Sandia National Laboratories

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

Albuquerque, New Mexico 87185- 0834

date: July 31, 2005
Original Feb 25, 2001; Revised Feb 27, 2001, DAL; Revised Jan 17, 2003, TAB; Revised July 2005, PRS, TAB;

to: Distribution

from: T.A. Baer, MS 0834, Org. 9114; Randy Schunk, MS 0834, 9114;

subject: Tutorial on Level Set Interface Tracking in GOMA (GT-020.3)

keywords: Level Set, Interface Tracking

input records: Level Set Interface Tracking, Level Set Length Scale, Level Set Renormalization Method, Level Set Renormalization Tolerance, Level Set Renormalization Frequency, Level Set Initialization Method, SURF, EQ, VOLUME_INT, Density, Navier-Stokes Source

Introduction

This memo is intended to accomplish two objectives: 1) serve as an introductory document to the very powerful technique of level set interface tracking, 2) provide a tutorial to new users on how to set up and run a couple of simple interface tracking problems using the level set algorithms currently (07/05) in GOMA. While we cannot promise that changes might occur to the algorithm or it's user interface which might put this document at odds with the current implementation, we feel at this point the LS capability is fairly stable. The implementation has changed somewhat since the first version of this memo dated back in 2001. The algorithm is more explicit, the surface tension terms are handled differently, and the robustness of GOMA has improved. We strongly encourage you to discard all old versions of this tutorial.

Theory: Level Set Embedded Interface Tracking

Interfaces occur rather frequently in many different problems. Free surface flows, melting/solidification, bubble dynamics, film growth, flame propagation, deposition are all problems in which a distinct boundary between regions of very different properties or even physics moves or evolves in time. Capturing the location of this interface and coupling its own distinctive physical properties to the other regions in the domain is a very big problem in computational science. For problems in which the interface can assume a complex shape, even to the extent of breaking up or, the converse, coalescing, a very useful technique for following it is an embedded interface method. In this class of method, a representation of the interface is tracked over time through a fixed mesh. The motion of the representation being a function of potentially other fields in the domain, for example, the velocity field of the fluid phases when tracking the interface between them as they flow through a geometry. Typically, the location of the interface is often abstracted within its representation and must be constantly reconstructed. This is one of the costs of using this class of

methods. In addition, because of this abstraction, properties that are associated with interfaces, such as surface tension or heats of melting, become problematic to apply without introducing some degree of spatial averaging in a region close to the interface.

Level set embedded interface tracking is a subset of these methods. It begins with the assumption that there is a smooth function $F(x,y)$ which is related to the interface location by the fact that the interface is the curve (or surface for three dimensional problems) in space that solves:

$$F(x, y) = 0 \qquad \text{(EQ 1)}$$

If the value of F is taken as height in the z direction, it is clear that the interface coincides with the zero level contour of the function F. This is the origin of the name "level set methods." The level set function, F, will evolve over time, but it is always required that its zero level contour will be the location of the interface. This is the nature of the level set abstraction of the interface: the explicit location of the one-dimensional interface is replaced by finding the zero contour of a two-dimensional function. This is done as needed by solution of Eq. (1).

The level set function, F, is a smooth function in the sense that its gradient in a finite region around its zero contour is continuous. This is an important feature because, among other things, the normal vectors to the interface and its curvature are thus well defined and readily determined. This property also has benefits when it comes to evolution of the level set field, as we'll see. This requirement in addition to Eq. (1) are the only constraints on F. We are free to choose it as we please. Indeed, far from the interface the actual nature of F becomes irrelevant. For simplicity of initialization (as much as anything), we have chosen to define F as the signed minimum distance from the interface. That is, at any point (x,y) the absolute value of F is the smallest distance from (x,y) to the interface. The sign of the level set function is based upon which side of the interface the point is on, which in turn is up to the analyst. Figure 1 shows how a circular phase boundary (1a) would be represented by such a level set function. Inside the circular region, the level function is positive, outside it is negative, so if the value of the level set function is taken as height above (or below) the z=0 plane, a picture of F emerges in Figure 1b. The zero contour of this function is the location of the interface. It should also be noted that by choosing F to be a distance function, the magnitude of its gradient vector should be unity at all points.

A static representation of an interface is of little interest. What is needed is a means to evolve the representation and the interface along with it. This is done by assuming a velocity field $u(x,y)$ which, like the level set function, is equal to the actual interface velocity at each point on the interface. For the case of tracking the boundary between two fluid phases, this velocity field is simply the velocity fields in the two fluids themselves. For problems of flame propagation or melting/solidification, however, the choice of velocity field is somewhat more problematic. We will not consider these latter situations in this tutorial.

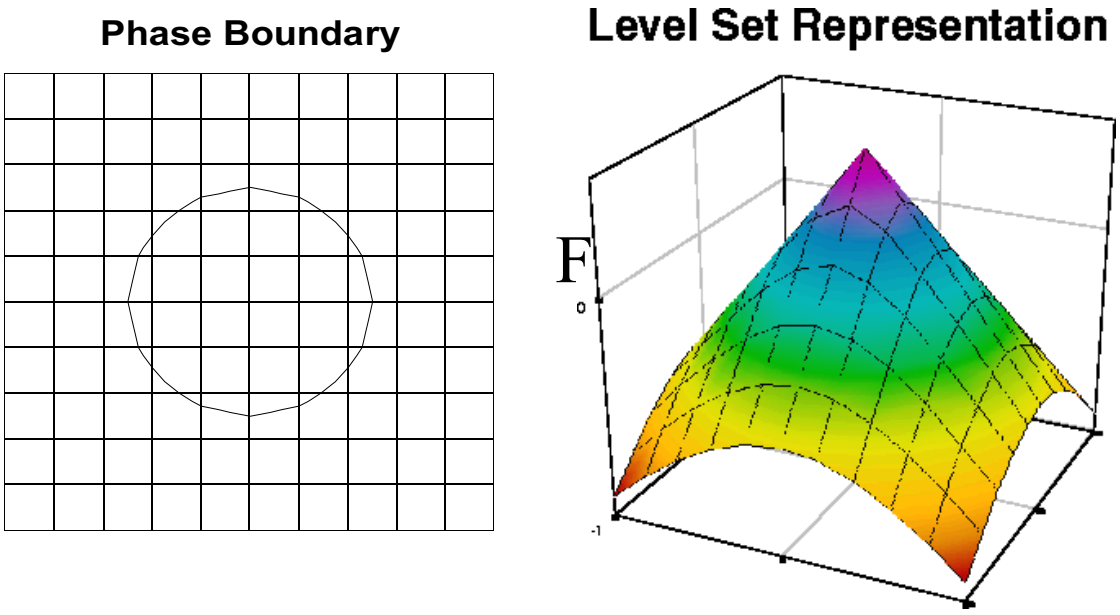

Figure 1. Example circular interface and corresponding level set representation

Evolution of the level set function can be accomplished by solution of a simple advection equation:

$$\frac{\partial F}{\partial t} + u \cdot \nabla F = 0 \qquad \text{(EQ 2)}$$

The reader recognizes this as a purely hyperbolic equation with characteristics being particle pathlines. Put another way, if the velocity field represents the motion of a fluid, each particle of fluid will retain the same value of F throughout the evolution (barring any dispersion that might result from approximation error). More importantly the fluid particles that start at the interface where F = 0 will retain this same value throughout. Since it is the location of these fluid particles that define the location of the interface (barring any mixing), finding the zero level contour of F at all times is equivalent to locating the interface. This is, of course, the requirement we imposed on the evolution scheme in the first place.

Solution of this equation for evolving the level set function is exactly what is done to evolve the color function in volume-of-fluid techniques. But here is where our smooth level set function helps us. Solution of this equation is done using the finite element method on a finite element grid. Finite element methods are not renowned for the applicability to purely advective equations. It has been our experience, however, that solving from smooth initial datum, as we are in the level set method, can be done using the finite element method without introduction of tremendous dispersion. This, of course, is not the case for VOF methods where advection of a discontinuous function via FEM is at best highly dispersive.

As noted above, if we restrict ourselves to computing the motion of an interface between two incompressible fluids, the velocity field used to evolve the level set function can be determined by coupled solution of the mass and momentum conservations equations for both fluid phases. In the current GOMA implementation, we solve a single set of equations for both phases but require that the material properties are functions of the value of the level set function. For example, we might say that in regions where the level set function is negative the fluid density is ρ_-, in positive regions the density is ρ_+. We assure a smooth transition between the two values across the zero level set contour by using, like Sussman, the following functional form for the density:

$$\rho(F) = (\rho_+ + \rho_-)/2 + (\rho_+ - \rho_-)\sin(\pi F/2\alpha)/2, \qquad -\alpha < F < \alpha \tag{EQ 3}$$

When F is outside the range $(-\alpha,\alpha)$, the density is constant at either ρ_+ or ρ_-. A similar relationship is used to represent the viscosity. The parameter α gives the length scale over which the transition occurs. The velocity field is determined by solution of a fluid momentum equation coupled to an incompressibility relation:

$$\rho(F)\frac{Du}{Dt} = -\nabla P + \nabla\bullet(\mu(F)\nabla u) + \rho(F)\vec{g} \tag{EQ 4}$$

$$\nabla\bullet u = 0 \tag{EQ 5}$$

It may seem strange that the velocity field follows a solenoidal restriction when the density is a clearly variable function of position, but because the level set function evolves as it does, it can be shown that the velocity field should remain solenoidal.

Using the velocity field of the fluids themselves to advance the level set function introduces a difficulty. Whereas, the level set function used might commence as a signed normal distance function from the interface, there is nothing inherent in the fluid velocity field which would preserve this property, that is, over time F will evolve into something that is not a distance function. Overall this is not necessarily a problem, since the actual value of F is more or less irrelevant away from the zero level set. However, if the situation is allowed to develop too far, it is possible for large gradients to appear in the level set function. Sharp gradients tend to degrade the accuracy of any numerical method devoted to solving the advection equation with the result that location of the interface determined from F in these regions is likely to be in error.

A solution is to periodically renormalize the level set function back to a distance function. In GOMA, a gradient norm in a region close to the zero level set is determined. If F has deviated from a distance function, this norm will likewise differ from unity. If this deviation is too large it triggers a renormalization step.

There are a number of ways that F can be renormalized. We have developed a simple algorithm approach which is fast, robust and reasonably accurate. Elements which are crossed by the zero level set are identified. On each of these elements, line segments are defined connecting adjacent local nodes. The sign of the level set function at each node

is used to decide which of these line segments the zero contour crosses. An iterative method on the parameterized line between two nodes is solved to precisely locate the point where the line segment intersects the zero level set contour. This is done for all appropriate line segments in the element and all elements that contain the zero level set contour. The result is a list of physical points that are on the zero level set contour. In some sense, this list represents a discretization of the interface curve.

This list is used to renormalize the level set function. The updated function value at a point is the smallest distance found between the point in question and all the points on the zero level set contour list. The sign at a point is unchanged before and after renormalization. Because the values obtained are distances from fixed points, the gradient magnitude of the level set function is returned to approximately unity.

As noted above inclusion of interface-related phenomena within an embedded interface scheme is somewhat problematic. This is certainly the case for inclusion of surface tension effects. In our current algorithm, the surface tension forces are present, not strictly as interfacial forces, but instead spread out as a body force in a relatively narrow region adjacent to the zero level set. The following body force tensor is added to the fluid momentum equation:

$$T_\sigma = \sigma \delta_\alpha(F)\left(I - \frac{\nabla F \nabla F}{|\nabla F|^2}\right) \qquad \text{(EQ 6)}$$

The term $\delta_\alpha(F)$ is a "smooth" Dirac delta function in the sense that it has the same area of a conventional delta function (namely, unity) but that area is spread over a finite region around zero of width α. The functional form for it is as follows;

$$\delta_\alpha(F) = \begin{cases} (1 + \cos(\pi F/\alpha))/(2\alpha), & |F| \le \alpha \\ 0, & \text{O/W} \end{cases} \qquad \text{(EQ 7)}$$

This method for applying surface tensions is reasonably convenient, but it can lead to problems, especially in cases of large surface tension and low viscosity and/or density in one of the phases. We have implemented a host of workarounds to this problem, including element enrichment strategies and subelement integration, etc. to help achieve accurate solutions and sharper interfaces. The problem is that the presence of this "smeared" force in regions close to the interface can result in non-physical motions of the fluid least able to resist such forces. The sharpening techniques and integration techniques have mitigated these problems somewhat. Discussion of those is beyond the scope of this report. The report does make mention of some of these strategies in the examples.

Examples

We next present two examples of simple level set calculations to showcase input deck and material file cards that are required.

Rising Bubble

The appropriate files for this test problem should be in a directory entitled "bubble". The bubble rise is a very simple application of the level set method. It is a model of a bubble gas rising in a liquid column. It illustrates some of the basics of the level set algorithm without introducing complicating factors such as contact and wetting line methods. Also by choice of the surface tension parameter it can be used to illustrate effectively the detrimental effects of distributed surface tension forces and the applications of remedies such as the extended finite element method.

The domain can be view by examing the file **`bubble.exoII`** with **`blot`**. It is a simple rectangulare region with the lower boundary (sideset 1) lying on the y = 0 plane. We are going to solve this problem as a 2D axisymmetric problem so this is a requirement. The dimensions of the domain are 5 cm x 10 cm.

We begin by looking at the input deck for this problem. Edit the file **`bubble.inp`**. For the most part, this file is boilerplate Goma-speak. The differences are mostly related to application of the level set method. We shall discuss some of these differences.

The first them is the level set-specific time integration method, specified on the following card:

```
Fill Weight Function                = Explicit
```

This card essentially lags the velocity/pressure field information used to update the level set function by one-time step. As one might guess, this is a purely explicit form for the time integration method of the level set equation. Note that it applies only to the level set function. The other fields in the problem are still being integrated by Goma's implicit time integrator.

Accompanying the explicit time integration method of the level set function are new restrictions on the time step imposed by the Courant number condition. This requirements are imposed in addition to the standard variable time step control algorithm by including the following card:

```
Courant Number Limit                = 0.1
```

In this case the parameter 0.1 controls the size of the time step. The smaller it is, the smaller the allowed time steps will be. The value 0.1 seems to offer good performance.

The level set length scale is set on this card

```
Level Set Length Scale = 0.25
```

The 0.25 value corresponds to directly to the element size.

An important part of solving the a problem using the level set method is initializing the level set field. In this tutorial example, we demonstrate initialization using the **`Surfaces`** method. This method uses primitive geometric objects (spheres, circles, planes etc.) which have analytic expressions for the minimum distance from a point to the object. In this case, we are using the CIRCLE primitive with radius of 1.5 and center at (-1,0).

```
Level Set Initialization Method  = Surfaces 1
```

```
SURF = CIRCLE -1.0 0 1.5
```

The integer following the Surfaces indicates that the intialization will involve only one geometric primitive. This integer may be more than 1 in which case that number of **SURF** cards with appropriate geometries must be included.

Renormalization of the level set function is a requirement for a problem involving such large motion of the interface. The following card sets the renormalization method to "**Huygens_Constrained**" which does a pointwise interface reconstruction and renormalizes using a Lagrange multiplier constraint that minimizes mass loss.

```
Level Set Renormalization Method = Huygens_Constrained
```

The frequency of renormalization is best decided by the quality of the solution. This is done by setting a renormalization tolerance as done on this card

```
Level Set Renormalization Tolerance = 0.25
```

What is done here is to compute the deviation of the average magnitude of the level set function near the zero level set contour. Theoretically, the value of this should be unity. The tolerance sets how far away from one this value is allowed to stray prior to renormalization. In this case, this error measure is allowed to deviate between 0.75 and 1.25 without renormalization be triggered. Renormalization frequency can be increased by decreasing this tolerance and of course the converse is also true.

It is also possible to require renormalization to occur every n time steps by setting this value on the following card:

```
Level Set Renormalization Frequency = -1
```

In this cases, setting the value to -1 implies that the frequency of renormalization is dependent solely on the renormalization tolerance.

The following card invokes a more accurate integration method for source terms located at the interface. In this case, it uses a subgrid integration scheme with the smallest grid size one-eighth of element size (2^2).

```
Level Set Subgrid Integration Depth = 2
```

Note we are using a direct solver based upon a multifrontal method:

```
Solution Algorithm                  = umf
```

Boundary conditions

The boundary conditions for this problem are quite straightforward. Basically, the transverse velocities on the two constant y nodesets (1 and 3) are set to zero. On nodeset 4, which is the boundary at x = 0, both velocity components are set to zero making this boundary a no slip surface.
The last boundary condition of interested is

```
BC = LS_CAPILLARY LS 0
```

This "boundary condition" is actually applies the surface tension source term to the fluid phase at the zero level set contour. It is something we have taken to calling and "embedded" boundary condition. The integer parameter on this card allows can take the values of -1, 0, or 1. For problems that use subgrid integration and non-zero values of the level set length scale, this parameter should always be set to zero. Problems that employ subelement integration conventionally set the level set length scale to zero. In these cases, the user must choose which phase the surface tension forces should be applied to. This is done by setting the third parameter to +1 or -1.

Equation Specs

The equation specification section of the input deck is relatively straightforward Goma-speak, save for the pressure interpolations.

```
EQ = momentum1   Q2 U1 Q2   1 1 1 1 1 0
EQ = momentum2   Q2 U2 Q2   1 1 1 1 1 0
EQ = continuity  Q1_XV P Q1_XV   1          0
EQ = level_set   Q2 F  Q2   1 1      1
```

In this case, we are employed the Q1_XV interpolation for pressures. This interpolation order is an extended finite element interpolation method which includes additional degrees of freedom that allow for representation of step changes *across the zero level set contour*. In cases in which surface tension forces are very significant, this interpolation order is critical to stable solution. In the problem presented here the capillary is approximately 2 so that using extend finite element interpolation is not as critical, but it is included here for illustration purposes

Materal File

First, note the use of the aprepro definitions and assignments to clearly identify the values of density and viscosity values be assigned to the appropriate phases. This is of course good practice.

The material file for this problem illustrates use of the **`Second Level Set`** syntax for specifying the different level set phase properties. Consider the cards that are used to set density and viscosity:

```
Density= CONSTANT  {rho_fluid}
Second Level Set Density = CONSTANT {rho_gas} NEGATIVE

Viscosity= CONSTANT{mu_fluid}
Second Level Set Viscosity  = CONSTANT {mu_gas} NEGATIVE
```

The first card for both of these properties is the conventional method that the material file specifies a property. In this case, we are specifying constant values of viscosity or density , but this need not be the case. Any of the other non-linear models present in Goma can be used in this setting. The indexing with respect to the level set function is invoked by the prescence of the **`Second Level Set`** card following the original material model card. These imply that the property in question is to vary according to the sign of the level set function. Indeed, the keyword on this card tells Goma that value immediately preceding the string will be applied to appropriate signed phase. In this case, the

negative side of the interface. The values that will be applied to the other side of the interface will be found via the model specified on the first card. Again in this case, this is a constant value, but it could also be from a non-linear viscosity or density model, for example.

The momentum source from gravitation is specified using the same type of syntax:

```
Navier-Stokes Source= CONSTANT  {-rho_fluid*980.} 0.0 0.0
Second Level Set Momentum Source =
                CONSTANT {-rho_gas*980.} 0.0 0.0 NEGATIVE
```

Note that we are specifying the actual momentum source vector and not just the gravitational vector. This is in contrast to the other method for setting the gravitational body using using the **LEVEL_SET** momentum source model.

Running the Problem

The problem is started by invoking **goma** from the command line thusly:

```
% goma -i bubble.inp -a
```

Unlike ALE problems, intervention by the user occurs much less often for LS problems. After the first, time step has completed it is always good practice to the blot the exodusII file to verify that the initial location of the level set function is what was anticipated. Further, one should also plot the velocity vectors to assure that boundary conditions are also specified correctly and indeed that the bubble appears to moving in the right direction. It is a fairly common error to swap density or viscosity values for the phases in which case the bubble would sink instead of rise. It always creates a good deal of frustration to return in the morning to a computation started the night before only to find that things are proceeding in the opposite direction to what was anticipated. It is also important to note while looking at the velocity vectors at this point that they are relatively smooth and of uniform size. That is to say, parasitic currents are not a significant problem.

Fortunately for us, the size of the bubble is large enough that the pressure rise across the capillary surface is not significantly larger than the hydrostatic pressure gradient along the length of the bubble. Consequently, including the extended finite element interpolation functions is not so critical for robust solution. If the bubble, however, were one-tenth the size, this would not be the case. In the current problem, the extended enrichment interpolations have the most benefit at the very start of the problem as the bubble is beginning to accelerate upwards. At this time the velocities are still small and as a consequence the instanenous capillary number is relatively small. However, as the computation continues the bubble accelerates and viscous and even inertial effects start to dominate surface tension forces.

Actually, this tutorial problem is quite sizeable and takes a considerable amount of time to run out. Overnight on a fast processor is typical. We have extract four time planes to illustrate the type of behavior that should occur. Note the considerable deformation of the bubble and the shedding of the smaller bubble into the trailing vortex.

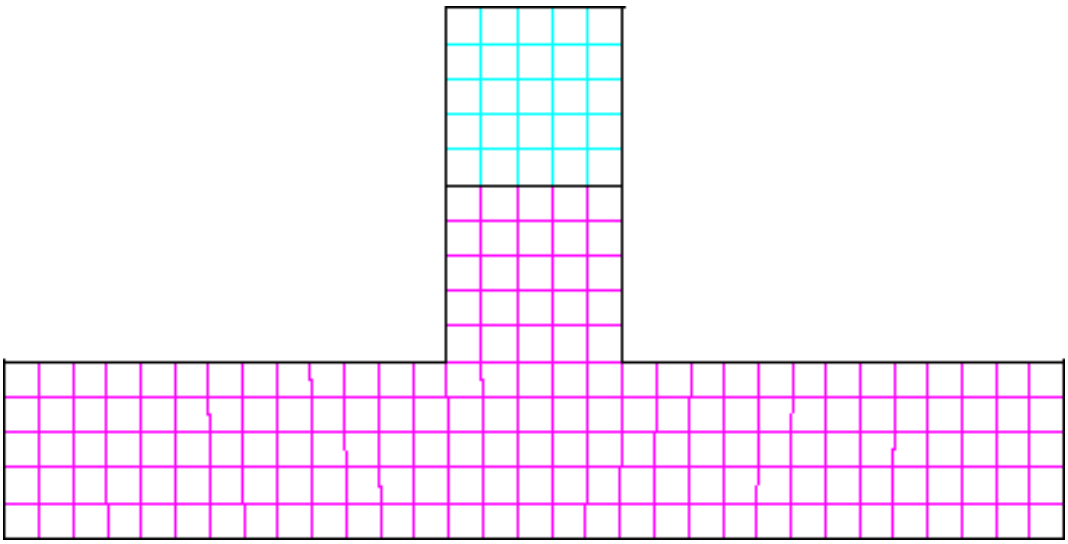

Figure 2. Simple slot flow geometry

Simple Slot Coater

The simple slot problem represents a application that is quick to run yet illustrates the requirements for including wetting line motion and contact using the level set method. The geometry can be viewed by "blotting" the file **`simple_slot.exoII`**. It consists of a "T" shaped geometry (see figure). Fluid is introduced from the top through sideset 1 eventually contacting the lower boundary, sideset 40, which is moving from left to right. Wetting line dynamics occur on the lateral surfaces, sideset 10 and 20 and contact occurs on sideset 40. The two vertical boundaries, sideset 50 and 60, are assumed to be fully-developed flow boundaries..

The mesh consists of two blocks, labeled 1 and 2. The same material files are applied to both blocks so there is no difference in material between them. The purpose of defining the mesh in this way is entirely for the initialization of the level set method. Nodeset 2 is at the boundary between the two block as this nodeset will mark the starting location of the level set interface. Note that instead of initializing the level set function in this way, a SURF = PLANE geometry initialization primitive could have been defined.

Edit the input deck, **`simple_slot.inp`**. Most of the cards in this file are fairly standard Goma fair. We shall only discuss some of the level set specific cards.

Level Set Method Cards

The first is the level set time integration method, specified on the following card:

```
Fill Weight Function                    = Explicit
```

This card essentially lags the velocity/pressure field information used to update the level set function by one-time step. As one might guess, this is a purely explicit form for the time integration method of the level set equation. Note that it applies only to the level set function. The other fields in the problem are still being integrated by Goma's implicit time integrator.

Accompanying the explicit time integration method of the level set function are new restrictions on the time step imposed by the Courant number condition. This requirements are imposed in addition to the standard variable time step control algorithm by including the following card:

```
Courant Number Limit                    = 0.1
```

In this case the parameter 0.1 controls the size of the time step. The smaller it is, the smaller the allowed time steps will be. The value 0.1 seems to offer good performance.

The level set length scale is set on this card

```
Level Set Length Scale = 0.2
```

The 0.2 value corresponds to directly to the element size.

The initialization method is spec ified on the following card as noted above

```
Level Set Initialization Method  = Nodeset NS 2 EB 1
```

This card requires two element blocks separated by a nodeset (nodeset 2 in this case). The nodes of the node set serve as a set of points on which to define the initial level set field. The string **`EB 1`** indicates that the values of level set function computed for nodes in element block 1 will be positive (those in block 2 will be negative).

Renormalization of the level set function is a requirement for a problem involving such large motion of the interface. The following card sets the renormalization method to "**`Huygens_Constrained`**" which does a pointwise interface reconstruction and renormalizes using a Lagrange multiplier constraint that minimizes mass loss.

```
Level Set Renormalization Method = Huygens_Constrained
```

We recommend this renormalization option, even though others are available.

The following card invokes a more accurate integration method for source terms located at the interface. In this case, it uses a subgrid integration scheme with the smallest grid size one-eighth of element size (2^3).

```
Level Set Subgrid Integration Depth = 3
```

Note we are using a direct solver based upon a multifrontal method:

```
Solution Algorithm                      = umf
```

Boundary conditions

The inflow over sideset 1 is applied using the notorious GD_# type of conditions

```
BC = GD_LINEAR SS 1 R_MOMENTUM2 0 VELOCITY2 0 0.0 1.0
BC = GD_PARAB SS 1 R_MOMENTUM2 0 MESH_POSITION1 0 1.5 0. -6
```

```
BC = U NS 1 0.0
```

The user manual contains many nice examples on how to set these boundary conditionis up. The requisite boundary conditions for a boundary already possessing a wetting line are illustrated by those applied to sideset 10:

```
BC = VELO_SLIP_LS SS 10 0.3 1.e-5  0. 0. 0. 1.e-8
BC = WETTING_SPEED_LINEAR SS 10 45.0 0.1 0. 0.001 0. 0. 0.
BC = VELO_NORMAL SS 10 0.0
```

The first is the **VELO_SLIP_LS** condition which in this context is actually used to impose the no-slip boundary condition, but to a varying degree depending upon proximity to the wetting line. The first parameter on this card defines this proximity. This is the thickness of the region around the wetting line in which the value of slip parameter, specified by the second parameter on the card, is applied. The third, fourth and fifth parameters is the velocity vector of the boundary (in this case it is not moving). The last parameter on this card is the value of the slip parameter applied on the boundary in the regions that are not near the wetting line. As one can see, this boundary condition applies a nearly no-slip condition (1.e-8) away from the wetting line, but this requirement is relaxed by three orders of magnitude near the wetting line.

The second card needed to apply wetting line motion is the **WETTING_SPEED_LINEAR** card. This card applies a wetting force in the region near the contact line that is related to the value of the static contact angle. This contact is the first parameter (in degrees subtended) on the card. The second parameter is a proportionality constant between the wetting line velocity and the difference of the cosine of the actual contact angle and cosine of the static contact angle. The third parameter is reserved for expansion and is not used. The fourth parameter is a distance scale value that issimilar in form to the slip coefficient appearing in the **VELO_SLIP_LS** boundary condition. The smaller this parameter, the larger is the magnitude of wetting line force applied. The last three parameters are once again the components of the boundary velocity.

Finally, the **VELO_NORMAL** boundary condition is employed to explicit enforce no penetration on these boundaries.

In the case of the contact boundary, sideset 40, certain modifications are needed to allow for the contact to proceed.

```
BC = VELO_SLIP_LS SS 40 0.3 1.e1  2. 0. 0. 1.e-8
BC = WETTING_SPEED_LINEAR SS 20 45.0 0.1 0. 0.001 2. 0. 0.
BC = VELO_NORMAL SS 40 0.0
```

Obviously, one can see the addition of the substrate velocity (2,0,0) in the **VELO_SLIP_LS** and the **WETTING_SPEED_LINEAR** boundary conditions. Note also however that the near-contact-line slipping parameter has been significantly increase (by six orders of magnitude). This permits significant slip to appear on this boundary as the level set interface approaches but prior to actual contact. It is this slipping that allows for the hydrodynamic forces in the problem to evacuate the gas liquid layer between interface and boundary which in turn allows for contact to occur.

The last boundary condition of interested is

```
BC = LS_CAPILLARY LS 0
```

This "boundary condition" is actually applied the surface tension source term in the fluid phase at the zero level set contour. It is something we have taken to calling and "embedded" boundary condition. The integer parameter on this card allows can take the values of -1, 0, or 1. For problems that use subgrid integration and non-zero values of the level set length scale, this parameter should always be set to zero. Problems that employ subelement integration conventionally set the level set length scale to zero. In these cases, the user must choose which phase the surface tension forces should be applied to. This is done by setting the third parameter to +1 or -1.

Equation Specs

The equation specification section of the input deck is relatively straightforward Goma-speak, save for the pressure interpolations.

```
EQ = momentum1   Q2 U1 Q2   1 1 1 1 1 0
EQ = momentum2   Q2 U2 Q2   1 1 1 1 1 0
EQ = continuity  Q1_XV P Q1_XV   1          0
EQ = level_set   Q2 F  Q2   1 1      1
```

In this case, we are employed the Q1_XV interpolation for pressures. This interpolation order is an extended finite element interpolation method which includes additional degrees of freedom that allow for representation of step changes *across the zero level set contour*. In cases in which surface tension forces are very significant, this interpolation order is critical to stable solution. In the problem presented here the capillary number is approximately 2 so that using extend finite element interpolation is not as critical, but it is included here for illustration purposes

Materal File

The material file for this problem is for the most part conventional Goma expect for the level set specific models for density, viscosity and fluid momentum source:

```
Density  = LEVEL_SET  0.05 0.0005 {0.1}

Viscosity  = LEVEL_SET  1.0 0.0005  {0.1}

Navier-Stokes Source  = LEVEL_SET    0. -20.0   0.0
```

In the case of the first two, the first two parameters are the values of the density or viscosity in the negative phase and positive phase, respectively. The third parameter is a length parameter which defines the thickness over which the transition between these values occurs. If it is set to zero, this thickness defaults to the the level set length scale specified earlier. The Navier-Stokes source cards

allows the user to define the components of the gravitational vector. This vector will be multiplied by the local density value internally to give the fluid body force. Note that the Density model must be **LEVEL_SET** and the momentum source model must also be **LEVEL_SET** for this multiplication to occur.

Running the Problem

Unlike ALE, problems that employ the level set method tend to be more "fire and forget." Launch this problem by invoking goma with the command line

```
% goma -i simple_slot.inp -a
```

After a few time steps have elapsed, **blot** the file **out.exoII**. Verify that the zero level set contour has been initialized properly by contouring the variable **F**. The zero contour should lie very near the boundary between the element blocks. It is also interesting at this point to look at the velocity field. Using the vectoring command in **blot**, examine the hedgehog plot for velocities. If things are operating properly the left to right motion of the substrate should be the dominant feature in the lower part of the geometry. The downward flow in the downcomer should appear clearly. Also the wetting line motion should be evident from velocity vectors on the vertical "no-slip" boundaries in the vicinity of the zero level set contour.

A little while longer check back and contour F again, and observe that the fill front is making progress down the inlet port and also that it is starting to outpace the wetting line. This is a consequence of the parameter choice on the wetting boundary conditions. By decreasing the length scale parameter on the **WETTING_SPEED_LINEAR** boundary condition card or increasing the near-contact-line slipping parameter, it might be possible restart the problem and have the wetting line move faster and keep up with the rest of the filling front. One must exercise care in this however since this adds momentum to problem and involves something of a balance between viscous and wetting line forces. If this balance is exceeded too much in the direction of the wetting line forces, the result can be what are essentially parasitic currents in the vicinity of the wetting line and a slow-to-advance computation.

As the computation approaches approximately two seconds, the filling front have entered the horizontal channel and is being swept downstream. Eventually, one will see the filling front contact the lower boundary. Once this has happened, one school of thought suggests that the computation should be stopped and restarted with the slipping parameter on the **VELO_SLIP_LS** card associated with the sideset 40 should be changed to a much smaller value since the large value was used only to ensure that contact occured. Be that as it may, we shall let to computation continue undisturbed. Eventually, we should see the upstream meniscus creep upstream along the upper surface, the downstream menicus will be swept more quickly downstream until it exits. On the substrate, the the free surface should periodically make contact with the moving substrate. Each of this events will create a bubble of light phase which is then carried out of the domain by the moving substrate. Seen over several cycles this gives the impression that we are able to model the complicated air entrainment process. As experienced analysts, we are all well aware that the models we are using for contact do

not possess that sort of physics. However, it does make a fine animation with which to dazzle those charged with distributing resources.

Summary

This memo should give the user a basic understanding of the level set method and the requirements for setting up a solution of this class in goma. Although a very powerful method for solving problems with interfaces and interfacial phenomena, the user is reminded of the caveats associated with the current implementation. The implementation is likely to change as better ways of doing things are introduced. This implementation is also new and relatively untested. Rigorously correct simulations should not be expected from it. Indeed, this algorithm should be considered in the same category as beta software releases: friendly users are encouraged to use it and should report problems and suggestions to the developers.

TAB:9114:tab

PRS:9114:prs