TAM
Theoretical and Applied Mechanics
University of Illinois at Urbana-Champaign

TAM Report No. 1040
UILU-ENG-2004-6001
ISSN 0073-5264

A hybrid level-set method in two and three dimensions for modeling detonation and combustion problems in complex geometries

by

Sunhee Yoo
D. Scott Stewart

February 2004

A Hybrid Level-Set Method in Two and Three Dimensions for Modeling Detonation and Combustion Problems in Complex Geometries

S. Yoo and D. S. Stewart§

Theoretical and Applied Mechanics, University of Illinois, 216 Talbot laboratory, 104 S. Wright St. Urbana, IL 61801, USA

Abstract. We present an accurate and fast wave tracking method that uses parametric representations of tracked fronts, combined with modifications of level-set methods that use narrow bands. Our strategy generates accurate computation of the front curvature and other geometric properties of the front. We introduce data structures that can store discrete representations of the location of the moving fronts and boundaries, as well as the corresponding level set fields, that are designed to reduce computational overhead and memory storage. We present an algorithm we call Stack Sweeping to efficiently sort and store data that is used to represent orientable fronts. Our design and implementation feature two reciprocal procedures. The first is called *forward* "front parameterization" and constructs a parameterization of a front given a level-set field. The second is called a *backwards* "field construction", and constructs an approximation of the signed normal distance to the front, given a parameterized representation of the front. These reciprocal procedures are used to achieve and maintain high spatial accuracy. Close to the front, precise computation of the normal distance is carried out by requiring that displacement vector from grid points to the front be along a normal direction. For front curves in a two-dimensional level-set implementation, a cubic interpolation scheme is used and G^1 surface parameterization based on triangular patches is constructed for the three dimensional level-set implementation to compute the distances from grid points near the front. For remote grid points in the band, a less accurate method is used for both implementations. Boundary conditions at wall and internal boundaries are implemented. We present examples from the resulting code to the applications of detonation shock dynamics and dendritic solidification.

1. Introduction

We have long-standing interests in simulating detonation and combustion phenomena, related to experimental configurations and engineering devices in complex two and three-dimensional geometry. A representative example of such an experiment and a corresponding simulation are shown in Figures 1 and 2. Figure 1 shows an assembly

§ To whom correspondence should be addressed (dss@uiuc.edu)

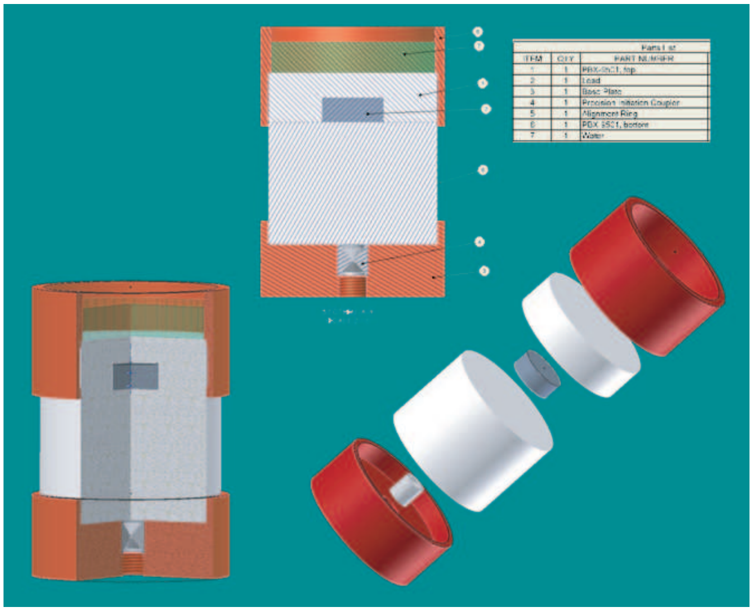


Figure 1. Assembly sketch of the DSD validation experiment at Eglin AFB carried by D. Lambert. Used with permission of Dr. Lambert

drawing of detonation ”wave shaper”, an explosive device designed by D. Lambert [32] to measure the detonation dynamics of the test explosive. The white material in Figure 1 is a condensed explosive like PBX9501. The bottom boundary is copper. The side boundaries are partially unconfined and copper. The top boundary is water. The gray disk embedded in the charge in this instance is lead. A small detonator and booster pellet is placed at the bottom of the charge to initiate the detonation.

In the experiment the detonation is ignited at the bottom by firing the detonator; the detonation shock front propagates through the explosive from the bottom and diffracts around the inert disk. As the detonation passes over the inert (lead) disk the detonation shock develops a hole. After the detonation passes over the disk, the hole heals itself and the subsequent oblique collision of the retarded portions of the detonation shock produces extraordinarily high pressures in the interior of the charge. The experiment records the times and positions when the detonation shock breaks out of the top of the charge. In the related applications, the high pressure generated at the center of the wave shaper is used to precisely cut or punch a hole through the object placed against it at the top of the charge.

Figure 2 shows the result of a 2-D axi-symmetric simulation of the same experiment that displays the shock fronts at fixed times (measured in microseconds). In this display the leading shock motion is of primary interest, but note that the detonation interacts with all of the confining materials at the explosive/confinement interface. If the shock speed is known for example, the shock relations (that use the equation of state for the explosive) can be used to calculate the shock pressure

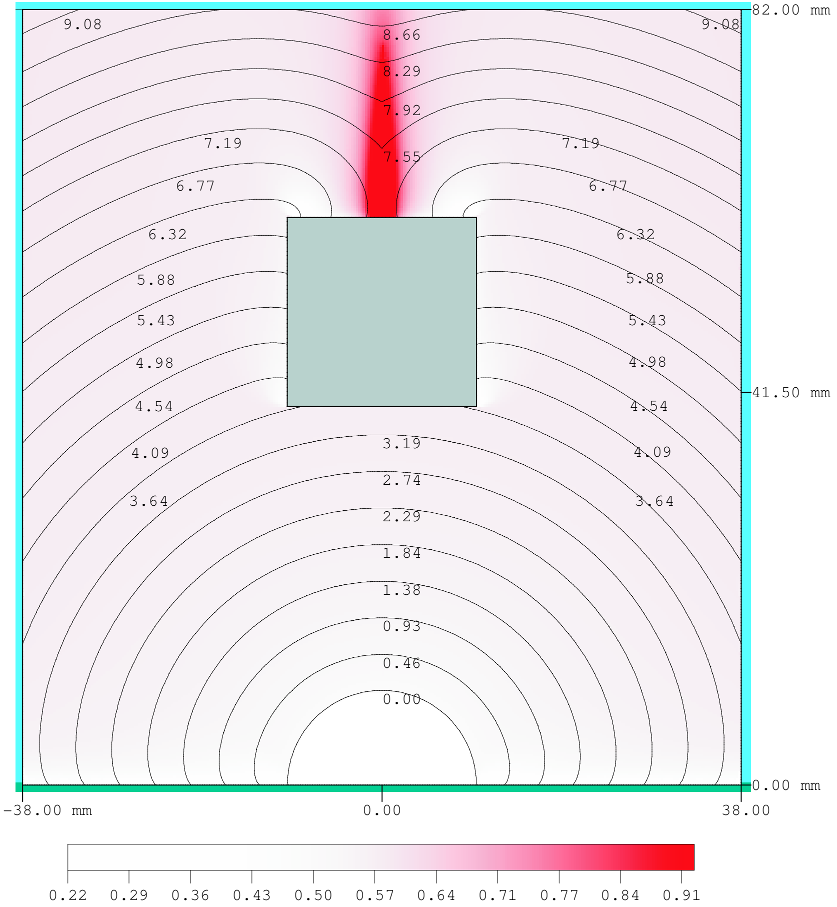


Figure 2. Shock time of arrival field for a wave shaper. The shocks are shown at the times labled, measured in microseconds. The shock pressure field is shown by the gray-scale contours, measured in units of 100 GPa.

right behind the shock as it crosses any point in the explosive charge. Hence one can construct a map of shock pressures generated in the interior of the charge and an example of such a map is shown in Figure 2.

Figures 3a-b and 4 show a sequence of the motion of the shock when the inert disk is placed off the axis of the cylinder, in which case 2-D symmetry is destroyed and 3-D simulation is required to model the experiment. Importantly, both 2-D and 3-D simulations show the change in topology of the shock front even for this simple experiment. The detonation wave tracking simulations shown in Figures 2, 3 and 4 are computed with the algorithms described in this paper.

In most applications it is important to track the shock or combustion front through the device and follow its interaction with other parts of the device that move or are static. In some cases one has an independent motion rule for the detonation shock or the combustion front (such as surface motion under the influence of curvature) and the calculation of the surface motion is called ”wavetracking”.

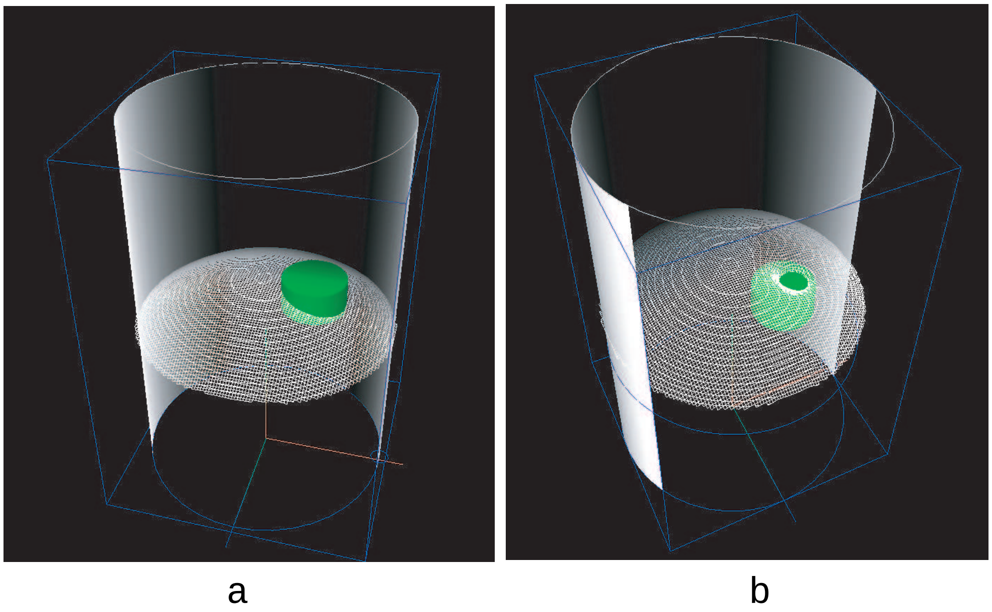


Figure 3. Detonation shock tracking simulation of the experimental device shown in Figure 1, for two different times

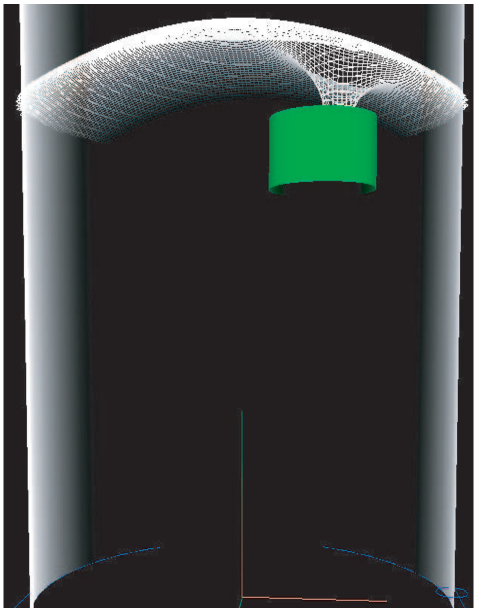


Figure 4. Detonation shock tracking simulation just after the shock passes the embedded disk

Sometimes the motion of the front is determined by solving a coupled problem where the interface separates two adjacent regions and one must solve field equations on either side of a flame or phase transformation interface across which jump condition must be satisfied. Some formulations for flame propagation and binary solidification lead to generalized Stephan problems that are representative mathematical models that require that the front motion be solved simultaneously with the evolution of the fields on either side.

Engineering devices that have explosive elements have many parts and layers of different materials that can compress and move. Each material interface (say), can be regarded as a front that must be tracked. In a compressible flow model, the motion of the material interface is specified by the normal velocity of the interface. (An expanded version of our introductory example is a multi-material simulation that solves for the motion of the detonation shock and the material states in the copper layer and embedded disk simultaneously.) High pressure, multi-material codes are called hydrocodes. Interface treatment is a fundamental part of a hydrocode. The highly accurate and efficient representation of complex interfaces are central to their improvement. Treatments of complex interfaces also arise in combustion applications associated with turbomachinery, or micro combustion devices of current interest that don't necessarily involve high pressures and large material deformation except in the fluids.

Interface representations to be used with higher-order schemes for the materials, must be higher order and have sufficient accuracy. Because of cost considerations in 3-D, it is important to use surface representations instead of a full-field representations. Thus we focus on narrow band level-set methods, where the surface is stored as a zero level-set function inside a narrow region of a fixed thickness. The thickness of the band, measured normal to the embedded front, is taken to be some integer multiple of the grid length. The band region can then also be used as a region of interpolation of field values into (ghost) boundary regions or to sample field data to extrapolate values to the front itself.

For 3D applications in particular, the memory savings achieved with narrow band methods are substantial. Consider a 3D sphere whose surface is near the boundary of the computational domain. At a resolution of $21^3 = 9206$ with $b = 4$ the narrow band takes up 99% of the total number of domain grid points. At the resolution of 201^3 with $b = 4$ the narrow band takes up 14.6% of the total number of grid points and at the resolution of $401^3 = 4,151,588$ with band width $b = 4$ the narrow band takes up 6.4% of the total number of grid points in $\mathcal{D}$. For a 3D simulation at the resolution of 401^3 is the modest resolution at best and the memory savings are dramatic. The use of narrow bands are essential for cost efficient computation in 3D.

1.0.1. A Hybrid Level-Set Method Our method uses parametric representations of tracked fronts, combined with modifications of the narrow band level-set methods and hence is a hybrid level-set method. Higher order representation of the front enables computation of the front curvature, and other geometric properties. We necessarily introduce data structures that can store discrete representations of the location of the moving fronts and boundaries, as well as the corresponding level set fields, that are designed to reduce computational overhead and memory storage. We developed an algorithm we call "Stack Sweeping to efficiently sort and store data that is used to represent orientable fronts and be very fast. A complete account intertwines numerical and software design issues such as fast search and storage procedures.

Our design and implementation specially feature two reciprocal procedures. The *forward* "front parameterization" constructs a parameterization of a front given a level-set field. The *backwards* "field construction", constructs an approximation of the signed normal distance to the front, given a parameterized representation of the front. These reciprocal procedures are used to achieve and maintain high spatial accuracy. Close to the front, precise computation of the normal distance is carried out by requiring that displacement vector from grid points to the front be along a normal direction. For front curves in a two-dimensional level-set implementation, a cubic interpolation scheme is used and G^1 surface parameterization based on triangular patches is constructed for the three dimensional level-set implementation to compute the distances from grid points near the front. For remote grid points in the band, the less accurate Fast Marching Method [21] is used for both implementations.

Let the forward parameterization be represented by an operator $\mathsf{G} : \mathcal{F} \rightarrow \mathcal{C}$, where $\mathcal{F}$ is a set of all (real valued) functions defined on the domain $\mathcal{D}$, and $\mathcal{C}$ is a set of all possible physically acceptable parameterized fronts embedded in the computational domain $\mathcal{D}$. In other words, given a field function ψ which embeds a curve Γ on the grid, one must construct highly accurate, discrete, approximate parameterization of the physical curve Γ. The *backwards* "field construction" is represented as $\mathsf{G}^{-1} : \mathcal{C} \rightarrow \mathcal{F}$ and assumes that we are given a parameterized curve Γ. This procedure constructs a field $\psi \in \mathcal{F}$ such that $\Gamma = \{p \in \mathcal{D} : \psi(p) = 0\}$. If we know Γ exactly, then its representation can be used to find the exact normal distance to Γ at all points inside the band domain $\mathcal{D}$. Successive application of these procedure is convergent on a limiting grid and can be used to define a fixed point mapping between Γ and ψ in the band. The reciprocal procedures are used to achieve and maintain high accuracy representation of Γ. In addition, if the signed normal function is perfectly calculated in the band, one has a perfect interpolation of field functions to Γ, with accuracy limited only by the grid density and the number of points used in the extrapolation stencil.

Figure 5 and 6 show an example of successive applications of the forward front parameterization and backwards field construction. The field function ψ is given

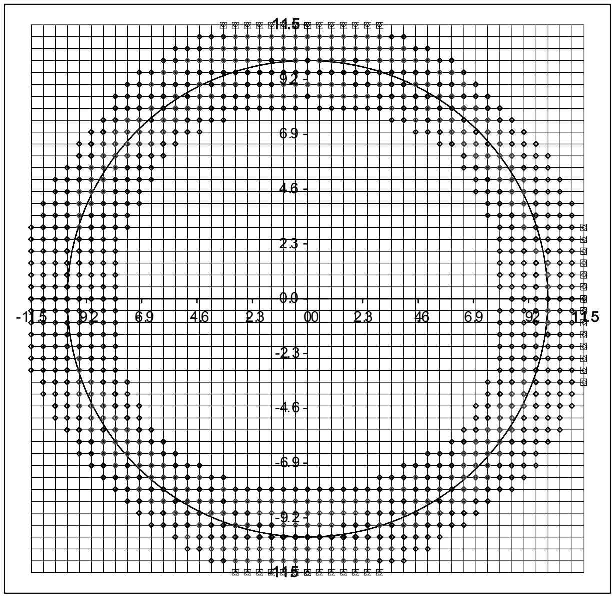


Figure 5. The computational domain and band shown for the circle example

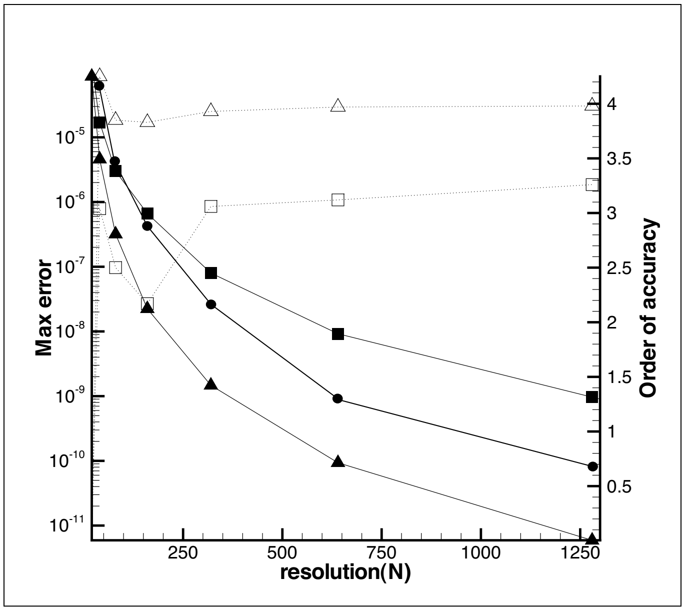


Figure 6. Error and numerical order of accuracy for a circle of radius $R = 10$ with bandwidth $\mathbf{b} = 4$.

exactly as a signed normal distance to a circle Γ of radius $R = 10$ centered at the origin. The narrow band has four grid points on each side of Γ. The computational domain is a rectangular domain $-11.5 \leq x, y \leq 11.5$ and the grid spacings considered have $dx = dy = 23.0/N$, where $N = 20, 40, 80, 160, 320, 640$, and 1280 were used. Nodal points on Γ are found by determining the zeros of ψ on grid lines that were approximated from cubic interpolations of the values of ψ in two coordinate directions. The details of these procedures are described in Section 4. The computation of the normal vector at the nodal points on Γ can be carried out in a similar way. The solid lines with rectangles and triangles show the maximum errors location and normal vectors. The solid line with circles shows that the maximum error of the computed ψ field in the vicinity of Γ, computed by composite action $\mathsf{G}^{-1} \circ \mathsf{G}$, decreases by an order of magnitude as the resolution is doubled. The dotted lines show the order of accuracy of the forward parameterization. For high resolution, the approximation of location is of order four, and that for the normal vector is of order three.

1.0.2. A brief review of related approaches Two methods have been widely used for wave tracking and surface descriptions, the marker-particle method and the level-set method. These methods differ primarily in the representation of the front location. In the marker-particle method the location of front is identified by a set of nodal points through which the front passes. The level-set method uses an approximation to a real-valued function (that ideally is the signed normal distance to the zero level-set that identifies the front) defined at grid points on a prescribed computational domain. The motion rule determines the normal velocities of the markers or level curves and is independent of the method.

While it is sufficient to define the velocity of the front for the marker-particle method, the front velocity must be defined on the entire computational domain for level-set formulation and an appropriate velocity field extension must be provided. To avoid irregular behavior of the level-set during the time integration, (re)-initializations of the distance function in the vicinity of front is required. The marker method can be made more accurate than the level-set method at least in a well-defined portion of computational domain where no topological change of the front occurs. The marker-particle method can be made fast since only the nodal points on the fronts are updated. In contrast all points in the domain must be updated in level-set method. However, the marker methods are harder to implement, especially for the surface tracking of merging and splitting fronts in 3-D domain.

Representative description of wave-tracking that uses marker methods can be found in recent papers by Tryggvason *et al* [14] and by Udaykummar *et al* [13]. These authors track the interfaces separating different regions given by the multi-phase Navier-Stokes equations. The markers are connected to each other by surface elements and stored as an ordered list of points. The surface elements constructed by surface points are triangular patches and are used to reconstruct the surface as simulation is advanced in time. Once the triangular surface patches are constructed, the transfer of the surface values such as surface tension is simple, even though the implementation and construction of the surface patch can be complicated. Udaykummar [13] gives a clear and systematic explanation of the assignment of values from fronts to grid points. In his version of 2-D marker method, the fronts are parameterized by quadratic local interpolations of marker points and then the normal distance function is computed at the nearest grid points to the front (curve) by using orthogonality condition between the displacement vector and the tangent vector of the curve. The transported variables are extrapolated along the computed normal directions. The marker method requires a complicated numerical procedure for the maintenance of equi-spaced marker points on fronts. For example, wave-tracking in the presence of a sharp corner is difficult and requires special treatment in order to supply sufficient number of markers around the corner.

Work by Adalsteinsson *et al* ([18] and [1]), Chen and Chopp, [3], [23] a recent book by Sethian [4], and reviews by S. Osher [22] and J. A. Sethian [20] contain descriptions of modern level-set method and its recent applications. Chopp et al. [15] and Keck [16] showed that accurate computation and parameterization of the zero level-set is required in order to accurately re-distance the level-set so as to maintain a robust and accurate method. Keck showed that accurate calculation of the normal distance from the nearest grid points to the front improves the accuracy of re-initialization. Chopp ([15]) used a cubic interpolation of level-set function (defined on grid points) around the front and used the orthogonality condition to compute the location of the front and developed a higher order modification of Sethian's Fast Marching, although it is computationally expensive, in three-dimensions. Our work is related to the approaches described above and like them can be described as a second generation front tracking method. We use a hybrid combination of the level-set ideas and and particle tracking methodology and we discuss the unique aspects below.

Our implementation captures the front accurately and constructs a parameterization of the front (surface and curve) similar to marker methods. We identify disjoint segments of the front if it is multiply connected. Once a front (i.e. all segments) is constructed, the orthogonality condition is used to compute the signed normal distance to grid points in the neighborhood of the front. In the 2-D implementation we place three layers of grid point around the front. The first layer of grid points are the nearest to the front are denoted V_Γ. The second layer of grid point is the is the set of grid points nearest to the first layer in the "vicinity of V_Γ" and we denote them as $\widetilde{\mathsf{V}}_\Gamma$. The third layer is the remaining set of points in the band domain and we refer to them as points "remote" to the front. Using the orthogonality condition we compute the exact signed distance function in the set V_Γ and its vicinity, $\widetilde{\mathsf{V}}_\Gamma$. Using the information in these two layers, the front construction and the level-set field

Mesh size	Chopp's method	WaveTracker
20×20	1.81×10^{-3}	3.47×10^{-4}
40×40	1.41×10^{-4}	7.28×10^{-5}
80×80	1.36×10^{-5}	2.92×10^{-6}
160×160	1.96×10^{-6}	1.43×10^{-7}
320×320	2.74×10^{-7}	9.33×10^{-9}

Table 1. Comparison of accuracy of distance function by WaveTracker with Chopp's result shown in paper [15] (Table 4.2) on the error measurements in L^∞ norm ,

construction are shown to be reciprocal procedures accurate to fourth-order. In the remote zone, we use a modified Fast-Marching method for approximate computation of the distance function to fill out the band.

The addition of an extra surrounding layer in which the exact normal distance to the parameterized zero level-curve is computed greatly increases the accuracy of the computation. Table 1 shows a comparison of accuracy of the distance to a circle using the Chopp's method that computed the exact normal distance only on the nearest grid points and our method were we use two layers of grid points surrounding the zero level-set including grid points nearest the front. Table 1, shows an order of magnitude decrease in the L^∞ error at the same grid resolution. In addition our method uses less grid points to locate the position of zero level-set and presumably uses less computation time. We find the front location by Newton iteration on a scalar function, whereas Chopp computes the distance to the zero level-set with formulation that uses sixteen algebraic equations at each grid cell two-variable polynomial.

Thus our hybrid algorithm uses exact computation nearest the front and approximate computation remote to the front for the re-distancing procedure to maintain a high accuracy front approximation. The reconstruction of narrow-band computational domain is carried out before the computation of the distance function and the procedure maintains an approximately equal number of grid points equal on both sides of Γ. The advantage of using the level-set-based parameterization of the front compared with the marker method is that the automatically captured nodal points are *always* fairly evenly spaced. Little computational over-head is associated with the reconstruction of fronts, unlike the general marker method. The difficulty in parameterization of fronts near corners is also diminished through the use of the local level-set method.

In three dimension, computation of the normal distance grid points (in the vicinity of V_Γ) to a given parameterized surface is straightforward, given good seed points for the numerical iteration on the orthogonality condition. But this procedure is quite expensive. Therefore in the 3-D implementation we use only two layers, V_Γ and the remote zone which leads to a 2nd order accurate computation of re-initialization or velocity extension, similar to the second order accurate computation obtained by Chopp [23] in his two-dimensional calculations.

The algorithms that parameterize a surface are well developed (for example, see [25]). Hence we focus on the description of our algorithm that constructs an ordered set of nodal points that represents the surface. To make the construction method fast and robust we devised a simple, generic, and robust technique called the Stack-Sweeping Method. The algorithm for a front in 2-D is quite simple but is more complicated for a surface in 3-D. The Stack-Sweeping Method uses two lists (address stacks) to store the memory addresses of nodal points on the surface and from a seed point, the surface is constructed on the surface the set of ordered nodal points that captures the surface is constructed by outward propagation and by the switching the address stacks (See section 4.1.5).

In previous a paper [9], we gave an application of level-set methods to detonation shock dynamics (DSD). An important feature of DSD applications is that the detonation shock is attached to internal and external boundaries at a prescribed angle. The angle attachment condition comes from physical considerations of confinement. Angle control condition is a novel method used in DSD, that is not a part of the standard level-set formulation. These angle boundary conditions are similar to angle conditions found in contact problems of immiscible fluids with surface tension. Hence the techniques for angle control, originally developed for DSD, are applicable to other physically important problems.

In Section 2. we describe briefly the basic theory of level-set method. In Section 3. we describe the code architecture and give definitions that define required data structures. In Section 4. we presented a detailed overview of the algorithms and give some of the implementation details needed to build the WaveTracker code. In Section 5. we present the results of tests and applications. Specifically we describe the applications to detonation wave front tracking, and applications to explosive engineering. We also describe an application to dendritic solidification, where the front moves according to a prescribed motion law, coupled to evolving temperature field solutions on either side. In a sequel to this work we will explain how the interface representations described in this paper can be integrated into a high-order multi-material hydrocode that in turn can be used to engineer complex devices that may contain energetic materials.

2. Preliminaries

Let ψ be a piecewise continuous function defined on the domain of propagation. Let the contour $\psi = 0$ represent the front or curve of physical interest at a given time; the same curve or front is denoted Γ. Let V_n be the normal velocity of the front. The level-set function ψ satisfies

$$\frac{\partial \psi}{\partial t} + V_n \, |\nabla \psi| = 0 \, , \tag{1}$$

where ψ is initially given as a monotonic function which is negative in the interior to Γ and positive exterior to Γ. Typically, the initial ψ-field is taken to be the signed normal distance function

$$\psi(\mathbf{p}) = sgn_{\mathbf{p}} \min_{s \in I} ||\mathbf{p} - \Gamma(s)|| \tag{2}$$

where $\Gamma(s)$ is a prescribed parameterization of front such that $\psi(\Gamma(s)) = 0$ and $sgn_{\mathbf{p}}$ is the sign ±1 is used to determine on which side of Γ the point $\mathbf{p}$ lies. The set I is the domain of parameter s of Γ and $||\mathbf{p} - \Gamma(s)||$ is the distance from a point $\mathbf{p}$ to a point $\Gamma(s)$. If one defines $T(x, y)$ as the crossing time field associated with the sweep of the front through the domain, (for DSD applications this is called the "burn time" and we use $t_b(x, y) \equiv T(x, y)$, with $V_n > 0$), then the boundary-value formulation, is alternatively given by

$$|\nabla T| V_n = 1 \, , \tag{3}$$

where the physical surface Γ is given by $\Gamma = \{(x, y) | \, T(x, t) = t\}$. The boundary data is a specification of the initial surface $T(x, y) = 0$.

For the level surface $\psi =$ constant, we define the normal to the surface, the total curvature and the normal velocity by

$$\hat{\mathbf{n}} = \frac{\nabla \psi}{|\nabla \psi|} \, , \quad \kappa = \nabla \cdot \hat{\mathbf{n}} \, , \quad V_n = -\frac{\partial \psi}{\partial t} \frac{1}{|\nabla \psi|} \, . \tag{4}$$

Since V_n on Γ is the normal velocity, we define its extension in the domain of computation such that its spatial gradient lies in the surface $\psi = constant$. Since $\nabla \psi$ is in the direction normal to the level surface, V_n satisfies

$$\nabla \psi \cdot \nabla V_n = 0 \, . \tag{5}$$

Sethian's fast marching method solves (3) in combination with (5). The method starts with the initial locus at $t = 0$, on which V_n is known to approximate the T-field off the curve, followed by using (5) to calculate an extension of V_n off the same curve. The solution then marches outward from the initial curve.

A narrow band is simply a domain of finite width that embeds the physical surface Γ. For a given grid the discrete version of the narrow band is defined as a collection of points inside the band. The level-set contours are maintained in the band. The evolution of the level-set contours, and hence the motion of the Γ defined by $\psi = 0$, is computed inside the band. If one is solving for the motion of a surface whose motion is defined by the values of field variables defined in the surface or jumps of the same across Γ, then the band is a region that is used for interpolating field variables or their derivatives onto Γ from either side, or both sides. In particular, interpolation of field values onto the band *requires* that the signed normal distance function from Γ be found exactly at the grid points in the band. We do this by using a parametric representation of the physical surface Γ and exact determination of the normal distance at the neighboring grid points. The velocity extension in the band is constructed in a manner similar to that described in [1].

3. Code Architecture: Definitions, Data Structures and Procedures

Here we give definitions and describe data structures and procedures that are needed to build the code. The WaveTracker can be used as a stand-alone application to compute static surfaces, (in which case the surface velocity is zero) the intrinsic motion of a surface that only depends on its current shape, or motion of fronts that is determined by the solution of moving boundary problems (such as dendritic solidification or flame propagation) that require that the field on either side of the interface be solved simultaneously with the motion of the interface. A motion rule must be specified that determines the normal velocity of the interface, V_n on Γ, and when the motion is coupled to the fields on either side of Γ, one-sided extrapolation of the field values to the boundary is required. Since we compute the exact normal distance to Γ in the nearest neighbor region, highly accurate extrapolation is possible.

From an architecture view, the field application always consists of at least two parts; a "Geometry Specifier" that specifies the domain, internal and external boundaries and a "Field Problem Solver" that solves for the fields and controls the location of the front and possibly requires an iteration of the position (or velocity) of the front as discussed above. The Field Problem Solver updates field values on internal and external boundaries and on the front as required, and controls the time stepping for the entire simulation. The "WaveTracker" code element constructs and maintains the front based on known values of a level-set evaluated on a computational domain and computes the signed minimum distance field to Γ. It enforces any front boundary condition at intersection with other internal and external boundaries as required by the formulation. Figure 7 shows a flow chart for a typical application program, with a detailed breakout for the WaveTracker. When Γ is determined by its intrinsic dynamics, the application is simply a driver. When Γ is found as a solution to a moving boundary problem, the WaveTracker interrogates the field on either side and evaluates a residual on Γ (or uses an equivalent procedure).

The WaveTracker consists of five basic procedures below:

(I) **Initialization** This procedure specifies geometry from the field application, defines data structures specifically associated with the narrow bands and allocates memory. (See section 3.1).

(II) **Surface Parameterization** This procedure computes the front parameterization Γ from an existing level-set ψ, represented by $\mathsf{G} : \mathcal{F} \to \mathcal{C}$. It exports the

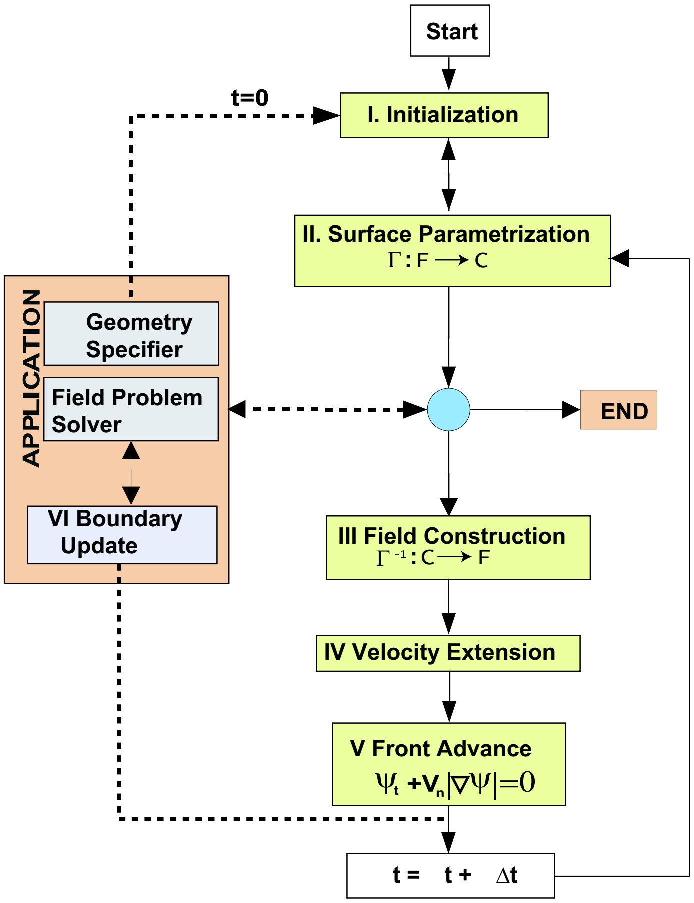


Figure 7. Flow chart of the WaveTracker interacting with a field application.

front information to the field application and defines the normal velocity on Γ. (See section 4.1).

(III) **Field Construction** This procedure computes the signed distance field, ψ in the narrow band once Γ has been specified, and is represented by $\mathsf{G}^{-1} : \mathcal{C} \rightarrow \mathcal{F}$. (See section 4.2).

(IV) **Velocity Extension** This procedure extends the normal velocity defined on Γ to all the band regions. (See section 4.3).

(V) **Front Advance** This procedure solves the level-set PDE according to the velocity extension defined in the band. The details of the update depend on the field application. Typically one needs a conventional upwinding strategy to solve the level-set PDE. The front advance may also be influenced by boundary conditions (See section 4.4).

(VI) **Boundary Update** This procedure enforces the front boundary conditions at the intersection internal and external boundaries, consistent with the application (See section 4.5).

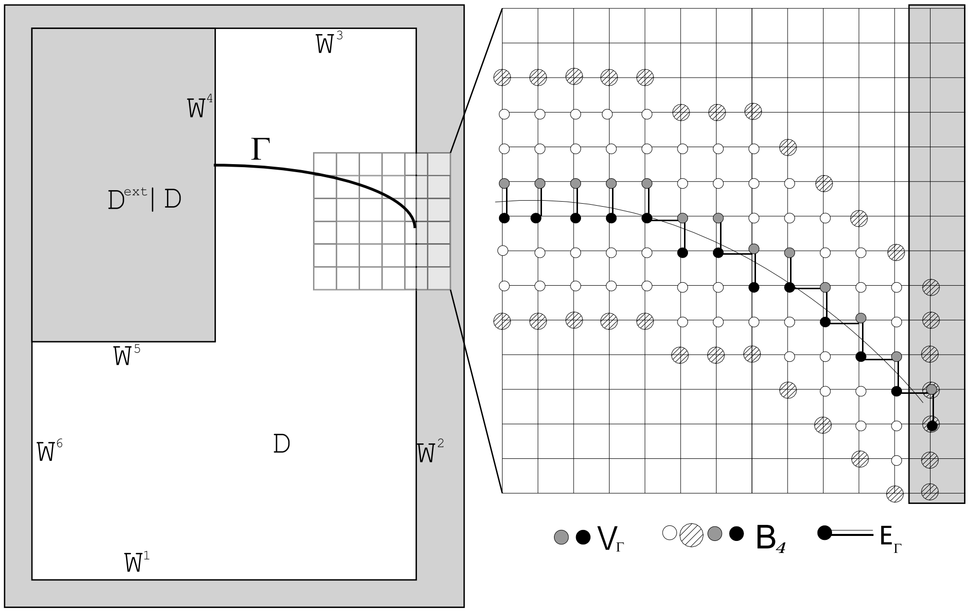


Figure 8. Sketch of the Physical Set up and continuous sets for the a simulation of the WaveTracker and the structure of BANDSET $\mathcal{B}_{\mathbf{b}}$ of bandwidth $\mathbf{b} = 4$.

3.1. Geometric Objects and Definitions

Here we define continuous and discrete objects that we use to represent the front, boundaries and so on. These definitions allow us to define and organize the data structures that are used to carry out the discrete computation.

The domain $\mathcal{D}$ is the region in which the front is allowed to propagate and may have embedded regions inside it through which Γ is not allowed to propagate. Hence $\mathcal{D}$ may be multiply connected. For convenience we take $\mathcal{D}$ to be an open set so that strictly speaking a point on the boundary of $\mathcal{D}$ does not lie in $\mathcal{D}$. The boundary of $\mathcal{D}$ is $\mathcal{W} = \partial\mathcal{D}$. The extended domain, $\mathcal{D}^{ext}$ is the computational domain that fully embeds $\mathcal{D}$ and it includes non physical regions where the surface is not allowed to propagate. We assume that $\mathcal{D}$ is discretized with a uniform rectangular mesh such that (in 2D) the grid points (x_i, y_j) defined by $x_i = x_{min} + i\,dx, y_j = y_{min} + j\,dy$ with $dx = (x_{max} - x_{min})/N$ and $dy = (y_{max} - y_{min})/M$ and $0 \leq i \leq N, 0 \leq j \leq M$.

The "wall" set $\mathcal{W}$ can always be decomposed into a collection of piecewise smooth, parameterized boundary curves $\mathcal{W}^i$ that define the shape of $\mathcal{D}$ and such that $\mathcal{W} = \cup_i \mathcal{W}^i$. The boundary curves $\mathcal{W}^i$ separate the computational domain $\mathcal{D}^{ext}$ into two domains $\mathcal{D}$ and $\mathcal{D}^{ext}|\mathcal{D} \equiv \mathcal{D}^{ext} - \mathcal{D}$. We use the level-set ϕ to represent the signed minimum distance to the wall $\mathcal{W}$, with the property that $\phi = 0$ corresponds to the boundary set $\mathcal{W}$. In many applications the boundary level-set ϕ remains static and then the set of parameterizations of wall curves $\mathcal{W}^i$ is an initial input.

We assume that Γ consists of disjoint segments such that $\Gamma = \cup_i \Gamma_i$ where Γ_i are piecewise smooth, parameterized segments of the front that must be specified inside the domain $\mathcal{D}$. The initial set of parameterizations for Γ_i are initial inputs. We use ψ to represent a set of level contours in $\mathcal{D}$ that embed the contour $\psi = 0$, that represents the front Γ. The level-set ψ is advanced as the front Γ moves. The front Γ splits the physical domain $\mathcal{D}$ into two domains $\mathcal{D}_- = \{\mathbf{x} \in \mathcal{D} | \psi \leq 0\}$ and $\mathcal{D}_+ = \{\mathbf{x} \in \mathcal{D} | \psi > 0\}$. If the front propagates forward into fresh material (as in the DSD application) one can think of $\mathcal{D}_-$ as being the "burnt' region and $\mathcal{D}_+$ as being the "fresh" region.

The set of regularly spaced grid points (x_i, y_j) is superimposed on $\mathcal{D}^{ext}$. We define an edge to be a straight line segment that connects adjacent grid points. The set $\mathbf{E}_\Gamma$ is the set of all edges in $\mathcal{D}$ through which Γ passes. The set $\mathbf{E}_\mathcal{W}$ is the set of all edges in $\mathcal{D}^{ext}$ through which $\mathcal{W}$ passes. Likewise, the set V_Γ is the set of all vertices of edges corresponding to $\mathbf{E}_\Gamma$, and the set $\mathsf{V}_\mathcal{W}$ is the set of all vertices of edges corresponding to $\mathbf{E}_\mathcal{W}$. Furthermore it is useful to define subets of $\mathsf{V}_\mathcal{W}$ such that $\mathsf{V}_\mathcal{W}^+ \subset \mathcal{D}$, and $\mathsf{V}_\mathcal{W}^- \subset \mathcal{D}^{ext} | \mathcal{D}$. Note that since $\mathcal{D}$ is an open set, vertex points that might lie exactly on the boundary $\mathcal{W}$ belong to $\mathsf{V}_\mathcal{W}^-$. We call $\mathsf{V}_\mathcal{W}^-$ the "outer" boundary points and $\mathsf{V}_\mathcal{W}^+$ the "inner" boundary points and they are disjoint. We also define the subets of $\mathbf{V}_\Gamma$ such that $\mathsf{V}_\Gamma^+ \subset \mathcal{D}_+$, and $\mathbf{V}_\Gamma^- \subset \mathcal{D}_-$. We call V_Γ^+ the "outer" front points and V_Γ^- the "inner" front points and they are disjoint as well.

A continuous narrow band can be defined as follows. In 2D, for each point on Γ, considers a circle of radius b centered on any point on Γ. Then the continuous narrow band is the union of all such sets, and in 2D the boundaries of the band set would be two curves parallel to Γ of width b on either side. An alternative is to construct the bands from a union of square bounding boxes of width $2\,b$. The discrete version of the narrow band set that we use is the set of vertex points related to the square boxes. One starts with the previously identified sets of vertex points that enclose Γ, V_Γ. We include in the discrete narrow band $\mathbf{B}_b$, those grid points (x_i, y_j) such that they are contained in the box $x_0 - (b-1)\,\triangle x \leq x \leq x_0 + (b-1)\,\triangle x$ and $y_0 - (b-1)\,\triangle y \leq y \leq y_0 + (b-1)\,\triangle y$, where $(x_0, y_0) \in \mathsf{V}_\Gamma$.

For the purpose of extrapolation of the value of the level-set function ψ throughout $\mathbf{B}_b$ one needs to identify a subset of $\mathbf{B}_b$ that are grid points that form the "boundary" of $\mathbf{B}_b$, which we refer to as $\partial\mathbf{B}_b$. Since the points are exterior points they always have a neighbor that does not belong to $\mathbf{B}_b$ and hence are easily identified by a simple test on nearest neighbors. In the next section, we describe computer data structures of band set $\mathbf{B}_b$.

3.2. Data Structures

The design of the data structure defines data storage and hence determines the efficiency of the overall computation. While computational domain $\mathcal{D}^{ext}$ and the physical domain $\mathcal{D}$ are generally fixed, the size and shape of the band domain and hence $\mathbf{B}_b$, changes dynamically during the computation. Memory should be allocated dynamically for efficiency. We introduce ADDRESMATRIX as a matrix with entries

$\mathcal{D}^{ext}$	:	Extended computational domain
$\mathcal{D}$	:	Computational domain in which surface propagates
$\mathcal{W}$	:	Wall surface enclosing $\mathcal{D}$
$\mathbf{B}_b$	:	Discrete band set of width b.
$\mathcal{D}^{ext}\|\mathcal{D}$	:	$\mathcal{D}^{ext} - \mathcal{D}$
$\partial\mathbf{B}_b$	:	Boundary grid points of band set $\mathbf{B}_b$
$\mathbf{E}_\Gamma$	:	The set of all edges in $\mathcal{D}$ through which Γ passes
$\mathbf{E}_\mathcal{W}$	:	The set of all edges in $\mathcal{D}^{ext}$ through which $\mathcal{W}$ passes
V_Γ	:	The set of all vertices of edges corresponding to $\mathbf{E}_\Gamma$
$\widetilde{\mathsf{V}}_\Gamma$	:	The vicinity of V_Γ defined in section 4.2.1
$\mathsf{V}_\mathcal{W}$	:	the set of all vertices of edges corresponding to $\mathbf{E}_\mathcal{W}$
V_n	:	Normal velocity function defined on $\mathcal{D}$
$\mathcal{D}_+$	:	$\{\mathbf{x} \in \mathcal{D} \| \psi > 0\}$
$\mathcal{D}_-$	:	$\{\mathbf{x} \in \mathcal{D} \| \psi \leq 0\}$
$\mathsf{V}_\Gamma^-(\subset \mathcal{D}_-)$	:	Grid points which are vertices of edges in $\mathbf{E}_\Gamma$ and in $\mathcal{D}_-$
$\mathsf{V}_\Gamma^+(\subset \mathcal{D}_+)$	:	Grid points which are vertices of edges in $\mathbf{E}_\Gamma$ and in $\mathcal{D}_+$
$\mathsf{V}_\mathcal{W}^-$	:	Inner boundary points
$\mathsf{V}_\mathcal{W}^+$	:	Outer boundary points
ψ	:	Level-set function such that $\psi = 0$ defines Γ
ϕ	:	Level-set function such that $\phi = 0$ defines $\mathcal{W}$
$\mathcal{F}$	:	The set of all possible ψ functions on $\mathcal{D}$
$\mathcal{C}$	:	The set of all possible Γ embeded in $\mathcal{D}$
$\mathsf{G} : \mathcal{F} \to \mathcal{C}$	:	Forward parameterization
$\mathsf{G}^{-1} : \mathcal{C} \to \mathcal{F}$	:	Backward field construction
ADDRESSMATRIX	:	Data structure for grid points in $\mathcal{D}_{ext}$
BANDSET	:	Data structure for narrow band $\mathbf{B}_b$
SURFACE	:	Data structure for the edges $\mathbf{E}_\Gamma$ and Γ

Table 2. Summary of Terms and Definitions

for each for each grid point in $\mathcal{D}^{ext}$, BANDSET which stores lists of grid points $\mathbf{B}_b$ and SURFACE which stores lists of edges $\mathbf{E}_{\boldsymbol{\Gamma}}$ with accompanying structured data entries. SURFACE stores the location of Γ. ADDRESSMATRIX points to the addresses of data entries that include the values of ψ and ϕ at the same grid points listed in BANDSET, $\mathbf{B}_b$. Since the two sets $\mathcal{D}$ and $\mathcal{D}^{ext}|\mathcal{D}$ are disjoint, ADDRESSMATRIX can be used to store the addresses of both the values of the level-set ψ and the boundary level-set ϕ simultaneously. In a similar manner, SURFACE is a structured list of edges that can be generated from BANDSET.

3.2.1. ADDRESSMATRIX In 2-D, ADDRESSMATRRIX is $(N+1) \times (M+1)$ and stores the information of the set membership and the addresses for the values of ψ

and ϕ at grid points in $\mathbf{B}_b$. Each entry of ADDRESSMATRIX is 4 bytes (32 bits). The first 4 bits are used to determine the set membership of grid points. The first bit of the status flag is 0 if the grid point is in $\mathcal{D}$ and 1 if the grid point is in $\mathcal{D}^{ext}|\mathcal{D}$. The second bit of the status flag is 1 if the grid point is used for computation, otherwise it is 0. The next two bits can be reserved for other program use and are specific to the application. The status flag is set to -1 if the grid point is not in $\mathbf{B}_b$.

The last 28 bits are used for the addresses of the memory locations of member of BANDSET and are divided into a 12 bit segment address segment and 16 bit offset address segment. Note that 16 bits of the offset address segment can represent up to $2^{16} = 65,536$ unique addresses and the 12 bit address segment can represent up to $2^{12} = 4,096$ blocks of address, each of which can have 64 K address locations. For a typical two dimensional computation, one or two segments of address allocation is enough to represent all of the grid point addresses in BANDSET. But for the 3-D example of the sphere at a resolution of 200^3, 19 (block) segments of addresses are required. The maximum number of addresses by this scheme is 4096 segments, and this simple method will allow addressing for as high resolution as 3200^3. One can extend this scheme further by using 64 bit words address scheme instead of 32 bit scheme explained here. As the computation progresses, we allocate in increments of 64 K units of structured memory as memory is needed. The data structure of BANDSET and SURFACE and the association with the addresses of ADDRESSMATRIX with the grid points is explained next.

3.2.2. BANDSET and SURFACE Each entry of BANDSET consists of five items. The first is the ordered pair for the grid point, (i, j) in 2-D. The second item stores the absolute value of floating point value for ψ. The third item stores the normal velocity at that point. If the grid point is in $\mathsf{V}_{\mathcal{W}}^{-}$, then the third item stores the value of ϕ.

The fourth item (in 2D) is used in the front parameterization and for the Front Advance when ψ evolves. For front parameterization, an ordered pair of 32 bit word addresses are stored as pointers to SURFACE. The first address is the address of the horizontal edge and the second is the address of a vertical edge. During the Front Advance step (Step V), the memory is used to hold a temporary value of ψ. The fifth item is a 1 byte (8 bit) status item that indicates the structure of $\mathbf{B}_b$. If the first bit is 1 then the grid point is a member of the boundary of the BANDSET, $\partial\mathbf{B}_b$. Otherwise the point is a regular interior member of BANDSET. The next three bits are used to store information for the surface reconstruction. The fifth bit is used for the sign of ψ at the grid points. The last 3 bits are held in reserve.

The data structure SURFACE stores the list of edges $\mathbf{E}_\Gamma$ and hence stores the location of Γ. Like BANDSET, each entry of SURFACE consists of five items. The first item contains an ordered pair (i, j) for a vertex (grid point) for an edge. In

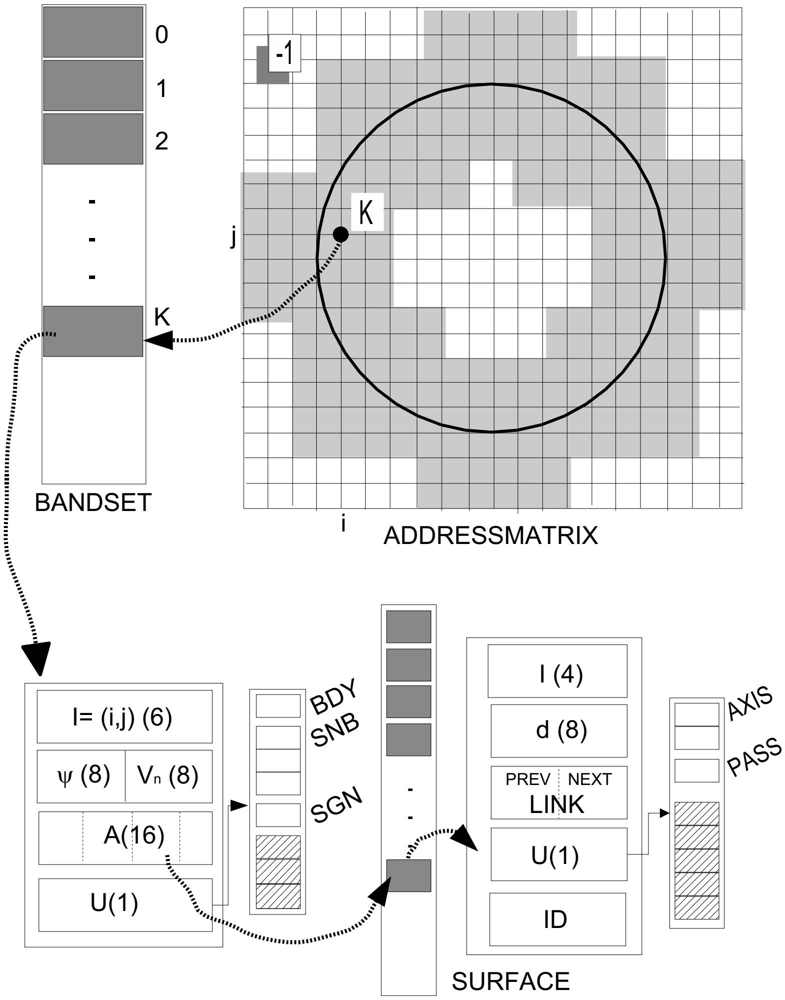


Figure 9. Data structures ADDRESSMATRIX, BANDSET and SURFACE

2-D the edge is either at the bottom (for a vertical face edge) or at the left edge (for a horizontal face edge). The next item is the directed distance starting from the vertex to the intersection of Γ with that edge. Note that these distances are enough to describe all of the intersections of Γ with the grid lines. The third item contains an ordered pair of addresses that are 32 bit words that serve as pointers to other entries of SURFACE. Our parameterization of Γ assumes that the curve (in 2D) is orientable, and that arclength along the curve increases. Therefore the addresses correspond to the locations of the adjacent intersection points of Γ with the grid lines. The first address corresponds to the intersection point to the left and the second address correspond to the intersection point to the right as arclength increases. The fourth item is again a 1 byte (8 bit) status item that consists of flags called 1 bit ”PASS”, 1 bit ”AXIS” that are used to store logical information used in the surface parameterization step. The last item is the identification of a curve segment. Recall that Γ may consist of several disjoint segments Γ_i. The total data storage for one entry of SURFACE is 22 bytes. Figure 9 shows the three main data structures and their relationship to each other.

4. Algorithms

In this section we describe the algorithms. Specifically these algorithms parameterize the front, find and order nodal points which are the intersections of the front with the grid, assign values to the data structures, interpolate values to the grid, reconstruct the normal distance field through the use of the orthogonality condition, carry out the velocity extension, advance the front advance and update boundary values.

4.1. Front Parameterization $\mathsf{G} : \mathcal{F} \rightarrow \mathcal{C}$

The *forward* "front parameterization" is one of the reciprocal operations. We start with a $\psi_{i,j}$ at grid location (i, j) stored in the first item of BANDSET. Generally Γ and hence the ψ field will change in time, but for these procedures one can think of Γ as static, and these algorithms could represent any surface. Next we identify the set V_Γ, which is the set of vertices corresponding to the set $\mathbf{E}_\Gamma$, which is the set of all edges through which Γ passes.

BANDSET is a list of structured items that can be indexed by $k = 0, 1, 2, \ldots$. But the entries of BANDSET are not strictly ordered, in that their order is possibly based on a previous sweeping operation through the grid and the order of entry on the list are influenced by that operation. Recall that if one gives $\psi_{i,j}$ in BANDSET, then one can initialize all the addresses that must be stored in ADDRESSMATRIX. Also recall that if some grid point of ADDRESSMATRIX is not in BANDSET, then one sets the address value equal to -1 as a status flag.

4.1.1. Constructing Sets $\mathbf{E}_\Gamma$ *and* V_Γ *in* $\mathbf{B}_b$. We start by checking an entry of BANDSET with index $k = 0$ (say), that corresponds to $(i, j) = (i^*, j^*)$ (say), and ψ_{i^*,j^*}. Since the value of k is known ($k = 0$ say) and corresponds to grid location (i^*, j^*), one can go to ADDRESSMATRIX and find the values of k for the two neighboring cell vertices $(i^* + 1, j^*)$ and $(i^*, j^* + 1)$. For neighbors in BANDSET one computes the products $\psi_{i^*,j^*}\ \psi_{i+1^*,j^*}$ and $\psi_{i^*,j^*}\ \psi_{i^*,j+1^*}$ respectively. If the sign is negative it corresponds to a sign change in the stored level-set values across a grid line and indicates that a transverse intersection of Γ has occurred with the respective grid line. This means that the k^{th} item (i^*, j^*) is in V_Γ.

We can identify crossing vertices and edges as follows. Recall that each entry of BANDSET has four items. The fourth item U (see Figure 9), consists of 8 bits, which is divided into 1 bit for BDY and 3 bits for SNB, 1 bit for SGN, and 3 bits that are reserved. If for example, $\psi_{i^*,j^*}\ \psi_{i+1^*,j^*} < 0$, then there is a horizontal crossing we set the first bit of SNB equal to 1, otherwise it remains 0. If $\psi_{i^*,j^*}\,\psi_{i^*,j^*+1} < 0$, then there is a vertical crossing and we set the second bit of SNB equal to 1; otherwise it remains 0. Finally if $\psi_{i^*,j^*} = 0$, then we set the third bit of SNB is set to 1. (This means that Γ lies on the grid point!). By using this status item in BANDSET we are able to identify both V_Γ and $\mathbf{E}_\Gamma$. We can repeat this procedure for all members of BANDSET by successive examination of each member of the list. Therefore we can count the number of all the edges and thus can allocate the exact amount memory needed for SURFACE.

Each edge stored in SURFACE represents a grid intersection point for Γ. Each entry in SURFACE corresponds to a member in set $\mathbf{E}_\Gamma$. Next we precisely determine the intersection points of Γ with each member of $\mathbf{E}_\Gamma$, and then we assign that distance information to corresponding item in SURFACE. By doing this, we will determine the distance from any vertex in V_Γ to Γ along the edge (in 2D this is either the horizontal or vertical distance as measured from the vertex point).

4.1.2. Constructing the Data Structure SURFACE. Each grid point in BANDSET that has non zero entries in SNB are members of V_Γ, and have either a horizontal or vertical intersection of Γ with the edge attached to that grid point vertex. As we pass through BANDSET, when we find a member of V_Γ, we copy information into an entry in SURFACE. If there are horizontal and vertical intersections, then we use one or two storage locations in SURFACE. We copy the (i,j) location of the vertex point into the first item of SURFACE, determine the intersection point on the edge for a given face and store that value as the second item of for the entry of SURFACE. (The method we use to compute the distance is described below.) We know the direction of the edge, i.e. whether it is an x or a y edge, and that information is stored at 2 bits item of the fourth item of SURFACE (in a status item, called AXIS, see Figure 9). If the value is 0 the edge is an x-edge; if the value is 1 the edge is a y-edge; if the value is 2, Γ passes through its one of the vertices.

4.1.3. Finding Nodal (Grid Intersection) Points for the Parameterization of Γ*.* The computation of the directed distance from a grid point to the intersection point x, on an edge is done by using a cubic interpolation of ψ values with four points (for the horizontal case) with one to the left, the vertex point itself and two to the right (in the vertical case, one below the vertex and two above). These four values can be used to fit a unique cubic interpolating polynomial for ψ. Then one solves $\psi = 0$ for value x, using that interpolation. By experimentation we found that an ordinary Newton method was fast enough so that three or four iterations obtain an absolute error accuracy of $\mathrm{O}(10^{-10})$. The computed value is stored in the item d of SURFACE.

A special implementation is required when two different segments of Γ are close

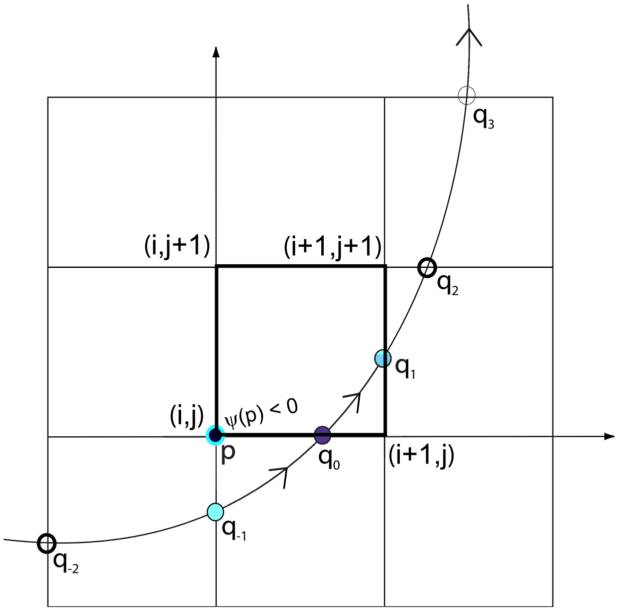


Figure 10. The neighbors of a point and the ordering of nodal points on curve Γ.

to each other. There is a discontinuity in the gradient of ψ between two segments, so cubic interpolation with a four point scheme is not appropriate. The segments can be identified by the procedure described below. One identifies non smooth regions of ψ by checking the number of zeros of the cubic interpolation of ψ in the range of the four node points. If two zeros occur, we put a marker indicating a possible singularity at a nodal point in SURFACE. Later the marked nodal points are associated with particular curve segments and the precise location of marked nodal points can be corrected by one-sided interpolation.

4.1.4. Ordering Nodal Points for the Parameterization of Γ *in 2D* At this point one has passed through BANDSET and has made entries into SURFACE that store Γ, (i.e. the grid location of points in $\mathbf{V}_\Gamma$) specifically the distance from that grid point to the intersection along edges, and a status item that indicate whether an edge is horizontal or vertical. In what follows, we refer to the intersection points of Γ with the grid lines as nodal points.

Next we order the entries in SURFACE so that we can construct a parameterization for ordered curve segments. We sort the nodal points needed to describe Γ (and stored in SURFACE) in order to generate an ordered, parameterizable interpolant for Γ. We do this by one pass through SURFACE. We make an assignment of the values of the third storage item, called ”LINK” in SURFACE and the identification of the segments of Γ into item ID, as described below, (see Figure 10). For the purposes of orienting the surface, we always assume that positive increments in the surface parameterization coordinates are such that in the (2-D) plane the motion associated with change of direction is counter-clockwise.

Assume for a moment that we have found a point on a Γ, say $\mathbf{q}_0$ as shown in Figure 10. The adjacent nearest neighbor nodal points of $\mathbf{q}_0$ on Γ are represented by $\mathbf{q}_{-1}$ and $\mathbf{q}_{+1}$. Note that $\mathbf{q}_0$ corresponds to a member of V_Γ, call it $\mathbf{p}$. One can find the nearest neighbors as follows. First note that $\mathbf{p}$ defines a rectangular cell with vertices (i, j), $(i + 1, j)$, $(i, j + 1)$, $(i + 1, j + 1)$ and such that $\mathbf{p}$ corresponds to the bottom, leftmost vertex (i, j) as the local origin. Suppose that $\psi(\mathbf{p}) < 0$ and that the edge is a horizontal edge. Then the curve Γ enters the cell from the bottom. Given reasonable smoothness and moderate variation assumptions for Γ with a sufficiently small grid spacing, one can assume that Γ must exit the same cell through only one other point on the cell boundary. From the vertex indices (i, j) one can go to the ADDRESSMATRIX to determine the addresses of the other cell vertices and their membership in BANDSET. From interrogation of BANDSET one can determine whether any other vertices are members of V_Γ and hence whether the curve Γ exits the cell.

If the curve exits then one knows the address of $\mathbf{q}_{+1}$ in SURFACE and writes that address to the second (right) item of LINK, for the entry for $\mathbf{q}_0$. If the curve enters the cell from below, then it must also be the case that Γ left the box below it. And in a similar manner one can interrogate the members of BANDSET of the box below, to find vertices that are in V_Γ. Once found, one writes that address to be associated with $\mathbf{q}_{-1}$ in the first item of LINK for the entry for $\mathbf{q}_0$ of SURFACE. Therefore for every member in SURFACE, with a small number of queries, one can determine the address of its nearest neighbor ahead of it, and behind it, in the sense of increasing arc length of Γ, defined by counter clockwise traverse where the region with $\psi < 0$ lies in the interior. If $\psi_{ij} = 0$, there is some ambiguity in determining the oriented direction of Γ. This is easily resolved by inspecting the larger rectangle centered at (i, j) with vertices at $(i \pm 1, j), (i, j \pm 1), (i \pm 1, j \mp 1), (i \mp 1, j \pm 1)$.

4.1.5. The Stack Sweep Algorithm to Order the Surface Entries Note that Γ may possibly be comprised of disjoint segments. Some of the segments may be closed. Some can terminate on boundaries. The next task is to develop a global sorting procedure that examines each entry of SURFACE and stores the necessary information such that we can determine ordered nodal points that correspond to continuous segments of Γ that can be parameterized by increasing arc length (say). We perform this procedure by using the "Stack Sweeping" algorithm and describe that next for the 2-D application. The algorithm has been extended to construct ordered 3-D surface segments and indeed, the 3-D application is the impetus for this invention.

Figures 11 and 12 show an example of the realization of the Stack Sweep algorithm in 2-D. Note that there are two segments shown, a closed segment and a

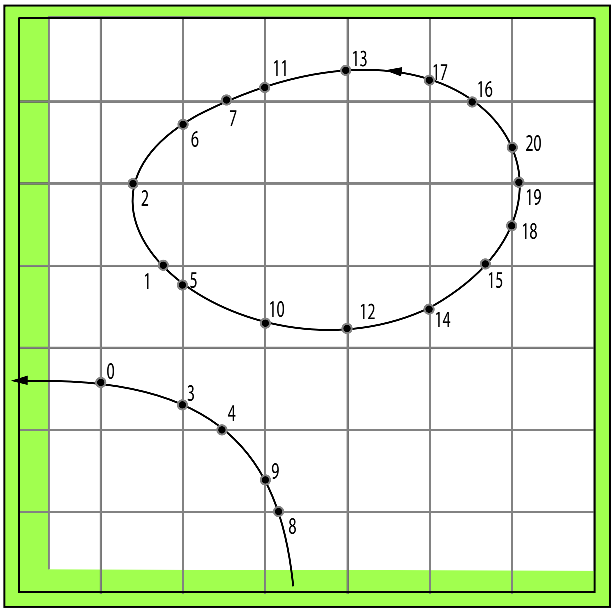
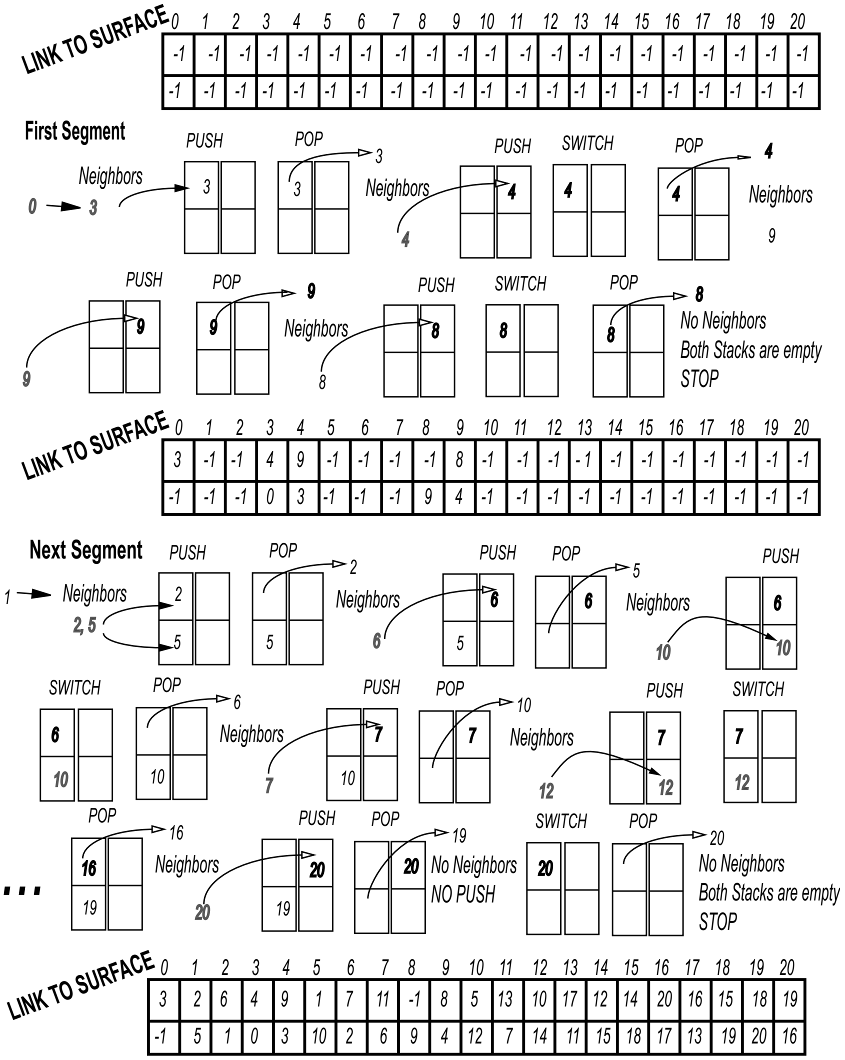


Figure 11. Segments of Γ and with the nodal label in SURFACE shown

segment that terminates on boundaries. Note that the order of entries in SURFACE are not likely to have a strict order that is associated with incremental storage of the nodes. The Stack Sweep algorithm starts with the first entry of SURFACE. Recall from the prior discussion above that by simple queries we determine nearest neighbor points on Γ. For example for entry 0, there is no next neighbor, but there is a nearest neighbor that corresponds to entry, where the sense of advance in the counter clockwise direction. For entry 3 there is a previous neighbor 4 and a next neighbor 0, in the sense of advance in the counter clockwise direction.

The Stack Sweep algorithm in 2-D uses a pair of memory storage stacks S0 and S1 (say), that are use to store temporary information as we sort through the list. We also introduce a set of simple operations on these stacks, which we have simply named, POP and PUSH and SWITCH. Figure 11 shows the list of LINK items of SURFACE. For 2-D, the item of each LINK has two components calls NEXT and PREV, (see Figure 9). Before the Stack Sweep, the list is initialized to - 1. There are 21 nodal points in this example of Figure 11 and hence there are 21 items in SURFACE. Normally those items carry address locations, with the exception that if there is no previous or next neighbor then the value remains at -1. Figure 12 shows 2 address stacks S0 (left) and S1 (right) that can store two address each on the top and bottom. Note that in 2D, the length of each stack is just two but in 3-D it can be of variable size.

One starts the sorting process with the very first entry labeled in 0 in Figure 11 and identified in Figure 12. We set an index for the segment identification to 0 for the first segment. Next we find the nearest neighbors of entry 0 in the manner



LINK TO SURFACE

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

LINK TO SURFACE

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
3	-1	-1	4	9	-1	-1	-1	-1	8	-1	-1	-1	- 1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	0	3	-1	-1	-1	9	4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

LINK TO SURFACE

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
3	2	6	4	9	1	7	11	-1	8	5	13	10	17	12	14	20	16	15	18	19
-1	5	1	0	3	10	2	6	9	4	12	7	14	11	15	18	17	13	19	20	16

Figure 12. Illustration of the Stack Sweep Algorithm that shows the assigment of the LINK entries of SURFACE that corresponds to Figure 11

described previously. In this case there is no next neighbor but there is a previous neighbor 3. One inserts the address for 3 into the previous (top) LINK item for entry 0. One "PUSHES" the address to node 3 to the top of stack S0. Then one "POPS" the address to node 3 out, and we set a flag called "PASS", which is stored in first bit of the status item of SURFACE for node 3 (See Figure(9)). Then the neighbors of 3 are sought. In Figure 11, one see that the previous neighbor of node 3 is 0 and the next neighbor of node 3 is 4. The addresses for 4, the previous and 0, the next are copied into the top and bottom of the third item of SURFACE for the entry for node 3. Note that node 0 has been previously used and the PASS flag for node 0 is set to true, (or 1, say). So the new, unexamined neighbor is 4. We then push the address of node 4 into the top of stack S1. Both entries of the first stack S0 is empty, so we then ”SWITCH”, the entries of the S0 and S1, stacks. The address for node 4 is now in the top of stack S0, so we ”POP” that address out. We again look for the neighbors of node 4 and find the previous node 9 and the next node 3. The addresses for 9, the previous and 3, the next are copied into the top and bottom of the third item of SURFACE for the entry for node 4.

This process continues to include nodes 9 and 8. However after we POP out the address for node 8, the check of both stacks S0 and S1, shows that both are simultaneously empty. This is a termination signal. This indicates that the segment has terminated. Also a check of node 8, shows that there is only one next neighbor 8 and no previous neighbor. Hence the addresses stored in LINK for node 8, have the address of node 9 in the bottom location, but a -1 in the top location. In constructing the segment the appearance of -1 signals termination at a boundary. Hence we have illustrated how the bottom segment shown in Figure 11 is sorted and how those addresses are assigned. The values stored in LINK of SURFACE are shown in the middle of Figure 12.

Once a termination is encountered, we increase the segment index by one and one checks the next entry of SURFACE. In this case it would be the entry for node 1. One also check whether the flag PASS is set to true or false, (0 or 1). If true, that means the node is part of another segment, hence one can advance to the next item in SURFACE. One continues to check each item in this way until the PASS flag is false.

We continue the same algorithm as before, except that we start with the entry for node 1. The segment index is now 1. Here PASS was set to false. We find the previous neighbor 2 and the next neighbor 5 of node 1. We copy those addresses into the top and the bottom of LINK of SURFACE for node 1. Both nodes 2 and 5 have not been previously used, so we PUSH the addresses of 2 and 5 into stack S0. Next we POP the address for node 2, and we find the neighbors of node 2 whose PASS flag is false. In this case there is a previous neighbor, node 6. We next PUSH the address of node 6 into the top of S1. Then we POP the address for node 5, to empty stack S0. Next we find the neighbors of 5 with a false PASS flag, in this case node 10. Since the stack S0 is empty we SWITCH the information in stack S0 and S1, such that S1 is empty. We can repeat this procedure in the same manner until we have node 20 in the top of stack S0. Then we POP out the address of node 20 and we find the neighbors of 20, which are nodes 16 and 19. But PASS flags are true, hence these nodes have been used. There are no new addresses to PUSH and both stack S0 and S1 are empty, which indicates a STOP for that segment.

Figure 12 shows the values stored in LINK of SURFACE at the bottom of the figure after the second segment has terminated. If desired one can use the bottom

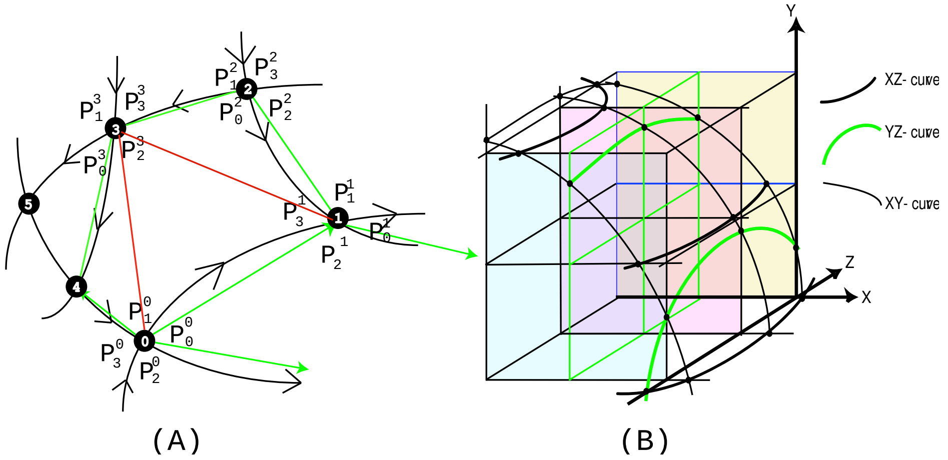


Figure 13. Structure of polygon patches.

addresses of LINK of SURFACE to list ordered nodes in the sense of increasing arclength. The Stack Sweep algorithm compactly constructs sequentially ordered list that identifies disjoint segments of Γ.

4.1.6. Ordering Nodal Points for the Parameterization of Γ *in 3D* If one gives two nodal points on Γ and assigned normal vectors at these points in 2-D, and three nodal points on Γ and assigned normal vectors at these points in 3-D, a smooth parameterization can be constructed with the methods described in Appendix 7.1. In the last section, we described an algorithm for ordering the nodal points on a curve to represent segments of Γ. Now we describe a similar algorithm for ordering the vertices on a given surface in 3-D. We first compute the location of nodal points (vertices) on surface in the same way as we did in the 2-D implementation. Note that in 3-D, two curves pass through each nodal point and each of these curves lie in one of the three orthogonal planes defined by the orthogonal grid system as shown in Figure 13(b).

The surface Γ is composed of a system of polygonal patches whose vertices are the intersection points of the surface and grid lines of the rectangular grid system. Once the system of polygonal patches with their associated normal vectors is determined, a smooth parameterization can be constructed such that normal vector field is continuous across the boundary curves of the patches of Γ, [29]. Since two boundary curves pass through each nodal point, a normal vector can be computed from the cross product of the two linearly independent tangent vectors of the curves passing through the point.

The goal is to represent the surface by an ordered list of the addresses of the vertices and normal vectors of the polygonal patches. The line segments connecting the vertices define the boundary of the patch. The segmented boundary curve of each patch is closed, and orientable. Any patch, say, P in the system of polygonal patches should contain the following information: The ordered vertices (nodal points), their normal vectors and the link information that identifies the neighboring patches that share boundary curves with P. Therefore a memory slot that points to the address of the polygon, P is added to the data structure of SURFACE.

A polygonal patch is not necessarily a triangular patch, see Figure 13. We only parameterize triangular patches so the polygonal patch can not be used directly if it is not triangular. Therefore splitting of a non-triangular polygonal patches into triangles is required if the polygon has more than three vertices by choosing one of the vertices of the polygon. We choose the splitting vertex for the decomposition to be the one whose incident angle is the largest, as shown in the Figure 13. In the figure, the vertex 3 is chosen to the splitting vertex. We set the splitting vertex to be the first item in the ordered list of vertices of polygon for simplicity. The construction of the list of polygonal patches needed for the surface is constructed with the Stack-Sweeping algorithm explained previously. We describe the first few steps of the algorithm for patches and the remaining steps follow recursively.

The boundary curves of polygonal patches lie on one of three planes, XY, YZ, or ZX. We construct the list of surface patches as follows: Figure 13, shows the relevant vertices marked with a number enclosed by a black filled, indexed circle and we refer to these numbers in what follows. For vertex **0**, (say) on Γ, there are four polygon patches that share vertex **0**. This follows from the fact that there are two plane curves with the directions shown in the figure 13. Recall that in 2D curve segments can be defined to have an orientation (i.e. a assigned direction) that is determined uniquely by the of the level-set function. That is also case for any of the plane curves. Hence the direction of the boundary curves of the polygon patches are uniquely specified.

A polygonal patch made of its list of addresses of the vertices and the address list is stored in SURFACE. The first address stored in the list is the address of splitting vertex if the number of vertices is greater than three. Next we explain how to construct the list of vertices. Note that initially a list of vertices exists, and is stored in SURFACE, but they are not necessarily ordered and associated with polygonal patches. The same procedures that were described in the 2D implementation are used in 3D. One vertex **0** (say) is the first item in the SURFACE. Next allocate memory for the four polygons that share vertex **0**, $[P_0^0, P_1^0, P_2^0, P_3^0]$ and store these addresses with vertex **0** in SURFACE. For each associated polygonal patch, put the address for vertex **0** in list of vertices. The vertex **0** has four neighboring vertices; two of them are marked in the Figure 13(a) as **1** and **4**.

For each vertex neighboring **0**, the following that identifies associated vertices is carried out: Given a vertex say, **1**, there is a plane boundary (edge) curve, e_0 that connects **0** and **1** and one knows that the polygons P_1^0 and P_0^0 share the edge e_0. We add the addresses of polygons, P_1^0 and P_0^0 to the list of polygons of the vertex **1**. The polygon P_1^0 contains the vertex **4** and there is a plane curve say e_4 connecting these two vertices, **0** and **4**. The vertex **1** has another plane curve e_1 transverse to

e_0. Assume without loss of generality that the vertex **4** is the previous point of the vertex **0** in the ordering of nodal points of e_4, as determined by the direction. Then if the dot product of the tangent of e_1 and e_4 is positive (i.e. its interior angle is less than 180 degrees), we add the previous vertex **2** of vertex **1**, as determined by the direction of e_1, to the polygon P_1^0. Otherwise we add the next vertex of vertex **1** to the polygon P_1^0. We repeat the same procedure for the other three polygons P_0^0, P_2^0 and P_3^0.

Recall that the Stack Sweep algorithm has two stacks, S_0 and S_1, say. Initially the address of vertex **0** is stored in S_0. We "pop" the address stored in S_0 and carry out the vertex identification as described above and mark the vertex **0** as COMPUTED. Since each vertex has two curves that passes through the vertex, there are four neighboring vertices associate with it. "Push" the addresses of neighboring vertices of vertex **0** into S_1 unless that vertex is not marked as COMPUTED. Then "switch" the two stacks and repeat the stack sweep procedures. The terminal condition is when both stacks are empty.

The lists of the vertices of the polygonal patches are complete at the end of the stack sweep. For each patch we can determine the splitting vertex and use that vertex as the first entry in the list of vertices. We further order the entries of the list of vertices so that it reflects the sequence of vertices encountered in a traverse around the boundary and this is easily accomplished since each vertex has the previous and next flags set on the boundary curves that pass through it. Then each polygonal patch is decomposed into triangular patches suitable for parameterization. Figure 13 (a) shows the vertex **2** with one line segment added for the subdivision. Each vertex of the subdivided triangular patch has one normal vector and it is parameterizable in terms of barycentric coordinates.

4.2. The field reconstruction $\mathsf{G}^{-1} : \mathcal{C} \longmapsto \mathcal{F}$

So far we have described the algorithm for the forward front parameterization which takes a discrete level-set field represented by BANDSET, sorts and constructs an ordered discrete version of Γ which is stored in SURFACE. Next we describe the reciprocal procedure, which we call backward field reconstruction, $\mathsf{G}^{-1} : \mathcal{C} \longmapsto \mathcal{F}$. This procedure assumes that we know exactly the zero level-set curve Γ in terms of a suitable parameterization which is used to find the exact normal distance to Γ at all points inside the band domain $\mathbf{B}_b$. We assume that all the information we need to describe Γ is stored in SURFACE.

The procedure $\mathsf{G}^{-1} : \mathcal{C} \longmapsto \mathcal{F}$ is based on the computation of distance from a given point in the set V_Γ to a local parameterization Γ by orthogonality as described in Appendix (7.2). As shown in the Figure (14) our computational band domain $\mathbf{B}_b$ consists of V_Γ, (its vicinity), and the remote field. In this section, we describe the computation of distance from a point in V_Γ and its vicinity (4.2.1) and the extension

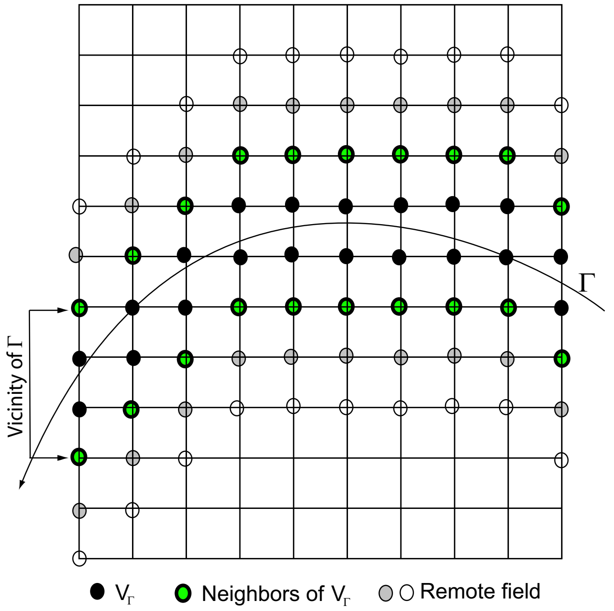


Figure 14. The direction of propagation of points of computation in stack sweeping.

of the computed function to the rest of the computational domain (4.2.2).

4.2.1. Computation of normal distance at points in V_Γ *and in its vicinity* The ψ field in the vicinity of Γ is computed as follows. Every nodal point on Γ that is stored in SURFACE has at most two grid points in V_Γ that are vertices of the edge through which a plane curve passes. Therefore any grid point in V_Γ is connected by one or two nodal points of Γ. The normal distance from the point to Γ can be computed easily using these connected nodal points for seed points for the iteration procedures described in section 7.2. If a vertex point in V_Γ is connected to two different segments of Γ, we take the minimum of all calculated distances from one vertex and attribute it to the ψ value at the point. Note that in even in such cases of multiple choices of origin, there is only one sign of ψ at that (vertex) grid point. After we finish the computation on V_Γ, we also calculate the normal distance at points in $\widetilde{\mathsf{V}}_\Gamma$, the vicinity of V_Γ.

A neighbor of a grid point, say, $\mathbf{p}$ located at (i, j) is by definition one of the grid points located at $(i \pm 1, j \pm 1)$. For all grid points in V_Γ, there are at most three neighbors which are not in V_Γ. For example, in Figure 21, the point $\mathbf{p}_0$ is in V_Γ and the point $\mathbf{p}_2$ is the only neighbor of $\mathbf{p}_0$ which is not in V_Γ. But a neighbor point of V_Γ can have more than one neighbor point in V_Γ. The grid point $\mathbf{p}_1$ in Figure (15) has two neighbors in V_Γ while $\mathbf{p}_2$ and $\mathbf{p}_3$ have only one neighbor each. For each neighbor $\mathbf{p}$ which are not in V_Γ, we find all its neighbors which are in V_Γ and collect all nodal points associated with the neighboring points of V_Γ.

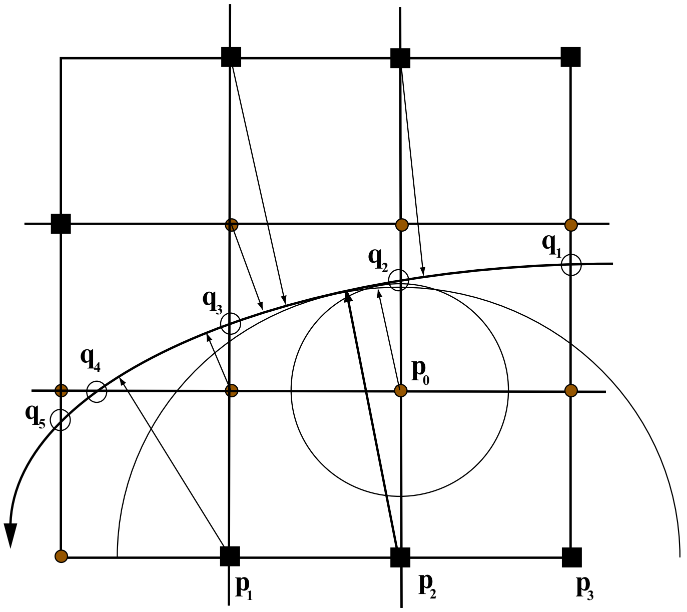


Figure 15. Normal distance from a point **p**, the points marked by small filled circles are in V_Γ and the points marked by black rectangles are their neighbors. The arrows drown from points to Γ show the closest points from these points.

In the example shown in Figure 15, $\mathbf{p}_1$ has two neighbors and the collected points are $\{\mathbf{q_3}, \mathbf{q_4}, \mathbf{q_5}\}$, $\mathbf{p}_2$ has one neighbor and the collected point is $\{\mathbf{q_2}\}$, and $\mathbf{p}_3$ has one neighbor and the collected point is $\{\mathbf{q_1}\}$. To determine an appropriate origin of a local coordinate system for interpolation, we find the closest nodal point among the collected nodal points. Once an origin is found, we use the interpolation method described before to find the normal distance from the point to the interpolation polynomial.

When the equation for the normal distance to Γ given by equation (38) in Appendix I is solved with initial guess given by the minimizing procedure discussed above, the initial guess x can be at the edge of the range of the interpolation appropriate for the selected local coordinate system. In such cases, with aid of the orientation of nodal points in SURFACE, we change the local coordinate system to that which origin is close to the initial guess. For example, if (x_{-1}, x_0, x_1, x_2) are the nodes associated with a chosen local interpolation, and x is close to x_2 or x_{-1}, we change the coordinate system with local origin x_{-1} , x_1 or x_2 according to the position of x_0. Note that the sign of distance function ψ at neighbors of grids in V_Γ is always the same as the sign of their neighbor points in V_Γ. So the sign of ψ field at points off V_Γ is determined by the sign of value of ψ on V_Γ.

4.2.2. Computation of Normal Distance at Points Remote from Γ Having computed the ψ field for all the neighbors of V_Γ in vicinity of Γ, with the orthogonality condition applied, we proceed to calculate the normal distance to additional grid points in the band that are located farther out. If we continue to use the orthogonality condition for those neighbors, the computational cost associated with finding the proper initial guess is too expensive. Instead, we use an approximate calculation of the signed normal distance to Γ for those points in $\mathbf{B}_b$, now denoted as the "remote" members. Figure 14 shows the grid points labeled as members of V_Γ, points in the vicinity and remote points of Γ. We use the Fast Marching Method starting with known values of ψ for points in the vicinity of V_Γ.

The Fast Marching Method solves the Eikonal equation

$$|\nabla\psi| = 1\,, \tag{6}$$

according to the discretization

$$\Big[\max(D_{ij}^{-x}\psi\,, -D_{ij}^{+x}\psi\,, 0)^2 + \max(D_{ij}^{-y}\psi\,, -D_{ij}^{+y}\psi\,, 0)^2\Big] = 1\,. \tag{7}$$

where

$$D_{ij}^{-x}\psi = \frac{\psi_{i,j} - \psi_{i-1,j}}{\triangle x} \quad D_{ij}^{+x}\psi = \frac{\psi_{i+1,j} - \psi_{i,j}}{\triangle x}\,. \tag{8}$$

Note that this discretization allows to solve for $\psi_{i,j}$ in the upwind direction, i.e. propagating away from Γ. Also note that (7) is a quadratic equation for an unknown value $\psi_{i,j}$ and can be solved simply.

4.3. Velocity Extension

For the applications that we are concerned with, one assumes that there is a motion rule that advances Γ. In a primitive physical formulation of a problem, this rule is given by specification of the normal velocity on Γ. But the level set method is an embedding technique and requires that the surface normal velocity V_n be extended continuously into regions on either side of Γ, and that Γ is uniquely embedded by a set of level contours or surfaces. It is always the case that one must implement a velocity extension to define V_n everywhere in the band surrounding Γ.

Given Γ with defined normal velocities V_n at each point on Γ (held either by an exact analytic description or by an interpolation held by a set of nodal points), the normal velocity is extended in such a way that it satisfies the following geometric condition on Γ itself, namely

$$\nabla V_n \cdot \nabla\psi = 0\,. \tag{9}$$

If $\psi = 0$ represents Γ then $\nabla\psi$ is normal to Γ, and similarly if V_n is the normal velocity defined on the surface then ∇V lies in the tangent plane of Γ. Equation (9) is simply an orthogonality condition that extends V_n to the band and guarantees that on each level curve $\psi = \mathit{constant}$ the gradient of V_n on each curve is orthogonal to its corresponding normal vector. Since we know ψ, we construct the extension of V_n to satisfy (9).

We explained how to use Stack Sweep Algorithm to determine the normal distance to Γ in the band. The first two steps of Stack Sweep uses an orthogonality condition to compute the normal distance to Γ from the points of the set V_Γ and the its nearest neighbors. Thus for any point we can, by interpolation, find the coordinates of the intersecting point on a line normal to Γ that passes through that point. Call the intersection point $\mathbf{p}$. For example, if the normal velocity on Γ is a prescribed function of the local curvature then one must compute the curvature and store its values (typically in SURFACE), along with the node values that define Γ, so that one can interpolate. Similarly, if V_n is defined by field or even derivatives that are extrapolated onto Γ, or jumps in those values across Γ, then the values that define the motion rule for the application must be stored.

The simplest velocity extension consistent with (9) is to set $V_n(\mathbf{q}) = V_n(\mathbf{p})$. Note that it follows simply that

$$\nabla V_n \cdot \nabla\psi = \left(\nabla V_n \cdot \frac{\nabla\psi}{|\nabla\psi|}\right) |\nabla\psi| = (\nabla V_n \cdot \hat{\mathbf{n}})\ |\nabla\psi|\,. \tag{10}$$

Since $\nabla V_n \cdot \hat{\mathbf{n}} = dV_n/dn$ (i.e. the directional derivative of $V_n$with respect to n), if V_n is extended as a constant along a normal then its corresponding directional derivative is zero and hence (9) is automatically satisfied. We use this method of extension for all points in V_Γ and their neighbors.

For the remainder of points in BANDSET we use a discretization of (9) to calculate the values for V_n since ψ is known. The Fast Marching Method is an upwinding method that propagates values out from Γ. The calculation of ψ that uses the min-heap data structure discussed above, and the computation of the velocity extension to the remote field shown in Figure 14, can be carried out simultaneously. One uses values closer to Γ to compute the velocities farther from Γ. For example, if $V_{n\,(i+1,j)}, V_{n\,(i+1,j)}$ and $\psi_{i+1,j}, \psi_{i,j}, \psi_{i,j+1}$ are known, then $V_{n\,(i,j)}$ can be computed from the upwind approximation of (9),

$$\frac{V_{n\,(i+1,j)} - V_{n\,(i,j)}}{dx}\ \frac{\psi_{i+1,j} - \psi_{i,j}}{dx} + \frac{V_{n\,(i+1,j)} - V_{n\,(i,j)}}{dy}\ \frac{\psi_{i,j+1} - \psi_{i,j}}{dy} = 0\,. \tag{11}$$

A detailed description of this upwinding scheme is found in Sethian's book [4].

4.4. Front Advance

From the previous subsection we can assume that V_n has been extended throughout the band and that the level-set PDE is given by

$$\frac{\partial\psi}{\partial t} + V_n|\nabla\psi| = 0\,. \tag{12}$$

Further we can always assume a velocity decomposition

$$V_n = V_0 + V_1 \tag{13}$$

where V_0 is a constant and V_1 has spatial variation that depends on location in the domain. Here we think of V_0 as being the dominant contribution in size to the velocity and V_1 as a correction, although this is not a required interpretation. In typical applications V_0 is the velocity of a steady traveling front, whose value is known. Also if V_1 were zero then the front would be advanced by a Huygen-construction and the underlying dynamics of the front advance is governed by a first-order hyperbolic PDE.

In the case of detonation shock dynamics, $V_0 = D_{CJ}$, where D_{CJ} is the Chapman-Jouguet wave speed. In other applications it would be the laminar flame speed or a value of a planar solidification front. In any event, the value V_0 is usually known, and hence V_1 is calculated either from the intrinsic properties of the ψ-field or by an interrogation of the field problem. Since we use the method of lines to integrate (12) we can break the updates for ψ into an advection operator $\psi_t + V_0|\nabla\psi|$, and a correction, $\psi_t = -V_1|\nabla\psi|$. Similar to our previous work on level-set method applied to detonation shock dynamics, [9] and [4], we use a straightforward, weighted second-order accurate ENO scheme for the advection update and use central differences to approximate the updates associated with $-V_1|\nabla\psi|$, for the diffusion operator. If, as explained above, V_0 is known, the velocity V_1 is defined by $V_1 = V - V_0$ where V is the velocity extension of V_n. The gradient of ψ associated with the correction is approximated with central differences.

Points of the difference stencil that lie in BANDSET are in the interior of $\mathbf{B}_b$ and are advanced in the standard way. Points in the stencil outside of BANDSET and in $\partial\mathbf{B}_b$ use quadratic extrapolation from the interior to define the values of the level-set. The maximum normal velocity on Γ is used to define a time step so that the motion of Γ is restricted to advance no more than one cell width.

4.5. Boundary Update

An angle boundary condition on the shape of Γ is enforced by the level- set evolution and ψ is extended to the exterior of wall boundary regions to satisfy the boundary condition. The implementation of ψ on $\mathbf{E}_{\mathcal{W}}$ is generally application specific. Here we review an example of the angle boundary condition that was originally implemented for detonation shock dynamics (DSD), but is general in nature. The boundary condition is enforced by specifying the ψ values on the corresponding discrete set $\mathsf{V}^-_{\mathcal{W}}$, (see the list of definitions). Therefore we construct the field ψ on $\mathsf{V}^-_{\mathcal{W}}$ such that the angle condition is satisfied not only at the intersection of the zero level curve and $\mathcal{W}$ but also at all intersections of level curve or surfaces of ψ and wall $\mathcal{W}$. Figure 16 shows the local coordinates and the stencil used for implementation of the boundary conditions and is adapted from [9]. We assume that the level set field ϕ on $V^-_{\mathcal{W}}$ is

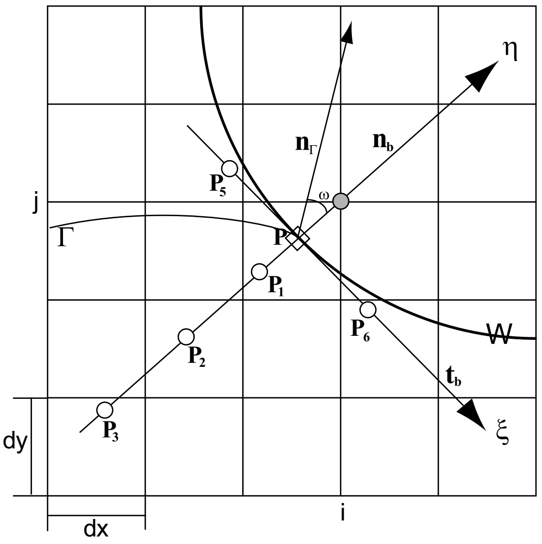


Figure 16. Local coordinates for application of boundary conditions adapted from [9]

already calculated and stored in BANDSET. Note that this calculation is done only once at the initialization step by using the same procedure $\mathsf{G}^{-1} : \mathcal{C} \longmapsto \mathcal{F}$. Therefore we can easily find the projection of $V_{\mathcal{W}}^{-}$ on $\mathcal{W}$.

We give the case is when the angle of intersection of Γ with the wall is prescribed. We assume $dx = dy = h$. Let $\hat{\mathbf{n}}_b$ be the unit normal vector to the wall boundary pointing away from the interior of $\mathbf{B}_b$. Let $\hat{\mathbf{n}}_\Gamma$ be the outer normal vector to Γ at the wall intersection point pointing in the direction of propagation. Note that $\hat{\mathbf{n}}_\Gamma = \nabla\psi/|\nabla\psi|$. Let ω be the angle between the two normal vectors such that at the wall they are calculated according to

$$\cos(\omega) = \hat{\mathbf{n}}_b \cdot \hat{\mathbf{n}}_\Gamma = \frac{\nabla\psi}{|\nabla\psi|} \cdot \hat{\mathbf{n}}_\Gamma = \frac{\psi_\eta}{|\nabla\psi|}\,, \tag{14}$$

where η is the local surface-attached coordinates in the normal direction as shown in Figure (16).

Let a point $\mathbf{q}$ be a point representing a grid point, say (i, j) in 2-D. Let the nearest point on the wall $\mathcal{W}$ to $\mathbf{q}$ be a point $\mathbf{p}$. Let the distance between these two points be ϕ_q. Then the vector $\hat{\mathbf{n}}_\mathbf{b} = (\mathbf{q} - \mathbf{p})/|\mathbf{q} - \mathbf{p}|$ is the normal vector to $\mathcal{W}$ at $\mathbf{p}$. According to the stencil shown, the quadratic approximation to the values ψ_η at point $\mathbf{p}$ are given as follows

$$\psi_\eta = \frac{\psi_2 - 4\psi_1 + 3\psi_q}{2h} - \frac{\psi_2 - 2\psi_1 + \psi_q}{h^2}\phi_q \tag{15}$$

where the values ψ_p, ψ_1 and ψ_2 are the values of the level-set function ψ evaluated at $\mathbf{q}$, $\mathbf{p}_1 = \mathbf{q} -\, h\hat{\mathbf{n}}_b$, and $\mathbf{p}_2 = \mathbf{q} - 2\,h\hat{\mathbf{n}}_b$ respectively. We use a bi-linear interpolation to evaluate ψ_1 and ψ_2.

From equations (14) and (15), the formula for ψ_q is

$$\psi_q \left(\frac{3}{2h} - \frac{\phi_q}{h^2} \right) = cos(\omega) |\nabla \psi| - \left(\frac{\psi_2 - 4\psi_1}{2h} - \frac{\psi_2 - 2\psi_1}{h^2} \phi_q \right) \tag{16}$$

where the gradient $|\nabla\psi|$ is evaluated at point $\mathbf{q}$ by using a linear interpolation of gradients defined at grid points. The gradients at grid points in the interior of the band-set is calculated when the level-set function ψ is updated as described in section 4.4. The gradient at points in $V_{\mathcal{W}}^-$ is computed by a linear extrapolation of $|\nabla\psi|$ defined on the interior domain. Equation 16 is a system of nonlinear algebraic equations for $\psi_q,\ q \in V_{\mathcal{W}}^-$ and is solved with a root solver as in [9]. Note that the discretization of the derivative along the tangential direction is not used here, unlike the method used in [9]. Our current method is more convenient in a 3-D implementation of angle boundary condition.

5. Tests and Applications

Here we present the results of testing the WaveTracker for representative applications. The first application is curvature dependent front propagation, illustrated for detonations. We develop exact solutions and compare with the computed results to measure the normal distance function, the front position and curvature.

The second application is to detonation shock dynamics, (DSD) whereby the motion of the front is determined from the geometry of the front and the angle at the intersection of the front (shock) with internal or external boundaries. Hence we include a description of the implementation of the boundary condition, which must be added to the WaveTracker. The third example is a Stefan problem that represents a model of dendritic solidification problem. In this example the zero level set represents the solid/liquid interface. We chose this test problem because it is an oft-computed test problem and it is closely related to combustion problems that treat the reaction front as a sharp interface. In both cases, the front must be located as a solution to a moving boundary problem, determined by continuous fields on either side of the front. Simple problems with exact solutions are available for each application.

5.1. Curvature Dependent Front Propagation

Consider a circle expanding according to the motion rule

$$D_n = D_{CJ}\,(1 - \alpha\,\kappa) \tag{17}$$

where D_n is the normal velocity, κ is the curvature of a moving front and D_{CJ} and α are constant. An exact solution of the problem for a given circular front of radius R

Res.\\Δt	Max. Error	O^n
100(0.4)	2.1×10^{-5}	-
200(0.2)	1.7×10^{-6}	3.61
400(0.1)	1.1×10^{-7}	3.91
800(0.05)	7.7×10^{-9}	3.90
1600(0.025)	6.6×10^{-10}	3.54

Table 3. Accuracy of distance function $\tilde{\psi}$ for the test problem in section5.1.

at time $t = 0$ is given by [9]:

$$t = \frac{1}{D_{CJ}}\left(r - R + \alpha \log\left(\frac{r-\alpha}{R-\alpha}\right)\right) \tag{18}$$

where r is the radius of the expanding circle. The numerical solution given by WaveTracker is compared to exact solution (18) to measure the error. The curvature is related to the radius by $r = 1/\kappa$ and the formula for the curvature a level curve in 2-D is given by

$$\kappa(x,y) = \nabla \cdot \frac{\nabla \psi}{|\nabla \psi|} = \frac{\psi_{xx}^2 - 2\psi_y \psi_{xy} + \psi_{yy}\psi_x^2}{(\psi_x^2 + \psi_y^2)^{3/2}}\,, \tag{19}$$

and central differences are used to compute derivatives for the error measurements. For the test we use a rectangular computational domain $[-20, 20] \times [-20, 20]$ with a circle of initial radius $R = 3.38$, with $D_{CJ} = 7.6$ and $\alpha = 0.14$.

The distance function measured from the expanding circle can be compared with in the band in both the nearest neighbors grid points V_Γ and their vicinity $\widetilde{\mathsf{V}}_\Gamma$ (ie. closest two layers to Γ). The orthogonality condition is use to compute the normal distance to the grid points the nearest neighbor region. The Fast Marching method is used to approximate the normal distance to the zero level-set in the remote regions. Our test uses a narrow band domain B_b of width $b = 5$. The distance function $\tilde{\psi}$ is defined only on the subset $\widetilde{\mathsf{B}}_b = \mathsf{V}_\Gamma \cup \widetilde{\mathsf{V}}_\Gamma$ of B_b.

Let $\tilde{d}^n_{(i,j)}$ be the distance approximation at grid (i, j) to the curve of interest and $d^{exact}_{(i,j)}$ be the exact distance from the point. Table 3 shows the result of error in the maximum norm $E^n = \max_{(i,j)\in\widetilde{\mathsf{B}}_b} |\tilde{d}^n_{(i,j)} - d^{exact}_{(i,j)}|$ computed with resolution $2^n \times 100$ with n-integer. Then E^1 stands for the error at resolution 100×100. The order of accuracy O^n is calculated according to the formula

$$O^n(t) = log_2\,(\,E^{n-1}(t)/E^n(t)\,)\,, \tag{20}$$

and measures the rate of reduction of error when the resolution is doubled. Table 3 shows that the order of accuracy of our field reconstruction is approximately equal to the order of accuracy of the forward front parameterization, the polynomial interpolation on the nodal points of zero level-set.

Res.\\Δt	Max. Error	O^n
100(0.4)	1.1×10^{-1}	-
200(0.2)	1.4×10^{-2}	2.97
400(0.1)	2.9×10^{-3}	2.27
800(0.05)	6.6×10^{-4}	2.14
1600(0.025)	1.5×10^{-4}	2.14

Table 4. Accuracy of distance function computed by 2nd-order Fast Marching Method for the test problem in section 5.1.

Table 4 shows the error measurement for the distance function in the remote region, $\mathsf{B}_b - \mathsf{V}_\Gamma \cup \widetilde{\mathsf{V}}_\Gamma$, computed with the Fast Marching Method. Note that the Fast Marching method is formally second order and the results Table 4 indicates the order estimate decreases from above toward two, as resolution increases.

Next we consider position and curvature errors measured on the interface of the expanding circle. As the front expands according to the previously defined motion rule, we measure the L^1 position error on the interface calculated with the WaveTracker against the exact solution at different times. The results are shown in Figure 17. The formula for the error is given by

$$E^n(t_n) = \frac{1}{N} \sum_{k \leq N} \left| r_k^n(t_n) - r(t_n) \right|, \tag{21}$$

where N is the number of nodal points on $r^n(t)$ at the time t and r_k^n is the distance from the origin to the $k-th$ nodal point on the computed interface at time t_n.

Table 5 shows the error measurement for the curvature computed from the distance function ψ in the band. The values listed in the column under κ_0 represent the maximum error for the curvature in the set (1st layer) V_Γ set, those under κ_1 represent the maximum error computed curvature in $\widetilde{\mathsf{V}}_\Gamma$ (in the vicinity of V_Γ) and those under κ_2 represent the maximum error of the computed curvature in the remote zone. In V_Γ, the computation of curvature is quite accurate and at least first order. In other outer regions the higher resolution does not improve the accuracy. In part, our measurement in the remote regions is flawed since we used a central difference formula to compute the curvature, which is inconsistent with the upwind Fast Marching Method. However in the nearest neighbor regions, we can justify using central difference scheme to compute curvature.

Figure 17 shows the L^1 error for three different resolutions and the corresponding order of accuracy comparisons as the circular interface expands. The figure shows the L^1 error of the location of the front on the left (shown with symbols), and the corresponding order of accuracy on the right computed with formula (20) (shown without symbols) as a function of time. The results show that the accuracy of the front position is at least second order but that the order of accuracy drops as the front

Res.	κ_0	κ_1	κ_2	O_0^n	O_1^n	O_2^n
100	2.7×10^{-3}	1.4×10^{-2}	3.8×10^{-1}	-	-	-
200	1.2×10^{-3}	1.0×10^{-2}	1.2×10^{-1}	1.17	0.512	1.7
400	5.6×10^{-4}	9.2×10^{-3}	1.2×10^{-1}	1.13	0.133	0.024
800	2.6×10^{-4}	1.0×10^{-2}	1.1×10^{-1}	1.07	-0.17	0.112
1600	1.3×10^{-4}	9.6×10^{-3}	1.1×10^{-1}	1.03	0.11	-0.0028

Table 5. Accuracy of curvature field after re- initialization

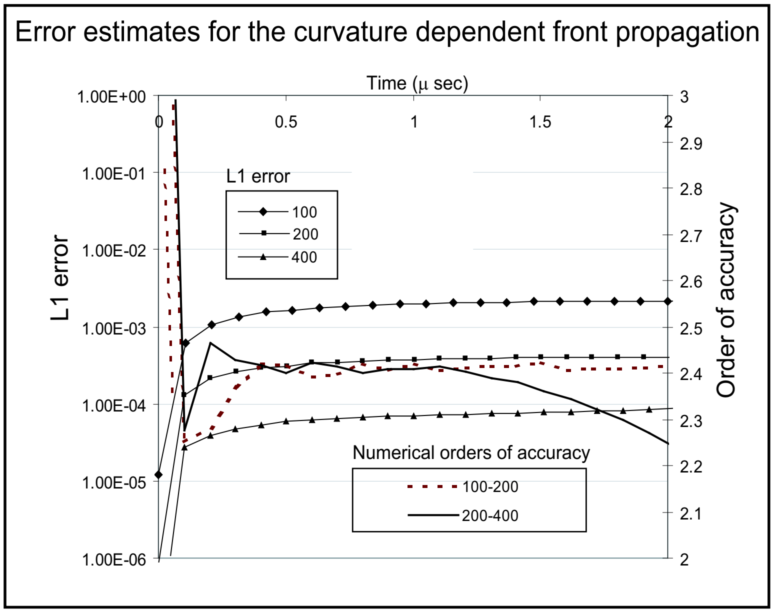


Figure 17. L^1 error of the location of front at time t. Numerical order of accuracy are shown in the graph.

approaches the approaches the boundary of the computational domain near the final time $t = 2.0$. This loss in accuracy is expected due to the fact that computation in parts of the band domain is affected by the one-side extrapolation near the rectangular computational boundary.

5.2. Detonation Shock Dynamics, DSD

Detonation shock dynamics is an asymptotic theory that describes the evolution of a multi-dimensional, curved detonation shock in terms of an intrinsic evolution equation for the shock surface. An overview of the theory can be found in [11]. One of the results of DSD theory provides a relation of normal velocity of the shock, $V \equiv D_n$ to the total shock curvature, κ. This $D_n - \kappa$ relation depends on the equation of state of the explosive and the rate law. The $D_n - \kappa$ relation is the motion rule for moving Γ. The simplest DSD model assumes the shock angles are specified at the shock–inert (wall) interfaces, and if the wall material changes then the shock angle intersection can change with the inert material. The first use of level-sets for shock dynamics and DSD was given in [9].

Therefore we assume that under suitable conditions the detonation shock propagates according to

$$D_n = D_{CJ}(1 - \alpha(\kappa)) \tag{22}$$

where D_n is the normal shock velocity (previously designated V_n), D_{CJ} is the CJ detonation velocity, and $\alpha(\kappa)$ is a function of the total shock curvature. We use a $D_n - \kappa$ relation that models the explosive PBX9501, consistent with Hull's experimental data([12]), as shown in Figure 2, and Lambert's experiment shown in Figure 1.

5.2.1. DSD Boundary Conditions The simplest DSD boundary conditions enforces an angle boundary condition at the intersection of Γ with the boundary. Let ω be the angle between the wall normal direction and a normal to the detonation shock. Let ω_c be a special sonic flow and let ω_s a different sub-sonic flow angle. Both are assumed to be constant with $\omega_c \leq \omega \leq \omega_s$. We implement the same conditions as in [9]. Quoting from the summary (on page 394 of that reference), "In summary, ... (i) when the flow in the explosive is supersonic (i.e. $\omega < \omega_c$) the continuation and outflow boundary condition is applied. This corresponds to extrapolating the front to the exterior without changing the angle at the boundary. (ii) When the flow turns sonic, $\omega = \omega_s$, two cases can arise: (a) The pressure induced by the inert is below that [found] immediately behind the detonation shock and the confinement has no influence on the detonation. The sonic boundary condition is applied, $\omega = \omega_s$: (b) The pressure induced in the inert above that immediately behind the detonation shock. The angle ω increases i.e., $\omega > \omega_s$ until the pressure in the inert and explosive are equilibriated. This angle $\omega = \omega_c$ is the equilibrium value for the angle and is regarded as a material constant that is a function of the explosive/inert pair. Thus the boundary condition recipe is as follows: (1) A continuous boundary condition is applied for supersonic flows, and (2) when the flow becomes either sonic or subsonic, ω is bounded from above by a critical angle ω_c (unique for each explosive/inert pair) that is determined using the above discussion."

5.2.2. DSD Modeling of a Detonation Wave Shaper One application of condensed explosive detonation is to cut materials through the concentrated effect of converging shock waves. A way to do this is to place an inert in an explosive charge, ignite the charge at the bottom, and diffract the detonation wave around an inert. The detonation diffraction results in the collision of detonation fronts to produce extraordinarily high pressures in the interior of the charge. When the detonation breaks out at the top of the charge it carries with it the same high pressure which in turn can be used

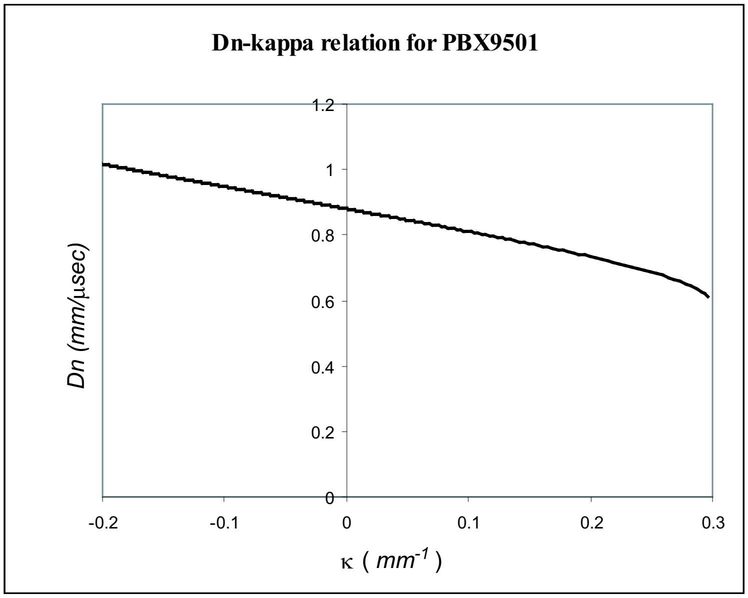


Figure 18. $D_n - \kappa$ relation that models the explosive PBX9501, consistent with Hull's experimental data ([12]).

to precisely cut the object placed against it.

Figure 1 is an example of such a device and the explosive material is taken to be PBX9501. The inert confinement on the bottom and sides is copper. The inert block in the center is lead. The top boundary is unconfined. Using the equation of states for unreacted PBX9501 and copper we compute angle boundary conditions by matching shock polars on a material interface, given a regular reflection of the detonation and inert shock waves at the interface. Therefore on the PBX9501/copper boundaries the confinement angle is $\omega_c = 75$ degrees. The sonic angle for the explosive is taken to be $\omega_s = 50$ degrees.

The charge is initiated at the bottom by a smaller cylindrical detonation at time $t = 0.0\,\mu\,sec$. Figure 2 shows an axi-symmetric 2-D, DSD calculation carried out with the WaveTracker. We use the $D_n - \kappa$ relation, as shown in Figure 18. The gray scale in Figure 2 shows the pressure measured on a 100 GPa scale, as calculated from a shock Hugoniot calculation. It indicates relative regions of high pressure experienced at the shock as the shock passes through the interior of the explosive, i.e. a shock pressure map of the interior. The solid lines show the shock location at various times, measured in μ second from $t = 0$. The calculation shown was for a 152×164 domain with a 2nd order Runge-Kutta time integration. The entire computation took about 40 seconds on a 1 Ghz Dell Inspiron 4100 (2001-model) laptop computer. This time estimate includes the graphics rendering as well as data recording and the actual compute time; without rendering, approximately 5 millisecond for every time step is required.

Next we illustrate the same simulation but in 3-D. We relocate the lead disk and place it off of the axis of symmetry of the charge so that the simulation is necessarily three-dimensional, for example compare Figure 2 with Figures 3 and 4. The 3-D simulation computes the exact normal distance to grid points in the band using the orthogonality condition only on V_Γ and the distance function is extended to remote points using the Fast Marching Method and the 3-D simulation is less accurate than 2-D simulation. The number of iterations needed for the root solver for this 3-D angle boundary condition was not significantly increased when compared with the corresponding 2-D implementation, even though the number of unknowns in the 3-D simulation is much bigger than that of the 2-D simulation.

5.3. Dendritic Solidification

Our second example is the application of the same algorithms to the classical model of dendritic solidification, whose formulation uses the Gibbs-Thompson relation at the solid/liquid interface. In this case the motion rule for the interface depends on the curvature of the front, the values of a field quantity (temperature) on the front and the jump in the derivatives (the normal temperature gradient) across the front. The dendritic solidification problem is a much studied and standard test case, and has similar complexity to problems that solve for the motion of flames.

The dendritic solidification problem is specified as follows. On either side of the solid-liquid interface, designated by Γ the heat equation holds

$$\frac{\partial T}{\partial t} = \nabla^2 T \,. \tag{23}$$

A heat balance across Γ is

$$V_n = [\nabla T] \cdot \hat{\mathbf{n}} \quad \text{across} \quad \Gamma \,, \tag{24}$$

where $[\,]$ designates a jump $[\,] = |_{solid} - |_{liquid}$, and $\hat{\mathbf{n}}$ is a unit normal vector pointing into the liquid. The Gibbs-Thompson relation holds on Γ prescribed by

$$T = -a\,\kappa - b\,V_n \,, \tag{25}$$

where a is the surface tension coefficient and b the molecular kinetic coefficient.

5.3.1. Self-Similar Solution: Frank Spheres Self similar solutions called Frank spheres form a class of exact solutions that are used to test the algorithms. The initial front is a circle of radius S with center at origin. The initial temperature distribution inside and on the front is equal to zero. The temperature is equal to a constant T_∞ in the far field. The solution when $a = b = 0$ can be found as follows. Let $r = \sqrt{x^2 + y^2}$ and $s = r/\sqrt{t}$. The phase change surface is located at $r = S\sqrt{t}$, where S is a constant. Then the heat equation is transformed into a ordinary linear differential equation

$$2sT^{''} + (2 + s^2)T^{'} = 0\,, \tag{26}$$

where $' = d/ds$. The general solution of this equation is

$$T(s) = C_1 E_i(s^2/4) + C_2\,, \tag{27}$$

where

$$E_i(x) = \int_x^\infty \frac{e^{-u}}{u} du\,. \tag{28}$$

By applying the boundary conditions $T(0) = 0$ and $T(\infty) = T_\infty$, we find that $C_1 = -T_\infty/E_i(s^2/4)$ and $C_2 = T_\infty$. The solution is re-written as

$$T = 0 \quad \text{for} \quad s < S \quad \text{and} \quad T = T_\infty \left(1 - \frac{E_i(s^2/4)}{E_i(S^2/4)}\right) \quad \text{for} \quad s > S\,. \tag{29}$$

Since $V_n = dr/dt = S/(2\sqrt{t})$ on the interface, and $[\nabla T] \cdot \hat{\mathbf{n}} = (dT/ds)/(\sqrt{t})$ evaluated at $s = S^+$ on the liquid side, then the condition $V_n = [\nabla T] \cdot \hat{\mathbf{n}}$ leads to the formula

$$\frac{S}{2}\left[E_i(S^2/4))/E_i^{'}(S^2/4)\right] = T_\infty\,. \tag{30}$$

The solution for the undercooled value $T_\infty = -0.5$ is $S = 1.56$.

5.3.2. Computation of heat equation It is assumed that Γ has been advanced by the previous step's determination of V_n and the heat equation is solved by an implicit scheme in $\mathcal{D}$, with the position of Γ_{n+1} frozen. The heat equation is time advanced as

$$T_{ij}^{n+1} = T_{ij}^n + \Delta t \;\; \nabla^2 T_{ij}^{n+1} \tag{31}$$

with boundary condition along Γ defined implicitly as follow.

$$T^{n+1} = -a\kappa - bV^{n+1} \qquad \text{and } V_n = \frac{\partial T}{\partial n}\Big|_\Gamma^{solid} - \frac{\partial T}{\partial n}\Big|_\Gamma^{liquid}\,. \tag{32}$$

A standard 5-point stencil is used for the discretization of $\nabla^2 T_{ij}^{n+1}$ so that

$$\nabla^2 T_{ij}^{n+1} = \frac{T_{i+1,j}^{n+1} - 2T_{i,j}^{n+1} + T_{i-1,j}^{n+1}}{h^2} + \frac{T_{i,j+1}^{n+1} - 2T_{i,j}^{n+1} + T_{i,j-1}^{n+1}}{h^2}\,, \tag{33}$$

where h is the grid size.

We use one-sided cubic polynomial extrapolations (from either side) in order to calculate the second order derivatives of the temperature on Γ and points in V_Γ. Then (31) becomes a system of nonlinear algebraic equations having the following general form

$$T_{i,j}^{n+1} = C_{i,j}\, T_{i,j}^n + f_{i,j}(T_{i,j+1}^{n+1}, T_{i,j-1}^{n+1}, ...) \tag{34}$$

where $f_{i,j}$ is a function that is determined by (33) or from one-sided extrapolation.

The unknowns are $T_{i,j}$ and the temperature on at the node point on Γ stored in SURFACE. A simple method, the Gauss-Seidal iteration scheme suggested by Chen *et al* [3], is used to find the solution. The stopping criterion is

$$\sum_{i,j=1}^{M}(T_{i,j}^{n+1,N+1} - T_{i,j}^{n+1,N}) < TOL, \text{ where } TOL = 1.0 \times 10^{-12} \tag{35}$$

where $T^{n+1,0} = T_{i,j}^{n}$ and M is the total number of grid points in the computational domain. Appendix III give details for the iteration procedure.

5.3.3. Comparison of the Computed Solution with Exact (Franks Spheres) Solution The computational domain is the rectangle $[-5,5]\times[-5,5] \subset R^2$. The exact solution of the heat equation for the Frank Sphere at time $t = 1.0$ is used as an initial condition for the temperature. The initial interface is a circle of radius 1.56 and the temperature in the far field is approximately -0.5 as calculated from equation (30).

Figure 19 we present results that show the numerical solution at time $t = 1.5$ for various resolutions and Table 1 displays our computed error estimates based on the exact solution of this model problem. The L^1 norm is about $O(10^{-4})$ for all resolutions and even moderate resolution (80×80) provides an accurate result. Our results show an improvement over those shown in a recent those found in recent papers [3] and [8] with roughly speaking our $N \times N$ resolution being as accurate as their $2N \times 2N$. Specific comparisons for N = 20, 40, 80 and 160, are shown in Figure 19.

5.3.4. Numerical Simulation of Unstable Dendritic Growth Figure 20 presents results for long time unstable growth of slight perturbations to Frank Sphere solution, computed with resolutions 100×100 and 200×200 respectively. We adapted Chen *et al* 's example (in section 5.2.1 [3]), and used the following initial conditions for the interface,

$$x(s) = (R+P\cos(8\pi s))\cos(2\pi s)\,,\; y(s) = (R+P\cos(8\pi s))\sin(2\pi s)\,, \tag{36}$$

where R = 1.57 and P = 0.08. The temperature distribution outside of the interface is the solution of the Frank Sphere at time t = 1.0 and zero temperature is set to the inside of the interface. We set $\Delta t = 0.001$, $a = 0.002$ and $b = 0.0$. Time levels shown are in increments of 0.5 up to the final time 9.0.

6. Conclusions

Motivated by the need for high-resolution representation of complex surfaces and front for applications to muti-dimensional detonation and shock physics, combustion

Res.\Δt	0.1	0.01	0.001	0.0001
80	1.917665	1.909899	1.909029	1.908939
160	1.920441	1.911303	1.910429	1.910339
320	1.92783	1.911274	1.910638	1.910545

Table 6. Average radius at time $t = 1.5$. Exact value is 1.910602

Res.\Δt	0.1	0.01	0.001	0.0001
80	7.06E-03	7.03E-04	1.57E-03	1.66E-03
160	9.84E-03	6.97E-04	1.73E-04	2.63E-04
320	1.72E-02	1.57E-03	1.81E-04	1.83E-04

Table 7. L^1 error for Radius at time $t = 1.5$

Res.\Δt	0.1	0.01	0.001	0.0001
80	9.74E-04	1.41E-04	1.57E-04	1.66E-04
160	1.04E-03	2.07E-04	1.64E-04	1.73E-04
320	1.24E-03	2.53E-04	2.03E-04	2.07E-04

Table 8. L^1 error for Temperature field.

and multi-material hydro-code simulation, we have developed an advanced, hybrid-level set method. We introduced algorithms that use parametric representations for the tracked fronts and interpolation to positions on the fronts. We used reciprocal procedures to generate the level-set fields in narrow bands, with the most accurate representation of the signed normal distance function near the front (on two layers of grid points surrounding the zero level-set). It is very important to have efficient sorting and storage procedures to generate segmented, parameterizable fronts, and they require well-designed data structures. We developed the Stack Sweeping algorithm reduced the complexity of our code implementation. We were able to show that we can successfully implement these algorithms to solve quite different moving boundary problems, including that where the front propagates according to intrinsic dynamics and others that depend on interpolation of field values to the front. By using the schemes described here, we obtain very high accuracy in the computation of distance function to the zero level-set, which is useful for re- initialization and essential in level-set method. We presented a clear description of the construction of a moving narrow band domain. Our description is robust, simple, and generic in that it can be used in 2-D and 3-D simulation with some obvious modification of the size of data structures in composing the narrow band domain. With our reciprocal procedures to generate the level-set fields in narrow bands, we have shown that the accuracy of the computation of distance function is nearly the same as the accuracy of the parameterization of the zero level-set. In a sequel to this paper we will show

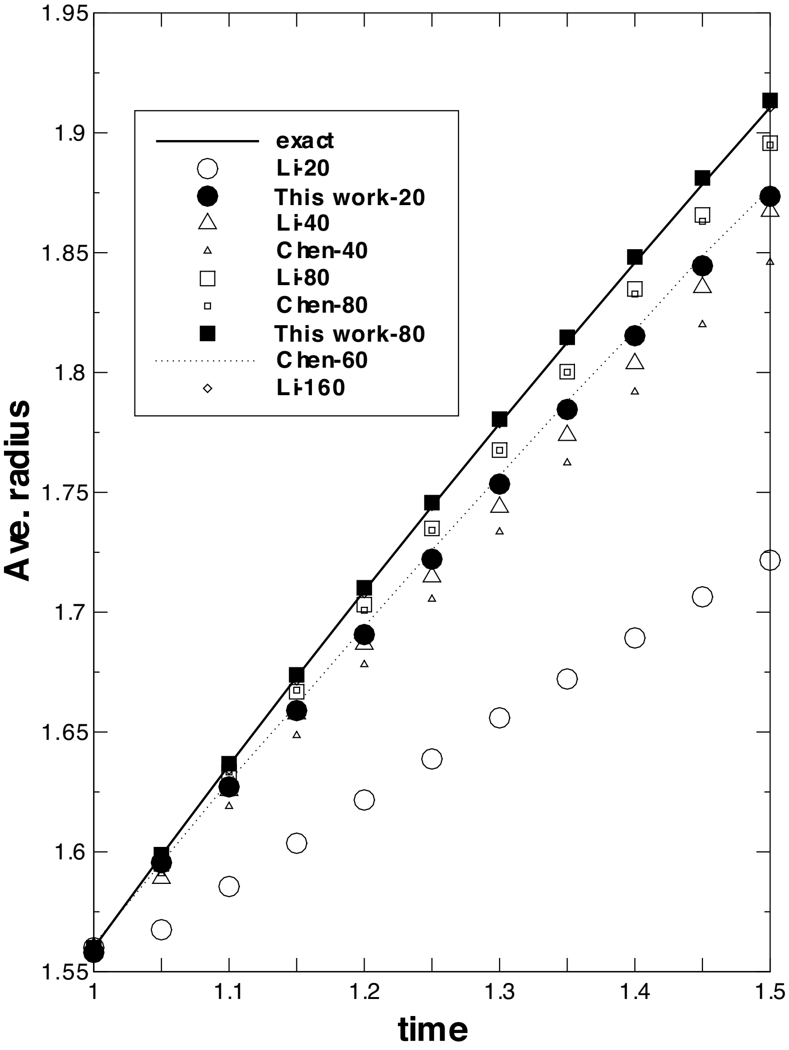


Figure 19. Comparison of radius at $t = 1.5$

how this interface strategy can be use to improve the accuracy of multi-material simulation.

Acknowledgements

This work was carried out with support from the US Air Force Wright Laboratories Armanent Directorate, Eglin Air Force Base F08630-00-1-0002, the US Air Force Office of Scientific Research, Physical Mathematics, F49620-00-1-0005 and support from the UIUC Center for Simulation of Advanced Rockets, U. S. Department of Energy through University of California subcontract B341494.

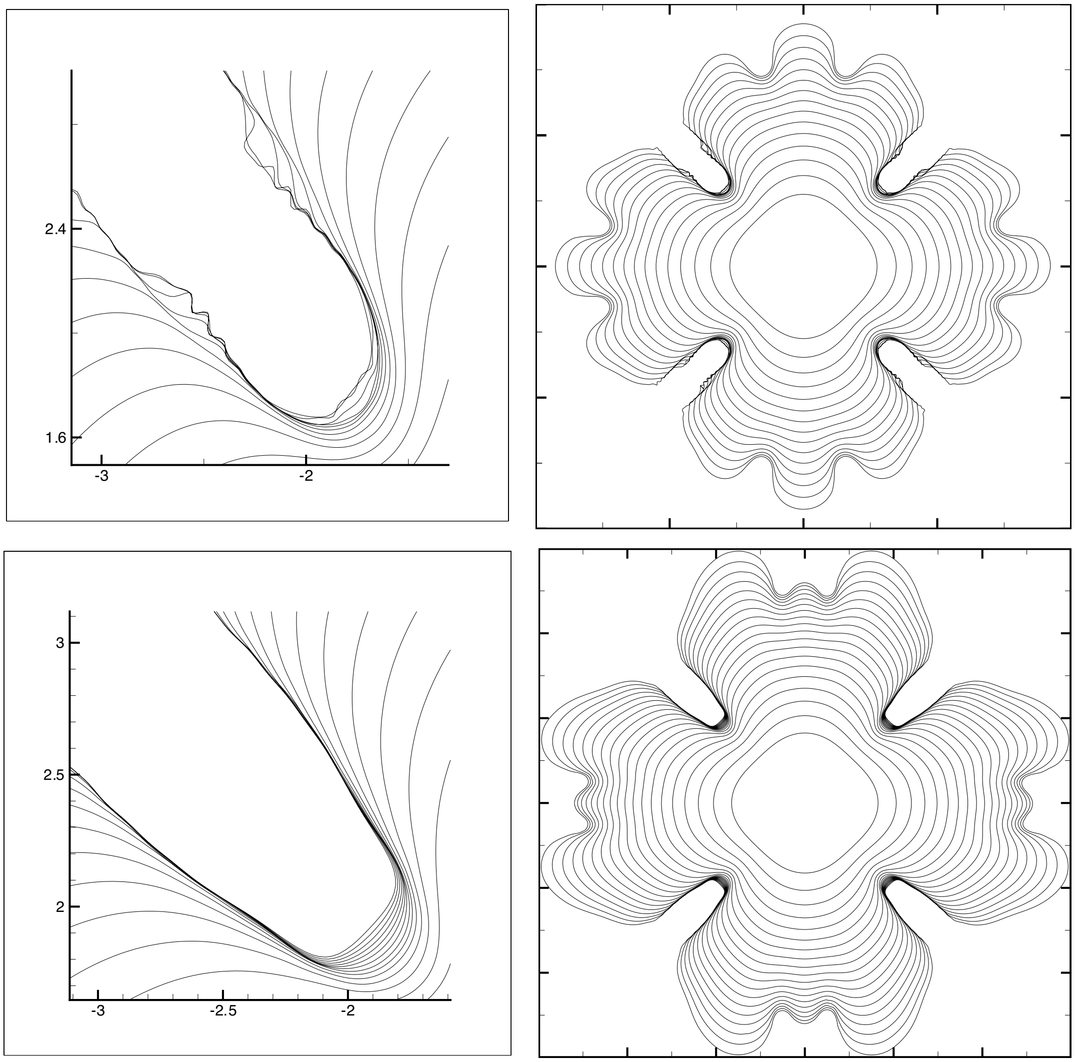


Figure 20. Comparisons of dendritic growth for grid resolutions 100×100 (bottom) and 200×200 (top)

7. Appendix I

7.1. Local Approximation Formula

Given three points $\mathbf{p}_0, \mathbf{p}_1$ and $\mathbf{p}_2$ and their associated unit surface normal vectors $\mathbf{n}_0, \mathbf{n}_1$ and $\mathbf{n}_2$, the local representation of surface is given by the formula

$$\mathbf{\Gamma}(\alpha) = \sum_{i=0}^{n} \alpha_i \mathbf{p}_i + h(\alpha) \sum_{i=0}^{n} \alpha_i \mathbf{n}_i \tag{37}$$

where $n = 1, 2$ and $\alpha = (\alpha_0, ...\alpha_n)$ is a barycentric coordinate, so $\sum_{i=0}^{n} \alpha_i = 1$. The function $h(\alpha)$ is a displacement function depending on the vectors $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{n}_0, \mathbf{n}_1$, and $\mathbf{n}_2$.

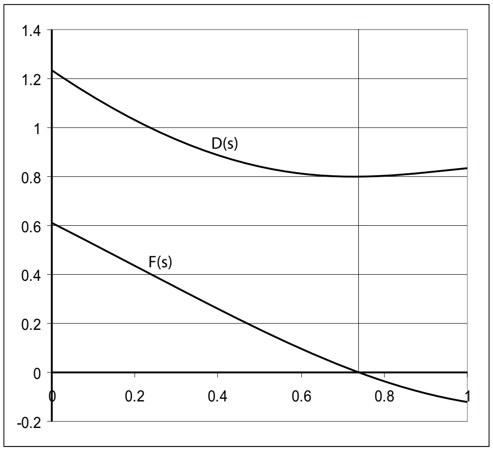


Figure 21. The distance function F and minimization function D.

For $n = 1$, the displacement function can be defined as a function of one variable t (say, by letting $\alpha_0 = 1 - t$ and $\alpha_1 = t$) as follow

$$h(t) = [\mathbf{p_0} - \mathbf{p}_1] \cdot \mathbf{n}_0 H_1(t) + [\mathbf{p_0} - \mathbf{p}_1] \cdot \mathbf{n}_1 H_2(t)$$

where $H_1(t) = (1-t)^2 t$, $H_2(t) = (1-t)t^2$, and $0 \leq t \leq 1$.

For $n = 2$, the displacement fuction $h(\alpha)$ is defined by loft operators and their blending operations as described by Smits *et al* , see [29].

7.2. Computation of Normal Distance Using Orthogonality

Let $\mathbf{p}$ be a point in V_Γ. The condition that the tangent vector and the displacement vector be orthogonal can be expressed as

$$F(s) \equiv (\mathbf{\Gamma}(s) - \mathbf{p}) \cdot \frac{d\,\mathbf{\Gamma}}{d\,s} = 0\,. \tag{38}$$

Then $F(s)$ is a polynomial of fifth degree and its zero gives $\bar{s}$ such that $\Gamma(\bar{s}) = \min_s ||\mathbf{\Gamma}(s) - \mathbf{p}||$. A Newton-Raphson iteration on the fifth order polynomial generally converges rapidly with only few iterations required to obtain an absolute error of $O(10^{-10})$ if the initial guess is sufficiently close to the solution.

Another way to compute the distance is to minimize the following function:

$$D(s) = ||\mathbf{\Gamma}(s) - \mathbf{p}||^2 \tag{39}$$

One method of the minimization is the successive iteration by quadratic interpolation of the function D (see [31]). The function D is a quadratic-like function as shown in figure 21. Given three values of parameters s_0, s_1 and s_2, the three-point quadratic formula $\tilde{D}$ of approximation to D is

$$\tilde{D}(s; s_0, s_1, s_2) = \sum_{k=0}^{2} \prod_{\substack{i=0 \\ i \neq k}}^{2} \frac{(s - s_i)}{(s_k - s_i)} D(s_k) \tag{40}$$

The value $\bar{s}$ that gives the minimum value of $\tilde{D}$ is obtained by the formula:

$$\bar{s} = \frac{1}{2}\frac{y_{12}D(s_0) + y_{20}D(s_1) + y_{01}D(s_2)}{s_{12}D(s_0) + s_{20}D(s_1) + s_{01}D(s_2)} \tag{41}$$

where $y_{ij} = s_i^2 - s_j^2$ and $s_{ij} = s_i - s_j$.

Therefore the procedure of the minimization can be summarized as: i) Initial step: Select three initial guess $s_0 = 0$, $s_1 = 0.5$ and $s_2 = 1.0$, ii) (Loop): compute $\bar{s} = \bar{f}(s_0, s_1, s_2)$, remove a parameter $s \in \{\bar{s}, s_0, s_1, s_2\}$ such that $D(s) = \max\{D(s_0), D(s_1), D(s_2), D(\bar{s})\}$, and set these remainders to $\{s_0, s_1, s_2\}$, iii) Repeat the step (Loop) four times.

For the computation of distance, we use a combination of the Newton method for $F(s)$ and the minimization method for the function $D(s)$. After four iterations of minimization method, the resulting value $\bar{s}$ is used to begin the iteration of Newton method for $F(s) = 0$ if $F(\bar{s}) > \epsilon$, where $\epsilon = 1.0 \times 10^{-8}$. Once a solution for s is obtained, the distance of $\mathbf{p}$ to Γ is known from evaluating the distance $||\mathbf{\Gamma}(\bar{s}) - \mathbf{p}||$ and is assigned to ψ at that grid point.

The direction of the tangent vector $\mathbf{t}$ is found unambiguously to be pointing along Γ in the direction of increasing arclength, with the interior (locally) on the left; $\mathbf{n}$ points to the exterior. (For example if $\mathbf{t} = (x, y)$, then the outward normal $\mathbf{n} = (-y, x)$.) Note that there can be more than one solution of the equation (38). For example, if Γ is a circle and the point y is the center of the circle, then all of the points on the circle are the solutions of (38). However, the issue of nonuniqueness can can generally be avoided by using sufficient resolution in the narrow band.

For the surface $\mathbf{\Gamma}(\alpha, \beta, \gamma)$ defined on a triangular patch in baricentric coordinates (α, β, γ), the corresponding orthogonality condition from a grid point $\mathbf{p}$ can be written as a system of two equations;

$$\left\{ \begin{array}{lcl} F(\alpha,\beta) & = & [\mathbf{\Gamma} - \mathbf{p}] \cdot \left(\frac{d\mathbf{\Gamma}}{d\alpha} - \frac{d\mathbf{\Gamma}}{d\gamma}\right)\Big|_{(\alpha,\beta,1-\alpha-\beta)} = 0, \\ G(\alpha,\beta) & = & [\mathbf{\Gamma} - \mathbf{p}] \cdot \left(\frac{d\mathbf{\Gamma}}{d\beta} - \frac{d\mathbf{\Gamma}}{d\gamma}\right)\Big|_{(\alpha,\beta,1-\alpha-\beta)} = 0. \end{array} \right. \tag{42}$$

To carry out the distance compuation, we first select a triangle $[\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}]$ and check if $\mathbf{q} = \mathbf{p} + d\,\mathbf{N}_{\mathbf{p_0},\mathbf{p_1},\mathbf{p_2}}$ is inside the domain. The normal $\mathbf{N}_{\mathbf{p_0},\mathbf{p_1},\mathbf{p_2}}$ is a normal vector of a plane determined by the three vertices $\{\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}\}$. The distance d from the point $\mathbf{p}$ to the triangle is given by a simple formula

$$d = \mathbf{N}/|\mathbf{N}| \cdot (\mathbf{p}_1 - \mathbf{p})\,. \tag{43}$$

The computed position $\mathbf{q}$ has a corresponding barycentric coordinate (α, β, γ) with respect to the triangle $[\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}]$. This barycentric coordinate (α, β, γ) can be computed by a simple system of equation (see [33])

$$\begin{pmatrix} m_{00} & m_{01} \\ m_{01} & m_{11} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} (\mathbf{p_0} - \mathbf{p_2}) \cdot (\mathbf{p} - \mathbf{p_2}) \\ (\mathbf{p_1} - \mathbf{p_2}) \cdot (\mathbf{p} - \mathbf{p_2}) \end{pmatrix} \tag{44}$$

where $m_{00} = (\mathbf{p_0} - \mathbf{p_2}) \cdot (\mathbf{p_0} - \mathbf{p_2})$, $m_{11} = (\mathbf{p_1} - \mathbf{p_2}) \cdot (\mathbf{p_1} - \mathbf{p_2})$ and $m_{01} = (\mathbf{p_0} - \mathbf{p_2}) \cdot (\mathbf{p_1} - \mathbf{p_2})$ and $\gamma = 1 - \alpha - \beta$.

If $0 \leq \alpha, \beta, \gamma \leq 1.0$, then the point $\mathbf{q}$ is inside the triangle $[\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}]$. The corresponding distance d is the distance from the point $\mathbf{p}$ to the surface patch to the first order of accuracy. We used a Broyden's method ([34]) to compute the equation (42) while the partial derivatives are computed by that of interpolation polynomials. The initial guess of the barycentric coordinate $(\alpha, \beta, 1 - \alpha - \beta)$ for the iteration in Broyden's method is given by the solution of equation 44.

We can use a minimization method to generate an initial guess for the Newton iteration for the distance to a surface, but we found that much more iteration is required to have a good guess. Instead of using a minimization procedure, we first compute the distance from a point to the boundary curves. Then we check the direction of tangents at the closest point of a curve, we choose a patch for the computation of the equation (42). In most cases, the closest point lies on the chosen patch. If not, we use a special subroutine to find another patch which is a neighbor of the previously chosen patch. If there is a corner on a boundary curve of a selected patch, we use a first order computation of distance around this corner.

7.3. Re-initialization of the level set fuction ψ by using the Stack Sweeping Algorithm

In this section we provide an example that describe an algorithm for re-initializing the level set function ψ in the band $\mathbf{B}_b$ by using our Stack Sweeping Algorithm. This discription is a variant of the algorithm for computing temperary ψ described by Sethian *et al* in the paper [4].

First we reserve two stacks, again named S_0 and S_1, whose lengths are twice the total number of nodal points in SURFACE.

STEP-I

(1) (*Initialization*) Starting with the first (0-th) item, say E_0 in SURFACE, get the vertices associated with the item E_0. Note that the vertices of E_0 (say) are the members of V_Γ. PUSH the addresses of all these vertices into S_0. Repeat this procedure until all items in SURFACE are used. After we have done this step, the stack S_0 contains all addresses of grid points in V_Γ.

(2) . (*Computation*) For each address in S_0 which represents a grid location, say (i, j), compute the normal distance from that point (i, j) to Γ by use of the orthogonal condition and save the distance into the corresponding item in BANDSET.

STEP-II:

(1) POP one address from stack S_0, if the stack is empty STOP.

(2) Find the grid location addressed by the POP-ed value say, (i, j).

(3) Find grid points among the neighbor of (i, j) i.e., $(i+1, j), (i-1, j), (i, j+1)$ or $(i, j-1)$, which are not in V_Γ but in $\mathbf{B}_b$.

(4) (*Computation*) If there are no such points, go back to step II-1, otherwise compute the normal distance at these points. Set the sign of $\psi_{(i\pm1,j\pm1)}$ at these points to the same sign of ψ_{ij}. Note that the location of nodes on Γ is determined partly by the sign of ψ at grid points in V_Γ. Set the result in the corresponding BANDSET and PUSH the address of BANDSET of the item into stack S_1.

(5) Go to II-1

STEP-III:
Switch the stack S_0 and S_1. If both stacks are empty, STOP.
STEP-IV:

(1) POP an address from S_0 If S_0 is empty then STOP.

(2) Find neighbors whose ψ field are not computed.

(3) (*Computation*) Compute ψ values by using the Fast Marching method, with known values at grid points determined in STEP-II.

(4) PUSH the addresses of neighbors into S_1.

(5) Go to IV-1.

STEP-V: Go to **STEP-III**

The Use of Min-Heap Storage for the PUSH and POP Operations For the PUSH operation that is used to fill stack S_1 at **STEP-II** and **STEP-IV**, we use the min-heap storage strategy to partially order the entries in S_1. Let us briefly review this sorting strategy. Assume that J is the total number of address entries in stack S_1 and for our current purpose the addresses of S_1 run from 0 to $J-1$. Initially $J = 0$, when the stack is empty. The ordered entries in S_1 can be indexed by j from $j = 0$ to $j = J - 1$. The stack is structured by the relation between child and parent entries. The relation between parent and child is as follows. A parent of the $j - th$ entry is the entry $(j-1)/2$. A child entry with index k is $2\,k + 1$.

The band-building PUSH operation has several steps. The first PUSH operation saves the PUSH-ed address in the first position of S_1 and J is increased by one. Let

A be the address to be PUSH-ed into S_1. Copy A into $J - th$ entry of S_1 (i.e. at the end of the stack). Compare the ψ-value of A with the ψ value of its parent node, which is $(J-1)/2$. If the magnitude ψ of A is smaller than the ψ value of its parent, then switch the contents of the child and parent entry in S_1. If the value ψ of A is larger or equal to ψ value of the parent node, then stop. If the $J - th$ entry is 0-th node, then stop. Otherwise, one repeats the process and continues till it terminates. Once the stack S_0 is empty, the stack S_1 is filled with structured addresses according to the min-heap tree structure. The stacks are SWITCH-ed and the addresses of S_0 reside in S_1 as shown in **STEP-III**.

The POP operation removes an item from stack S_0, but since we use the min-heap algorithm, the POP operator in the loop **STEP IV** breaks the tree-valence and so after having removed the first entry of S_0, we must "re-balance" the stack S_0. If j is the S_0 location of the item that is POP-ed, compare that ψ (i.e the parent) value with ψ values of its children nodes, i.e. those stored at $2\,j+1$ and $2\,j+2$. Next we copy the address of the child with the smallest magnitude of ψ to the $j - th$ storage location in S_0 and we remove the child node by setting its entry to the value -1. We carry out the same removal procedure for the child entry which was copied to the parent entry. And we repeat the removal until we encounter the end of the tree structure, i.e. when both children are -1.

Appendix III: Iteration Procedure for the Heat Equation

Step A Initialize the temperature $T_{i,j}^{n+1,N} = T_{i,j}^{n}$, with $N = 0$.

Step B For each (i,j), compute the left hand side of the equation (34) and update $T_{i,j}^{n+1,N}$ according to

$$T_{i,j}^{n+1,N} \leftarrow C_{i,j} T_{i,j}^{n,N} + f_{i,j,N}(T_{i,j+1}^{n+1}, T_{i,j-1}^{n+1}, ...)\,. \tag{45}$$

Step C Increment N: $N \leftarrow N+1$. If the criteria (35) for N and $N-1$ steps is not satisfied, Goto **Step B**; otherwise STOP.

We use the Stack Sweeping method described in section 3.3.4 to carry out the Gauss-Seidel iteration of **Step B** for temperature in the whole field. We use the same steps (without **STEP-II**) described previously. Instead of ψ (marked *computation* in section 3.3.4), we compute the temperature T, according to (34). The modified computation for **STEP-I** must include the procedure of extrapolation of temperature to Γ. The PUSH and POP operations simply copy and remove addresses from the stacks and the min-heap sorting is not required. We found that this conditioning of the Gauss-Seidal iteration leads to accelerated convergence to the updated solution $T_{i,j}^{n+1,\infty}$ for the temperature at the time step $t = t_{n+1}$.

Therefore our numerical procedure can be summarized as follows (see the flow chart in Figure 7)

(1) Initialize the temperature field and initial location of the front at time $t = t_0$.

(2) Start the WaveTracker.

(3) When WaveTracker requests the computation of the normal velocity on Γ at $t = t_{n+1}$, carry out the following two steps; i) Use the Gauss-Seidal iteration to advance the temperature on both side of the front. ii) Compute the normal velocity at all points on Γ using (32b).

References

[1] Adalsteinsson, D. and Sethian, J. A., *The fast construction of extension velocities in level set methods*, *J. Compu. Phys.*, **148**, 2-22, 1997

[2] de Boor, C. and A. Ron., *Computational aspects of polynomial interpolation in several variables.*, Math. Comp., 58, 705-727, 1992

[3] Chen, S. and Merriman, B. and Osher, S. and Smereka, P., *A simple level set Method for Solving Stefan Problems*, *J. Compu. Phys.*, 135, 8-29, 1997

[4] J. A. Sethian, *Level set Methods and Fast Marching Methods*, Cambridge, 1999,

[5] Thomas Sauer and Yuan Xu, *On Multivariate Lagrange Interpolation.*, Mathematics of Computation, **64**, 211, 1147-1170, Jul., 1995

[6] David Levin, *Mesh-independent surface interpolation.*, page 46, In 4th International Conference on Curves and Surfaces, Saint-Malo, 1999

[7] David Levin, *The approximation power of moving least-squares.*, Math. Comp., **67**, 224, 1517-1531, 1998

[8] Li-Tien Cheng and Ronald P. Fedkiw and Frederic Gibou and Myungjoo Kang, *A Symmetric Method for Implicit Time Discrization of the Stefan Problem*, CAM Report, UCLA, Oct. 2000

[9] Aslam, Tariq D. and Bdzil, John B. and Stewart, Scott, *Level set methods applied to modeling detonation shock dynamics.*, *J. Compu. Phys.*, **126**, 390-409, 1996

[10] Xu, Shaojie and Aslam, Tariq D. and Stewart, Scott, *High resolution numerical simulation of ideal and non-ideal compressible reacting flows with embedded internal boundaries.*, Combust. Theory Modeling, **1**, 113-142, 1997

[11] Stewart, D. S., *The shock dynamics of multi-dimensional condensed and gas phase detonations.*, Proceedings of the Combustion Institute, **27**, 2189-2205, 1998

[12] Hull, L. M., *Mach reflection of spherical detonation wave.*, Tenth Symposium(International) on Detonation, **1**, 9, 11-18, 1990

[13] H. S. Udaykummar and R. Mittal and Wei Shyy, *Computation of Solid-Liquid Phase Fronts in the Sharp Interface Limit on Fixed Grids*, Journal of Computational Physics, 153, 535-574, 1999

[14] G. Tryggvason and B. Bunner and A. Esmaeel and D.Juric and N. Al-Rawahi and W. Tauber and J. Han and S. Nas and Y.-J. Jan, 2001, *A Front-Tracking Method for the Computations of Multiphase Flow*, Journal of Computational Physics, 169, 708-759

[15] Chopp, D. L., *Some improvements of the fast Marching Method*, SIAM J. SCI. COMPUT, 23, 1, 230-244, 2001

[16] Keck, R., *Reinitialization for level set Methods*, Progress in Industrial Mathematics at ECMI 98, 10, 255-262, 1999

[17] Sussman, M. and Smereka, P. and Osher, S., 1994, *A level set Approach for Computing Solutions to Incompressible Two Phase Flow*, Journal of Computational Physics, 114, 146-159,

[18] Adalsteinsson, D. and Sethian, J. A., 1995, *A fast level set method for Propagating Interfaces*, Journal of Computational Physics, 118, 269-277,

[19] C. W. Shu and S. Osher, 1988, *Efficient Implementation of Essentially Non-Oscillatory Shock Capturing Scheme*, Journal of Computational Physics, 77, 439-471

[20] Sethian, J. A., *Evolution, Implementation, and Application of level set and Fast Marching Methods for Advancing Fronts*, Journal of Computational Physics, 169, 503-555, 2001

[21] Sethian, J. A., *Fast Matching Methods*, SIAM Review, 41, 2, 199-235, 1999

[22] Stanley Osher, 2001, *Level set Methods: An Overview and Some Recent Results*, Journal of Computational Physics, 169, 463-502

[23] Chopp, D. L., *Computing minimal surfaces via level set curvature flow*, @JCP, 106, 77-91, 1993

[24] Zhao, H., *A fast sweeping method for Eikonal equations*, CAM report, UCLA, , , 2001

[25] Jean-Daniel Boissonnat and Frédéric Cazals, *Smooth Surface Reconstruction via Natural Neighbour Interpolation of Distance Functions*, Comp. Geometry Theory and Applications, 185–203, 2002

[26] P. Alevizos and J-D. Boissonnat and M. Yvinec, *Non convex contour reconstruction*, Journal Symbolic Comput., 1990, **10**, 225-252

[27] Gary D. Knott *Interpolating cubic splines* Progress in Computer Science and Applied Logic, Birkhauser **18** 2000

[28] Zoltan J. Cendes and Steven H. Wong, *C^1 Quadratic Interpolation Over Arbitrary Point Sets.*, IEEE Computer Graphics & Applications, November, 1987

[29] Brian Smits, Peter Shirley and Michael M. Stark, *Direct ray Tracing of Smoothed and Displacement Mapped Triangles*, Technical Report UUCS-00-008, University of Utah

[30] Gerald Farin, *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*, Academic Press, Inc., San Diego, CA, 1990

[31] Hongling Wang, Joseph Kearney, and Kendall Atkinson, *Robust and Efficient Computation of the Closest Point on a Spline Curve*, Curve and Surface Fitting: Saint-Malo, 2002

[32] D. E. Lambert, D.S. Stewart, and S. Yoo, *Experimental Validation of Detonation Shock Dynamics in PBX-9501*, To be appeared, 2004

[33] David Eberly, *C^1 Quadratic Interpolation of Meshes*, Magic Software, Chapel Hill, NC,

[34] W. H. Press and S. A. Teukolsky and W. T. Vetterling and B. P. Flannery, Numerical Recipes in C., Cambridge University Press, Cambridge, 1992

List of Recent TAM Reports

No.	*Authors*	*Title*	*Date*
955	Fried, E.	An elementary molecular-statistical basis for the Mooney and Rivlin–Saunders theories of rubber-elasticity—*Journal of the Mechanics and Physics of Solids* **50**, 571–582 (2002)	Sept. 2000
956	Phillips, W. R. C.	On an instability to Langmuir circulations and the role of Prandtl and Richardson numbers—*Journal of Fluid Mechanics* **442**, 335–358 (2001)	Sept. 2000
957	Chaïeb, S., and J. Sutin	Growth of myelin figures made of water soluble surfactant—Proceedings of the 1st Annual International IEEE–EMBS Conference on Microtechnologies in Medicine and Biology (October 2000, Lyon, France), 345–348	Oct. 2000
958	Christensen, K. T., and R. J. Adrian	Statistical evidence of hairpin vortex packets in wall turbulence—*Journal of Fluid Mechanics* **431**, 433–443 (2001)	Oct. 2000
959	Kuznetsov, I. R., and D. S. Stewart	Modeling the thermal expansion boundary layer during the combustion of energetic materials—*Combustion and Flame*, in press (2001)	Oct. 2000
960	Zhang, S., K. J. Hsia, and A. J. Pearlstein	Potential flow model of cavitation-induced interfacial fracture in a confined ductile layer—*Journal of the Mechanics and Physics of Solids*, **50**, 549–569 (2002)	Nov. 2000
961	Sharp, K. V., R. J. Adrian, J. G. Santiago, and J. I. Molho	Liquid flows in microchannels—Chapter 6 of *CRC Handbook of MEMS* (M. Gad-el-Hak, ed.) (2001)	Nov. 2000
962	Harris, J. G.	Rayleigh wave propagation in curved waveguides—*Wave Motion* **36**, 425–441 (2002)	Jan. 2001
963	Dong, F., A. T. Hsui, and D. N. Riahi	A stability analysis and some numerical computations for thermal convection with a variable buoyancy factor—*Journal of Theoretical and Applied Mechanics* **2**, 19–46 (2002)	Jan. 2001
964	Phillips, W. R. C.	Langmuir circulations beneath growing or decaying surface waves—*Journal of Fluid Mechanics* (submitted)	Jan. 2001
965	Bdzil, J. B., D. S. Stewart, and T. L. Jackson	Program burn algorithms based on detonation shock dynamics—*Journal of Computational Physics* (submitted)	Jan. 2001
966	Bagchi, P., and S. Balachandar	Linearly varying ambient flow past a sphere at finite Reynolds number: Part 2—Equation of motion—*Journal of Fluid Mechanics* **481**, 105–148 (2003) (with change in title)	Feb. 2001
967	Cermelli, P., and E. Fried	The evolution equation for a disclination in a nematic fluid—*Proceedings of the Royal Society A* **458**, 1–20 (2002)	Apr. 2001
968	Riahi, D. N.	Effects of rotation on convection in a porous layer during alloy solidification—Chapter 12 in *Transport Phenomena in Porous Media* (D. B. Ingham and I. Pop, eds.), 316–340 (2002)	Apr. 2001
969	Damljanovic, V., and R. L. Weaver	Elastic waves in cylindrical waveguides of arbitrary cross section—*Journal of Sound and Vibration* (submitted)	May 2001
970	Gioia, G., and A. M. Cuitiño	Two-phase densification of cohesive granular aggregates—*Physical Review Letters* **88**, 204302 (2002) (in extended form and with added co-authors S. Zheng and T. Uribe)	May 2001
971	Subramanian, S. J., and P. Sofronis	Calculation of a constitutive potential for isostatic powder compaction—*International Journal of Mechanical Sciences* (submitted)	June 2001
972	Sofronis, P., and I. M. Robertson	Atomistic scale experimental observations and micromechanical/ continuum models for the effect of hydrogen on the mechanical behavior of metals—*Philosophical Magazine* (submitted)	June 2001
973	Pushkin, D. O., and H. Aref	Self-similarity theory of stationary coagulation—*Physics of Fluids* **14**, 694–703 (2002)	July 2001
974	Lian, L., and N. R. Sottos	Stress effects in ferroelectric thin films—*Journal of the Mechanics and Physics of Solids* (submitted)	Aug. 2001
975	Fried, E., and R. E. Todres	Prediction of disclinations in nematic elastomers—*Proceedings of the National Academy of Sciences* **98**, 14773–14777 (2001)	Aug. 2001

List of Recent TAM Reports (cont'd)

No.	*Authors*	*Title*	*Date*
976	Fried, E., and V. A. Korchagin	Striping of nematic elastomers—*International Journal of Solids and Structures* **39**, 3451–3467 (2002)	Aug. 2001
977	Riahi, D. N.	On nonlinear convection in mushy layers: Part I. Oscillatory modes of convection—*Journal of Fluid Mechanics* **467**, 331–359 (2002)	Sept. 2001
978	Sofronis, P., I. M. Robertson, Y. Liang, D. F. Teter, and N. Aravas	Recent advances in the study of hydrogen embrittlement at the University of Illinois—Invited paper, Hydrogen–Corrosion Deformation Interactions (Sept. 16–21, 2001, Jackson Lake Lodge, Wyo.)	Sept. 2001
979	Fried, E., M. E. Gurtin, and K. Hutter	A void-based description of compaction and segregation in flowing granular materials—*Continuum Mechanics and Thermodynamics*, in press (2003)	Sept. 2001
980	Adrian, R. J., S. Balachandar, and Z.-C. Liu	Spanwise growth of vortex structure in wall turbulence—*Korean Society of Mechanical Engineers International Journal* **15**, 1741–1749 (2001)	Sept. 2001
981	Adrian, R. J.	Information and the study of turbulence and complex flow—*Japanese Society of Mechanical Engineers Journal B*, in press (2002)	Oct. 2001
982	Adrian, R. J., and Z.-C. Liu	Observation of vortex packets in direct numerical simulation of fully turbulent channel flow—*Journal of Visualization*, in press (2002)	Oct. 2001
983	Fried, E., and R. E. Todres	Disclinated states in nematic elastomers—*Journal of the Mechanics and Physics of Solids* **50**, 2691–2716 (2002)	Oct. 2001
984	Stewart, D. S.	Towards the miniaturization of explosive technology—Proceedings of the 23rd International Conference on Shock Waves (2001)	Oct. 2001
985	Kasimov, A. R., and Stewart, D. S.	Spinning instability of gaseous detonations—*Journal of Fluid Mechanics* (submitted)	Oct. 2001
986	Brown, E. N., N. R. Sottos, and S. R. White	Fracture testing of a self-healing polymer composite—*Experimental Mechanics* (submitted)	Nov. 2001
987	Phillips, W. R. C.	Langmuir circulations—*Surface Waves* (J. C. R. Hunt and S. Sajjadi, eds.), in press (2002)	Nov. 2001
988	Gioia, G., and F. A. Bombardelli	Scaling and similarity in rough channel flows—*Physical Review Letters* **88**, 014501 (2002)	Nov. 2001
989	Riahi, D. N.	On stationary and oscillatory modes of flow instabilities in a rotating porous layer during alloy solidification—*Journal of Porous Media* **6**, 1–11 (2003)	Nov. 2001
990	Okhuysen, B. S., and D. N. Riahi	Effect of Coriolis force on instabilities of liquid and mushy regions during alloy solidification—*Physics of Fluids* (submitted)	Dec. 2001
991	Christensen, K. T., and R. J. Adrian	Measurement of instantaneous Eulerian acceleration fields by particle-image accelerometry: Method and accuracy—*Experimental Fluids* (submitted)	Dec. 2001
992	Liu, M., and K. J. Hsia	Interfacial cracks between piezoelectric and elastic materials under in-plane electric loading—*Journal of the Mechanics and Physics of Solids* **51**, 921–944 (2003)	Dec. 2001
993	Panat, R. P., S. Zhang, and K. J. Hsia	Bond coat surface rumpling in thermal barrier coatings—*Acta Materialia* **51**, 239–249 (2003)	Jan. 2002
994	Aref, H.	A transformation of the point vortex equations—*Physics of Fluids* **14**, 2395–2401 (2002)	Jan. 2002
995	Saif, M. T. A, S. Zhang, A. Haque, and K. J. Hsia	Effect of native Al_2O_3 on the elastic response of nanoscale aluminum films—*Acta Materialia* **50**, 2779–2786 (2002)	Jan. 2002
996	Fried, E., and M. E. Gurtin	A nonequilibrium theory of epitaxial growth that accounts for surface stress and surface diffusion—*Journal of the Mechanics and Physics of Solids* **51**, 487–517 (2003)	Jan. 2002
997	Aref, H.	The development of chaotic advection—*Physics of Fluids* **14**, 1315–1325 (2002); see also *Virtual Journal of Nanoscale Science and Technology*, 11 March 2002	Jan. 2002

List of Recent TAM Reports (cont’d)

No.	*Authors*	*Title*	*Date*
998	Christensen, K. T., and R. J. Adrian	The velocity and acceleration signatures of small-scale vortices in turbulent channel flow—*Journal of Turbulence*, in press (2002)	Jan. 2002
999	Riahi, D. N.	Flow instabilities in a horizontal dendrite layer rotating about an inclined axis—*Journal of Porous Media*, in press (2003)	Feb. 2002
1000	Kessler, M. R., and S. R. White	Cure kinetics of ring-opening metathesis polymerization of dicyclopentadiene—*Journal of Polymer Science A* **40**, 2373–2383 (2002)	Feb. 2002
1001	Dolbow, J. E., E. Fried, and A. Q. Shen	Point defects in nematic gels: The case for hedgehogs—*Proceedings of the National Academy of Sciences* (submitted)	Feb. 2002
1002	Riahi, D. N.	Nonlinear steady convection in rotating mushy layers—*Journal of Fluid Mechanics* **485**, 279–306 (2003)	Mar. 2002
1003	Carlson, D. E., E. Fried, and S. Sellers	The totality of soft-states in a neo-classical nematic elastomer—*Journal of Elasticity* **69**, 169–180 (2003) with revised title	Mar. 2002
1004	Fried, E., and R. E. Todres	Normal-stress differences and the detection of disclinations in nematic elastomers—*Journal of Polymer Science B: Polymer Physics* **40**, 2098–2106 (2002)	June 2002
1005	Fried, E., and B. C. Roy	Gravity-induced segregation of cohesionless granular mixtures—*Lecture Notes in Mechanics*, in press (2002)	July 2002
1006	Tomkins, C. D., and R. J. Adrian	Spanwise structure and scale growth in turbulent boundary layers—*Journal of Fluid Mechanics* (submitted)	Aug. 2002
1007	Riahi, D. N.	On nonlinear convection in mushy layers: Part 2. Mixed oscillatory and stationary modes of convection—*Journal of Fluid Mechanics* (submitted)	Sept. 2002
1008	Aref, H., P. K. Newton, M. A. Stremler, T. Tokieda, and D. L. Vainchtein	Vortex crystals—*Advances in Applied Mathematics* **39**, in press (2002)	Oct. 2002
1009	Bagchi, P., and S. Balachandar	Effect of turbulence on the drag and lift of a particle—*Physics of Fluids*, in press (2003)	Oct. 2002
1010	Zhang, S., R. Panat, and K. J. Hsia	Influence of surface morphology on the adhesive strength of aluminum/epoxy interfaces—*Journal of Adhesion Science and Technology* **17**, 1685–1711 (2003)	Oct. 2002
1011	Carlson, D. E., E. Fried, and D. A. Tortorelli	On internal constraints in continuum mechanics—*Journal of Elasticity* **70**, 101–109 (2003)	Oct. 2002
1012	Boyland, P. L., M. A. Stremler, and H. Aref	Topological fluid mechanics of point vortex motions—*Physica D* **175**, 69–95 (2002)	Oct. 2002
1013	Bhattacharjee, P., and D. N. Riahi	Computational studies of the effect of rotation on convection during protein crystallization—*Journal of Crystal Growth* (submitted)	Feb. 2003
1014	Brown, E. N., M. R. Kessler, N. R. Sottos, and S. R. White	*In situ* poly(urea-formaldehyde) microencapsulation of dicyclopentadiene—*Journal of Microencapsulation* (submitted)	Feb. 2003
1015	Brown, E. N., S. R. White, and N. R. Sottos	Microcapsule induced toughening in a self-healing polymer composite—*Journal of Materials Science* (submitted)	Feb. 2003
1016	Kuznetsov, I. R., and D. S. Stewart	Burning rate of energetic materials with thermal expansion—*Combustion and Flame* (submitted)	Mar. 2003
1017	Dolbow, J., E. Fried, and H. Ji	Chemically induced swelling of hydrogels—*Journal of the Mechanics and Physics of Solids*, in press (2003)	Mar. 2003
1018	Costello, G. A.	Mechanics of wire rope—Mordica Lecture, Interwire 2003, Wire Association International, Atlanta, Georgia, May 12, 2003	Mar. 2003
1019	Wang, J., N. R. Sottos, and R. L. Weaver	Thin film adhesion measurement by laser induced stress waves—*Journal of the Mechanics and Physics of Solids* (submitted)	Apr. 2003
1020	Bhattacharjee, P., and D. N. Riahi	Effect of rotation on surface tension driven flow during protein crystallization—*Microgravity Science and Technology* **14**, 36–44 (2003)	Apr. 2003

List of Recent TAM Reports (cont'd)

No.	*Authors*	*Title*	*Date*
1021	Fried, E.	The configurational and standard force balances are not always statements of a single law—*Proceedings of the Royal Society* (submitted)	Apr. 2003
1022	Panat, R. P., and K. J. Hsia	Experimental investigation of the bond coat rumpling instability under isothermal and cyclic thermal histories in thermal barrier systems—*Proceedings of the Royal Society of London A*, in press (2003)	May 2003
1023	Fried, E., and M. E. Gurtin	A unified treatment of evolving interfaces accounting for small deformations and atomic transport: grain-boundaries, phase transitions, epitaxy—*Advances in Applied Mechanics*, in press (2003)	May 2003
1024	Dong, F., D. N. Riahi, and A. T. Hsui	On similarity waves in compacting media—*Horizons in Physics*, in press (2003)	May 2003
1025	Liu, M., and K. J. Hsia	Locking of electric field induced non-180° domain switching and phase transition in ferroelectric materials upon cyclic electric fatigue—*Applied Physics Letters*, in press (2003)	May 2003
1026	Liu, M., K. J. Hsia, and M. Sardela Jr.	In situ X-ray diffraction study of electric field induced domain switching and phase transition in PZT-5H—*Journal of the American Ceramics Society* (submitted)	May 2003
1027	Riahi, D. N.	On flow of binary alloys during crystal growth—*Recent Research Development in Crystal Growth*, in press (2003)	May 2003
1028	Riahi, D. N.	On fluid dynamics during crystallization—*Recent Research Development in Fluid Dynamics*, in press (2003)	July 2003
1029	Fried, E., V. Korchagin, and R. E. Todres	Biaxial disclinated states in nematic elastomers—*Journal of Chemical Physics* **119**, 13170–13179 (2003)	July 2003
1030	Sharp, K. V., and R. J. Adrian	Transition from laminar to turbulent flow in liquid filled microtubes—*Physics of Fluids* (submitted)	July 2003
1031	Yoon, H. S., D. F. Hill, S. Balachandar, R. J. Adrian, and M. Y. Ha	Reynolds number scaling of flow in a Rushton turbine stirred tank: Part I—Mean flow, circular jet and tip vortex scaling—*Chemical Engineering Science* (submitted)	Aug. 2003
1032	Raju, R., S. Balachandar, D. F. Hill, and R. J. Adrian	Reynolds number scaling of flow in a Rushton turbine stirred tank: Part II—Eigen-decomposition of fluctuation—*Chemical Engineering Science* (submitted)	Aug. 2003
1033	Hill, K. M., G. Gioia, and V. V. Tota	Structure and kinematics in dense free-surface granular flow—*Physical Review Letters*, in press (2003)	Aug. 2003
1034	Fried, E., and S. Sellers	Free-energy density functions for nematic elastomers—*Journal of the Mechanics and Physics of Solids*, in press (2003)	Sept. 2003
1035	Kasimov, A. R., and D. S. Stewart	On the dynamics of self-sustained one-dimensional detonations: A numerical study in the shock-attached frame—*Physics of Fluids* (submitted)	Nov. 2003
1036	Fried, E., and B. C. Roy	Disclinations in a homogeneously deformed nematic elastomer—*Nature Materials* (submitted)	Nov. 2003
1037	Fried, E., and M. E. Gurtin	The unifying nature of the configurational force balance—*Mechanics of Material Forces* (P. Steinmann and G. A. Maugin, eds.), in press (2003)	Dec. 2003
1038	Panat, R., K. J. Hsia, and J. W. Oldham	Rumpling instability in thermal barrier systems under isothermal conditions in vacuum—*Philosophical Magazine* (submitted)	Dec. 2003
1039	Cermelli, P., E. Fried, and M. E. Gurtin	Sharp-interface nematic–isotropic phase transitions without flow—*Archive for Rational Mechanics and Analysis* (submitted)	Dec. 2003
1040	Yoo, S., and D. S. Stewart	A hybrid level-set method in two and three dimensions for modeling detonation and combustion problems in complex geometries—*Combustion Theory and Modeling* (submitted)	Feb. 2003