Sandia National Laboratories

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

Albuquerque, New Mexico 87185-0826

date: 11 January 2006

to: Distribution

from: P. R. Schunk, 9114, MS0834; E. D. Wilkes, Gram Inc.

subject: GOMA's Overset Mesh Method: User Tutorial (GT-026.4)

keywords: Lagrange Multipliers, Applying boundary conditions to level set surfaces

input records: BC = DX_USER, BC = DY_USER, BC = DZ_USER, BC = BAAIJENS_SOLID_FLUID, BC = BAAIJENS_FLUID_SOLID BC = LAGRANGE_NO_SLIP, BC = LS_NO_SLIP

PREFACE

This tutorial assumes the user has gone tLhrough the beginner's training tutorial on GOMA (GT-001.4) and the level-set usage tutorial (GT-020.2). If you would like copies of these tutorials, contact Duane Labreche (dalabre@sandia.gov), Randy Schunk (prschun@sandia.gov), or Tom Baer (tabaer@sandia.gov). You also need to have CUBIT to complete most of these exercises.

EXAMPLE PROBLEMS COVERED IN THIS MEMO:

- Bent blade problem
- Roll gear problem
- Roll-gravure problem with partial overlap

Key Limitations of the Algorithm Discussed here (as of 3/29/2005): serial processing only, two-dimensions only.

To run these tutorial problems you need a Goma version at least post-dating March 30, 2005. Otherwise you should revert back to GT-026.2.

NOTE 1/11/2006: Also, please see important remarks at the beginning of the Examples section regarding best choices of bases functions, interpolations and LS fields to deploy for fluid-structure interaction problems with and without capillary surfaces.

INTRODUCTION

Many physical processes, such as coating flows, involve both fluid and solid mechanics within a coupled framework. The challenges associated with obtaining accurate numerical solutions to this class of problems are numerous and depend on the frame of reference in which each phase is treated. The most expedient treatment employs computational Lagrangian framework for the solid phase and an computational Eulerian framework for the fluid phase, mainly because of the natural variables of momentum transport for each phase, viz. displacement fields for solids and velocity fields for fluids. This work illustrates just such a treatment, with a finite element algorithm deploying topologically independent meshes for each phase and a bordering algorithm imposing fluid-structure interaction by means of Lagrange multiplier constraints between phases. This method uses a level set interface in the fluid phase to track the effect of the motion of the solid boundary on the flow and to transmit stresses back to the solid. Moreover, the algorithm accommodates all necessary Jacobian sensitivities for convergence of Newton's method. Validation of this approach is accomplished with a simple falling sphere example, and used to investigate solid motion through fluid channels.

There are several types of physical processes which involve mutually induced motion of both solids and fluids; some examples are fluidized-bed chemical reactors, coating flows, and particle settling in fluid containers. Such processes may consist of a solid body which is set into motion by a surrounding fluid (e.g. fluidization) or a fluid which is forced to flow in response to the motion of a solid body (e.g. a pump impeller). Moreover, the solid object may be either rigid or deformable. In any of these cases, the physics of the two phases are strongly coupled, and the extent of relative motion between the phases can be large.

It is highly desirable to be able to model such fluid-solid processes with robust numerical algorithms such as the finite element method. However, the nature of these problems present some challenges. One such challenge is to maintain acceptable element quality during mesh motion when the fluid-solid interface displacement becomes large relative to the size of the computational domain. This is illustrated in Figure 1 with an example of a solid ball falling downward through a column of fluid; here, the meshes for both phases are prescribed to undergo arbitrary Lagrangian-Eulerian (ALE) motion such that both domains conform to the motion of the interface. This widely-used method simplifies the task of tracking the transient interface location and the specification of interaction boundary conditions (e.g. continuity of velocity and stress), and works well for small relative motion between phases. This method will ultimately fail, however, when the degree of fluid domain deformation reaches a certain point (also illustrated in Figure 1). Continuation of the transient solution beyond this point would typically require frequent (and often undesirable) remeshing and remapping steps.

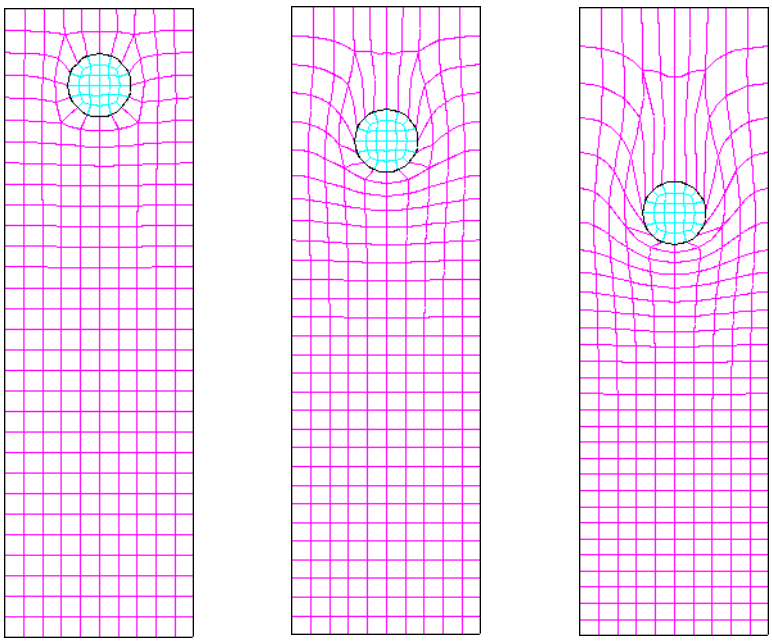


Figure 1: Evolution of conforming ALE meshes for a falling ball.

The approach taken in this work is to construct two independent but overlapping meshes, requiring that only the solid mesh conform to the moving boundary. Doing so allows the solid-phase variables (e.g. solid displacement and stress), which are typically Lagrangian, to be solved on a mesh with Lagrangian mesh motion, while the fluid variables (e.g. velocity, pressure) may be solved on an Eulerian mesh. The fluid mesh can then be fixed; in lieu of solving equations for mesh motion, and a level set interface is used to track the location of the solid boundary from within the fluid. This interface provides a locus within a fluid element along which liquid-side contributions to the interaction conditions are to be imposed, while the corresponding solid-side contributions are simply integrated along the element sides which comprise the moving solid boundary. This approach is similar to the fictitious domain/mortar element method of Baaijens[1]

As suggested above, the key to this method is the ability to impose kinematic-type constraints (viz. impenetrability constraints) which require contributions from the primitive variables in both phases. Achieving an algorithm that accounts for the sensitivities of these conditions to the variable degrees of freedom, as in a Newton approach, will have to involve a dynamic data structure with a degree-of-freedom dependency map that changes as the two materials move relatively to one another. Most data-structure formats supported by matrix-solver packages (e.g. Aztec, Sparse 1.3, Umfpack, etc.) do not readily support such changes and for that matter nor do most legacy finite element codes. This is a particular concern for the full-Newton finite element method, whose convergence relies heavily on having a complete set of these sensitivities.

In the present overlapping grid algorithm, the interfacial constraints are imposed as augmenting conditions to the base matrix system, and are assembled separately from the remainder of the equations. The augmenting conditions are solved simultaneously with the base matrix system using a bordering algorithm (cf. Chan and Resasco 1986). This algorithm effectively adds equations to the

overall system, which then requires a corresponding unknowns to maintain closure. Hence, the Lagrange multiplier field is treated as an additional primitive variable on one of the two phases, as will be explained in Section 2. This Lagrange multiplier treatment is consistent with the equation system in Baaijens' [1] paper, although no mention is made of how his interaction constraints were handled or how (if at all) cross-mesh sensitivities were included.

Below we outline the system of equations and constraints which comprise the fully-coupled problem, including the handling of Lagrange multiplier constraints. Details of the finite element implementation used in this work are also given as are simple validation results and additional sample results.

NUMERICAL FORMULATION

As indicated in the previous section, the class of problems considered here consist of two phases which are coupled at a common interface, but otherwise would act independently except that one phase is displaced volumetrically by the second.

Governing equations

In this work, the governing equations consist of a fluid momentum balance:

$$\int_V \phi_v{}^i \left[\rho_f \frac{Dv}{Dt} + \nabla \bullet \tau + \rho_f \underline{F} \right] dV - \oint_\Gamma {}^i\gamma_k d\Gamma = 0 \tag{1}$$

a mass balance:

$$\int_V \phi^i [\nabla \bullet v] dV = 0 \tag{2}$$

and a solid momentum balance:

$$\int_S \phi_x{}^i [\rho_s \ddot{x} + \nabla \bullet \sigma + \rho_s \underline{F}\,]\, dV + \int_\Gamma \phi_k{}^i \gamma\, d\Gamma = 0 \tag{3}$$

The kinematic constraint at the fluid-solid interface is:

$$\int_\Gamma \phi_\gamma{}^i [\, \dot{x} - v] d\Gamma = 0 \tag{4}$$

and the level set function is evaluated at each fluid mesh node by:

$$f = \Theta \left\| \underline{x}_i - x_s \right\| \tag{5}$$

Equations (1) through (4) are written in a Galerkin/Finite form, with ϕ^i representing the weighting functions at node *i*. Equations (1) through (3) are enforced at all nodes *i* that contain the appropriate degrees of freedom (viz. solid or fluid dofs). Equation (4) applies at the solid-liquid interface. ρ_f and ρ_s are the fluid and solid material densities, respectively, $\underline{v}$ is the fluid velocity, $\underline{F}$ represents any body forces such as gravity, τ is the fluid stress tensor, $\underline{\gamma}$ is the Lagrange multiplier vector unknown, $\underline{x}$ is the solid displacement vector unknown, σ is the solid stress tensor, f is the level set unknown, Θ is a step function which is -1 for points within the region occupied by the solid and +1 outside this region, $\underline{x}_i$ and $\underline{x}_s$ are the position vectors of a fluid node and of the closest point to it on the solid

boundary, respectively, V is the fluid volume domain, S is the solid volume domain, and Γ is the solid boundary (interface) surface domain.

Boundary conditions at locations other than on the solid boundary (interface) will depend on the specific problem being solved; these may include no-slip and no-penetration at walls, fully-developed inflow profiles, axi-symmetry conditions, etc. Solid-fluid interfacial conditions are discussed in the following subsection.

Lagrange multiplier constraints

In this formulation, the Lagrangian solid mesh frame of reference follows the motion of the solid while the fluid mesh maintains a fixed frame of reference. As previously indicated, it is desirable to first assemble the independent equation systems for each individual phase, then augment both with the necessary interfacial constraints. Following the formulation of Baaijens[1], Equation (4) is applied as a Lagrange multiplier constraint on the fluid and solid velocities, such that they are required to be equal on the interface Γ. The new unknown which is introduced along with this constraint is then the Lagrange multiplier vector $\underline{\gamma}$.

The physical problem also requires a balance of stress across the interface. This is implemented here by augmenting the momentum equation in each phase with a surface term which represents the change in total stress due to interaction with the other phase; these are the surface integral terms in Equations (1) and (3). More formally, these terms can be derived by applying the principles of the calculus of variations to the problem (cf. Finlayson). In summary, the kinematic constraint [Equation (4)] is used to solve for the interfacial stress correction value which results in continuity of phase velocity. Thus, the physical meaning of $\underline{\gamma}$ is clear along the solid boundary. Although $\underline{\gamma}$ is defined throughout one of the two domains, it is not significant away from the interface. This Lagrange multiplier treatment allows for a complete, fully coupled problem statement; details of the solution method are discussed in the following section.

COMPUTATIONAL METHOD

The algorithm described in Section 2 was implemented in *Goma*, a full-Newton finite element program developed at Sandia National Laboratories which contains the necessary mechanics equations, boundary conditions, transient nonlinear solver, and bordering algorithm to accommodate the features of this approach.

Equation assembly

This algorithm involves assembly of the primary equation terms separately from the augmenting condition terms. Equations (1), (2), (3), and (5) are assembled on their respective volume domains in residual form:

$$\underline{R} = (\underline{R}_m, \underline{R}_c, \underline{R}_x, \underline{R}_f)^T = 0 \tag{6}$$

Equation (6) lacks a Lagrange multiplier residual because its terms are handled in the augmenting conditions. Any boundary conditions other than those at the fluid-solid interface are then applied to these residuals. As each residual term is assembled, the unknowns upon which it depends are noted, and corresponding sensitivity terms for unknowns in the same phase are assembled to construct the Jacobian matrix $\underline{\underline{J}}$:

$$[J] = \frac{d\underline{R}}{dx} \tag{7}$$

However, sensitivities of the surface integral terms of Equations (1) and (3) are also deferred to the augmenting conditions.

These are the steps which are normally used for simple Newton's method applications, and could be used to solve a problem with two uncoupled, non-interacting phases by iteratively solving:

$$[J](-\Delta x) = \underline{R} \tag{8}$$

Augmenting condition assembly

Terms which may involve unknowns from both phases are more convenient to include in augmenting conditions. To do this, Equation (4) is assembled as an augmenting constraint which is discretized on each element on or overlapping the fluid-solid interface. The solid-side contributions are evaluated along the element side(s) which coincide with the interface. In the fluid elements, the level set function *f* is used to determine if the interface passes through the finite element: if *f* changes sign anywhere on the element, then it contains part of the interface. In this case, the nodal values of *f* are used to construct the locus of the interface within the element, and the fluid-side term contributions are evaluated along this locus.

The overlapping-grid problem consists of *N* unknowns (all variables including the Lagrange multipliers discretized over both domains) and *M* augmenting constraints. As indicated above, the main residual is a vector of length *N* and the Jacobian is an *N*x*N* matrix. The augmenting condition residuals are loaded into vector $\underline{G}$ of length *M*. The sensitivities of these residuals to all variables on both meshes are computed and loaded into submatrix $\underline{\underline{C}}$ of dimension *M*x*N*. Finally, the sensitivities of the surface integral terms in Equations (1) and (3) to the Lagrange multiplier unknowns are computed and loaded into submatrix $\underline{\underline{B}}$ of dimension *N*x*M*. This yields the following augmented system:

$$\begin{bmatrix} J & B \\ C & D \end{bmatrix} \begin{bmatrix} -\Delta x \\ -\Delta y \end{bmatrix} = \begin{bmatrix} R \\ G \end{bmatrix} \tag{9}$$

where Δy is the vector of updates to the Lagrange multiplier unknowns. Submatrix D is not relevant to this algorithm, so it is loaded with zeros.

Augmented system solution

The augmented system [Equation (9)] is then solved for base-unknown updates, Δx, and augmenting unknown updates, Δy, at each Newton iteration by means of a block-elimination bordering algorithm similar to that proposed by Chan and Resasco[2], which is incorporated into the *Goma* nonlinear solver. One Newton iteration then consists of the initial *N*x*N* solve of $\underline{\underline{J}}$ [equation (8)], *M* resolves of $\underline{\underline{J}}$ using right-hand sides taken from the columns of $\underline{\underline{B}}$, one solve of the *M*x*M* Schur complement $\underline{\underline{S}}$, and updates to all unknowns. This is done within the framework of a predictor-corrector strategy: an initial solution guess is predicted, usually from the previous time step, then Newton iterations are perform to correct the current solution until the residual norm and/or solution update norm fall below specified tolerances.

Because the algorithm performs *M* resolves per Newton iteration, the primary concern in the selection of a linear solver is the ability to perform fast resolves of a given matrix after the initial solve.

Experience indicate that direct sparse-matrix solvers such as UMFPACK[3] have the best resolve performance, in which case time taken for the resolves and other steps of the bordering algorithm is typically on the same order as the initial solve for $M \leq 50$. Iterative linear solver algorithms such as GMRES take considerably longer to perform the resolves. In any case, regardless of the linear solver, this algorithm appears to exhibit superlinear ($\alpha \sim 1.5$) convergence of the nonlinear iteration procedure, similar to that reported by Baaijens for his algorithm[1].

EXAMPLE PROBLEMS

NOTE: AS OF 26 MARCH 2005 THE INPUT DECK FORMAT FOR THE OVERSET CAPABILITY HAS CHANGED CONSIDERABLY. THE PHASE-FUNCTION LEVEL-SET FIELDS ARE NOW DEPLOYED TO HANDLE THE OVERSET GRID. PLEASE NOTE THAT THE OVERSET BALL PROBLEM AND THE SIMPLE ONE-WAY COUPLING BLADE PROBLEM NOW USE "PHASE FUNCTIONS" RATHER THAN THE LEVEL-SET FIELDS FOR THIS PURPOSE.

FURTHER NOTE: AS OF 11 JANUARY 2006 GOMA WAS GENERALIZED SO THAT BOTH PHASE-FUNCTION FIELD OR ORIGINAL LEVEL SET FIELDS CAN BE SLAVED TO SOLID BOUNDARY SIDE SETS. ALL FUNCTIONALITY THAT EXISTED IN GT-026.3 IS FULLY FUNCTIONAL HERE, BUT NEW VERIFICATION STUDIES HAVE SHOWN THAT THERE MAY BE BETTER COMBINATIONS OF INTERPOLATION FUNCTIONS AND ELEMENT TYPES WHICH LEAD TO BETTER MASS CONSERVATION. UNFORTUNATLY, THE FULL COMPLIMENT OF SHARP-XFEM CAPABILITIES ARE NOT AVAILABLE FOR BOTH LEVEL-SET CAPILLARY FREE SURFACES AND FOR FLUID STRUCTURE SURFACES. ***PLEASE SEE DISCUSSION ON 'USER-TIPS' BELOW.***

The results presented here illustrate the use of the overlapping grid algorithm for various fluid-solid interaction configurations. In each example, an ALE fixed mesh is used for the fluid domain, while a dynamic Lagrangian mesh [Equation (3)] is used for the solid domain. In each case, a compressible neo-Hookean solid moves through an incompressible Newtonian fluid.

Spherical ball falling through fluid due to gravity: Validation Problem

A very simple example of a fluid-solid interaction is the settling of a solid sphere under gravity. We use this problem to verify/validate the mechanics of the formulation as there exist an analytical solution for a limiting case. *Setting up and running a problem using the overset grid method is discussed in the next section.* This problem can be found in the directory "**`balldrop_tutorial`**". In that directory you will see two subdirectories, **`ale`** and **`overset`**. If in the **`overset`** directory you do not have an input file entitled **`ball_Rphase.input`**, then you need to get updated. Please contact authors. The problem in the **`ale`** directory corresponds to the demonstration problem in the first figure of this tutorial, and is used to verify the overset grid implementation. The problem in **`overset`** corresponds to the verification problem run here. NOTE: To run the overset grid problems it may be necessary to increase the maximum number of augmenting condition global variables, viz. the compile parameter **`MAX_NGV`**, to be greater than the number of surface elements. This particular problem requires Goma to be compiled with **`MAX_NGV=21`** (the default value is 10).

Here, the fluid is contained in a vertical cylinder with a radius of $4R_s$, where R_s is the sphere radius. The cylinder has a closed bottom.The solid material density is 10 times that of the fluid.

Running this problem is fairly straightforward. The key elements of the input deck are all follows:

```
Number of phase functions = 1
Phase Function Slave Surface          = yes
 $$Phase Function Initialization Methods = Nodeset NS 1 EB 1
```

```
 Phase Function Initialization Method = Surfaces 1
    SURF = SS 1
---
---------------------------------------
Augmenting Conditions Specifications
---------------------------------------
Number of augmenting conditions = -1
AC = OV 1 2 1 2
END OF AC


BC =  BAAIJENS_SOLID_FLUID SS 1 2 1
BC =  BAAIJENS_FLUID_SOLID PF 1 2 1
BC = LAGRANGE_NO_SLIP SS 1 2 1

<       EQ = phase1     Q2 F1  Q2   1 1    1
```

Note that the phase function capability is deployed for the overset grid imprint. The method of initialization is the Surface capability and it is tied to Sideset 1. The phase function is "slaved" to this sideset as it imprints the background fluid. The augmenting condition specifications tie the Lagrange multiplier field to the solid meshes, as explained above. Notice the type of AC is "OV", for "overlap".

The three boundary conditions that are applied to SS1 and the phase function basically handle the two-way coupling and the kinematic BC. Note that the SOLID_FLUID coupling is applied to SS 1 which is tied to the solid ball mesh facets, as is the LAGRANGE_NO_SLIP kinematic condition. The phase function type (PF) is used for the fluid-stress that drive flow, and are applied to the level-set surface of the fluid phase. Basically these are the boundary stress integrals on the fluid momentum equations. Finally, note the "**`phase1`**" equation type.

Figure 2 shows fluid streamlines for the fluid moving around the sphere, which illustrate the recirculation regions which form on the sides of the sphere as the fluid flows with a net upward motion; this recirculation region travels downward with the sphere. Figure 2 also shows contours of the shear component of the solid stress (σ_{rz}), which prove that the algorithm allows the stress due to fluid motion to be transmitted to the solid, which undergoes a small degree of deformation in the process.

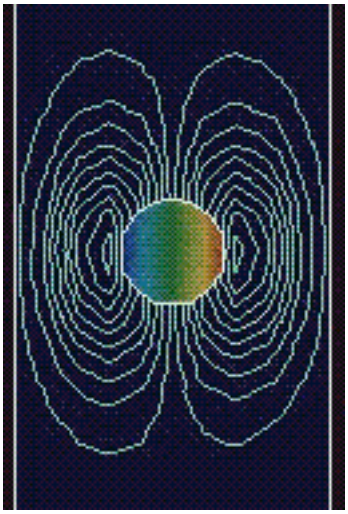


Figure 2: Streamlines and solid shear stress for ball dropping in a cylinder of radius $4R_s$.

To validate the accuracy of the algorithm advocated here, one specific case was run in *Goma* with the already validated ALE approach for comparison. In this case, the fluid is taken to have properties of water, the cylinder radius is taken as 1 cm and the ball radius as 0.05 cm such that the radius ratio is 20; all other conditions are as above. The Reynolds number for this case is about 5.0e-5, or low enough to simulate creeping flow, such that an estimate of the sphere's terminal velocity can be obtained for both algorithms before element distortion becomes significant.

The comparison case employs moving ALE meshes for both meshes, as shown in Figure 1. Therefore, a mesh equation similar to Equation (3) is also needed for the fluid. In this case, the interfacial constraints are applied directly on the boundary shared by the meshes, so the Lagrange multiplier and level set equations are not necessary.

Figure 3 shows the evolution in time of the sphere settling velocity for the two mesh configurations. Although there is a slight difference in the transient velocities approaching the terminal velocity, the terminal velocity was found to be 5.534e-3 mm/s for the ALE case and 5.475 mm/s for the overlapping grid case (less than about 1% difference). This result indicates that likely that the same problem physics is being solved in both case. For the creeping-flow regime, the correlations of Bird *et. al.*[4] predict a terminal velocity of 5e-3 mm/s, which is within about 10% of the computed value by either method. So at least in this very limited regime we can conclude that we are capturing the relevant physical phenomena. Also noteworthy is that no significant difference in the transient flow evolution was observed between the two cases.

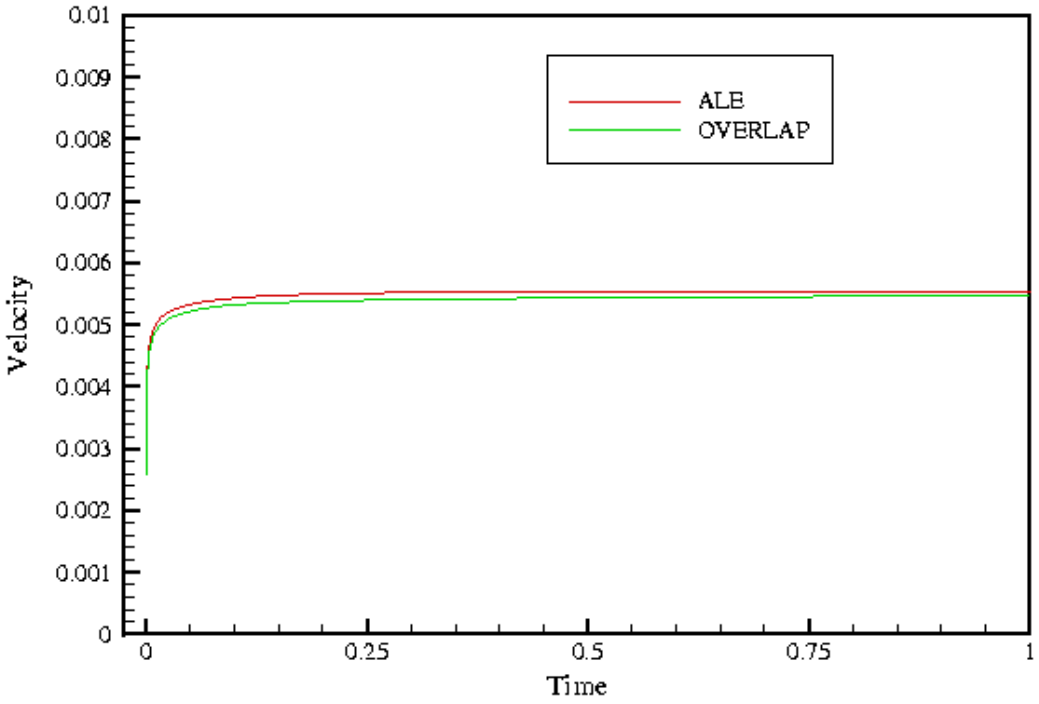
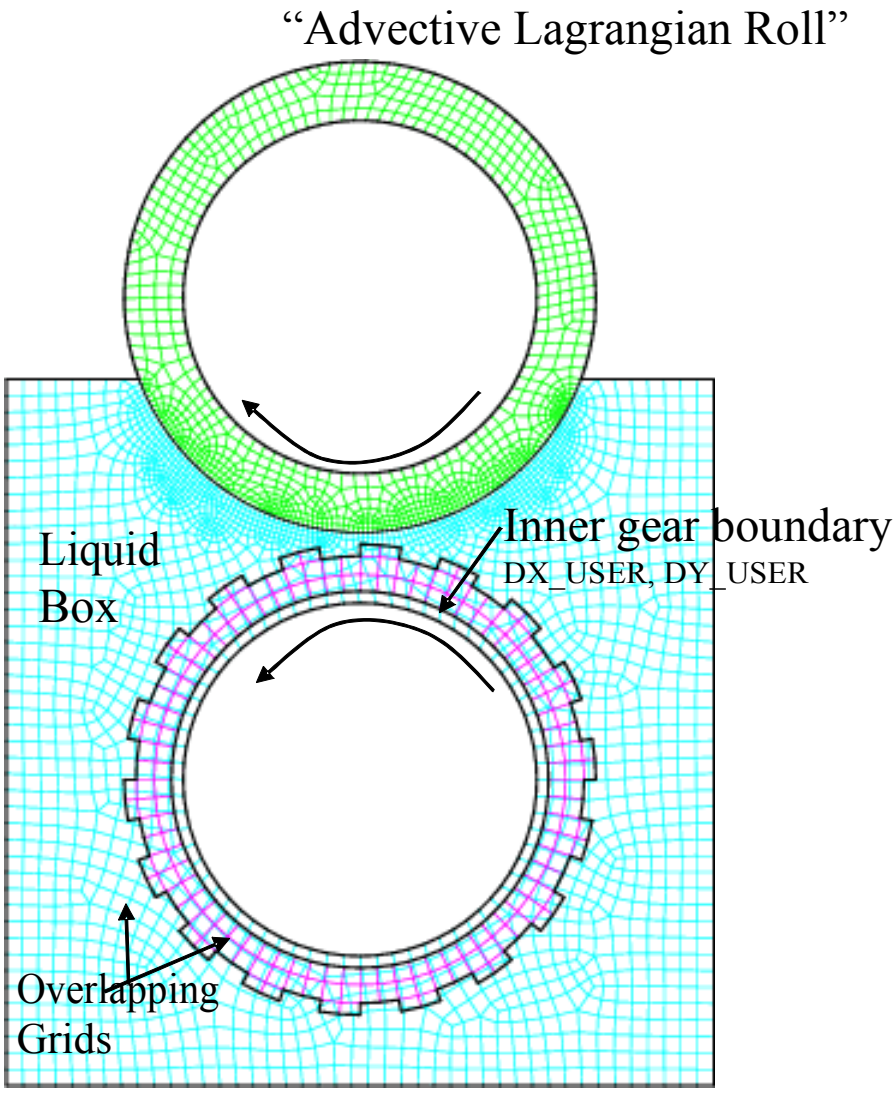


Figure 3: Velocity profiles for falling spherical ball using fixed and conforming fluid meshes.

By way of additional guidance for running this problem, you need to be careful how you specify the body force field on the fluid and the solid. It is important that the fluid phase that is masked (or in reality displaced) by the solid is not influenced by the fluid hydrostatic field. One way to accomplish this is to set the fluid density to zero and then set the solid density to achieve the desired density difference. Unfortunately this is not always practical, especially if more than two phases are present. Another way to accomplish this is to deploy a new capability (see Level-Set Tutorial, GT-020.3) that allows for one to specify material property changes across the level set interface. In this case you can specify a zero body force for the negative level-set side of the interface in the fluid material file. For this capability to be accurate, you also need to use subelement integration.

Gear-Fluid Interaction: Usage Tutorial

This problem (pictured below) is used to test the efficiency and usability of the overset grid method on a problem that more closely resembles a coating flow technology. Unlike the previous example which deploys the method on a solid-fluid-system in which both the solid motion and deformation are a priori unknown, we prescribe the solid motion but allow for solid-stresses to develop from hydrodynamic forces. The geometry consists of a toothed gear undergoing prescribed solid-body rotation in a head-box full of liquid. All related files can be found in the directory entitled "**`overlap_gravure`**". Running this problem as it stands in this directory requires a executable compiled with **`MAX_NGV`** at least 400. To make the problem more interesting, an elastic, deformable roll is pressed from the top toward the gear so as to create a roller-nip in which hydrodynamic forces will be heightened, with the mesh resolution requirements greater. Moreover, the deformable roll is modeled as a computational Lagrangian solid and hence requires a moving mesh in the solid and



Gear/roll test problem for overset grid method. Problem found in "**`overlap_gravure`**" directory.

liquid (Lagrangian mesh motion in the solid and Arbitrary mesh motion in the liquid). The presence of this roll in no way affects the way in which we contend with the gear-like roll using the overset grid method, viz. the upper roll is used to test the method in the presence of moving background grids.

The mesh was generated with Cubit Version 8.0 in a rather crude fashion. A cutting plane (web-cutting) was rotated and used to cut an otherwise smooth annular region. These and other geometry and meshing operations can be found in a file named **`gear.jou`**. The geometry creation commands in this file could easily be parameterized with a tooth-pitch and depth so that more gravure-roll like meshes can be generated. It is important to note that Cubit entity block 1 is the gear and it is meshed independently of the rest of the domain, greatly simplifying the meshing process relative to building a completely contiguous mesh (another advantage of the overset grid approach). Also note that the element sizes deployed for the overlapping regions are taken as the same, viz.

```
surface 442 size 0.07

surface 328 368 408 431 121 161 201 241 281 321 361 401 418 128 168 208 248 288
size 0.07
```

for the gear and

```
surface 502 size 0.07
```

for the background fluid. We discuss underpinning reasons for this choice/requirement further in the trouble-shooting tips below.

As for the Goma input deck, **`gear_Rphase.input`**, the noteworthy features are as follows (note that this is different from the old **`gear.input`** as the overlap field is deployed on the **`phase1`** equation). If you do not have this file, contact the author of this memo. The transient integration specifications and the level-set specifications are given as

```
Time Integration Specifications
 ---
Time integration              = transient
delta_t                  = 1.0e-6
Maximum number of time steps     = 1200
Maximum time                 = 2.5e+4
Minimum time step              = 1.0e-10
Maximum time step              = 1.e-2
Time step parameter = 0.0
Time step error              = -10000.0 1 1 0 0 0 0
Printing Frequency = 1
Fill Subcycle               = 1
Number of phase functions = 1
Phase Function Slave Surface        = yes
$$Phase Function Initialization Methods = Nodeset NS 1 EB 1
Phase Function Initialization Method = Surfaces 1
   SURF = SS 2
Phase Field Length Scale         = 0.
```

The lines in bold warrant discussion. First, the problem must be solved as a transient, which is necessitated by the fact that there is no inertial frame of reference in which all moving surfaces (i.e. both roll surfaces) will not sweep out volume. Actually, it is this fact that makes ALE moving mesh approaches impractical due to the frequent need for remeshing. Also, note that the initial time step was set arbitrarily, but initially very small at **`1.e-6`**. We have noticed that this is necessary when impulsively starting up a moving surface without a smooth velocity ramp in time. The actual time step size required for this problem varies with gear speed and is usually much larger than **`1.e-6`**, as computed by the time it takes a gear-surface element to sweep through the smallest finite element on the background grid. For whatever reason we found it necessary to increase the time-step error for variable time stepping (viz. the **`Time step error`** card) to a large value. At the time that this tutorial was written it was suspected that the time-step error tolerance did not account for the predicted error correctly for this algorithm. As for the level-set specifications, we start with the cards which activate level-set tracking and that which slaves the level set to a specified solid-fluid imprinted surface (which would be the gear-tooth imprint in this problem). Note that rather than using the "Level Set" cards, we are using the "Phase Function" cards. Phase function fields were added as a part of Goma's

multiple level-set field capability. Basically, the original Level-Set cards have enormous capability, and the Phase-Function cards activate a place-holder for other uses of level-set fields. One such use is the overlapping grid capability. This capability is built into the first phase field card. Notice how we instruct Goma to slave the first phase function to a surface with the **`Phase Function Slave Surface = yes`** card. The initial imprinted surface shape is set with the **`Phase Function Initialization Method`** card, and in this particular case is specified to be of type **`Surfaces`** and consist of **`1`** surface (last **`int`** on the card). That one surface is then specified on the next line to be defined by a side set, specifically by Side Set 2 (the gear tooth surface). Note that you could put multiple connected or disconnected surfaces here defined by side sets or other geometry descriptions available in the CGM implementation in Goma (more in upcoming tutorials). The last phase function card, viz. **`Phase Field Length Scale = 0`**. is not needed as it is assumed that subelement integration is always used for slaved phase functions.

The next set of cards that warrant discussion are found in the solver-specification section of the input file:

```
Solution Algorithm              = umf
...
Pressure Stabilization          = yes
Pressure Stabilization Scaling = 0.1
PRESSURE DATUM                  = 448 0.
```

Note that we have chosen the matrix solver as **`umf`**. Right now (11/19/2003) we recommend not using **`front`** or any iterative solver. **`front`** has not been thoroughly tested and iterative solvers are very inefficient for these problems due to the large number of matrix resolves. Note that we also deploy pressure stabilization so we can use a computationally convenient **`Q1Q1`** mesh. You can use the popular **`Q2`** velocity interpolation and **`Q1`** pressure interpolations combination as well. Make sure that if you use **`Q2`** velocities that you also use **`Q2`** displacements in the solid region. The reasons for this requirement are also explained below. Also note the pressure datum specification. The head-box of fluid has no inflow or outflow or free surfaces at which a pressure datum would be naturally set. Without a pressure datum most direct solvers will fail in their pivoting algorithms. ***Actually, we have found the Q2Q1 velocity-pressure element without pressure stabilization to be superior, and your version may be set up for that.***

The heart of the approach hinges on the use of Goma's augmenting condition capability needed to enforce the Lagrange multiplier constraint equation at the surface of the gear. A special type of augmenting condition constraint (viz. a counterpart to VC for volume constraint and BC for boundary condition constraint) is defined for this purpose: OV, for overlapping volume. The augmenting condition specification section for this problem appears as

```
-------------------------------------
Augmenting Conditions Specifications
-------------------------------------
Number of augmenting conditions = -1
AC = OV 2 1 2 1
END OF AC
```

The number of augmenting conditions will be equivalent to the number of nodal points on the surface of the solid. In this case the surface of the solid is defined by side-set 2. The integers appearing after **`AC=OV`** correspond respectively to 1) the side set ID of the solid-fluid surface, 2) the solid material

element block ID, 3) the fluid element block ID, and 4) the element block ID on which the Lagrange multiplier equations are to be defined. Note here we chose to define the augmenting conditions on the solid element block with ID 1.

The boundary conditions are specified as:

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

$Bounding Box
#BC = U NS 3 0.


BC = DX_USER SS 1 0.8 10.e-0
BC = DY_USER SS 1 0.8 10.e-0
$BC = DX NS 1 0.
$BC = DY NS 1 0.


BC =  BAAIJENS_SOLID_FLUID SS 2 1 2
BC =  BAAIJENS_FLUID_SOLID PF 1 1 2
BC = LAGRANGE_NO_SLIP SS 2 1 2

BC = SOLID_FLUID SS 3 3 2
BC = NO_SLIP SS 3 3 2


#########
END OF BC
#########
```

For such a seemingly complicated problem it is interesting that there are so few boundary conditions. The boundary conditions worthy of mention start with the **`DX_USER/DY_USER`** pair. These boundary conditions were added to Goma in order to solve this problem. The user-definitions of these boundary conditions are programmed into the **`dx_user_surf`** and **`dy_user_surf`** subroutines in there **`user_bc.c`** file. The continuous motion of the inner surface receives a boundary condition from the user-defined subroutines. For example, in the **`dx_user_surf`** subroutine the displacement is prescribed through the function "**`func`**" in the following excerpt:

```
if(fv->x0[1] >= 0.)
  {
    theta = acos(fv->x0[0]/u_bc[0]);
  }
else if (fv->x0[1] < 0. && fv->x0[0] < 0. )
  {
    theta = -asin(fv->x0[1]/u_bc[0]) + M_PIE;
  }
else if (fv->x0[1] < 0. && fv->x0[0] >= 0.)
  {
    theta = asin(fv->x0[1]/u_bc[0]) + 2.*M_PIE;
  }

*func = fv->d[0]-(u_bc[0]*cos(u_bc[1]*time + theta) - fv->x0[0]);
```

```
d_func[MESH_DISPLACEMENT1] = 1.;
```

Sideset 1 is the inner surface of the gear, and these BCs are used to prescribe a continuous rotational motion of that surface, as if it were guided by a roller bearing. Note that the displacements are always evolving in time and follow the entire history of the material points. The x- and y-coordinates at **t=0** help set the phase-angle of the displacement. In any case, applying these conditions results in a solid-body rotation of the gear, or effective solid-body rotation, because the elastic constants are taken high enough that the gear is undeformed by fluid stresses. In some sense this is a "rich-man's" way of user-prescribed solid-body rotation. These kinematics could be specified purely geometrically, without mechanics, if the capability existed to do this in goma.

The two boundary conditions applied to SS 2, viz.**BAAIJENS_SOLID_FLUID** and **LAGRANGE_NO_SLIP**, together with the embedded phase-function boundary condition **BAAIJENS_FLUID_SOLID** (appled to set type "**PF**") basically provide the fluid/solid coupling. Sideset 2 recall is the gear-fluid interface. The first two enforce the traction vector balance between the phases, and the last is used to enforce the kinematic condition. **LAGRANGE_NO_SLIP** is applied to sideset 2 and is used to enforce the Lagrange multiplier constraint Eq. (4) above. Note that this is a vector condition and the corresponding unknowns are the vector Lagrange-multiplier field $\underline{\gamma}$. We take the approach of defining the Lagrange multiplier field on the solid material, but **LAGRANGE_NO_SLIP** is enforced only on the solid-surface degrees of freedom. As it happens, away from the interface, we simply set the field to zero and trivialize the equations so that no updates result in the Newton scheme. This is all taken care of automatically in Goma.

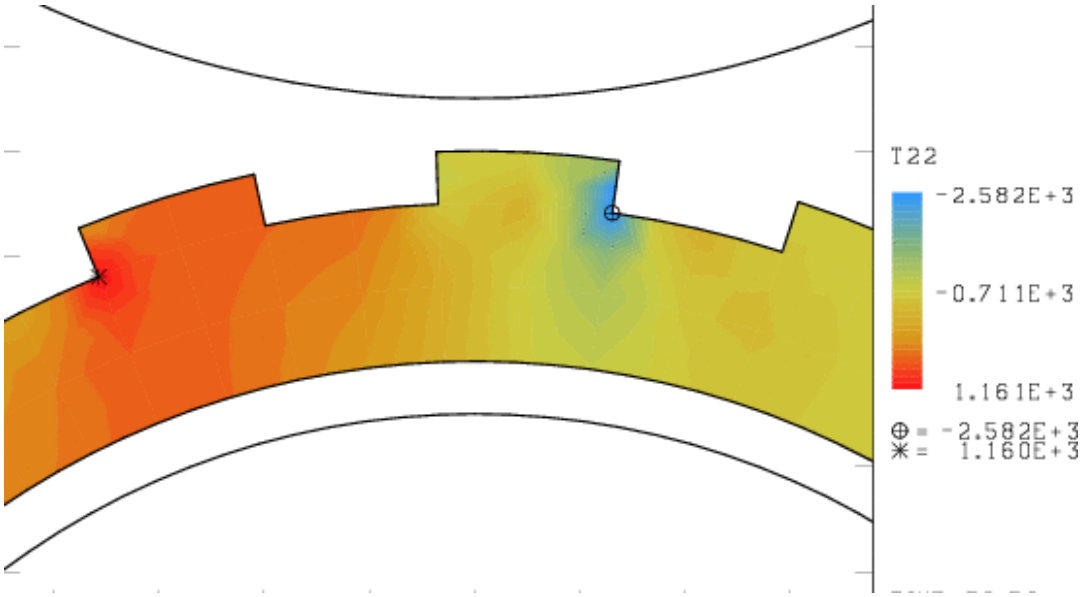
BAAIJENS_SOLID_FLUID is applied also to side set 2, which is attached only to the solid, and is used to add the weak-form boundary stress term on Equation 3 (viz. the elasticity equation). This boundary term you will notice is simply an integral of the Lagrange multiplier vector. Notice that the term is actually an integral over the solid-fluid interface, which is trivially performed when on element boundaries; in fact this boundary condition is applied the exact same way as the **CAPILLARY** and other related boundary conditions. I**BAAIJENS_FLUID_SOLID** is applied to the Navier-Stokes residual equation (1) also as a weak-form boundary condition. This condition is a bit harder to implement, as the integral is performed in the fluid phase along a level-set (or phase-field) imprint contour, or actually the first-phase-function imprint contour, (hence the "**PF**" boundary type on the card). Nonetheless, the details of performing this integral are transparent to the user. In summary, effectively these two boundary conditions result in a vector stress balance between the fluid and the solid, and in fact the Lagrange multiplier vector field represents a traction field over the interface that indicates the fluid traction resultant on the solid (or likewise the negative of the solid-traction on the fluid).

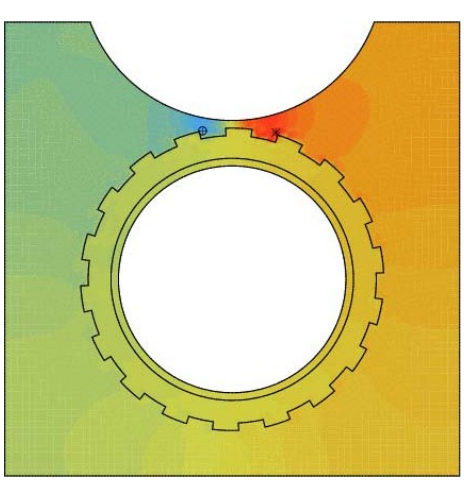
The remainder of the input deck is rather straight forward for users that are familiar and experience in analysis with Goma. Basically there are two materials (actually three, if one includes the upper roller which is handled in a Lagrangian mesh-motion framework). The background fluid material (**liq.mat**), involves no special properties or equations, with the representative mechanics consisting of the ALE formulation of the Navier Stokes equations (recall the mesh in this problem moves as a result of the deform ability of the upper roll, but not as a result of the turning gear). That is, in 2D, two fluid momentum equations, a continuity equation, and two mesh-motion equations are solved simultaneously in the fluid domain. The gear material (**gear.mat**) is basically an elastic solid and

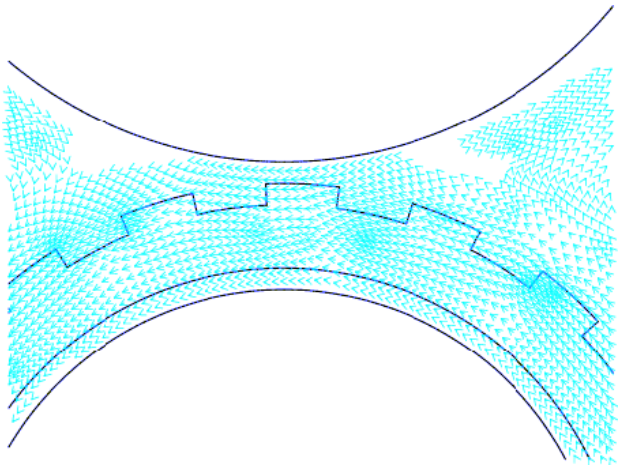
requires only the **LAGRANGIAN** form of the mesh-motion equations, hence the **mesh1** and **mesh2** equations.

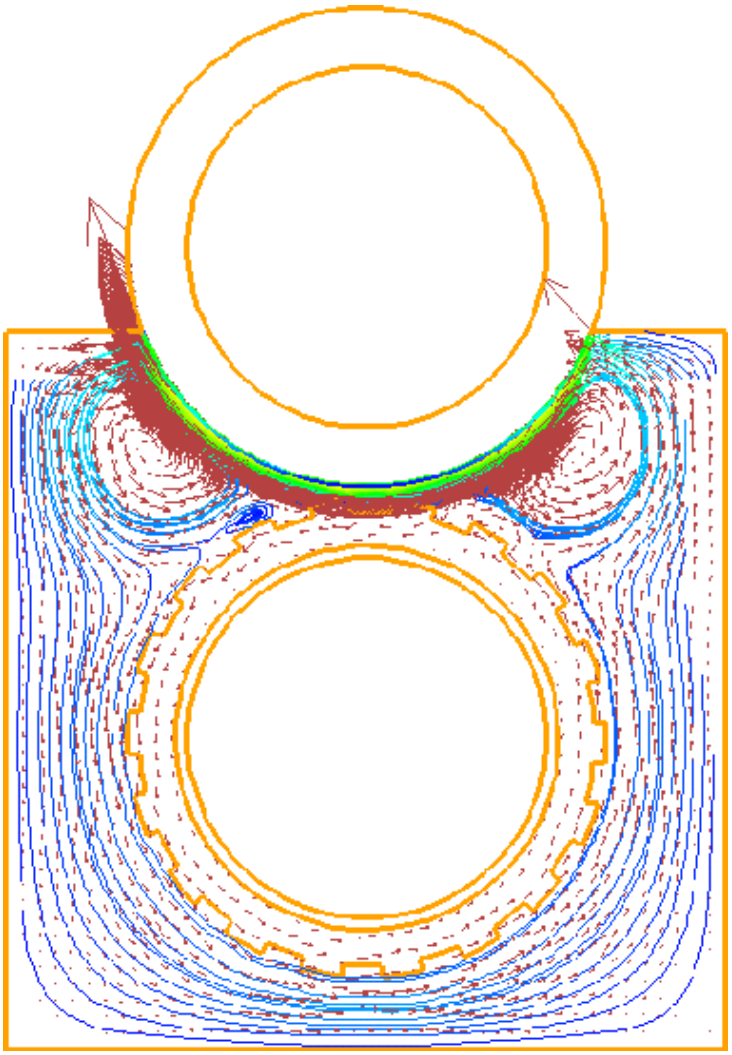
Running this problem is straight forward. Currently (3/29/05) it suffers from the following limitations: 2D only, serial processing only, umfpack linear solver only, and no streamlines. All three of these limitations are surmountable with additional work.

Below we show some sample results for one particular set of operating conditions. First we show the y-component of the normal stress tensor in the gear, indicating the stresses imparted by the hydrodynamics. Next we show the pattern of flow (velocity vectors) and the pressure field in the case that contains no second deformable roll. Finally we show a flow field snap shot of the roll-gear-fluid three-material problem.


T22
-2.582E+3
-0.711E+3
1.161E+3
⊕ = -2.582E+3
✳ = 1.160E+3

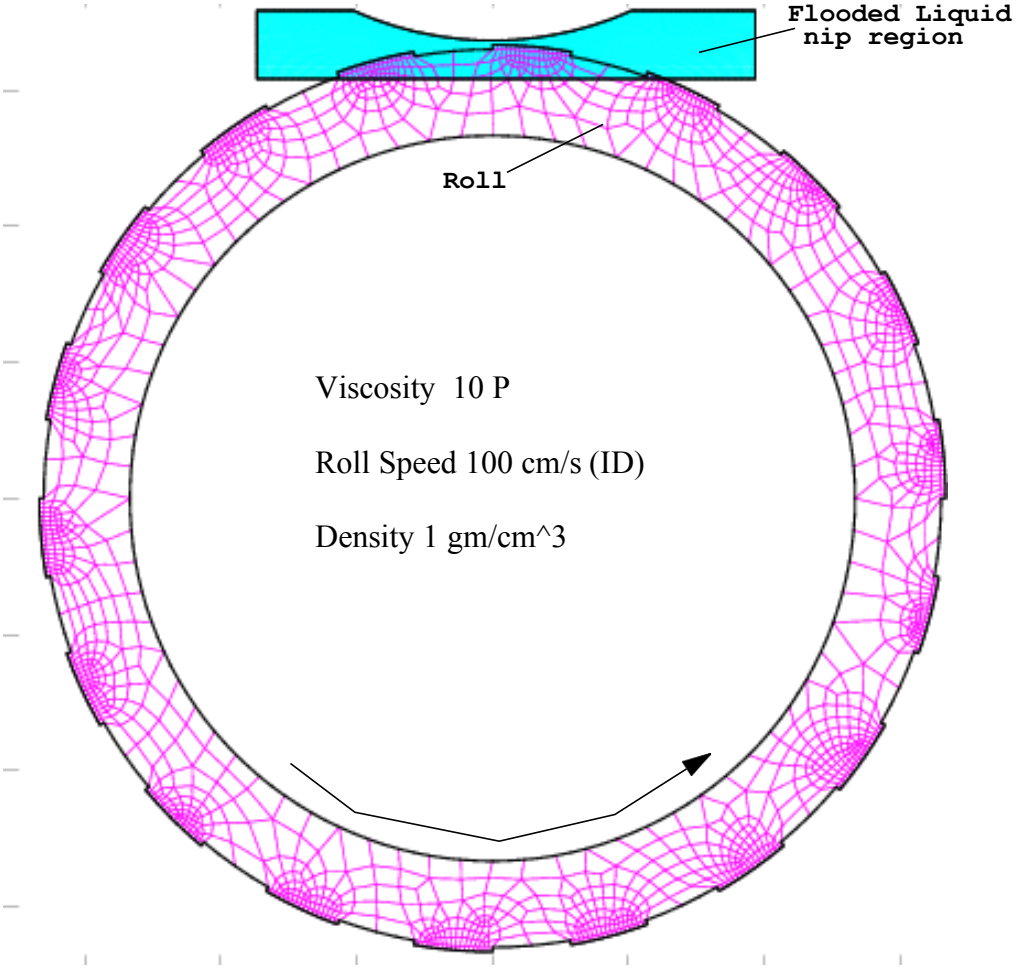






Roll-Gravure problem with partial overlap AND capillary free surface

This problem (pictured below) is used to test the efficiency and usability of the overset grid method



Gravure/roll test problem for overset grid method. Grid with free surface. Problem found in "**`overlap_gravure_coarse`**". Input file is **`gear_partial_with_flooded_nip.input`**. Mesh file is **`gear_partial_fluid.jou`**.

on a problem approaches a real gravure roll coating operation.. Unlike the previous example which which involved larger roll-feature sizes and a completely immersed roll, we test the algorithm in a more practical situation. Here we model the fluid-structure interaction over a smaller "action-zone" region, or coating-nip region, allowing the roll to enter and leave the domain. Where the roll is not "imprinting" the fluid region, the mechanics are simply solid-body rotation. In the overlap region the fluid-structure interaction is accommodate. At the intersection of the roll and fluid region the fluid boundaries are taken to be "open-flow" boundaries so that fluid is entrained into and out of the fluid region. Ultimately we will deploy this method together with a level-set representation of a liquid layer and associated capillary free surfaces. In this case the inflow and outflow fluid may be gas or will require some provisions to accommodate an incoming liquid layer.

Furthermore, we also deploy a second level-set field, or the original level-set field, to represent a capillary free surface. So, we are solving for two level-set fields in total, the **`phase1`** field for the imprinted solid roll on the background fluid (known as the "**`F1`**" variable), and the second representing a capillary free surface (known as the"**`F`**" field variable).

The geometry consists of a toothed gear/roll undergoing prescribed solid-body rotation. As you can see the roll enters the fluid domain on the upper right and leaves on the upper left. The input files are for the most part exactly the same as in the previous example, and are in fact bundled in the same directory entitled "**`overlap_gravure_coarse`**". The input file is **`gear_partial_with_flooded_nip.input.`**

The major difference in this problem is that we choose not to model the feedback stresses of the fluid on the roll (viz. we neglect the boundary term on Eqn 3. This greatly simplifies the problem as we no longer need augmenting conditions to help accommodate the application of this term and all its sensitivities, i.e., the augmenting condition section is commented out:

```
-------------------------------------------
Augmenting Conditions Specifications
-------------------------------------
$$Number of augmenting conditions = -1
$$AC = OV 2 1 2 1
$$END OF AC
```

With this change the problem runs much faster and can be pushed to much higher resolutions. With the mesh pictured above, the number of unknowns is about 100K, and a direct solver UMF solves the system in under 30s on a 2003 vintage Linux workstation. Additional uknowns for the second level set field do slow down the problem somewhat.

One other difference in the problem setup concerns the definition of the Lagrange multiplier equations and the boundary conditions required to enforce the kinematic constraint (Eqn. 5) and the boundary stress term in the fluid (Eqn. 1). When the feedback stresses are included, as in the previous gear-fluid example, no explicit Lagrange multiplier equations are required in either phase as the constraints are taken care of with the augmenting conditions. In this case we need to provide the problem with Lagrange multiplier equations in the fluid phase, viz.

```
MAT =   sample    2
   Coordinate System= CARTESIAN
   Element Mapping= isoparametric
   Mesh Motion= ARBITRARY
   Number of bulk species= 0
   Number of EQ= 7
   EQ = momentum1 Q2 U1 Q2 1 1 1 1 1 0
   EQ = momentum2 Q2 U2 Q2 1 1 1 1 1 0
   EQ = continuity  P0 P  P0   1        0
    EQ = level_set   Q2 F  Q2   1 1     1
    EQ = lagr_mult_1 P0 LM1 P0  1
   EQ = lagr_mult_2 P0 LM2 P0  1
   EQ = phase1      Q2 F1  Q2   1 1     1
```

Note that we require a level-set equation for the capillary free surface, a second level set equation, or a "**`phase1`**" equation for the roll-imprint on the background fluid, and a vector lagrange multiplier field. The Lagrange multiplier field provides a mechanism to enforce the stress the roll imparts on

the fluid (boundary term in Eqn. 1) and enforce the kinematic constraint (or no slip constraint given by Eqn. 5). These two conditions are applied with the following two boundary conditions:

```
BC =  BAAIJENS_FLUID_SOLID PF  1 1 2
BC =  LS_NO_SLIP PF 1 1 2
```

Notice that the set type for these boundary conditions is "**PF**" and that they are both applied to the phase 1 field, as is instructed by the first data integer. The last two data integers on each card are not currently used. Compare these boundary conditions with the previous case in which we accommodated roll-stress feedback.

Also note that the level-set specification section looks like:

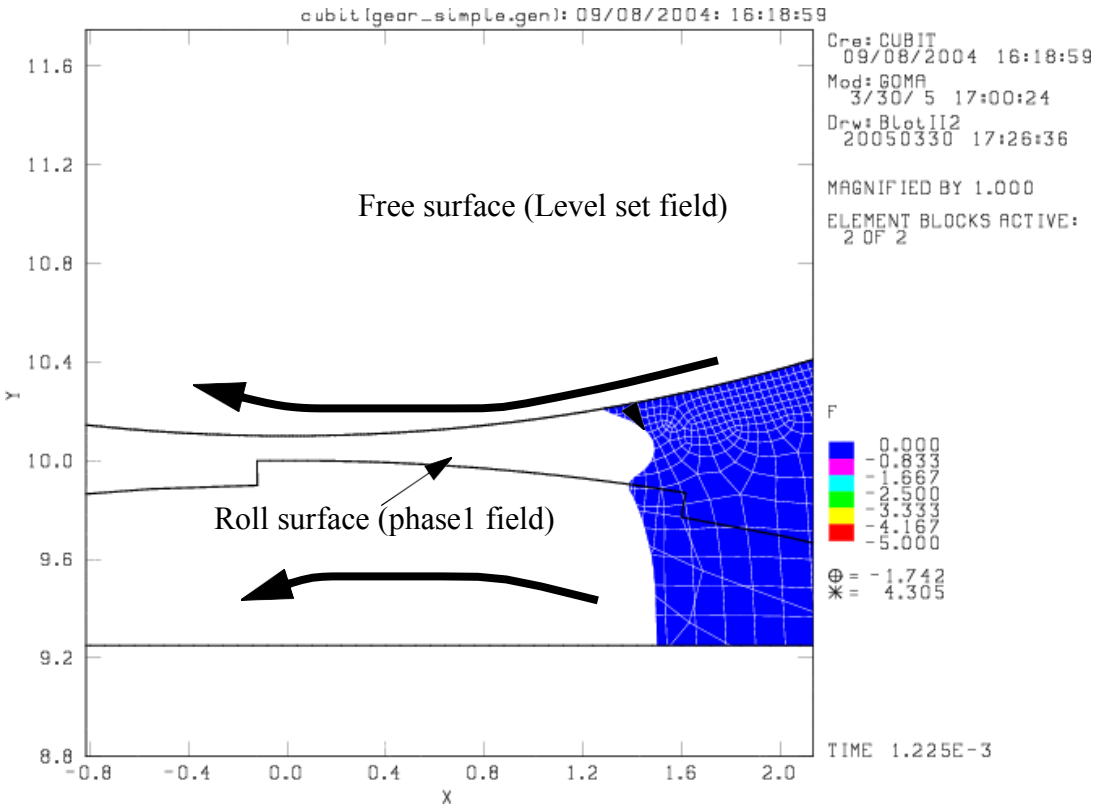
```
Level Set Interface Tracking     = yes
$Forward
$Level Set Initialization Method  = Surfaces 1 SURF = PLANE -1. 0. 0. -2.5
$Reverse
Level Set Initialization Method  = Surfaces 1 SURF = PLANE -1. 0. 0. -1.5
$Level Set Initialization Method  = Exodus

Level Set Renormalization Method = Huygens_Constrained
Level Set Length Scale           = 0.05
$Level Set Renormalization Tolerance = 0.250000
Level Set Renormalization Frequency = 5

Number of phase functions = 1
Phase Function Slave Surface          = yes
$$Phase Function Initialization Method  = Nodeset NS 1 EB 1
Phase Function Initialization Method   = Surfaces 1
    SURF = SS 2
Phase Field Length Scale           = 0.
```

The first level-set field is used for the capillary free surface and is initialized using a plane function so that the nip on the right-hand-side is partially flooded at the start. The phase-function field is used as before, viz. to imprint the roll shape onto the fluid to drive flow.

Sample results are shown in the figure below. The capillary free surface level-set field is contoured over the fluid and solid grid. The value in the solid grid is meaningless to the problem.


cubit[gear_simple.gen]: 09/08/2004: 16:18:59
Cre: CUBIT
09/08/2004 16:18:59
Mod: GOMA
3/30/ 5 17:00:24
Drw: BlotII2
20050330 17:26:36
MAGNIFIED BY 1.000
ELEMENT BLOCKS ACTIVE:
2 OF 2
Free surface (Level set field)
Roll surface (phase1 field)
F
0.000
-0.833
-1.667
-2.500
-3.333
-4.167
-5.000
⊕ = -1.742
✳ = 4.305
TIME 1.225E-3
11.6
11.2
10.8
10.4
10.0
9.6
9.2
8.8
Y
-0.8
-0.4
0.0
0.4
0.8
1.2
1.6
2.0
X

TROUBLESHOOTING

Running this problem is straight forward. Currently (4/29/2005) it suffers from the following limitations: 2D only, serial processing only, umfpack linear solver only, and no streamlines. All three of these limitations are surmountable with additional work.

Also, the following outstanding items need to be considered when running all of these problems (30 March 2005)

-Overset grid slaved level set field is tied to the first phase-function field, period. That is, the boundary conditions BAAIJENS_FLUID_SOLID and LS_NO_SLIP are applied to that field now.

-To run the phase-function capability with NO base level-set field, viz. no ls->var = FILL, but rather pfd->ls[0]->var = R_PHASE0, you need to build with the compile flags COUPLED_FILL and PHASE_COUPLED_FILL.

-To run the real old FILL capability not using level-sets (like the fill1 test suite problem), you must build with COUPLED_FILL.

-Currently field renormalization is disabled for phase-functions

-PF_CAPILLARY Term in mm_bc.c needs to be fixed up.

-Must change all BAAIJENS_FLUID_SOLID and LS_NO_SLIP BCs to set type “PF”, from “LS”.

-NOTE: BAAIJENS_FLUID_SOLID and LS_NO_SLIP BCs must have the first DATA Integer set to 1 so that they refer to PF function 0. Otherwise the PF field won’t be recognized.

User Tips

When running overset grid problems it is important to strive to keep mesh element sizes in the fluid and in the solid as close in value as possible. We find that disparate sizes lead to strange behavior and inaccuracies in the calculation. The user should be aware of these and other guidelines, summarized as follows:

- **Problems are sensitive to relative mesh size between fluid and solid—troubleshooting tips required**
- **Seem to work correctly for both Q2 and Q1 element types**
- **Works in conjunction with moving mesh**
- **Require second level-set field to work with fluid-fluid free surfaces. Or at least kill off the level-set dependence for the imprinted solid—needs testing**
- **Practical for real-life coating situations, given the overhead? Potential, yes!**
- **3D and parallel issues are substantial.**

It is also important to avoid perfect colinear mesh lines between overlapping grids, as often happens when meshing up both solid and fluid regions that share the same symmetry plane. When mesh lines on the symmetry plan from the solid mesh and the fluid mesh perfectly align, integration errors are common. In fact, if you receive the following message:

```
number of edges = 3
Silly me, I thought this couldn't happen.
```

then your mesh likely posseses this pathology.

Best Choices of momentum equation basis functions (velocity representations) and XFEM features for certain problem classes:

- **Fluid-Structure Interaction only (viz. a structure moved with prescribed motion or with communication with hydrodynamic forces only, with NO capillary surfaces)**

  ----For best mass conservation use Slave LS field and NO phase field functions together with **Q1_GP** velocity representation or **Q2_GP** velocity representation with P0 pressures and either lagrange multiplier BCs (like BAAIJENS_FLUID_SOLID and LS_NO SLIP).

  ```
  EQ = momentum1 Q1_GP U1 Q1_GP 1.0 1.0 1.0 1.0 1.0 0
  EQ = momentum2 Q1_GP U2 Q1_GP 1.0 1.0 1.0 1.0 1.0 0
  ```

  ----For real good mass conservation you can do exactly as above and use **LS_U BC** which is in place of the Lagrange multiplier approach to applying the kinematic boundary condition on the moving structure. This is somewhat limited in that the motion of the structure has to be compatible with these Dirichlet surface BCs on the structure surfaces.

  ---For OK mass conservation use Phase field 1 to SLAVE to structure surface together with **Q1** velocity representation with P0 pressures and either lagrange multiplier BCs (like BAAIJENS_FLUID_SOLID and LS_NO SLIP). Note that all XFEM-like basis functions

for sharp interfaces are not available in the phase-functions. Also note that this is the approach used in the example problems above. For most problems, Q2 velocity basis functions will lead to deleterious mass losses/gains without the sharp interface XFEM stuff like Q1_GP provides. The reasons for this are that the higher-order bubbles can accommodate a mass rarefaction in one element, viz. a mass loss/gain.

- **Fluid-Structure Interaction with second level set field for a capillary free surface (viz. a structure moved with prescribed motion or with communication with hydrodynamic forces in one part of the flow and a fluid/gas free surface in the same flow)**

  ---This class of problem was solved in the examples above using the phase function 0 to represent the fluid structure boundary (through slaving), and the base level set field for the capillary free surface. This is still advocated but please keep in mind that you cannot use XFEM like basis functions on the PF surface, and hence you can get severe mass errors. We recommend that you use Q1P0 velocity pressure elements so that you get reasonable mass conservation at the solid fluid boundaries. Note that in some cases the Q2P0 may work. You can still use all the XFEM sharpening capabilities for the capillary free surfaces.

  --Now, if you would like to capture mass accurately at the solid/fluid surface, use the base level-set field to represent it, viz. slave the sideset representing that surface to the level-set field. Then use the phase fields to representing your capillary surfaces. The caveate here is that the phase fields cannot be sharpened in any way, and with the exception of a few BCs like PF_CAPILLARY, you cannot apply and other wetting BCs, etc.

CONCLUSIONS

This work has demonstrated an algorithm which accommodates finite element solution of fluid-structure interaction problems by allowing each phase to have an independent mesh suited to the variables which are to be defined on it. Full coupling of the phases is achieved through a Lagrange multiplier formulation for the interfacial interaction terms. All necessary Jacobian sensitivities, including cross-mesh terms, are included through the use of augmenting conditions. The algorithm was validated against both theory and a fully-conforming mesh result using the classical falling sphere problem. Investigation of other fluid-solid configurations shows that this algorithm is capable of capturing many physical phenomena of coupled fluid and solid mechanics. The algorithm is versatile enough to be extended to a wide variety of such problems, and its numerical performance is shown to be at least equal to Baaijens'[1] algorithm.

Keep in mind that if feedback stresses to the structure are not required, viz. you wish to prescribe solid-body motion (translation or rotation) and model the associated fluid mechanics, a much simpler and more efficient version of the algorithm can be deployed, as the last example demonstrates.

REFERENCES

[1] Baaijens, F. P. T. "A fictitious domain/mortar element methodfor fluid-structure interaction,"*Int. J. Num. Meth. Fluids* **35**(7), 2001, 743.

[2] Chan, T. F. and Resasco, D. C. "Generalized deflated block-elimination," *SIAM J. Numer. Anal.* **23**, 1986, 913.

[3] Davis, T. A. and Duff, I. S. "An unsymmetric-pattern multifrontal method for sparse LU factorization," *SIAM J. Matrix Analysis and Applications* **18**(1), 1997, 140.

[4] Bird, R. B., Stewart, W. E. and Lightfoot, E. N. *Transport Phenomena*, Wiley, New York, 1960, 192.