Sandia National Laboratories

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

Albuquerque, New Mexico 87185- 0834

*date:* December 8, 2003

*to:* Distribution

*from:* T.A. Baer, MS 0834, Org. 9114

*subject:* **New and improved level set tutorial in which many of the additions developed in 2003 are detailed and described so that the uninitiated person might be enlightened in their use.**

## Introduction

The original level set method for tracking interfacial boundaries through static meshes has seen a great deal of enhancement during 2003. A great number of these were directed solely at improving the performance of the method at tracking fluid/fluid interfacial boundaries in the presence of large surface tension forces. This tutorial will serve the multiple purposes of documenting the mathematical form of these changes, discussing the manner in which they can be invoked through the input deck file, and provide a tutorial problem as a concrete example that employs most of the new features.

## New additions for 2003

The following is a list of added enhancements to the level set interface tracking method in *Goma*

1) Near-interface variable field enrichment via *i)* the extended finite element formulation, *ii)* the ghost field formulation, *iii)* variable field enrichment via Hughes-type element enrichment

2) More accurate integration methods for the near-interface region via *i)* recursive, non-conforming subgrid element division (subgrid integration), *ii)* non-recursive, conforming subelement element division (subelement integration).

3) New formulation for the surface tension momentum source term based upon a new normal vector field obtained via least squares projection of the level set gradient vector.

4) Extensional velocity computation via the "orthogonality" constraint for renormalization-free level set advection.

5) Instrumentation of *Goma* non-linear material models for use with level set interface tracking.

6) Strongly set velocity "embedded" boundary conditions.

7) A "sharply-set" wetting line force boundary condition.

## Theoretical Descriptions

In this section, we will present the mathematical descriptions of some of the enhancements mentioned above.

### Field enrichment:

One of the suspected sources of the spurious currents, the so-called "parasitic" currents, is the mismatch between the pressure field and the surface tension source terms. One of the chief sources of this mismatch is the inability of the standard Galerkin FEM shape functions being able to represent accurately the rapid change in pressure found in the vicinity of an interface surface possessing significant surface tension. Very often substantial oscillations in the pressure field occur at these points. Since the source term field in general is smoother than the pressure field, it does not oscillate, and it is theorized that this mismatch accounts for a larger portion of the parasitic current activity.

New techniques are evolving for locally enriching FEM shape functions so they can present richer behavior as required. One of these is the extended finite element (XFEM) approach. A one-dimensional prototype of this method demonstrated that it could permit an exact representation of a pressure step change *within an element* without oscillations. We implemented a higher-dimensional version in *Goma*.

Briefly, the XFEM method identifies the elements through which the interface surface passes. The shape functions associated with the nodes that belong to these elements (the "interface" nodes) are enhanced by an extended shape function, $\Psi_j$, so that the interpolation in elements that share each of this nodes is given by:

$$T = \sum_i N_i T_i + \sum_j \Psi_j a_j \tag{1}$$

where the j index extends over the number of interface nodes shared by the element. Note the additional degrees of freedom that appear with the method, the $a_j$ coefficients. Note also that an element might possess these additional degrees of freedom even though the interface itself passes through a neighboring element. Finally, the support of $\Psi_j$ is restricted by decomposing it in the following manner,

$$\Psi_j = N_j g_j(\boldsymbol{x}) \tag{2}$$

where $N_j$ is the original FEM shape function and $g_j(\boldsymbol{x})$, the extending function, is a global function of position that possesses the desired enhancing behavior. For example, if the desire is to represent a

step change then $g_j(\boldsymbol{x})$ would be some version of a Heaviside function. In the examples, described below the form for $g_j(\boldsymbol{x})$ was

$$g_j(\boldsymbol{x}) = H(\phi) - H(\phi_j) \tag{3}$$

The XFEM method can also be used to introduced jumps in field gradients instead of field values. In this case, the extending function takes the form:

$$g_j(\boldsymbol{x}) = |\phi| - |\phi_j| \tag{4}$$

A separate method for local enrichment is the ghost fluid method. In this approach, interface nodes posses two potential values of the field unknown, $T^+$ and $T^-$. Field interpolation on the negative side of the interface is done using the negative nodal degree-of-freedom values while interpolation on the positive side is done using the positive nodal values, vis, for $\phi < 0$

$$T^- = \sum_j N_j T^- = \sum_j N_j (T_j + a_j H(\phi)) \tag{5}$$

From this form, you might recognize that it can be reorganized to fit the XFEM format and allow us to recognize that the two methods can be related with the extending function for the ghost fluid method taking the form:

$$g_j(\boldsymbol{x}) = H(-\phi \text{ sgn } (\phi)) \tag{6}$$

Indeed, numerical results indicate that XFEM value enrichment gives equivalent answers to the ghost fluid enrichment choice.

Finally, the Hughes-based element enrichment is a piecewise discontinous enrichment of elements that span the interface. There are versions that allow a discontinous value, gradient, or both. For an element containing a jump in value, the unknown field, T, would be interpolated thusly:

$$T = \sum_j N_j T_j + \left\lfloor H(\phi) - \sum_j N_j H(\phi_j) \right\rfloor \langle T \rangle \tag{7}$$

where $\langle T \rangle$ is the jump in field value in the element.

## Improved integration accuracy:

An additional reason identified for the presence of the spurious currents is related to the numerical integration rules employed in the finite element method. The body force terms used to apply surface tension terms in an embedded method typically do so using a smoothed Dirac function. This type of function, even a smoothed function, is not integrated accurately by the standard low order Legendre integration rules that are *de riguer* in the finite element method. Instead, we have implemented separate integration rules for interface elements which are to a certain extent adaptive to the location and shape of the interfacial curve.

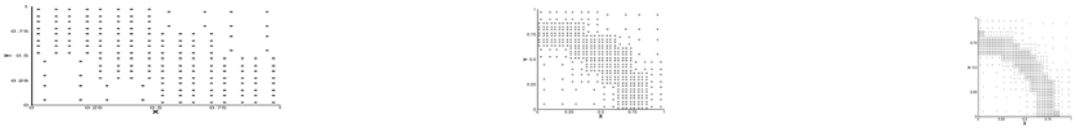

Figure 1. Distribution subgrid integration points around a circular arc centered at (0,0) with radius 0.75 for subdivision depths of 2, 3, and 4.

*Subgrid Integration: Recursive, non-conforming subdivision*

A simple idea that is easy to implement and relatively robust first divides each interface element into four (eight in 3D) equal-sized subelements that are aligned along the coordinate axes of the master element (these are referred to as subgrids). The subgrids through which the interface curve passes are subdivided once again. In this recursive fashion, the original element is subdivided into a set of subgrids until a given depth of division is reached. The Legendre integration points are then determined on each of the subgrids and all combined yield a large number of integration points that conform more and more to the shape and dimension of the interface zone the deeper the subdivision is conducted. Figure 1 shows an example of three different subdivision depths of 2, 3, and 4 applied to a circular arc centered at (0,0) with radius 0.75.

*Subelement Integration: Non-recursive, conforming subdivision*

While subgrid integration may be robust it isn't very efficient, especially in three dimensions. Furthermore, it requires a non-zero interface thickness. A different integration method was also developed this year that is both more efficient and also can be applied to problems in which the interface thickness doesn't enter as a parameter, *i.e.* it can be set to zero. This has been termed subelement integration and involves, first, halving of the quadrilateral interface element into two triangles according to the orientation of the interface in the element. These triangles are then divided again into smaller triangles such that one of the sides of the smaller triangles lies (approximately) along the interface. The picture that emerges is sketched in Figure 2.

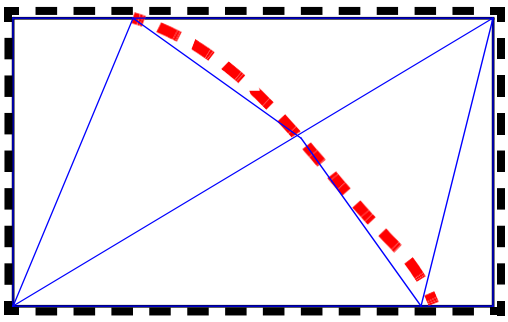

Figure 2. Sketch of subelement tesselation of interface element. Interface curve; dashed red, original element; dash black, subelement triangles; solid blue.

The advantage of this method is that subelement boundaries now directly conform with the interface contour. This permits direct integration of the line integrals that result in the level set formulations

when the width of the interface tends to zero, Hence an actual zero value for the level set interface width parameter is used when applying this integration method. The primary drawback to this method is that it effectively requires a reconstruction of the interface to apply. This is relatively simple in two dimensions, but grows in complexity for three dimensions. Therefore, the current implementation of this integration method is limited to two-dimensional problems.

## New Surface Tension Formulations:

The rational for reformulating the surface tension source terms in the momentum equation was based upon a recent paper by Renardy and Renardy [1]. The success that they report is derived from applying a higher order (quadratic) interpolation to the curvature of their interface representation (VOF in their case). While their ideas are not directly applicable to finite element analysis, they do suggest that curvature interpolation can have a large impact on magnitude of spurious currents. Toward that end, we sought to improve the effective smoothness of the curvature field used to apply the surface tension.

This can be done in a number of ways. The simplest to implement was to introduce an additional unknown field, the normal vector the level set field, into the problem. This vector, $\boldsymbol{n}$, can be computed by the process of least squares projection of the level set gradient to give the weak form:

$$\int N_i(\boldsymbol{n} - \nabla\phi)d\Omega = 0 \tag{8}$$

With this vector unknown present several new potential capillary source terms become available.

1) Divergence form:

$$\sigma\delta_\alpha(\phi)\nabla \cdot \boldsymbol{n} \tag{9}$$

2) Surface divergence form:

$$\sigma\delta_\alpha(\phi)\nabla_s \cdot \boldsymbol{n} \qquad \nabla_s = (I - \boldsymbol{nn}) \cdot \nabla \tag{10}$$

3) Improved curvature projection:

$$\sigma\delta_\alpha(\phi)\kappa \qquad \kappa = \nabla \cdot \boldsymbol{n} \tag{11}$$

4) Conventional form:

$$\sigma\delta_\alpha(\phi)\kappa \qquad \kappa = \nabla^2\phi \tag{12}$$

The primary difference between the third and fourth form is that in the latter the curvature is obtained from the second derivative of the level set function; a quantity that for quadratic level set interpolation is approximately piecewise constant. Whereas, in the third formulation the curvature is obtained via the divergence of a piecewise continuos field and therefore should result in a smoother curvature.

## Extension velocity projection:

The previous year saw introduction of the extension velocity. The extension velocity the velocity of the level set function normal to itself, that is, in the direction of its gradient and it is a scalar quantity. It can be be used to evolve the level set function in such a way as to avoid renormalization. In addition, it is indispensable for use in problems in which no global velocity field is present such as in film-growth and melting/solidification problems. Evolution of the level set function when an extension velocity field is present is via the equation:

$$\frac{\partial \phi}{\partial t} + e = 0 \tag{13}$$

where $e$ is the scalar extension velocity value. The extension velocity is tied to the fluid velocity field only at the interface where it is required to take the value of the local interface normal advance rate. At the current time, the implementation sets the extension velocity at the interface through an embedded boundary condition,

$$\int_I N_j(e - u \cdot n) dI \tag{14}$$

where $I$ is the interface curve and then solves for the extension velocity field elsewhere via finite element solution of the "orthogonality" constraint,

$$\mathrm{sgn}(\phi) \nabla e \cdot \nabla \phi = 0 \tag{15}$$

This constraint ensures that the lines of constant $e$ are perpendicular to lines of constant $\phi$ and consequently the level set function should not depart from a distance function as it evolves. The sgn($\phi$) multiplier is required so that characteristic information propagates outward from the interface curve [2]. While this multiplier is not required from a strict mathematical sense, it is required during the numerical solution process to ensure information is propagated in the correct direction. The boundary condition for $e$ at the interface is a form of a strongly integrated embedded condition.

## Instrumenting non-linear material models for level set tracking

*Goma* has a wealth of non-linear material models for viscosity, heat capacity, density, thermal conductivity, diffusivity etc. Previously, only models that had constant values for these parameters had been set up to transition between them when crossing the interface. What is desired is a means of doing this for any non-linear material model. We have a implemented a simple idea. Imagine that a material property is given in one phase by the material model $P_1(\alpha_1,\beta_1,\gamma_1,..;$ **v**,T,C..) and in the other phase by the model $P_2(\alpha_2,\beta_2,\gamma_2,..;$ **v**,T,C..) where $\{\alpha_1,\beta_1,\gamma_1,..\}$ and $\{\alpha_2,\beta_2,\gamma_2,..\}$ are the respective model parameters. Provided that both of these material models have finite values at all points within the problem domain a simple means to modulate the value of this property with respect to the level set function is:

$$P = P_1(\alpha_1, \beta_1, \gamma_1 \ldots; v, T, C)(1 - H_\alpha(\phi)) + P_2(\alpha_2, \beta_2, \gamma_2 \ldots; v, T, C) H_\alpha(\phi) \tag{16}$$

## Strongly set embedded velocity boundary conditions

Reconstructing the interface for subelement integration opens up the possibility of applying boundary conditions on velocities as line integrals much in the same way *Goma* conventionally uses the **VELO_NORMAL** boundary condition. For example, it is desired to set the fluid velocity along the interface curve to $(u_0, v_0)$. This can be done when using subelement integration by replacing the momentum equations of some of the nodes on interface elements with the following line integral constraints:

$$\int_I N_j(u - u_0)dI = 0 \tag{17}$$

$$\int_I N_j(v - v_0)dI = 0 \tag{18}$$

where $I$ is the interface curve. Subelement integration supplies line segments along the interface curve that allows for these line integrals to be approximately evaluated.

## Sharply-set wetting line force boundary condition

The conventional wetting line model used up to now in level set applications computes a wetting line force from the relation:

$$F_{wet} = \sigma(\boldsymbol{t}_b \cdot \boldsymbol{n})(\boldsymbol{n}_b \sin(\theta_s) + \boldsymbol{t}_b \cos(\theta_s)) \tag{19}$$

where $\boldsymbol{t}_b$ and $\boldsymbol{n}_b$ are the normal and tangent to the wetting substrate and $\boldsymbol{n}$ is the normal to the interface. This force is applied to the boundary by adding this vector integral to the fluid momentum equation:

$$\int_\Gamma \delta_\alpha(\phi) N_j \boldsymbol{F}_{wet} d\Gamma \tag{20}$$

where $\delta_\alpha$ is the smoothed Dirac delta function and $\Gamma$ is the wetting boundary surface. Now it is not hard to imagine allowing $\alpha$ to go to zero and this surface integral transforms into a line integral along the wetting line, $\omega$:

$$\int_\omega N_j \boldsymbol{F}_{wet} d\omega \tag{21}$$

Finally, in a two-dimensional problem $\omega$ can be represented as a single point, $\boldsymbol{x}_\omega$ and this boundary condition reduces to:

$$N_j(\boldsymbol{x}_\omega)\boldsymbol{F}_{wet}(\boldsymbol{x}_\omega) \tag{22}$$

Furthermore, it is a relatively simple task to actually determine $x_\omega$ and this is the theory behind the boundary condition **SHARP_CA_2D** which will be discussed below.

## Syntax changes and additions for new features

### Compiler defines: COUPLED_FILL

All of the encouraging results reported here and elsewhere are predicated on the use of fully-coupled level set evolution. That is, the level set equation is completely coupled with the other degrees of freedom in the problem and there is no subcycling time stepping. This feature is activated, not through the input deck, but rather through the makefile. This is done with compiler define:

```
-DCOUPLED_FILL
```

which must be included as part of the DEFINES variable to activate this feature. If this step is not taken the resulting behavior has not been defined.

### Field enrichment:

Invoking the new enriched interpolants is done when specifying the weight and interpolation functions in the equations section of the input deck. The conventional labels for these functions, Q1, Q2, etc., are replaced by the following new labels:

1) XFEM value enrichment: **Q1_XV, Q2_XV**
2) XFEM gradient enrichment: **Q1_XG, Q2_XG**
3) Ghost fluid value enrichment: **Q1_G, Q2_G**
4) Hughes-based element value enrichment: **Q1_HV, Q2_HV**

Value enrichment indicates that the field itself is expected to have a step change in value across the interface while gradient enrichment indicates that the gradient with respect to position is expected to have a step change across the interface. For example, across a capillary free surface the pressure field would likely have a step change in its value. The equation card that would invoke this is:

```
EQ = continuity Q1_XV P  Q1_XV   1         0
```

### Improved integration accuracy:

*Subgrid Integration: Recursive, non-conforming subdivision*

To have the interfacial source terms integrated by subgrid integration is a simple matter of including the following card in the level set section of the input deck:

```
Level set subgrid integration depth = n
```

where *n* is the maximum level of recursive subdivision, that is, the edge length of the smallest subgrids used will be $2^{-n}$ times that of the original element. The level set length scale parameter should always be non-zero when using this method of integration

*Subelement Integration: Non-recursive, conforming subdivision*

To invoke the subelement method of interface element integration the following card should be appear in the level set section of the input deck:

```
Level set subelement integration = {on|off}
```

by default this method is **off**. When using this integration method the level set length scale parameter should be set to precisely zero with the standard card in the input deck

```
Level set length scale = 0.0
```

## New Surface Tension Formulations:

The additional surface tension formulations are invoked via new "embedded" boundary condition cards in the BC section: These are as follows. Note the string "LS" in the place of the conventional SS or NS strings. This has come to designate a boundary condition (source) applied along an embedded interfacial curve:

1) Divergence form:

```
BC = LS_CAP_DIV LS {m}
```

2) Surface divergence form:

```
BC = LS_CAP_DIV_S LS {m}
```

3) Improved curvature projection:

**BC = LS_CAP_CURVE LS {m}** + normal vector degrees of freedom in the equation section

4) Conventional form:

```
BC = LS_CAP_CURVE LS {m}
```

The first three of these all require the normal vector degree of freedom to operate. This is invoked as a new equation type in the equation section, to wit in two dimensions:

```
EQ = normal1     Q1 N1 Q1     1    1
EQ = normal2     Q1 N2 Q1     1    1
```

The integer parameter *m* in these cards takes on a new role when using extended enrichment. Formerly, it was set to zero and used as a placeholder. Now, however, it can take on the values of (-1, 0, 1). Zero has been the conventional choice and is still the choice when conventional interpolation is used. However, with the advent of extending degrees of freedom this integer assumes a more important role. When extending degrees of freedom are present, this integer must be -1 or 1.

Nominally, the value of this integer identifies which phase the source term while be applied to, positive or negative. In the case where the physics of the phases are coupled, this choice is not important, but it must be -1 or 1 in order to avoid having the source term applied twice. It only has relevance to problems in which the physics in both phases has been disconnected by embedded Dirichilet boundary conditions.

## Extension velocity projection:

Evolution of the level set function by extension velocity is activated and employed by adding three cards to the input deck. First, in the time stepping section, the user must specify the type of weighting method that will be used on the level set function. This is done with this card:

```
Fill Weight Function = {Galerkin | GLS | SC}
```

This card in essence establishes the formulation used to advect the level set equation with only an extension velocity (**`Fill`** refers to the existing VOF method. We are overloading this card for the level set method). All the options listing will work with extension velocity, but this card must be present with one of them for extension velocities to work.

The second added card is a "embedded" boundary condition that sets the extension velocity value at the interface. This is a strongly integrated condition so it will replaced at some nodes near the interface the orthogonality condition used to project the extension velocities. The syntax of this card is given by

```
BC = LS_EXTV_FLUID_SIC LS 1 -1
```

The two integers following LS have important meanings. The first integer identifies from which phase the fluid velocity is taken to compute the extension velocity at the interface. If the velocities are permitted to have value steps across the interface, as when solving phase-decoupled problems, this parameter is important; otherwise, whether its value is 1 or -1 shouldn't play a huge role in the outcome. The second integer identifies which nodes will receive the strong boundary condition: a value of 1 indicates that nodes in the interface elements on the positive side of the interface will receive the boundary condition while a value of -1 means the negative side nodes will receive the boundary condition. Note that because this boundary condition involves a line integral along the interface curve it *should only be used in conjunction with subelement integration*.

Finally, an equation card must be supplied that invokes the orthogonality constraint on the extension velocity so it will be projected from the interface normal to constant level set curves:

```
EQ = ext_v Q1 EXT_V Q1 1  1
```

## Instrumenting non-linear material models with level set

Modulating the value of property with respect to the level set function in the manner described above is a simple matter of adding one additional card. We choose the viscosity property to illustrate this card:

```
Second Level Set Viscosity      = CONSTANT {value} {POSITIVE|NEGATIVE}
```

The presence of this card in the **.mat** file will invoke level set modulation of the viscosity. The constant value for viscosity will be applied to the phase sign set by the **POSITIVE** or **NEGATIVE** string at the end of the card. The other phase's viscosity will be computed from a different material model that must be specified in the same **.mat** file. For example, to have a Carreau model assigned to negative level set phase and a constant viscosity in the positive phase, the set of material file cards would be:

```
Liquid Constitutive Equation    = CARREAU
Low Rate Viscosity              = CONSTANT 2000.5
Power Law Exponent              = CONSTANT 0.17
High Rate Viscosity             = CONSTANT 61.5
Time Constant                   = CONSTANT 220.56
Aexp                            = CONSTANT 2
Second Level Set Viscosity      = CONSTANT 0.2500625 POSITIVE
```

Note that the cards that specify the non-linear Carreau model do no need to be modified to operate in a level set context. Note also that the current implementation allows for only a **CONSTANT** model for the second level set parameter value. The property material model types that can currently be modulated in this fashion are: viscosity, density, heat capacity, thermal conductivity, and fluid momentum source.

## Stongly set embedded velocity conditions

The syntax for this type of embedded boundary condition is as follows:

```
BC = LS_U LS {m} {u0}
BC = LS_V LS {m} {v0}
```

Here **m** is -1 or 1 and determines whether the positive or negative interface nodes will have their momentum equation replaced by the embedded boundary condition. The float values of **u0** or **v0** are the degree of freedom values. These boundary conditions can be used only with subelement integration along with **_XV**(or **_G**) enrichment for both pressure and velocity fields. An exercise the reader might attempt is to adapt the bubble rise tutorial that is presented below to a problem of transient flow past a cylinder where the cylinder is represented by the level set function and these boundary conditions are employed to satisfy the no slip condition on it.

## Sharply set wetting line forces

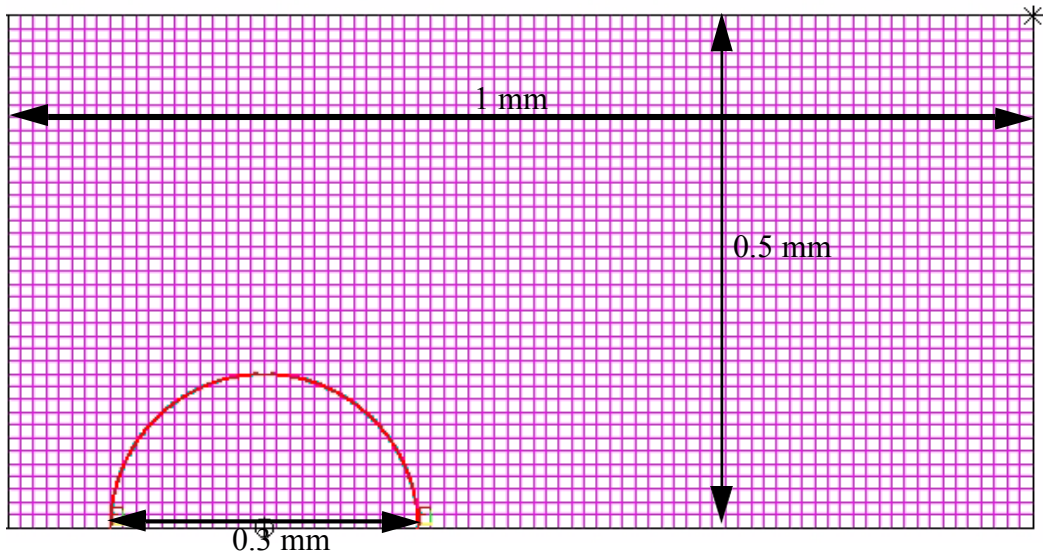


Figure 3. Tutorial problem domain, intial bubble shape and position and dimensions

These are applied using the boundary condition:

```
BC = SHARP_CA_2D SS {m} {angle}
```

where **m** is the external sideset number and **angle** is the static contact angle value. It should be noted that this boundary condition, like **FILL_CA**, must be used in conjunction with the **VELO_SLIP_FILL** boundary condition because it relies on a region of slipping behavior near the interface in order to apply the wetting line force. For this reason, the level set length scale should be non-zero when using this boundary condition, because the latter boundary condition relies upon this value to determine the width of this slipping region. The only situation where you might apply this boundary condition with a zero interface width parameter are those in which the wetting boundary itself is completely slipping, for example, a centerline boundary as we'll demonstrated shortly.

## Tutorial Problem: Rising bubble

As an illustration of some of these new features we return to a problem first dealt with in the original level set tutorial: a bubble of light fluid rising in a container of heavier fluid. The domain is the simple rectangular region depicted in Figure 3 with the dimensions shown. In addition, the starting position of the light bubble is also depicted. This is a small scale problem where surface tension forces are very significant and time scales are quite short. It is a challenging problem for any numerical method. Both the input deck and the material file used are included at the end of this memo. The complete tutorial setup including starting exodusII file and CUBIT journal file should accompany this memo in the tarfile: bubble_rise.tar.

The material properties for this problem are as follows:

$\mu_{heavy} = 1.0$ P ; $\mu_{light} = 0.001$ P

$\rho_{heavy}$ = 1.0 g/cm$^3$ ; $\rho_{light}$ = 0.001 g/cm$^3$

$\sigma$ = 2 dyn/cm

given that the initial radius of the bubble is 0.15 cm, we compute the *Oh* for the light phase to be,

$$Oh = \frac{\mu_-}{\sqrt{(\sigma \rho_- R)}} = 0.057$$

which is perhaps twenty times the corresponding air/water value, but still sufficiently small to provide an adequate example of the improvements.

At this point we shall go through the input deck and highlight the changes and additions that invoke the new features. Refer to the input file attached to this memo. To begin with, we note that at the start of the input deck we have put the aprepro variable definition statement:

```
{Use_Subelement = 1}
```

This is a flag that permits the input deck to use subelement integration (**Use_Subelement = 1**) or subgrid integration (**Use_Subelement = 0**) as options.

Proceeding through the input deck we see that level set interface tracking is first activated and then an initialization method is described with the cards:

```
Level Set Initialization Method  = Surfaces 1
                                 SURF = CIRCLE 0.25 0.0 0.15
```

that impose the initial circular level set contour on the axis of the domain. This initialization method was described in the original tutorial and it has not been modified since then. Below this card, however, is an **Ifdef/Else/Endif** aprepro clause on **Use_Subelement** that fixes the level set length scale, the integration method, and the renormalization method. In case you have forgotten, the aprepro preprocessor evaluates this **Ifdef/Else/Endif** clause and only keeps the true branch in the final input deck supplied to *Goma*. If the case of **Use_Sublelement = 1**, would see the following:

```
Level Set Length Scale = 0.0

Level Set Subelement Integration = on

Level Set Renormalization Method = Huygens
```

The first of these sets the level set length scale exactly to zero. This is the necessary choice when employing the subelement method of integration which is activated by the second of the cards. The third card sets the renormalization method to standard Huygens renormalization. The details of this renormalization method were also detailed in the original tutorial and won't be repeated. In the current version of *Goma*, it is a requirement that this renormalization method be used whenever subelement integration is employed. This is only a temporary restriction and will be removed in later versions.

The other half of the **Ifdef/Else/Endif** aprepro clause is used if **Use_Subelement = 0** so that subgrid integration is the method of choice.

```
Level Set Length Scale = {0.0125/4.0}

Level Set Subgrid Integration Depth = 4

Level Set Renormalization Method = Huygens_Constrained
```

In this case, the level set length scale has been set to one-fourth of the element size. This is appropriate since the subgrid integration method permits interface widths that are narrower than the elements. The subgrid integration method is invoked by the second card and the maximum division depth is set to 4. This means that the original element will be subdivided into smaller subgrids that are no smaller that $1/2^4$or 1/16th of the original element size. It is worth pointing out that the choice of subgrid depth is somewhat related to the value of the interface thickness. We have found through experience that a subgrid depth which results in *four* subgrid elements across the interface gives satisfactory results.

The two remaining cards in the level set section of the **bubble.inp** set the renormalization tolerance to be 1.0 and the renormalization frequency to -1. The latter setting means that renormalization will be controlled solely by the renormalization tolerance and not occur after a fixed number of successful time steps. The role of the renormalization tolerance was described in the early tutorial, but a review here is a appropriate. In a region around the zero level set contour that is twice the size of the level set length scale, an area-averaged value for the magnitude of the level set gradient is computed. If the level set function is a perfect distance function, this average value should be unity. How this value is allowed to deviate from unity before a renormalization is triggered is the tolerance that is set on the renormalization tolerance card. In this case, renormalization will be triggered if this area-averaged gradient magnitude exceeds 1.5 or is less that 0.5.

The solver section of the input deck invokes a UMFPACK direct solver. Recent improvements to the UMFPACK library have made this solver choice a much more viable one.

Proceeding to the boundary condition section, we note four Dirichlet conditions on velocity. These set the transverse velocity ($v_r$) on the constant r surfaces to zero and both velocity components on the lower z surface to zero making it a not slip surface. The upper z surface has natural boundary conditions applied in both directions.

Of note are two new boundary conditions recently added. The first

```
BC = SHARP_CA_2D SS 1 90.0
```

applies a contact angle force on **SS 1** at the intersection of the zero level set contour with this sideset. The nature of this boundary condition was discussed above. The reason for including it is ensure that the lower $r = 0$ surface is a true symmetry boundary. Note that because this is a complete slipping boundary we don't need a **VELO_SLIP_FILL** card to establish a slipping region near the interface, and consequently, the interface width may also be set to zero.

The other boundary condition is one of the new "embedded" type of boundary conditions that can be identified by the string "**LS**" in the place usually afford **SS** or **NS**:

```
BC = LS_CAP_DIV_N LS 1
```

This boundary condition is not so much a boundary condition however as it invokes a surface tension source term on the zero level set curve. We discussed these new source terms previously and if you refer back you'll see that this boundary condition is associated with *div(**n**)* type of surface tension source term. Note the integer 1 following the **LS** string. When the extended finite element method is not being used, this integer is nothing more than a place holder and is usually set to zero. However, in this case we will be using XFEM and the role of this integer is more important. Nominally, it indicates the phase to which the source term is to be applied with -1 = negative phase and +1 = positive phase. In this case, in which the two-phases are coupled it is only important that this integer be -1 or 1 and not zero. Setting it to zero will cause the source term to be applied twice to the embedded interface.

The Problem Description section contains a few noteworthy changes. First, we are using a **CYLINDRICAL** coordinate system. The **EQ** cards for the momentum equation degrees of freedom are pretty standard *Goma* fair, but look at the card for the continuity equation. Now the weight for this equation and the interpolation function for the pressure have been set to **Q1_XV**. We described what this interpolation type was above, but here it is being used in action. The reason for this choice of course is that we anticipate a significant jump in pressure across the capillary surface and we want to include this type of behavior in the interpolating functions. Now one might note correctly, that there will also be a jump in viscous stress so that it would also be appropriate to use **Q2_XG** to weight and interpolate the velocity equations and unknowns which would be able to introduce jumps in the velocity gradient field. This is not done for three reasons. First, it adds degrees of freedom to the problem and makes it bigger, second, a jump in gradient value is a "less severe" discontinuity in the FEM world than a value jump so the standard GFEM interpolating functions are more reliable when it comes to representing such behavior, and third, the ?**_XG** type of interpolation has the unfortunate trait that it "bleeds" into elements that surround the interface element but don't necessarily have the interface curve in them. This has undesirable effects on the solution and as a result at the current time we don't recommend using this type of extended interpolation type.

The other new addition to the Problem Description section are the **EQ** cards for the normal to level set vectors:

```
EQ = normal1     Q1 N1 Q1     1    1
EQ = normal2     Q1 N2 Q1     1    1
```

These add the least squares projection of the level set gradient onto the vector degree of freedom fields (**N1,N2,N3**) as discussed above. These degrees of freedom are necessary to employ the capillary source term we did in the boundary condition section. You'll be able to view these fields, as you might expect, in the exodusII field as well. Notice that these degrees of freedom are interpolated by Q1 polynomials while the level set field is interpolated by Q2 polynomials. Since the former is the derivative of the later, this seems like a reasonable choice. However, we are not aware of any LBB-like restriction on least squares projection so it is also perfectly reasonable to have the

level set function and the normal vector degrees of freedom interpolated at the same polynomial order.

Finally at the end of the input deck is a volumetric integration post-processing card:

```
VOLUME_INT = NEGATIVE_FILL 1 0 negf.dat 0.0
```

This computes the volume of the negative level set phase and writes it at each time point to the file **negf.dat**. This is a useful quantity to track since the degree to which mass is conserved is a good indicator of the overall solution quality.

Turning briefly to a discussion of the material file, **newt.mat** which also appears at the end of this memo. We won't discuss the material file in great detail. It is assumed the user is very familiar with it by now. We will only discuss it in the context of new things.

The first few cards in this file are aprepro definitions that set the phase property values:

```
$ rho+ = {rho_fluid = 1.0  }
$ mu+ =  {mu_fluid =  1.0  }


$ rho- = {rho_gas = rho_fluid/1000.0 }
$ mu-  = {mu_gas  = mu_fluid/1000.0 }
```

These should be self explanatory. These definitions are used to set the viscosity, density, and momentum source model parameters via the cards for density:

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

for viscosity:

```
Liquid Constitutive Equation = NEWTONIAN
Viscosity = CONSTANT {mu_fluid}
Second Level Set Viscosity = CONSTANT      {mu_gas} NEGATIVE
```

and momentum source:

```
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
```

Notice how the Second Level Set cards are used to set the values of the gas phase properties as well as assign these properties to the negative level set phase. The details of this were discussed above. Note also that **CONSTANT** models were used in every phase, however, it would not be hard to use a non-linear viscosity model in the heavy phase for example.

The example is run with the standard *Goma* command:

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

which pipes the stderr and stdout output to **serr** and **sout** respectively. After a few time steps have passed you should look at the solution in **out.exoII** using **blot**. If you invoke DETOUR and type the following:

```
DETOUR> vect vx vy
DETOUR> plot
```

you'll see a vector plot of the initial velocity field. Two things are noteworthy. First, the vortex being shed off the bubble as it starts its initial rise is very obvious, and second, the velocity field itself is quite smooth and free from burbles. This can be seen even more clearly by looking at the vx velocity contours:

```
DETOUR> contour vx
DETOUR> plot
```

These contours are quite smooth and free from oscillations. You might let the computations run for awhile (several hours) and then track the motion of the zero level set with:

```
DETOUR> contour F
DETOUR> crange 0 0
DETOUR> plot
```

It is interesting to observe the axial elongation of the bubble as it starts to accelerate away from the no slip boundary at the left side of the domain. It appears this boundary acts as a bit of drag on the lower (gravity) side of the bubble elongating it.

It would be quite illustrative to compare the behavior just described to that of the former algorithm. In **bubble.inp**, you might change the continuity interpolation and weight from **Q1_XV** back to standard **Q1** and rerun the problem. After only a few time steps the very considerable spurious currents in the velocity field should be quite evident. Addition of XFEM enrichment has a very clear positive effect.

## Final Words

This memo/tutorial has described a significant number of changes and additions to the *Goma* level set capability. The reader is justified in being somewhat confused at this point. *In truth, many of these additions are still under development and may not have reached a final usable form*. However, if one is searching for a message to take away from this document it is this: addition of **_XV** interpolation enrichment to the pressure field is effective at suppressing the parasitic currents. Try it and find out. The specialized integration rules for interface elements are also important in dealing with this problem, especially subelement integration. Both enrichment and one of the integration method should be employed for high surface tension problems for best results.

## References

1. Renardy, Y. and M. Renardy, "PROST: A parabolic reconstruction of surface tension for the volume-of-fluid method," *J. Comp. Phys*., **183**, 400-421, (2002)

2. Chessa, J., P. Smolinski, and T. Belytschko, "The extended finite element method (XFEM) for solidification problems," *Int. J. Numer. Meth. Engng.*, **53**, 1959-1977, (2002).

## Input deck: bubble.inp

```
{Use_Subelement = 1}

FEM File Specifications
-- ---- --------------
FEM file                          = bubble.exoII
#FEM file                         = contin.exoII
Output EXODUS II file             = out.exoII
GUESS file                        = contin.dat
SOLN file                         = no
Write intermediate results        = no
General Specifications
------- --------------
Number of processors              = 1
Output Level                      = 0
Debug                             = 0
Initial Guess                     = read_exoII


Time Integration Specifications
---- ----------- --------------
Time integration                  = transient
delta_t                           = 0.00001
Maximum number of time steps      = 500
Maximum time                      = 2.0
Minimum time step                 = 1e-9
Maximum time step                 = .0005
Time step parameter               = 0.0
Time step error                   = 0.1e-1 0 1 0 0 0 0 0
Printing Frequency                = 1
Initial Time                      = 0.00
Fill Subcycle                     = 1
#Fill Weight Function             =  GLS
Level Set Interface Tracking      = on

Level Set Initialization Method  = Surfaces 1
                                SURF = CIRCLE 0.25 0.0 0.15
                                   SURF = PLANE {cosd(0.1)} {sind(0.1)} 0.0 -3
#Level Set Initialization Method  = Exodus

{Ifdef(Use_Subelement)}

   Level Set Length Scale = 0.0
   Level Set Subelement Integration = on
   Level Set Renormalization Method = Huygens

{Else}
```

```
   Level Set Length Scale = {0.125/8.0}
   Level Set Subgrid Integration Depth = 4
   Level Set Renormalization Method = Huygens_Constrained

{Endif}

Level Set Renormalization Tolerance = 0.5
Level Set Renormalization Frequency = -1

Solver Specifications
------ --------------
Solution Algorithm               = umf
Preconditioner                  = dom_decomp
Matrix Scaling                   = row_sum
Matrix subdomain solver          = ilut
Matrix reorder                   = rcm
Matrix ILUT fill factor   = 3.0
Matrix drop tolerance            = 1.e-9
Matrix residual norm type        = r0
Matrix factorization overlap     = none
Matrix drop tolerance            = 0
Matrix polynomial order          = 3
Size of Krylov subspace          = 200
Orthogonalization                = modified
Maximum Linear Solve Iterations  = 200
Number of Newton Iterations      = 10
Newton correction factor         = 1
Normalized Residual Tolerance    = 1e-7
Residual Ratio Tolerance         = 1e-7
Pressure Stabilization           = no
Pressure Stabilization Scaling   = 0.01


Boundary Condition Specifications
-------- --------- --------------
Number of BC = -1

BC = U NS 4 0

BC = V NS 1 0
BC = V NS 3 0
BC = V NS 4 0

BC = SHARP_CA_2D SS 1 90.0

BC = LS_CAP_DIV_N LS 1

#BC = LS_U LS -1 0.0
#BC = LS_V LS -1 0.0
```

```
#BC = LS_EXTV_FLUID_SIC LS 1 -1

END OF BC


Rotation Specifications =


END OF ROT


Problem Description
------- -----------
Number of Materials= 1
MAT = newt 1
   Coordinate System= CYLINDRICAL

   Element Mapping= isoparametric
   Mesh Motion= ARBITRARY
   Number of bulk species= 0
   Number of EQ= 6

   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
   EQ = normal1     Q1 N1 Q1     1    1
   EQ = normal2     Q1 N2 Q1     1    1
   #EQ = ext_v       Q1 EXT_V Q1  1    1

Post Processing Specifications
---- ---------- --------------
Viscosity                      = yes
Density                        = yes
Stream Function= no
Streamwise normal stress= no
Pressure contours= no
Second Invarient of Strain= no
Mesh Dilatation= no
Navier Stokes Residuals= no
Moving Mesh Residuals= no
Mass Diffusion Vectors= no
Mass Fluxlines= no
Energy Conduction Vectors= no
Energy Fluxlines= no
Time Derivatives= no
Mesh Stress Tensor= no
Fill contours  = no
```

```
Post Processing Volumetric Integration =
   VOLUME_INT = SPEED_SQUARED 1 0 speed.dat 0.0
END OF VOLUME_INT
```

## Material File : newt.mat

```
$ rho+ = {rho_fluid = 1.0  }
$ mu+ =  {mu_fluid =  1.0  }

$ rho- = {rho_gas = rho_fluid/1000.0 }
$ mu-  = {mu_gas  = mu_fluid/1000.0 }

Material Data File



---Physical Properties
Density    = CONSTANT  {rho_fluid}
Second Level Set Density = CONSTANT {rho_gas} NEGATIVE

---Mechanical Properties and Constitutive Equations
Solid Constitutive Equation = NONLINEAR
Convective Lagrangian Velocity  = NONE
Lame MU    = CONSTANT 1.
Lame LAMBDA = CONSTANT 10.
#Lame LAMBDA = CONSTANT 1.
Stress Free Solvent Vol Frac= CONSTANT0.
Liquid Constitutive Equation = NEWTONIAN
Viscosity = CONSTANT {mu_fluid}
Second Level Set Viscosity      = CONSTANT      {mu_gas} NEGATIVE
Polymer Constitutive Equation   = NOPOLYMER

Surface Tension                 = CONSTANT 2.0

---Thermal Properties
Conductivity  =  CONSTANT  1.
```

```
Heat Capacity= CONSTANT.
Volume Expansion= CONSTANT1.
Reference Temperature= CONSTANT0.
Liquidus Temperature= CONSTANT1.
Solidus Temperature= CONSTANT1.

---Electrical  Properties
Electrical Conductivity= CONSTANT1.

---Microstructure Properties
Media Type              = CONTINUOUS
Porosity        = CONSTANT    0.0
Permeability = CONSTANT1.
FlowingLiquid Viscosity   = CONSTANT    1.0
Inertia Coefficient       = CONSTANT    0.0

---Species Properties
Diffusion Constitutive Equation = FICKIAN
Diffusivity  = CONSTANT  0   0.00
Species Time Integration = STANDARD 0
Latent Heat Vaporization= CONSTANT  0   0.
Latent Heat Fusion= CONSTANT  0  0.
Vapor Pressure  = CONSTANT  0   0.
Species Volume Expansion= CONSTANT  00.
Reference Concentration         = CONSTANT  0   0.
********Species Number**********************|

----Source Terms
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
Navier-Stokes Source= CONSTANT     0.0  0.  .0
Solid Body Source= CONSTANT0.  0.  0.
Mass Source= CONSTANT    0.
```

```
Heat Source= CONSTANT    0.
Species Source= CONSTANT  0   0.
Current Source= CONSTANT   0..
```