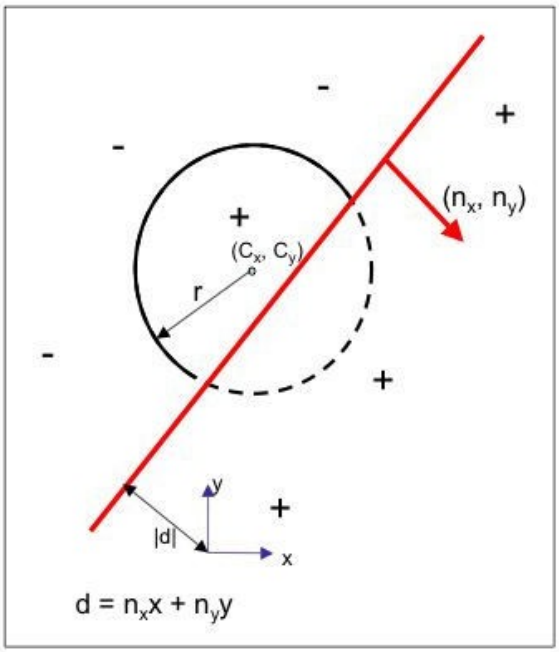
```
Level Set Interface Tracking     = on
Level Set Length Scale         = {elem_size}
#Level Set Initialization Method  = Exodus
Level Set Initialization Method  = Surfaces 1
             SURF = ARC {cx} {cy} {r} {nx} {ny} {d+elem_size}
Level Set Renormalization Method = Huygens_Constrained
Level Set Renormalization Tolerance = 0.25
```

```
Force Initial Level Set Renormalization = yes
Level Set Subgrid Integration Depth = 3
#Level Set Subelement Integration = on
```

The first of these pretty clearly indicates that level set tracking is active. This is required for any problem using level unknowns. The second card sets a distance parameter referred to as the level set length scale. This is used to set, primarily, the width of the interfacial region surrounding the zero level set surface for use in applying surface tension forces or transition between material property values.

The third card is commented out in the example (#). It would become active when restarting a from a previous level set computation in which it is desired to initialize the level set field from the exodus file.

The fourth card specifies the active level set initialization method. Initialization of the level set field is a non-trivial problem and we have implement various possibilities (SEE OTHER TUTORIALS). The one presented here was recently added and allows the user to specify a starting level set field that is the arc of circle. Since the equilbrium configuration (no flow) of a capillary free surface in a planar channel is the arc of a circle, this is the appropriate choice for the current problem. Note the set of parameters following the `SURF =` delimiter. The first is the string "`ARC`" which identifies the geometric primitive as the arc of a circle. The parameters appearing on the remainder of the card are described in the following sketch



The center of the arc is at (c_x, c_y) and the radius of the arc is r. The vector (n_x, n_y) and the parameter d define the "knockout" plane. As can be seen in the sketch above only those points on the circle that are on the "negative" side of this plane are retained in defining the arc primitive. The points indicated by the dashed segment above are not included in the arc. From this arc the level set function is initialized by finding the minimum distance to the points on this arc with the sign assignments as shown in the sketch.

The next card of the level set section sets the renormalization method. To those not familiar with the level set method, this is a procedure that is applied periodically to return the level set function to its original distance function form. It is a non-physical procedure and can potentially introduce progress mass loss. The `Huygens_Constrained` methodology uses Lagrange multipliers in the renormalization process to mitigate this mass loss and it has emerged as the preferred renormalization procedure. Nevertheless, another rule has also emerged which states that renormalization should not be conducted frequently. Controlling the frequency of the renormalization procedure is the function of the next 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 0.75 to 1.25.

The `Force Initial Level Set Renormalization` card indicates that a renormalization will occur prior to the first time step regardless of the value of the gradient norm. We find that this is often a good way to encourage a level set problem to continue if it has unaccountably drifted off into a region of small time steps and/or poor convergence. A simple restart with this card turned on will often turn the problem away from its former state of computational sin and debauchery and place it back on the path to numerical righteousness.

The last level set –specific card, `Level Set Subgrid Integration Depth = 3`, sets the depth of subgrid integration to the third level. By association it also turns on subgrid integration whenever this card is present with an integer parameter different than zero. A integration depth set to three indicates that the smallest grid used to integrate the interfacial region will be $1/2^3$ the size of the original element.

Matrix solver

The matrix solver used in this problem is the multi-frontal solver package, `umf`. The latest version of this has proven to be quite fast and reliable and it has evolved as the solver of choice for two-dimension problems. Diverting briefly to three-dimensional problems, we note that direct solvers like `umf` can't really be used effectively for this class of problems. Instead, we rely upon iterative solver algorithms which have the advantage of being faster and using less memory in three dimensions than direct solvers but with the accompanying disadvantage of being considerably less robust and stable. In particular, we have found that incompressible fluid mechanics problems are difficult for most iterative solvers to solve well. We have also discovered that problems involving dramatic changes in values of density and viscosity across an interface a particularly hard for iterative solvers to deal with. This is proving to be a significant barrier to robust three-dimensional level set solutions.

Boundary conditions for Wetting Line motion

Introducing wetting line motion in the context of level set interface tracking turns out to be somewhat involved. The details of the technique that we have evolved requires three separate conditions on a boundary were wetting line motion is expected to occur. The first imposes no-penetration of fluid velocity through the boundary. This may be done in general with a `VELO_ NORMAL` condition or a Dirichlet velocity condition depending upon the orientation of the boundary.

The second required boundary condition allows for the no slip condition to be altered to a greater or lesser degree in the region near the contact line. This is accomplished though use of the `VELO_SLIP_LS` boundary condition. This boundary condition imposes a Navier-slip stress condition at the boundary with the addition that the β parameter can different near the the wetting line.

The third boundary condition need to induce wetting line motion applies a stress (in addition to the underlying viscous stress) at the contact line. This stress is dependent upon the discrepancy of the current, actual contact angle and an assumed static contact angle. The greater the discrepancy the greater the stress. The combination of Navier-slip and this "extraordinary" wetting line stress can induce, with the right choice of parameters, a wetting line velocity that is consistent with the remainder of the flow field. At the moment, this is all we can ask of a wetting line model.

Let us consider how this is accomplished in the example at hand. In the boundary condition section, we see these three boundary conditions applied to sideset 10:

```
BC = VELO_NORMAL SS 10 0.0
BC = VELO_SLIP_LS SS 10 {2.0*elem_size} 1.e-5 0. 0. 0. 1.e-6
BC = WETTING_SPEED_LINEAR SS 10 30.0 0.1 0.0 0.001 0. 0. 0.
```

The first one, of course, is the `VELO_NORMAL` condition discussed above and its parameter list is familiar to even a novice Goma user.

The second boundary condition is the `VELO_SLIP_LS` condition applied to the sideset. The first parameter in its list is the width of interval around the contact line in which a different value of β is applied that on the rest of the boundary. This of course allows for a different level of slip to applied in this "special" region near the wetting line. The second parameter on this card is this special value of β. The next three parameters on the line are the components of the substrate speed velocity. These are zero for stationary boundaries, but they may be different from zero in, for example, slot coater types of problems. Finally, the final parameter on this card is the β value that is applied over the rest of the side, i.e. those parts of the sideset that are not in the "special" region near the contact line. It is also important to note that since the contact line is in motion, the special region will also move so that a region on this sideset my very well see both values of β applied to it over the course of the computation.

The last piece required for a moving contact line is the `WETTING_SPEED_LINEAR` boundary condition. This boundary condition computes a wetting line speed from a linear relation ship of the form:

$$V_{wet} = \frac{1}{c_T}(\cos(\theta) - \cos(\theta_s))$$

where c_T is a constant of proportionality, θ is the contact angle, and θ_s is the static contact angle parameter. This velocity can then be related to the stress required to produce it on a couette liquid layer of unity viscosity and thickness δ, to wit,

$$\tau_{wet} = \frac{1}{c_T \delta}(\cos(\theta) - \cos(\theta_s))$$

From which it can be seen that δ can be used to set the level of stress for a given contact angle deviation. This extraordinary wetting line stress is added to the fluid in the region near to the contact line. Since the fluid is permitted to slip in this region to the extent specified by the VELO_SLIP_LS boundary condition, the net result is that a velocity will be induced near the contact line.

The parameters on the `WETTING_SPEED_LINEAR` boundary condition are first the static contact angle in degrees, second the value for c_T, the third parameter is unused, the fourth parameter is the value for δ, and last three parameters are the components of the substrate velocity.

A similar set of boundary conditions is applied to the other wetting boundary sideset 20.

The surface tension source terms are included by including the boundary condition:

```
BC = LS_CAPILLARY LS 0
```

The integer that appears on this card is set to zero because we are using subgrid integration of this source term. A different choice is made when using subelement integration. This boundary condition applies as a body force term to the fluid momentum equation which scales with the surface tension value specified in the material file. This source is also scaled by the smoothed Dirac function centered on the zero level set surface with a length scale parameter equal to the level set length scale set previously. As a consequence, surface tension forces are applied only in the vicinity of the zero level set surface.

Equations

The next section in the input deck specifies the weight and interpolation functions for the several fields. The only thing that is not typical is the extended interpolation functions assigned to the pressure field. These are designated as Q1_XV interpolation. These interpolation functions essentially add pressure degrees of freedom to nodes that are in the vicinity of the zero level set surface. The shape functions associated with these degrees of freedom are step functions in the level set field value with compact support provided by an additional weighting by the standard node based shape functions. The net effect is to permit better representation of the step change in pressure that occurs across capillary free surfaces.

Subelement Integration

Next consider the subelement integration version of the input file we just discsussed: `microfill_subelem.inp`. We shall not conver it in the same amount of detail as previously. Only the salient differences will be identified.

The first primary difference with the subgrid integration cases is that the level set length scale is set to zero when using subelement integration. This is possible because the subelement integration scheme actually produces a geometric representation of the zero level set surface on which exact line integrals of the surface tension source term can be done. As an aside, it is also important to note because of this construction of a in-element interface meshing to find this representation, subelement integration cannot be used currently for three-dimensional problems. Subgrid integration is applicable to three-dimensional problems, but it is somewhat inefficient.

Further on in the level set parameter specification section the subgrid integration depth card has been commented out and the card:

```
Level Set Subelement Integration = on
```

has been substituted in its place. This is a required card for using subelement integration. Note also in this section that the renormalization method has been changed to simply `Huygens` instead of `Huygens_Constrained`. The current version of Goma does not support the latter renormalization method with subelement integration.

In the boundary condition section, the major changes are that the `WETTING_SPEED_LINEAR` boundary conditions have been replaced by `SHARP_WETLIN_VELOCITY` boundary condition. The latter boundary condition is identical in form to the former condition, but it differs in its means of application. The `SHARP_WETLIN_VELOCITY` imposes the identical wetting stress as a point stress at the contact line directly. The `WETTING_SPEED_LINEAR` boundary condition instead weigths the wetting stress by the smooth Dirac function and hence imposes it as a distributed stress around the zero level contour. The parameter lists for boundary conditions are identical. It should also be pointed out that the `SHARP_WETLIN_VELOCITY` boundary condition is applicable again only to two-dimension problems.

Finally, the surface tension source terms are applied differently when using subelement integration. Consider the card:

```
BC = LS_CAPILLARY LS -1
```

Whereas, this card applies again the same formulation of surface tension source term, because of the subelement in-element meshing these terms are computed, not as volumetric body forces, but as actual line integrals. Because of this it is necessary to identify which phase this line integral force is to be applied to. This is the function of the last integer. In this case, it indicates that the integral be applied to the negative phase. If it were 1, then the opposite would be true.

Material File

The material properties for this problem are specified in the file `newt.mat`. This is a Newtonian fluid problem with constant values for density and viscosity in each phase. The values for each phase are specified using the `Second Level Set` card format. In this format, the material property parameters for one of the phases are specified via the eonventional Goma material parameter cards. Indexing of the (in this case) viscosity or density with respect to the level set function is invoked by adding after the conventional cards, `Second Level Set Viscosity` or `Second Level Set Density` cards. On the latter cards a constant value for the appropriate property is specified as well as the string "`POSITIVE`" or "`NEGATIVE`" which identifies which side of the interface this value is assigned. The opposite sign will compute viscosity or density via the models specified previously.

In the current case, the conventional Goma models used for density and viscosity are simple constant value models. This example is intended to show the nature of this format without introducing to much complexity. Note that for this case, exactly the same model would be produced if the density and viscosity models were chosen to be `LEVEL_SET` with the appropriate parameter choice.

The remainder of the material file is conventional Goma and requires no elaboration in this advanced tutorial.

Running the Problem

As is the case with most level set problems, the actual details of running the problem are not much more elaborate than invoke the executable from the command line:

```
% goma -i microfill_subgrid.inp -a -se serr -so sout &
```

In this case we have use command line options to read from the specified input file (instead of the default `input`), invoked `aprepro` preprocessing, piped both the `stderr` stream and the `stdout` stream to the files `serr` and `sout`, respectively. Now we sit back and wait.

Take a look at `serr` after a bit and make sure that the code started smoothly. After a time step or two has passed it is also adviseable to examine the exodus output file with `blot`. Contouring the level set function (variable F) should reveal the zero level set contour is an arc about midway between the inlet and the square feature.

More waiting. Take a look at the exodus output again. Vector plot the velocity field. One should see the developing parabolic flow field in the channel plus on the boundaries near the contact line there should be a region with non-zero velocity. Here we see evidence that the wetting line models are working.

Back to waiting. Another look at the exodus file and the fill contours shows that the interface has crept slightly towards the square feature. Hmmmm, it seems this is going to take a long

time. Its time to find something else productive to do and let this problem go. Come back tomorrow and lets take a look at what's happened.

Well, we're back and it's a beautiful day, but never mind that how did our computation do? If all went well, we should be able to follow the interface as it swept through the entire domain. For this particular parameter set in this geometry, it should be the case that the upper wetting line traveled down the channel, turned the upstream corner, wetted up the wall, encountered the upper horizontal surface and at that point accelerated rapidly drawing fluid into the feature behind. The result should be a completely filled feature.

However, it is also possible that some time during the night the computation went south. Often this is manifested by the variable time step algorithm driving the time step to very small values at which point Goma quits. This we refer to jokingly as the level set "death spiral." In the current version of the algorithm it is nowhere near as common an occurrence as it used to be, but it seems to be related to the appearance of very low time scale events in the velocity field which are trying to be resolved. One potential solution to this is to use this card:

```
Minimum Resolved Time Step = <float>
```

in the time stepping section. This sets the smallest time step that the analyst feels represents the smallest time scale for physical events. Any time step smaller is assume to be in the domain of numerical noise. Hence, as the computation proceeds the time step controller will not reject a time step taken at a Δt less than this value regardless of what the error norms indicate. The assumption is that the error norms are reflecting only numerical noise and not physics. Admittedly, this could be a risky strategy because explicit schemes (as we are using here) are prone to dramatic instability if the time steps are not kept small, so care should be executed in using this card.

Be that as it may, we are currently in the position of having a half-completed computation that hit the death spiral some time in the wee hours of the morning. What to do ? The procedure that follows works the majority of the time:

1) copy the output exodus file to another file name
2) rename the output exodus file to contin.exoII
3) change the FEM file in the input deck to contin.exoII
4) change the initial time in the input deck from zero to the last time step in contin.exoII
5) comment out the Surfaces initialization method in the input deck
6) uncomment the card:
   `Level Set Initialization Method = Exodus`
7) Make certain the `Force Initial Level Set Renormalization` card is set to yes.
8) Restart the computation.

Generally, thing will pick up where they left off and proceed smoothly for quite some time afterwards. Although, it might proved necessary to repeat this restart procedure a few more times to achieve a final result.

The final result for this computation is shown in a sequence of pictures in the following figure: