SANDIA REPORT

SAND2005-0149
Unlimited Release
Printed January 2005

LDRD Final Report: On the development of hybrid level-set/particle methods for modeling surface evolution during feature-scale etching and deposition processes

Lawrence C. Musson, Sandia
Rodney C. Schmidt, Sandia
Corey L. McBride, Elemental Technologies

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

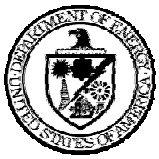
Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online



SAND2005-0149
Unlimited Release
Printed January 2005

LDRD Final Report: On the development of hybrid level-set/particle methods for modeling surface evolution during feature-scale etching and deposition processes

Lawrence C. Musson and Rodney C. Schmidt
Computational Sciences Department, Dept. 9233
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0316
lcmusso@sandia.gov
rcschmi@sandia.gov

Corey L. McBride
Elemental Technologies
357 North 50 South
American Fork, UT 84003
mcbride@elemtech.com

Abstract

Two methods for creating a hybrid level-set (LS) / particle method for modeling surface evolution during feature-scale etching and deposition processes are developed and tested. The first method supplements the LS method by introducing Lagrangian marker points in regions of high curvature. Once both the particle set and the LS function are advanced in time, minimization of certain objective functions adjusts the LS function so that its zero contour is in closer alignment with the particle locations. It was found that the objective-minimization problem was unexpectedly difficult to solve, and even when a solution could be found, the acquisition of it proved more costly than simply expanding the basis set of the LS function. The second method explored is a novel explicit marker-particle method that we have named the grid point

particle (GPP) approach. Although not a LS method, the GPP approach has strong procedural similarities to certain aspects of the LS approach. A key aspect of the method is a surface rediscretization procedure—applied at each time step and based on a global background mesh—that maintains a representation of the surface while naturally adding and subtracting surface discretization points as the surface evolves in time. This method was coded in 2-D, and tested on a variety of surface evolution problems by using it in the ChISELS computer code. Results shown for 2-D problems illustrate the effectiveness of the method and highlight some notable advantages in accuracy over the LS method. Generalizing the method to 3D is discussed but not implemented.

Contents

Figures

LDRD Final Report: On the development of hybrid level-set/particle methods for modeling surface evolution during feature-scale etching and deposition processes

1 Introduction

The problem of representing accurately the temporal evolution of a moving interface is a frequent one when modeling many different physical phenomena including interfacial fluid mechanics, and materials science just to name two. The interest here is the need to model accurately and reliably the evolving solid/gas interfaces in MEMS fabrication processes. What's modeled is the formation of usually a single component, partially constructed, of a device called a feature. The manufacturing processes in question are classed as either deposition or etch processes. The classes are identified by the net addition or subtraction of material though often in each case there is a competition between material deposition by chemistry and material removal by chemistry and ion sputtering. In this report we describe results from a small LDRD-funded research effort that has explored several ideas for creating a hybrid level-set (LS) / particle method for modeling surface evolution during feature scale etching and deposition processes.

1.1 Motivation

Theoretical modeling of the detailed surface chemistry and concomitant surface evolutions during microsystem fabrication processes has great potential for improving surface micro-machining (SMM) based process fabrication technologies. A fundamental aspect of this problem is the ability to model accurately large changes in surface topology that occur during etching and deposition processes. The level-set method is one viable method due to its ability to model such changes reliably without user interaction or other ad hoc treatments. A potential disadvantage of the LS method is that the spatial accuracy is such that very refined grids in regions of high curvature must be used to maintain model fidelity.

1.2 Quck Review of Free-Boundary Modeling

A variety of both explicit and implicit methods for modeling interface evolution have been developed over the years. Explicit, or so-called Lagrangian methods represent the interface by a collection of discrete points or material filaments which share material coordinates with the surface, and which are advected with the velocity of the surface. These methods can be classified as

either front-tracking methods [2]—where the surface is represented by contiguous material filaments, or marker-point methods [9]—where the surface is represented by a collection of points on it. Implicit, or so-called Eulerian methods, such as the volume-of-fluid [7] and the level-set method [9], define the interface implicitly by a scalar quantity from which the interface can be deduced locally on a stationary grid. Each of these approaches has particular advantages and disadvantages, and the problem of interface tracking continues to be an area of active research [8].

For problems with large topological changes, such as may occur in the feature length-scale modeling of MEMS and microprocessor fabrication processes, the level-set method has distinct advantages. In particular, the merging or pinching-off of colliding surfaces is handled naturally without *ad hoc* rules or the necessity of user interference.

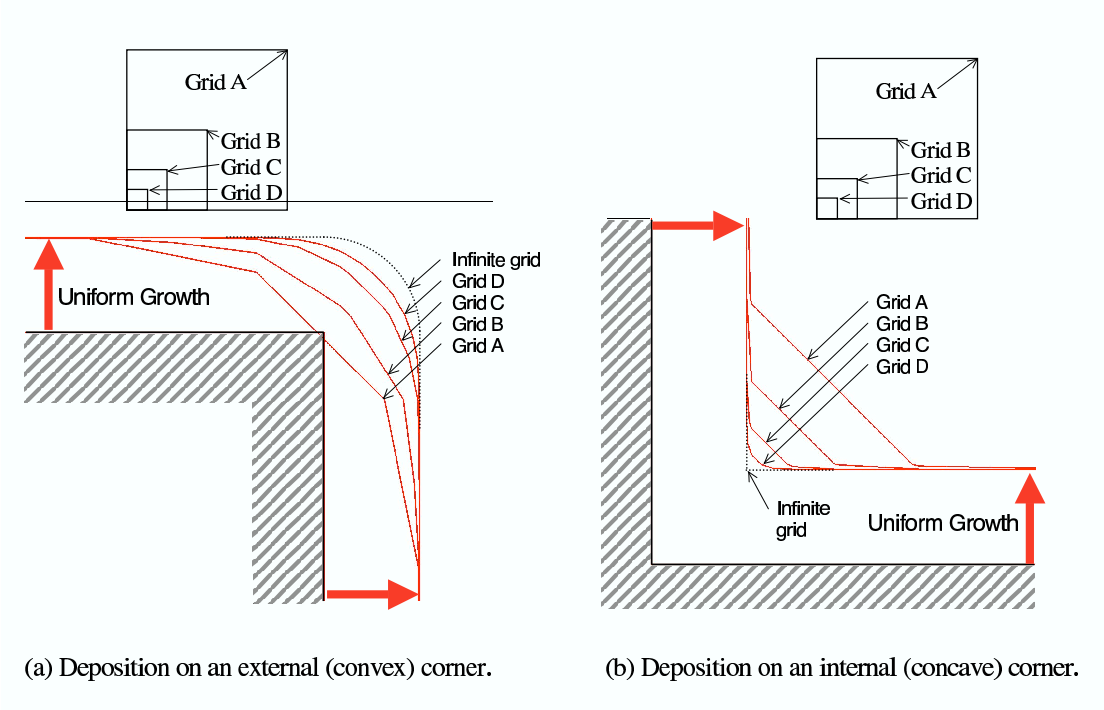
In the level-set method, a domain-spanning signed distance or level-set function, ϕ, is defined; the zero-value contour, or level set, of which conforms to the feature surface. The level-set function is evolved by solving the scalar partial-differential equation,

$$\frac{\partial \phi}{\partial t} + \mathbf{v} \cdot \nabla \phi = 0 \tag{1}$$

over the volume and integrating through time. The velocity, $\mathbf{v}$, in Equation 1 is called the extension velocity and is defined over the entire domain. The extension velocity must be chosen so that the level set of ϕ evolves in such a way that it remains true to the evolution of the physical surface; *i.e.* it is chosen based on the velocity of the surface—the deposition or etch rate in our case. The level set method avoids the debilitations of the explicit methods because the mesh which is used to solve Equation 1 does not deform, so grid-distortion issues are avoided. Likewise, because a volume-defined function is evolved, merging surfaces do not create problems in the method.

When the level-set method is employed, errors can accrue in the computed shapes and locations of the evolving surface from two sources. First, when the signed distance function is represented by a finite set of basis functions, as it must be in computer implementations of the method, insufficient resolution from the use of too few basis functions of the signed distance function can result in an inaccurate resolution of the surface. This can manifest itself as an artificial rounding or smoothing of corner regions, and can only be improved by the costly addition of more basis functions,*i.e.* the mesh must be refined. Figure 1 illustrates this for the idealized problem of uniform deposition near convex and concave corner regions. Likewise, if the extension velocity is not chosen perfectly, the interface will not evolve faithfully to that which the physics demands. Lagrangian approaches, such as front tracking or marker-point methods, can represent regions of high surface curvature and evolve surfaces with greater accuracy, but require complicated *ad hoc* treatments when surfaces merge or pinch-off. This leads to reliability problems on implementation that can require significant user interaction.

To model feature-scale MEMS fabrication processes accurately, it is imperative that both the high-curvature and the merging/pinch-off events be accurately represented. However, as described above, no one method is completely satisfactory under these constraints. In a recent Sandia project, these issues were carefully considered and the level-set method was chosen as the best alternative for a new feature-scale model called ChISELS of both 2D and 3D MEMS fabrication technologies. To capture the evolution of sharp corners and edges, very fine meshes created using locally adaptive



(a) Deposition on an external (convex) corner.

(b) Deposition on an internal (concave) corner.

Figure 1. Illustration of the effect of grid refinement on level-set error near corners.

meshing techniques must be generated to minimize the effects of artificial rounding. Experience with ChISELS suggests that the resolution required by the level-set method to capture the sharp corner regions is much higher than the resolution needed by the species transport and reaction models to resolve spatial variations in the surface growth rate. At present, accurately resolving many 3D problems of interest requires considerable computationally cost. To be parsimonious with computing resources, it is desirable to develop a method that represents the feature with the fewest number of surface elements required to model accurately the surface's evolution while still resolving regions of high curvature.

1.3 Research Goals

Recently, a hybrid particle/level-set method was developed by [1] to improve the accuracy of the standard level-set method. In their approach, a large number of Lagrangian marker particles are randomly placed on each side of an interface. If marker particles initially seeded on one side of the interface are found on the opposite side after a brief interval of time integration, they are flagged. The flagged particles are used to correct the level-set function based on their location relative to the interface as represented by the uncorrected level-set function. Their hybrid method, however, can be quite expensive requiring thousands of particles to work well, and would not reduce computational cost appreciably, if at all, in the applications of interest here. The thrust of the LDRD research described here was to explore the hybridization concept further in hopes of developing an improved approach for modeling feature-scale etching and deposition processes.

After evaluating a variety of possible strategies, two ideas were developed and tested. The first method, described in Section 2, supplements the LS method by introducing Lagrangian marker points in regions of high curvature. Once both the particle set and the LS function have been advanced in time, minimization of certain objective functions adjusts the LS function so that its zero contour is as close as possible to the particle locations. Unfortunately, it was found that the objective-minimization problem was often unexpectedly difficult to solve. Even when a solution could be found, the acquisition of it sometimes proved more costly than simply expanding the basis set of the LS.

The second method explored is a novel particle method called the grid point particle (GPP) method. This method is described in Section 3. Although not a LS method, the GPP method has steps that are similar to some steps of the LS method. A key aspect of the method is a surface rediscretization procedure employed at each time step that maintains a more accurate representation. A 2D version of the method was developed in the ChISELS computer code and tested on a variety of surface evolution problems. Results shown for 2-D problems illustrate the effectiveness of the method and highlight some significant advantages in speed and accuracy over the standard LS method. Generalizing the method to 3D is discussed but not implemented.

2 Method One

The first method to hybridize the level-set and particle methods borrows ideas from curve fitting and optimization. Whereas the method remains substantially a level-set method, fix-ups to the level-set function are made to help alleviate some of the inaccuracies in it; *viz.*, it alleviates inaccuracies in integrating the first-order wave equation, which models the evolution of the level-set function, and those due to the construction of the so-called extension velocity field in Equation 1. The following discussion assumes knowledge of the level-set method and how it is employed in ChISELS; details are available in [6].

When modeling MEMS fabrication processes by the method described in [6], the surface must be rendered for the transport calculations. This is done by representing the surface by elements as a contiguous set of line segments or triangular patches. At the onset of the computation, the surface elements are user provided—the initial level-set function is computed from them. Otherwise, the collection of surface elements is computed by solving for the zero contour of the level-set function. In either case, once an explicit representation of the surface has been made, specified regions of the surface—namely those where curvature is high compared to some tolerance—can be seeded with particles at any density. So the surface has two representations: that of the level-set function and locally that by a collection of surface-embedded particles.

A velocity is assigned to each surface element. The magnitude of the velocity is equal to the local growth or etch rate and the velocity's direction is parallel to the normal vector to the surface element. Particles are assigned a velocity identical to that of the surface element in which it is embedded.

Once a surface velocity field has been been assigned, the level-set function is advanced by one time step by the method reported in [10]. Simultaneously, particles are relocated according to

$$\mathbf{x}_p(t+\Delta t) = \mathbf{x}_p(t) + \mathbf{v}_p \Delta t \tag{2}$$

where $\mathbf{x}_p$ is the particle's location, $\mathbf{v}_p$ is the particle's velocity and Δt is the time elapsed in the time step. In general, the new location of the particles will not lie on the new level set. As it is assumed here that particle representation of the surface is more accurate than the level-set representation, a method has been devised to correct the level set so that it is in the closest possible alliance with the particle locations.

The level-set function is brought into closer accord with the particle locations by minimizing a pair of quadratic objective functions that are subject to a single, linear inequality constraint. the first objective function is a least-squares fit of the level set to the particle locations, *viz.*

$$f_1 = min \sum_{p=0}^{np-1} \left(\psi_p - \phi(\mathbf{x}_p)\right)^2 \tag{3}$$

where np is the number of particles, ϕ_i is the level-set function evaluated at the spatial location of the particle p. ψ_p is the desired value of the level-set function at the particle location, which, of course, equals zero. This objective function forces the zero value contour of ϕ to be as close

as possible—in a least-squares sense—to the locations of the particles, but it also admits a trivial solution where $\phi = 0$ for all values of $\mathbf{x}$. This malady is cured by a second objective function.

The second objective function is the integral over the volume of the square of the so-called Eikonal equation, *viz.*

$$f_2 = min \int_V (\|\nabla\phi\|_2 - 1)^2 \, dV. \tag{4}$$

this function forces the gradient of the level-set function to have as near as possible a unit magnitude throughout the domain. In regions where the gradient does have exactly a unit magnitude, it meets the definition of a signed-distance function [9]. It eliminates everywhere the possibility of a trivial solution.

One linear constraint is added to ensure that ϕ has a unique sign. The constraint is

$$\int \nabla\phi^0 \cdot \nabla\phi dV > 0 \tag{5}$$

where ϕ^0 is the uncorrected level-set function. In order to solve the problem with this constraint, a slack variable is added to Equation 5:

$$g = \int \left(\nabla\phi^0 \cdot \nabla\phi - s^2\right) dV = 0 \tag{6}$$

From Equations 3, 4 and 6, an equation set is formed and solved to get the corrected level-set function. The equation set is

$$\omega\nabla_\phi f_1 + (1-\omega)\nabla_\phi f_2 - \lambda\nabla g = 0 \tag{7}$$

$$g = 0 \tag{8}$$

where ω is a user-prescribed weight factor, λ is a Lagrange Multiplier and only solutions with $s > 0$ are accepted.

Ultimately, the end goal of the hybridized method was to improve upon inaccuracies attendant to the level-set method when a finite basis set is used to represent the level-set function. Thus coarser discretizations, i.e. fewer basis functions, could be enabled and corresponding CPU savings realized. As it turns out, however, the objective-minimization problem proved unexpectedly difficult to solve. Even with Newton linearization and an analytical Jacobian matrix, solutions often could not be found. In addition to the newly written Newton solver, NOX [3], a Sandia-produced nonlinear equation solver, a part of the Trilinos suite, was employed, with equal success. It was determined that, even when a solution could be found, the acquisition of it proved so costly as to make it not a viable alternative to expanding the basis set of the level-set function representation.

3 A Grid-Point Particle Method for Modeling Surface Evolution

In this section an explicit particle method is described for modeling the evolution of surfaces when the surface velocities are always normal to the surface. It is called the grid-point particle (GPP) method because at each time step new surface marker particles are scribed on the surface at the nearest point to the grid points on a uniform grid. A high level overview of the method is first provided in section 3.1, followed by a detailed description in Section 3.2. In this section the method is illustrated in a sequence of three time steps. In the first timestep, the basic algorithm is described by which a surface in motion is advanced in the model. The subsequent two timesteps show how the method handles two special cases that arise when modeling MEMS fabrication processes.

3.1 Model Overview

The GPP method is defined by a sequential process that incorporates, at various points in time, the following key operations:

1. *Surface discretization:* A novel technique described here for creating a unique discrete approximation of any arbitrarily defined surface based on a given background mesh.
2. *Calculation of surface element velocities:* This consists of using an appropriate physics model to calculate normal velocities on all discrete surface elements. In problems of interest here, these are etching and/or deposition models based on surface and gas?phase chemical reactions.
3. *Element-to-point velocity transformation:* A scheme for converting surface element velocities to equivalent surface point velocities.
4. *Constrained Lagrangian movement:* The displacement of surface points during a discrete time step as constrained by certain geometric limits.
5. *Surface reconciliation:* A check for surface overlap locations and, if they exist, the reconciliation of the new topology through local Boolean operations.

To evolve the surface topology in time, a calculation proceeds through a series of these steps. Each calculation has two phases, an initialization phase and a time integration phase. The steps associated with each of these phases are summarized below.

GPP Initialization Phase:

1. Input problem description and run parameters

- **S**: The initial surface
- h: The background mesh length scale
- ε: The subgrid resolution length scale

2. Perform surface discretization (Before evolving in time the initial surface must be approximated by performing an initial surface discretization.)

GPP Time Integration Phase:

Begin time step

1. Calculate surface-element velocities
2. Convert surface-element velocities to point velocities
3. Constrain time-step with local information. Move point locations based on velocity and time step.
4. Reconcile surface topology if surface overlap is detected
5. Construct a new discrete approximation of the surface by performing a surface discretization operation.

End time step

3.2 Details of the GPP method

Details of the GPP method will be described by reference to a series of figures illustrating each substep taken in the overall algorithm as it proceeds during an illustrative 2D calculation. The calculation will begin with an initilization phase during which the user-specified intial state of the surface is discretized based on the background mesh resolution. The calculation then proceeds through three sequential time steps. The first time step illustrates the basic method by which a surface is advanced in the model. The second two time steps illustrate how two important special cases are handled; (1) how to treat multiple surfaces in a cell, and (2) how to treat multiple surfaces in a cell when the surfaces merge or intersect, creating a major change in topology.

3.2.1 Initialization phase

Substep 1 A problem is initialized as illustrated in Figure 2. The boundary between two regions is defined by a fully-resolved surface indicated by the blue line. A background mesh of size h is shown by the vertical and horizontal grey lines. We also define a subgrid resolution length

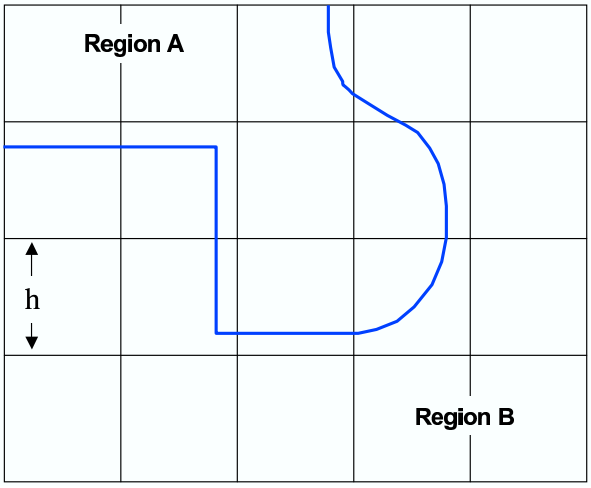


Figure 2. Initialization of illustrative problem

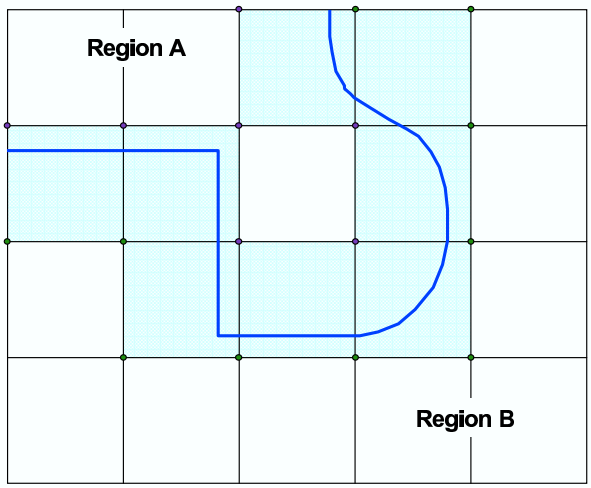


Figure 3. Surface-containing mesh cells

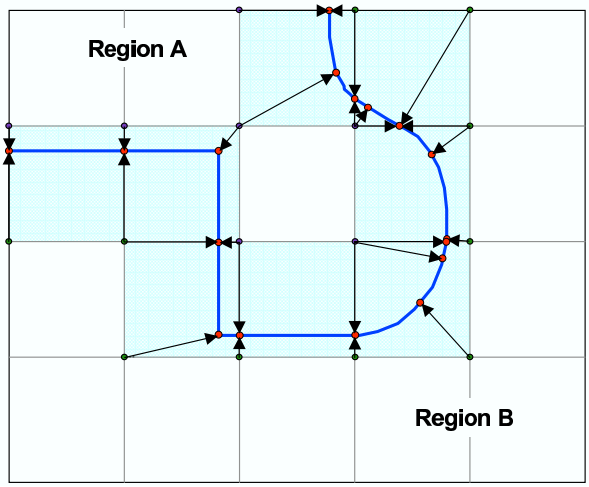


Figure 4. Finding surface discretization points.

scale ε. The value of ε must be smaller than h, and for all calcultions performed here ε = 0.1h. As will be described below, the values of h and ε affect the resolution of the surface discretization algorithm.

Substep 2 (Figure 3) Every cell in the background mesh through which the surface passes is identified and placed in a sequential list. If the surface passes through a cell more than once, the cell information is ”cloned”. Since this does not occur here, a description of what this entails and its purpose will be discussed later.

Substep 3 (Figure 4) In each surface-containing cell, we conceptually draw a line from each corner mesh point to the nearest location on the surface within the cell. Because neighboring cells share common corner points, multiple lines might be drawn from a single mesh point. However, each of these lines is associated with a different surface-containing cell. Note that at this stage multiple surface points may lay on top of each other.

Substep 4 (Figure 5) Next we create a sequential set of unique surface points and associated line segments (surface elements) that define a new (red) surface. In this process, any surface points that are within ε of each other are consolidated into a single point. This creates our discrete approximation to the original (blue) surface, which is now discarded.

Substep 4 completes the initialization phase.

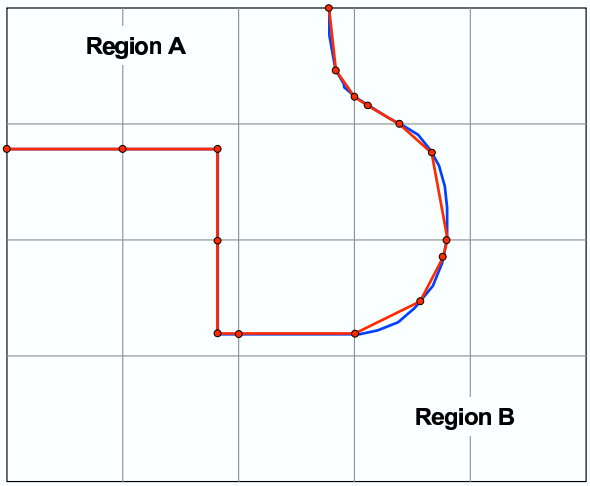


Figure 5. Surface discretization.

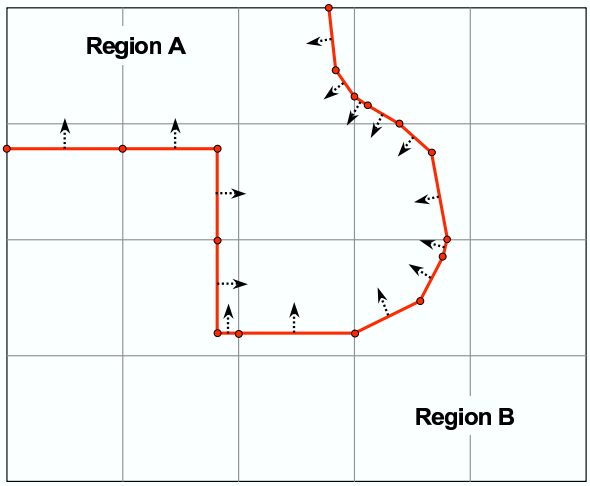


Figure 6. Compute surface-element velocities.

3.2.2 Illustrative time step one

Substep 5 (Figure 6) We begin the time step by computing the velocity on each surface element based on the physics of the problem at hand. For the surface chemistry involved in etching and deposition of microsystems, the direction of each surface velocity is always normal to the surface element.

Substep 6 (Figure 7) In this substep velocity vectors are calculated for each surface element endpoint. These must be calculated as a function of the surface element velocities, and a variety of alternatives can be constructed for doing this. Tests from several choices showed that the procedure used to compute these velocities is important to the overall accuracy and robustness of the method. In all our tests, we found the method described in the following subsection to be very satisfactory for 2D problems.

A method to convert surface-element velocities to point velocities in 2D The method begins by conceptually moving each line segment to a new location based on its specified normal velocity.

If neighboring line segments intersect, then the intersection point defines the vector direction and magnitude of the point shared by the two line segments.

Else,

we compute a velocity vector at each line segment endpoint as the vector average of the velocities on each of the two line segments (A and B) that share the point. This defines the vector direction.

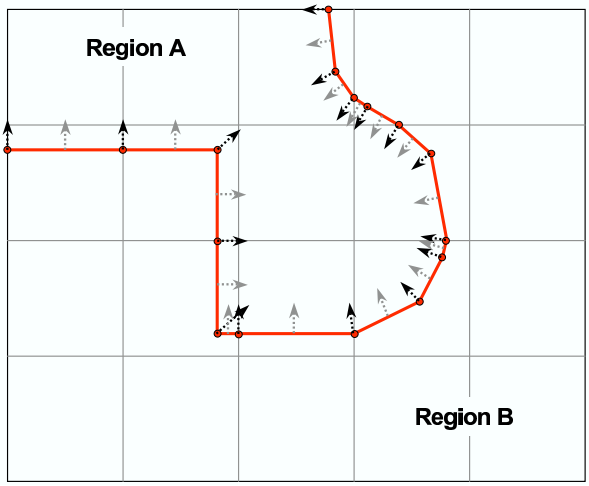


Figure 7. Compute surface-point velocity vectors from surface-element velocities

The vector magnitude, M, is computed as

$$M = \frac{(|V_A| + |V_B|)}{2} + ABS\,(sin(\phi)) * \left[MAX\,(|V_A|, |V_B|)) - \frac{(|V_A| + |V_B|)}{2} \right] \tag{9}$$

where ϕ is the angle formed between line segments A and B, and $|V|$ denotes the magnitude of velocity vector V.

Substep 7 (Figure 8) At this stage we loop over each point on the surface and find if the velocity vector at this point intersects any of its neighboring-point velocity vectors. From this we can compute a maximum time step for surface advancement based on the shortest distance to any intersection point that may have been found.

Substep 8 (Figure 9) All surface points are now moved in a Lagrangian fashion to new locations based on the time step and their respective velocities. In this process element-to-point connectivity is maintained, and a new surface location (here denoted by blue line segments) is thereby obtained. Also, any neighboring points that are within ε of each other are consolidated into a single point.

Substep 9 (Figure 10) Every cell in the background mesh through which the surface now passes is identified and placed in a new sequential list. If the surface passes through a cell more than once, the cell information is cloned. Once again, since this does not occur here, a description of this entails will be deferred.

This is our preliminary discrete approximation for the surface at time t + Δt. Up to this point, the method can be thought of as essentially a string method (see e.g. [cite Sethian]). However, we

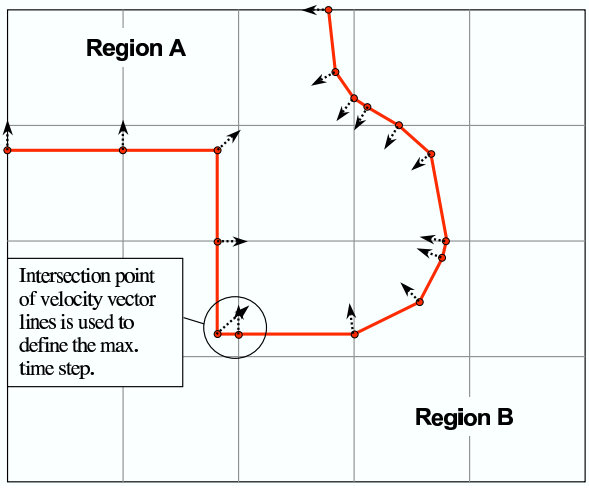


Figure 8. Constrain time step based on geometry.

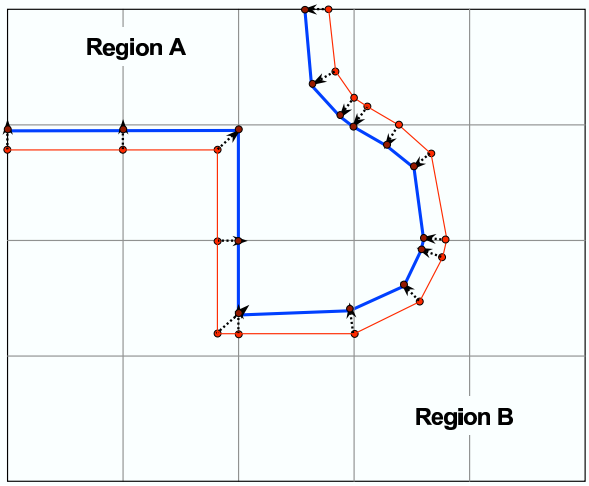


Figure 9. Lagrangian movement of points

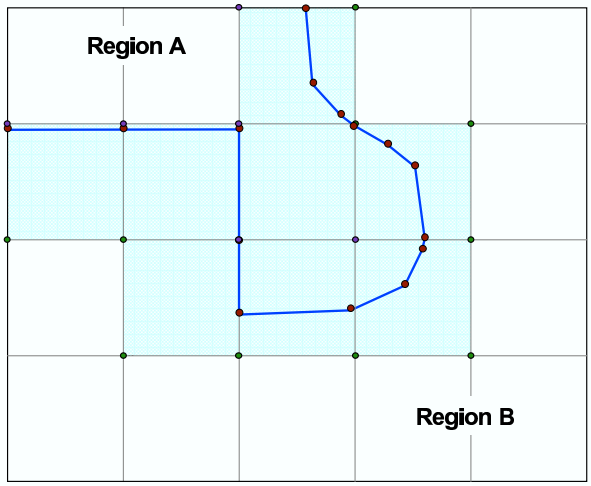


Figure 10. New surface-containing mesh cells.

have not yet adjusted the surface representation based on its relationship to the underlying grid, nor described how this process and the underlying grid is used to address major topological changes as they occur.

Substep10 (Figure 11) In each surface-containing cell, we conceptually draw a line from each corner mesh point to the nearest location on the surface within the cell. Because neighboring cells share common corner points, multiple lines might be drawn from a single mesh point. However, each of these lines is associated with a different surface-containing cell. Note that at this stage multiple surface points may lay very close to or on top of each other.

Substep 11 (Figure 12) The surface discretization is completed after consolidating any surface points that are within ε of each other. This new discrete representation of the surface is shown in red. The blue preliminary representation of the surface (mostly hidden in Figure 12), is now discarded.

The first time step in the time integration phase of the method is now finished.

3.2.3 Illustrative time step two

Substeps 12-14 (Figure 13) The next three steps are now a repeat of what has been described previously in time step 1, and include calculating surface element velocities based on the problem physics, converting the surface element velocities to equivalent surface point velocities, and the determination of a maximum time step for surface advancement based on the shortest distance to any intersection point that may have been found. For brevity, these are illustrated together in Figure 13.

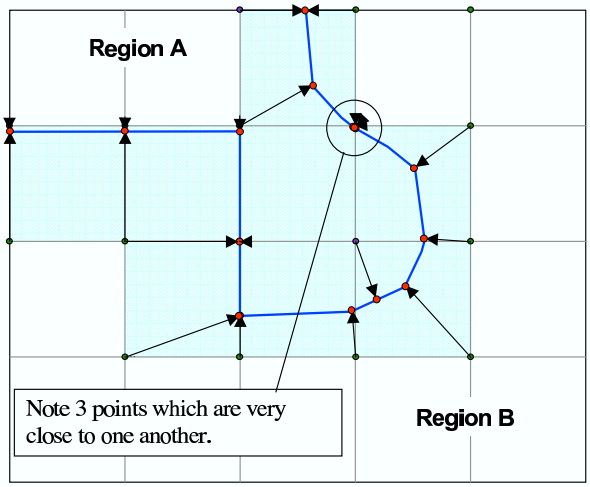


Figure 11. New surface discretization points.

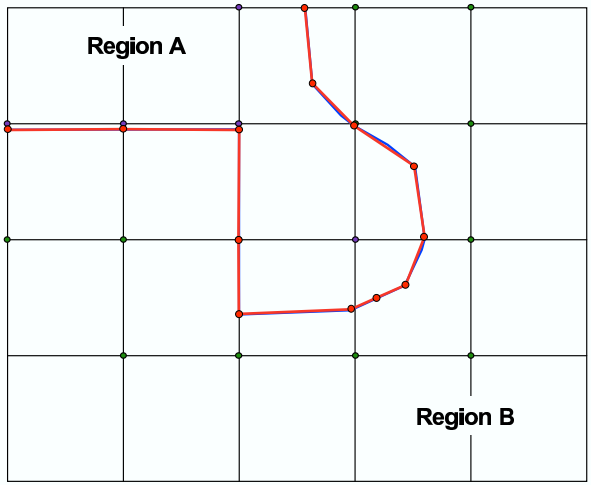


Figure 12. New surface discretization after point consolidation.

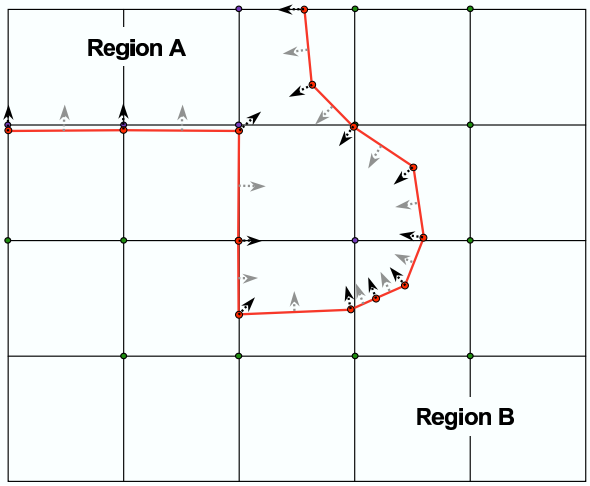


Figure 13. Calculation of surface velocities, point velocities, and maximum time step (time step 2).

Substep 15 (Figure 14) All surface points are once again moved in a Lagrangian fashion to new locations based on the time step and their respective velocities. Because element-to-point connectivity is maintained, a new surface location is obtained as denoted by blue line segments. Also, any neighboring points that are within ε of each other are consolidated.

Substep 16 (Figure 15) Every cell in the background mesh through which the surface now passes is identified and placed in a sequential list. If the surface passes through a cell more than once, as occurs twice in Figure 16, the cell information is cloned, and the cell is marked as a 2-surface cell. Its information is now found twice in the sequential list. If the surface passes into a cell more then twice, the cloned cell is simply created every additional time. Each cloned cell is treated as unique for the purposes of the next substep in the algorithm.

Substep 17 (Figure 16) In each surface-containing cell, we once again conceptually draw a line from each corner mesh point to the nearest location on the surface within the cell. Cloned cells are treated as distinct in this step, and the portion of the surface that is associated with each cloned cell is not seen by its other copies. As before, corner points shared by neighboring cells are treated as distinct so that multiple lines might be drawn from a single mesh point. However, each of these lines is associated with a different surface-containing cell.

Substeps 18 and 19 (Figure 17) Substep 18 consists of consolidating any surface points that are within ε of each other. The new discrete representation of the surface is shown in red. Because cloned cells are present, we must now check for overlap of surfaces. In substep 19 we check to see if any line segments from surfaces in clones cells intersect each other. In this case no overlap exists.

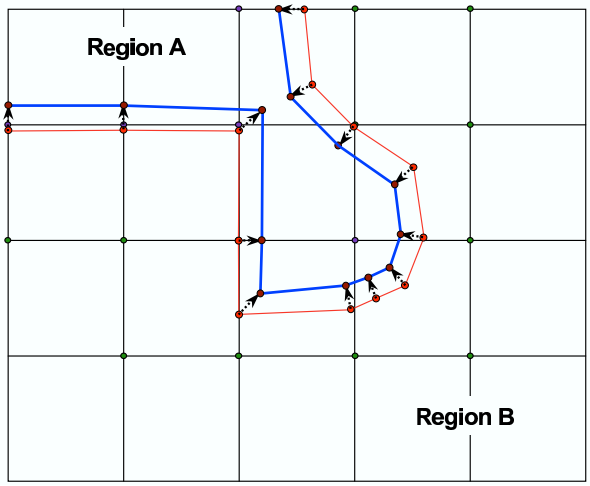


Figure 14. Lagrangian movement of points (time step 2).

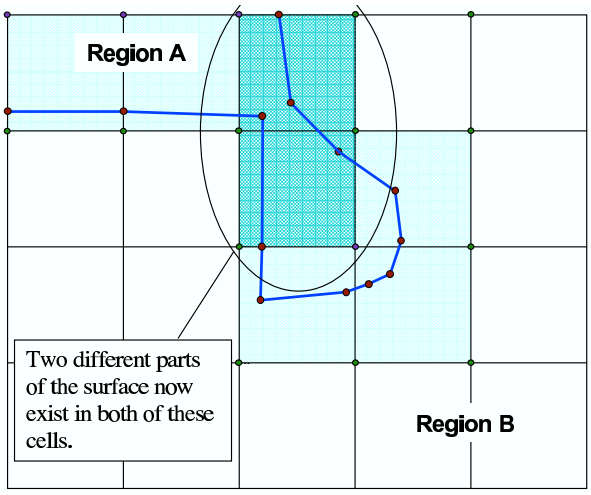


Figure 15. Identification of surface-containing mesh cells and the need for cell cloning.

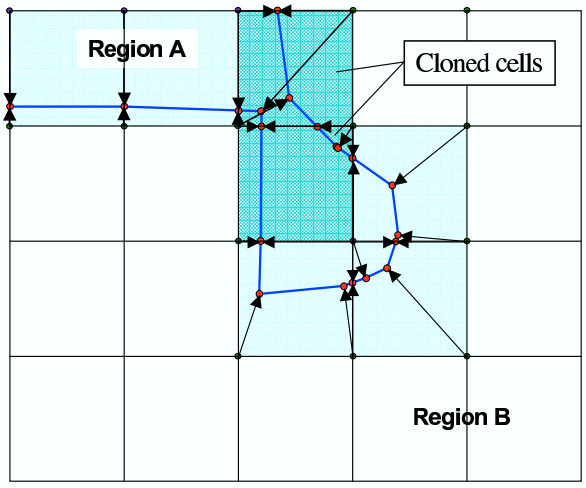


Figure 16. New surface-discretization points.

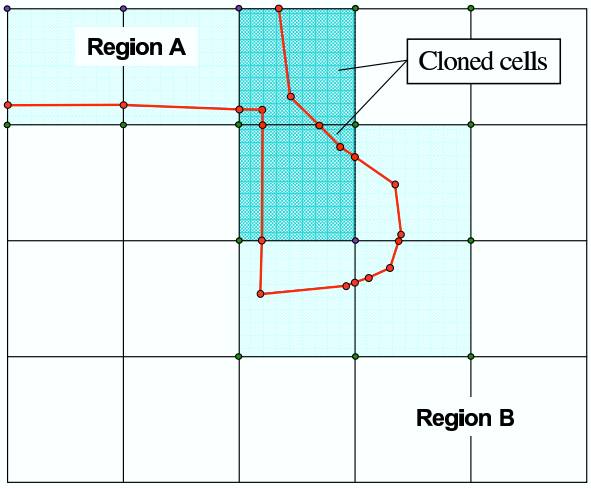


Figure 17. New surface discretization after point consolidation.

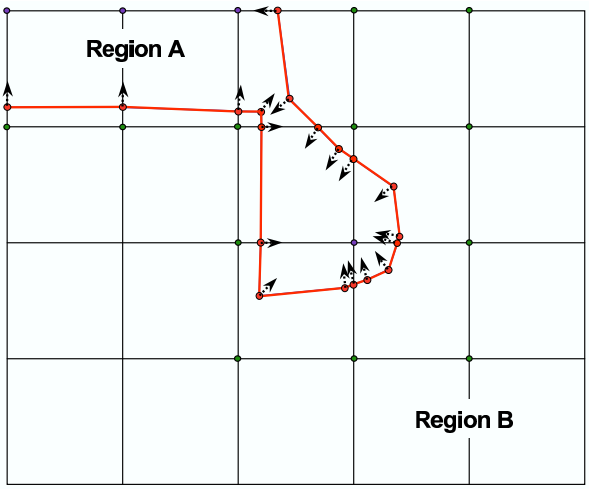


Figure 18. Calculation of surface velocities, point velocities, and maximum time step (time step 3).

The second time step in the time integration phase of our illustrative calculation is now complete.

3.2.4 Illustrative time step three

Substeps 20-22 (Figure 18) The next three substeps include calculating surface element velocities based on the problem physics (not shown here in Figure 18 because of the small element sizes), converting the surface element velocities to equivalent surface point velocities, and the determination of a maximum time step for surface advancement based on the shortest distance to any intersection point that may have been found.

Substep 23 (Figure 19) All surface points are once again moved in a Lagrangian fashion to new locations based on the time step and their respective velocities. Because element-to-point connectivity is maintained, a new surface location is obtained as denoted by blue line segments. Also, any neighboring points that are within ε of each other are consolidated.

Substep 24 (Figure 20) Every cell in the background mesh through which the surface now passes is identified and placed in a sequential list. If the surface passes through a cell more than once the cell information is cloned, the cell is tagged, and its information is now found multiple times in the sequential list. As explained before, each cloned cell is treated as unique for the purposes of the next substep in the algorithm.

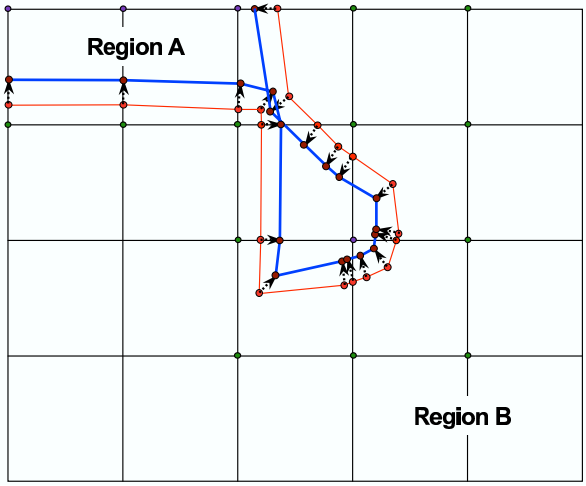


Figure 19. Lagrangian movement of points (time step 3).

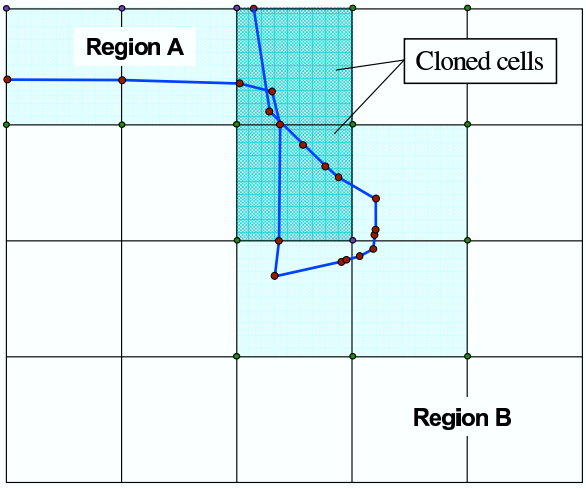


Figure 20. Identification of surface-containing mesh cells and the need for cell cloning.

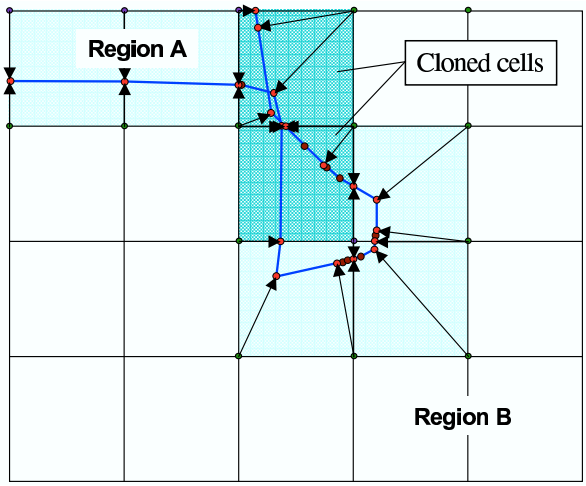


Figure 21. New surface-discretization points.

Substep 25 (Figure 21) New surface discretization points are found by conceptually drawing a line from each corner mesh point to the nearest location on the surface within each surface containing cell. Cloned cells are treated as distinct in this step, and the portion of the surface that is associated with each cloned cell is not seen by its other copies. The distinction between the old Lagrangian particle points (to be discarded) and the new discretization points is particularly clear in Figure 21.

Substep 26 and 27 (Figure 22) Substep 26 consists of consolidating any surface points that are within ε of each other. The new discrete representation of the surface is shown in red. Because cloned cells are present, we must now check for overlap of surfaces. In substep 27 we check to see if any line segments from surfaces in clones cells intersect each other. In this case two such intersection points are found, indicating that surface overlap now exists.

Substep 28 (Figure 23) All surface elements contained in the overlap loop are removed through a 2D Boolean operation. The operation consists of first creating two closed loops, one from each cloned cell, that are formed by the surface line segments and the cell boundary. When they are overlaid on top of each other the two loops intersect in space and three new loops can be formed. The center loop corresponds to the overlap region, and from a computational standpoint, can now be discarded. The interior elements of the remaining two loops define the two new sets of surface elements that are needed. In the calculations performed here, these simple 2D boolean operations are performed by calling appropriate subroutines in the 2D boolean library described in [4]. Note that because the number of surface elements is always very small, these operations are computationally very fast.

This substep marks the completion of the third time step in the time integration phase of our illustrative calculation. Figure 24 provides a comparison of our discrete representation of the

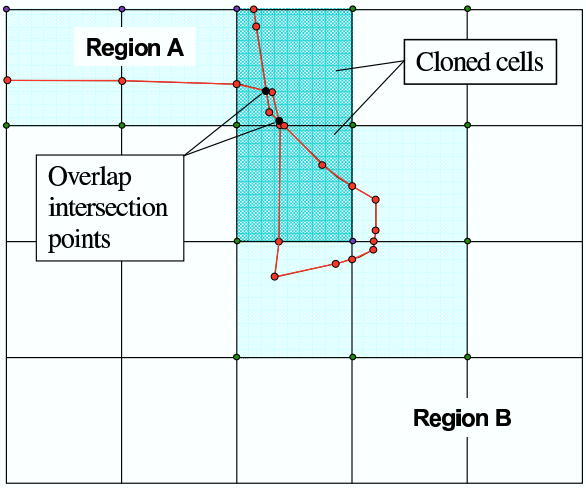


Figure 22. Consolidation of discretization points and identification of overlap intersection points.

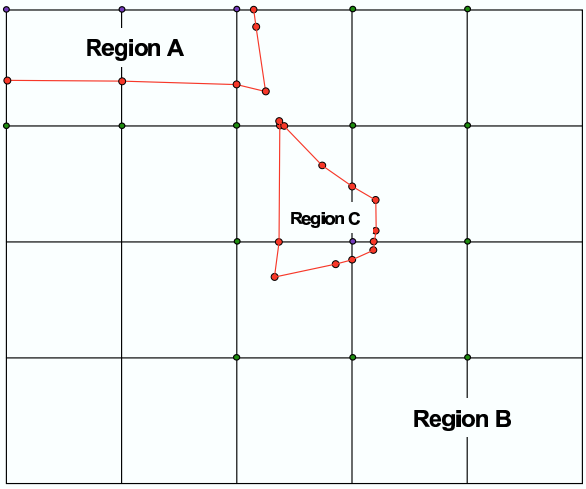


Figure 23. Removal of overlap surface elements to create final surface discretization.

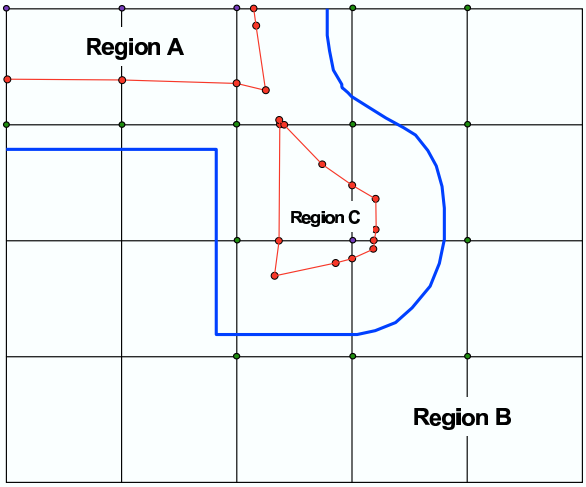


Figure 24. The discrete surface representation at the end of time step 3 compared to the original surface.

surface at this point with the original surface shown in Figure 2.

3.3 Example calculations with the GPP method

The GPP method was coded and implemented as an option in ChISELS [5], a Sandia code for modeling the evolving surface topology during MEMS fabrication. ChISELS was originally written to use the LS method, so adding the GPP method facilitated comparing its performance in a very direct way with that of the GPP model.

3.3.1 Uniform deposition on and in a 2D notch

The initially specified surface is a simple 2D notch (or trench), as illustrated by the black surface seen in Figure 25. An idealized surface deposition process that yields extremely uniform surface growth rates is specified. Calculations are run using both the level set method and the GPP method, with varying degrees of mesh resolution. Figure 25 shows the predicted state of the surface after deposition has continued for a suficient time to build up a fairly significant growth layer.

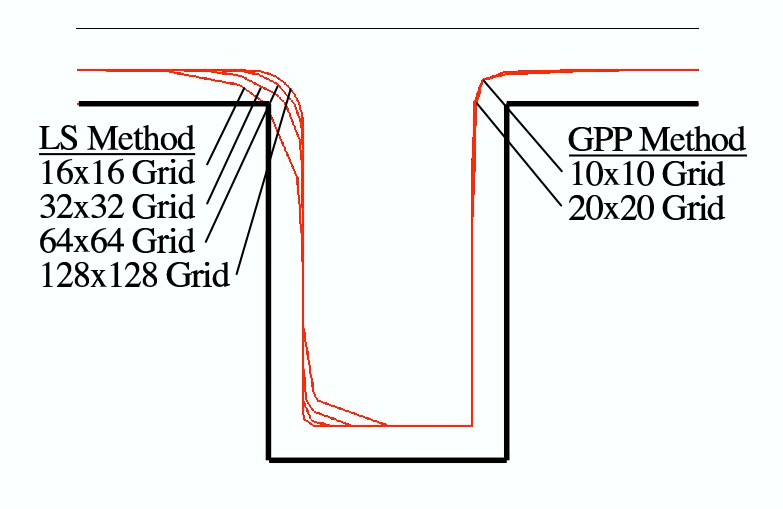


Figure 25. The effect of grid resolution in test problem 1.

Surface contours obtained using the LS method with varying degrees of mesh refinement are shown on the left side of the figure. As the numerical mesh is refined, the error introduced near corners is reduced. But only the calculation with the finest mesh refinement (corresponding to an equivalent uniform mesh of 128x128) appears to be nearing a mesh-converged solution.

Surface contours obtained using the GPP method are shown on the right side of this figure. In the GPP case only two levels of mesh refinement are shown, corresponding to a 10x10 and a 20x20 uniform background mesh. Here we can observe that the solution is very nearly a mesh-converged solution with the 20x20 grid, as the only differences between the two runs occurs in the region around the upper corner of the notch, and these differences are very small.

This test calculation demonstrates the ability of the GPP method to more accurately represent the deposition process near sharp corner regions. This is primarily due to the ability of the GPP

method to pick out the points on the surface where large surface curvature is present, and place surface elements in these regions in such a way that the curvature is captured in a fairly optimal way. Thus for the same level of accuracy, many fewer surface elements are required with the GPP method than with the LS method.

3.3.2 Uniform deposition in a complex 2D geometry

In this second test problem the initially specified surface is a more complex 2D geometry. Once again, an idealized surface deposition process that yields extremely uniform surface growth rates is specified and calculations are run using both the level set method and the GPP method. In this case, the LS method was run with a mesh resolution equivalent to a 32x32 background mesh. (Adaptive mesh refinement of the LS grid enables the code to refine to this level only near the surface itself.) The GPP method was run using a somewhat coarser 28x28 grid so that the total number of surface-elements, about 90, would be essentially the same for both methods.

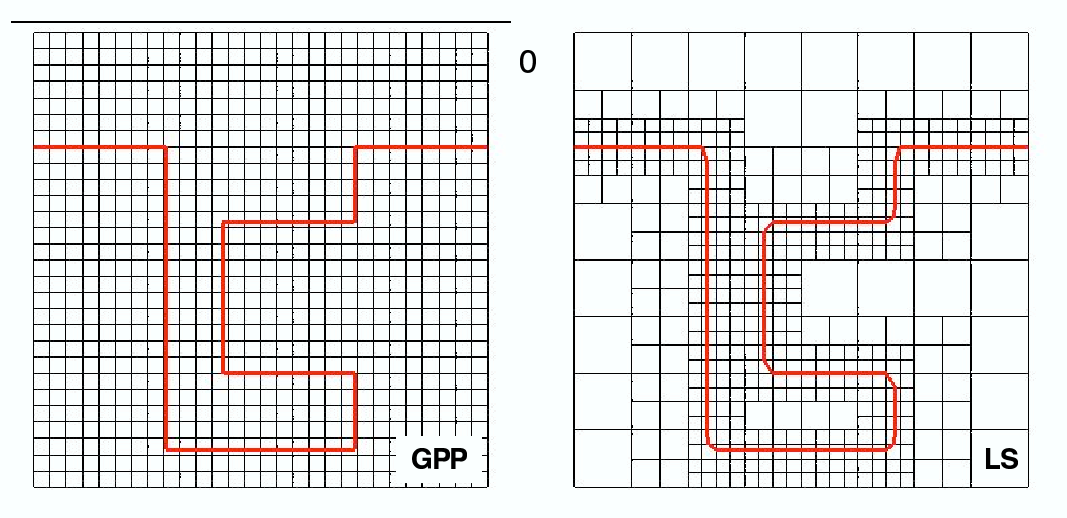


Figure 26. GPP and LS discretized surfaces after reconstruction of the initial surface in test problem 2.

Figure 26 shows the respective background meshes and the initial surface representation for this problem using both methods – the GPP method on the left, and the LS method on the right. Here we can see that in reconstructing the initial idealized surface, the GPP method captures the sharp corners exactly, while the LS method effectively rounds the corners at a scale proportional to the mesh size.

The next five figures show the background mesh and the state of the evolving surface using both methods at five subsequent points in time in the calculation.

In Figure 27 we see the surface at step 25. The major differences seen here are the radius of curvature at corners.

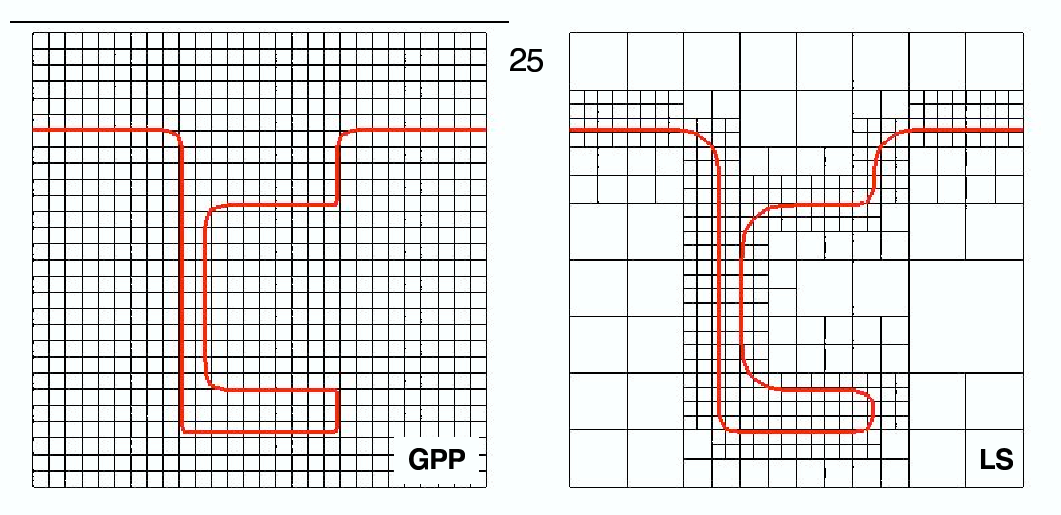


Figure 27. GPP and LS discretized surfaces after step 25 in test problem 2.

Figures 28 and 29 correspond to time steps 30 and 35. Of note here is that portions of the left and right vertical sides are approaching each other. In the GPP method, the right vertical side remains vertical. In the LS method an artificial bulging of the right side-wall is observed as the spacing between the left and right sidewalls becomes less than the mesh spacing. This happens because the surface reconstruction method, which relies on interpolation from a signed-distance function known at discrete points, cannot accurately reconstruct the separation distance between the two approaching walls when the distance between them approaches the mesh size. As can be seen, the GPP method does not suffer from this problem.

Figures 30 and 31 correspond to time step 40 and 45 in the calculation. Here the surface geometry has pinched off, and a small void region is left isolated while the upper surface continues to move with the deposition. Note that both the timeing of when the pinch-off occurs, as well as the shape of the resulting voided region are very different because of the numerical error introduced by the LS method.

This test problem and the figures illustrating the results demonstrate that, for the same number of surface-elements, the GPP method provides a more accurate representation of the temporal surface evolution process than the LS method. One reason for this is that the GPP method automatically concentrates surface elements in regions where large surface curvature (relative to the grid)) is present. Another important factor is the surface reconstruction method. In LS, the surface must be found by interpolation from the local mesh point values of the signed distance function. This introduces a numerical smearing of order the mesh size when surface regions merge. In GPP, the surface is reconstructed directly from current surface-element location information, using background mesh points only as geometric reference points.

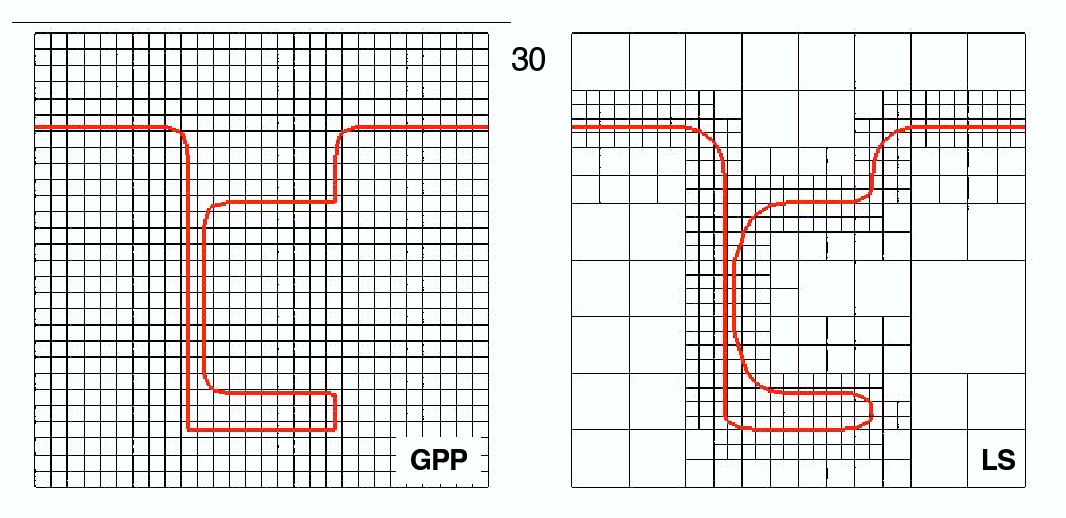


Figure 28. GPP and LS discretized surfaces after step 30 in test problem 2.

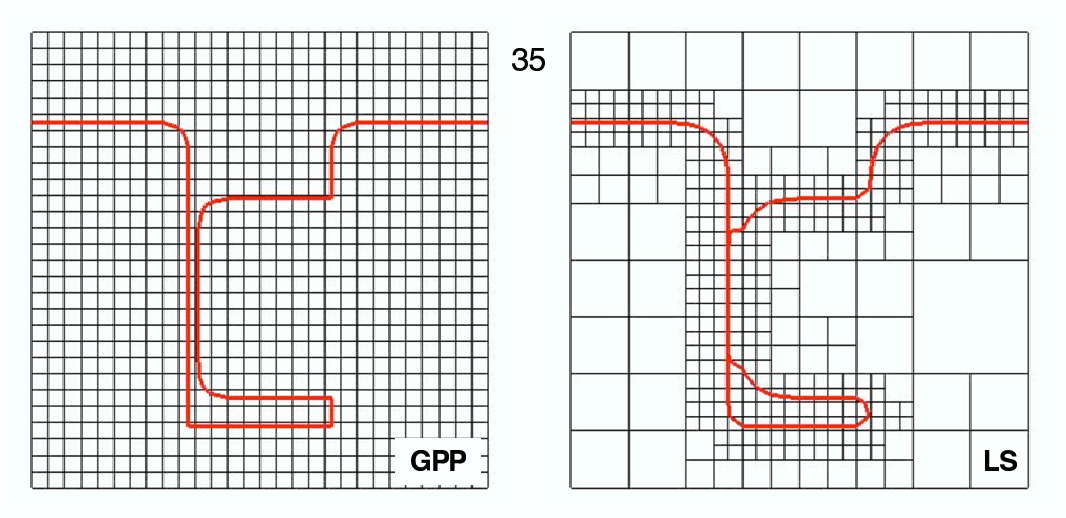


Figure 29. GPP and LS discretized surfaces after step 35 in test problem 2.

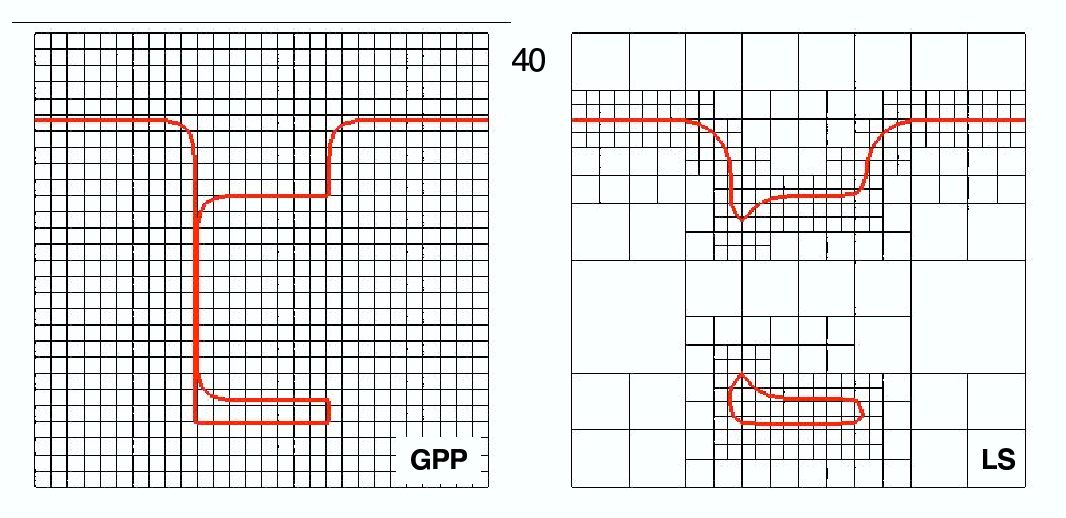


Figure 30. GPP and LS discretized surfaces after step 40 in test problem 2.

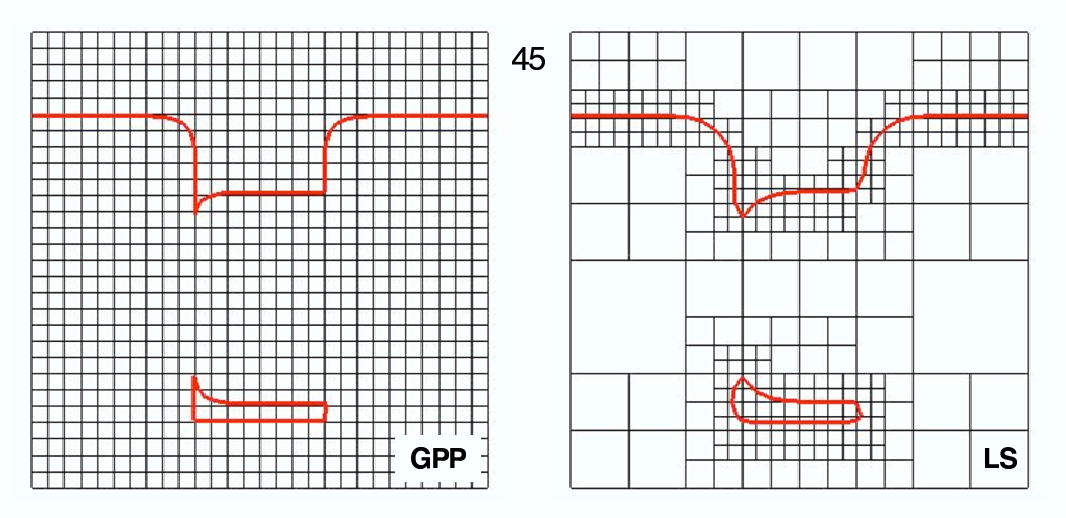


Figure 31. GPP and LS discretized surfaces after step 45 in test problem 2.

3.4 Extending the GPP method to 3D

In this report the GPP method has been formulated in 2D. Issues and ideas for extending the method for use in 3D are discussed here.

A 3D GPP method would consist of basically the same sequence of conceptual substeps as those for 2D problems. However, certain important differences in the geometric nature of the problem must be addressed in 3D. The first difference concerns the 3D discretization process.

In 2D, the surface discretization step identifies (A) the set of four surface points in each 2D cell that are closest to the four cell corners, and (B) all intersection points between the surface and the 2D cell side faces. These points are often the same, thus the number of unique points may be as few as two.

In 3D, the surface discretization must also have two parts. In part A, we must compute a set of eight points in each 3D cell that are closest to the eight cell corners. In part B, the intersection of the surface which each face of the 3D mesh must be identified. In 2D, the intersection is only a point, but in 3D this intersection is a 2D curve. Thus for each face intersected by the surface, a 2D surface discretization must also be performed to properly discretize the side-face intersection curve. Given the collection of points so identified, non-unique points are discarded, leaving the remainder as the basis for defining the discrete representation of the surface within the 3D cell.

The second important difference is faced at the end of the discretization process. In 2D, the discretization points are easily connected into a sequential set of up to 3 line segments. In 3D, a more complex algorithm must be used to assemble the points into a contiguous set of surface triangles. The well known Delaunay triangulation method is an obvious candidate for this, as it is robust and well understood. However, it should be noted that when the process is complete for all cells, the collection of surface-containing cells cannot be stored as a simple 1D list of sequential cells. Instead, the cell information must be stored in an unstructured data format that contains surface connectivity information – such as is maintained for finite elements analysis.

In section 3.2.2 above, the calculation of point velocities for the 2D algorithm was described. In 2D only two surface elements can share a given point, but in 3D many surface elements can share the same point. For 3D, Equation (9) must therefore be generalized to account for contributions from each surface element sharing a given point.

In 2D, neighbor point entanglement is easily prevented by limiting the time-step as described in section 3.2.2. This is straightforward because the velocity vectors all exist in the same 2D geometric plane. However, this is not true 3D and therefore this same method cannot be used to limit the time-step in 3D. An appropriate explicit solution to this issue in 3D has not been worked out by the authors. Without an explicit formula, the time step would need to be estimated, and the resulting surfaces checked for entanglement after the fact. If entanglement was detected, the time step would have to be reduced and the advection substep repeated.

When a given cell is intersected more than once by a given surface, the process of determining if the different surface parts intersect changes from being one of checking for the intersection of

line segments in 2D, to the process of checking for the intersection of triangular surface elements in 3D.

When 3D cloned cells exist and a surface intersection is determined, the resulting boolean operation that must be performed to change the topology is also 3D in nature. These 3D operations are more expensive than their 2D analogs. However, because the number of surface elements is bounded by a fairly small number, the cost should still be reasonable.

Although the additional complications just noted are non-trivial, the numerical operations required are all commonly performed in other contexts, and the remaining challenges noted do not appear insurmountable. The motivation for pursueing this approach in 3D is that the key advantages evident in 2D would be expected to also extend into the 3D realm. In particular, the number of surfaces required to model the evolution of a surface with sharp corners and edges would be significantly reduced because the method automatically finds these regions of high curvature and places marker points on them. The most important question to be answered is whether the additional complexities mentioned could be addressed in a completely robust way, so as to handle all cases in all situations without running into numerical exceptions that would cause the application code to fail.

References

[1] Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comp. Phys*, 183:83–116, 2002.

[2] Brian Gavin Higgins. *Capillary Hydrodynamics and Coating Beads*. PhD thesis, University of Minnesota., 1980. Published by University Microfilms International, Ann Arbor, MI.

[3] http://software.sandia.gov/nox.

[4] C. L. McBride and V. Yarberry. A 2D geometry Boolean library Rev. 1. Report No. SAND2005-XXXX, Sandia National Laboratories, To be published in 2005.

[5] Lawrence C. Musson, Steven J. Plimpton, and Rodney C. Schmidt. MEMS fabrication modeling with ChISELS: A massively parallel 3D level-set based feature scale modeler. In *NanoTech2003*, 2003.

[6] Lawrence C. Musson, Steven J. Plimpton, and Rodney C. Schmidt. Length-scale modeling of low-pressure and plasma-enhanced cvd mems fabrication processes. In *Technical Proceedings of the 2004 Design, Test, Integration, and Packaging of MEMS / MOEMS Conference*, 2004.

[7] W. Noh and P. Woodward. A simple line interface calculation. In *Fifth International Conference on Fluid Dynamics*, 1976.

[8] Stanley Osher and Ronald P. Fedkiw. Levelset methods: An overview and some recent results. *J. Comp. Phys*, 169:463–502, 2001.

[9] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, New York, USA, 1999.

[10] John Strain. Semi-lagrangian methods for level-set equations. *Journal of Computational Physics*, 1999.

DISTRIBUTION:

1	Corey L. McBride Elemental Technologies 357 North 50 South American Fork, UT 84003	
5	MS 0316	Lawrence C. Musson, 9233
2	MS 0316	Rodney C. Schmidt, 9233
1	MS 0316	Sudip Dosanjh, 9233
1	MS 1110	Steve Plimpton, 9212
1	MS 0123	LDRD office (Donna L. Chavez), 01011
1	MS 9018	Central Technical Files, 8940-2
2	MS 0899	Technical Library, 4916
2	MS 0619	Review & Approval Desk, 4916