Journal of Computational Physics **157,** 683–706 (2000)
doi:10.1006/jcph.1999.6389, available online at http://www.idealibrary.com on IDEAL®

A New Eulerian Method for the Computation of Propagating Short Acoustic and Electromagnetic Pulses

John Steinhoff, Meng Fan, and Lesong Wang

The University of Tennessee Space Institute, Tullahoma, Tennessee 37388
E-mail: {jsteinho, mfan, lwang2}@utsi.edu

Received November 2, 1998; revised October 6, 1999

A new method is described to compute short acoustic or electromagnetic pulses that propagate according to geometrical optics. The pulses are treated as zero thickness sheets that can propagate over long distances through inhomogeneous media with multiple reflections. The method has many of the advantages of Lagrangian ray tracing, but is completely Eulerian, typically using a uniform Cartesian grid. Accordingly, it can treat arbitrary configurations of pulses that can reflect from surfaces and pass through each other without requiring special computational marker arrays for each pulse. Also, information describing the pulses, which are treated as continuous surfaces, can be available throughout the computational grid, rather than only at isolated individual markers. The method uses a new type of representation, which we call "Dynamic Surface Extension." The basic idea is to propagate or "broadcast" *defining fields* from each pulse surface through a computational grid. These fields carry information about a nearby pulse surface that is used at each node to compute the location of the pulse surfaces and other attributes, such as amplitude. Thus the emphasis is on the dynamics of these propagating defining fields, which obey only local Eulerian equations at each node. The Dynamic Surface Extension representation can be thought of as dual to level set representation: The defining fields involve single valued variables which are constant at each time along lines that are normal to the evolving surface, whereas level set techniques involve a function which has constant values on the evolving surface and neighboring surfaces. In this way the new method overcomes the inability of level set or Eikonal methods to treat intersecting pulses that obey a wave equation and can pass through each other, while still using only single-valued variables. Propagating thin pulse surfaces in 1-D, 2-D, and 3-D that can reflect from boundaries and pass through each other are computed using the new method. The method was first presented as a new, general representation of surfaces, filaments, and particles by J. Steinhoff and M. Fan (1998, Eulerian computation of evolving surfaces, curves and discontinuous fields, UTSI preprint).

Key Words: numerical analysis; optics, electromagnetic theory.

0021-9991/00 $35.00

1. INTRODUCTION

The main subject of this paper is the computation of short acoustic or electromagnetic pulses that propagate over very long distances and can reflect from solid surfaces, refract, and pass through each other without interaction. These pulses are taken to be smooth, thin surfaces that, in the short wavelength limit, propagate along local normals according to geometrical optics. The computation involves a new representation of surfaces, which we call "Dynamic Surface Extension." This was first described as a new representation in a more general context in Ref. [1].

Currently existing methods for treating propagating pulse problems include:

(1) Direct discretization of the governing (i.e., acoustic, electromagnetic, etc.) partial differential equations on a fixed (Eulerian) computational grid [2].

(2) Similar discretization with a stable nonlinear "negative viscosity" or "confinement" term added to the equations [3, 4].

(3) Direct discretization of the governing equations on an adaptive grid [5].

(4) Eikonal equation and "level set" methods [6–10].

(5) Lagrangian "ray tracing" methods [11].

(6) Green's Function based methods, such as Kirchhoff's in [12].

Of these, schemes (1), (3), and (6) treat the wave (i.e., short wave equation pulse) as having a dynamical internal structure, which is defined by variables that obey the governing equations even inside the pulse. The other methods typically capture the pulse and treat the internal structure as having no relevant degrees of freedom: (2) results in a solitary wave representation while (4) and (5) treat the pulse as a (zero thickness) surface.

All of these methods have well-known advantages and deficiencies. These are outlined below:

(1) Direct Discretization. Since a linear wave equation pulse, unlike a shock, does not have characteristics pointing inward, a relatively large number of points (of the order of 10 or more) across the pulse are required to reduce numerical error and induced spreading to get resonable accuracy for the internal structure. Resulting computer cost is too high to make many problems feasible.

(2)—(1) with "Confinement." Here, a nonlinear term is added to the discretized wave equation which results in a solitary wave-like internal structure only $\sim$2 grid cells thick, which maintains a fixed shape as the pulse propagates over arbitrarily long distances. This new formulation still allows pulses to pass through each other with essentially no interaction.

As in shock capturing, accurate results are obtained outside the pulse since the method conserves variables integrated through the pulse. In many cases the internal structure of the pulse is not important since it is thin and this method is both accurate and economical. However, care must be taken to avoid numerical instabilities along the surface in some multidimensional problems. This method has also been extended to compute (multivalued) Eikonal functions for waves that pass through each other [3].

(3)—(1) with Adaptive Grids. Here, as in (1), the internal structure is resolved, but with less computer storage due to adaptive grid compression. However, the adaptation is complex, especially for unstructured grids, and precludes simple highly parallel computational approaches. Since the problems involve moving pulses, this adaptation must be done continually, causing these methods to be inefficient. Further, for realistic, complex problems involving many pulses, the computer requirements can greatly increase.

(4) Eikonal and Level Set Methods. These methods are efficient for many problems with non-intersecting or annihilating pulses, treating them as well-defined zero thickness surfaces. However, unlike all of the other methods, these generate viscosity solutions that cannot easily accommodate surfaces that intersect and pass through each other without interaction. This makes the methods complex for realistic problems involving a number of acoustic or electromagnetic pulses that obey a linear wave equation and pass through each other without interaction. Attempts have been made to use multi-valued variables to treat these problems [8]. These involve additional complexities. The problem is related to the computation of a distance function for intersecting curves: while straightforward for single propagating curves or pairs that annihilate, curves that pass through each other are difficult to treat.

(5) Ray Tracing. If the initial pulse locations (surfaces) are known beforehand and do not change topologically, arrays of computational markers can be assigned to them and they can be tracked in a Lagrangian sense. However, when pulses are absorbed or new ones created, this method is not effective as a general one since the originally assigned markers must be changed. Further, the markers typically diverge as they propagate, and they may not be useful for computating thin surfaces that can stretch across the grid, except at the particular locations of the markers. Although additional markers can be added in an adaptive sense, these methods then develop drawbacks such as in (3).

(6) Green's Function Based Methods. These methods rely on a closed form Green's function which is integrated over a fixed surface where the amplitude is known initially, to describe the propagation to a given observer. Typically, these methods can only be used in free space and cannot account for inhomogeneous media or reflections, since a closed form Green's function is not then available.

In this paper we describe the application of a new method that treats short wave equation pulses which propagate according to geometrical optics as zero-thickness surfaces on an Eulerian grid, as in Eikonal or Level Set schemes (4). However, the method automatically treats pulses that pass through each other, as in the other methods. The method is based only on local equations of motion for discretized fields and can treat inhomogeneous media and reflections. None of the above methods can apparently do this in a general, efficient way.

The method utilizes a regular computational grid with no adaptation (typically uniform Cartesian) with no markers and can treat arbitrary configurations of (zero thickness) pulse surfaces, as long as each is smooth enough along its surface to be resolved by the grid.

The method involves a "Dynamic Surface Extension" (DSE) representation that is apparently new and different from the above schemes. It will be described in Section 2. A basic 1-D implementation will be discussed in Section 3, 2-D implementations in Section 4, and 3-D in Section 5. Section 6 will describe some generalizations, possible extensions, and problems still to be addressed and Section 7 will present the conclusions. An initial description and implementation of the method for general propagating surfaces, filaments, and particles were given by the first two authors in Ref. [1].

Error estimates are given for our method by comparing the results with those of a Lagrangian marker based method, both graphically and numerically in table form.

The cases treated in this paper involve only constant index of refraction. For these cases, which represent an important class of problems, a very efficient implementation is described which does not require interpolation, but only averaging of some of the variables. Variable index of refraction would require an extension of the method which would involve, among

other additions, interpolation onto the location of the surface. This interpolation would require logic to avoid interpolating between data on neighboring nodes if they correspond to different surfaces. This logic is similar to that which the current method uses when computing averages, but other extensions are required. We are currently developing an extended version of the method to treat these cases. Also, such an extension is treated in a subsequent work [13].

2. OUTLINE OF THE METHOD

We first mention the variables used by the various methods to define the pulse location so that they can be contrasted to the DSE representation.

In all of the conventional Eulerian methods mentioned (1)–(4), a field is defined whose values, in some way, determine the location (and possibly other properties) of the wave equation pulse. For (1) and (3), the actual pulse is resolved, including the dynamics of the internal structure. For (2), the computed function defines only the centroid surface of the pulse and its amplitude (integrated through the pulse at each point on the surface). For (4), level sets of a smooth function determine the location of the surface at each time. By contrast, the values of the particle trajectories of the Lagrangian method (4) *directly* determine the location of points or markers on the pulse surface (as well as total amplitudes). These values are stored as separate arrays in the computer for each pulse surface.

In the Dynamic Surface Extension method, we treat each pulse surface as defined by fields discretized only on an Eulerian grid as in (1)–(4), and as in (2), (4), and (5), do not compute details of the internal structure. Also, as in (4), the pulse is treated as a zero-thickness surface. However, unlike in those methods, we do not use any variable whose centroid or level set determines the location of the surface. Instead, we propagate *defining fields* through the grid (at least in the vicinity of the surface) whose values can be used at each node to compute the location (as well as other attributes) of the pulse surface. An important property is that the governing equations for propagating these new fields are local. As in method (2) (Refs. [3, 4]), the method uses nonlinear difference equations for computing variables at each grid node and does not involve any approximate discretization of a partial differential equation.

The key to the success of the DSE method is that the defining fields are constant along lines normal to the surface. To propagate them away from a surface (in the presence of other intersecting but non-interacting surfaces) is thus easier than computing a distance function, as in level set schemes, which has a specified gradient. This property of DSE can be thought of as “dual” to level set representations, where a function is defined which takes constant values on the pulse surface and neighboring surfaces (see Fig. 1). While it is easy to compute level set functions such as signed distance for isolated surfaces or ones that annihilate each other, this is difficult in the region between surfaces passing through each other, and DSE variables that are constant along normals associated with each surface seem to have an advantage.

The important point is that even though the quantities that we transfer still vary along the surface, we have no variation in the normal direction, which is the direction of motion of the surface. We use this transferred information together with the known position of each grid node to compute algebraically a distance function. Level Set schemes, by contrast, attempt to compute this distance by solving a finite difference equation that specifies its

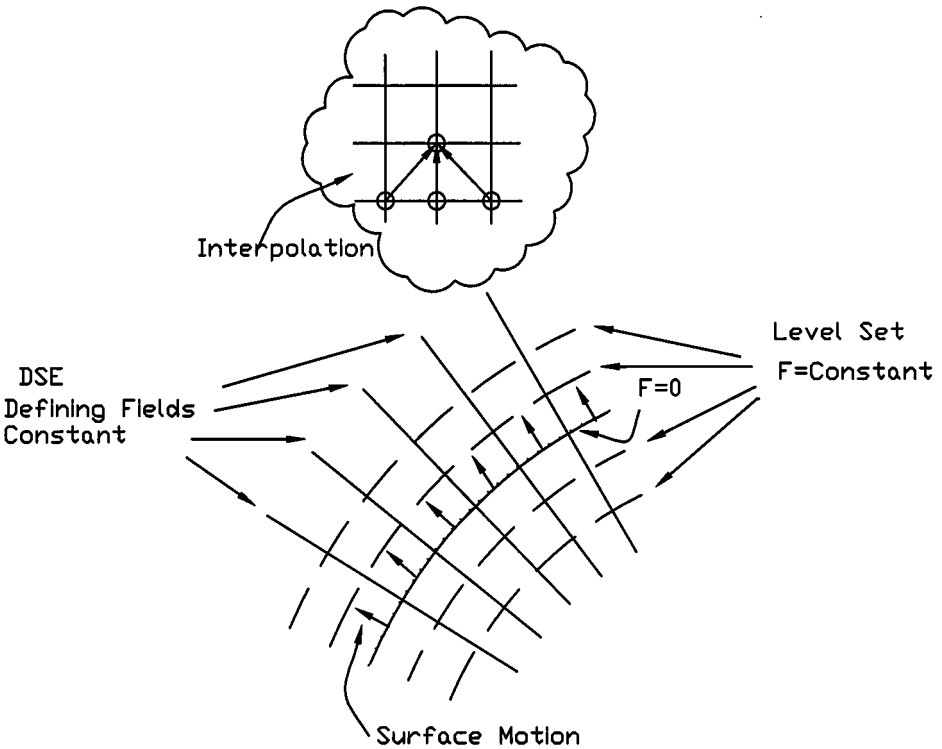

FIG. 1. Surface definition and propagation.

variation in the normal direction, when computing the function in a new region where it was not previously computed. This difference is important when there are intersecting non-interacting surfaces and the particular surface that a distance function refers to at a particular grid point can change from one time step to the next.

It is important to realize that all of the quantities available in level set methods are available in DSE. Even though different variables are propagated from node to node, distance functions are algebraically computed from these, as well as unit vectors normal to the surface, etc.

3. THE 1-D DESCRIPTION

3.1. *Basic Description*

The simplest example that can be used to describe the method involves parallel planar surfaces propagating along their normals and passing through each other. These pulses can be defined by the coordinates and velocities along the (common) normal, and the problem is one-dimensional. We feel that this digression to a much simpler problem is important for explaining the basic idea behind the representation, since it is unconventional. The multidimensional applications involve essentially the same idea.

Although in 1-D each surface is effectively a particle, we still call each a surface. Here, the Lagrangian description would be trivial: it would involve an array of position variables, $\tilde{y}_k(t)$, and velocity variables, $\tilde{u}_k(t)$ for each surface (labled k).

In this section a "verbal" description of the method will be given. In the next, an algorithmic one.

In the traditional Eulerian descriptions, scalar fields would be defined on a computational axis, which would be concentrated near the $\tilde{y}_k$'s, the centroid values. For the Eikonal or Level Set methods separate signed distance functions would be defined from each grid point to the $\tilde{y}_k$'s at each time. The Dynamic Surface Extension method involves elements from both Lagrangian and Eulerian descriptions: Instead of the above scalar fields, the *actual* values of the surface velocity and other attributes are propagated or "broadcast" through the grid (at least in the region within a few cells of each surface, where they are required), from point-to-point, at a speed greater than that of the moving surfaces. The surface coordinates can then either be computed at each (nearby) node by integrating velocity or computed only on nodes near each surface and also propagated. What results is a "cloud" of variables representing features of each surface (including the current coordinates) that follow each surface on the surrounding nodes as it moves along the axis. Although the actual surface coordinates and velocities are used in this example, the general idea is to use *any* variables that can, in turn, be used at each node to compute the coordinates. Other examples of such variables will be used in the 2-D and 3-D propagating surface examples, treated in Sections 4 and 5.

Since, in general, there can be a number of surfaces, a rule must be implemented to select the variable set for each grid node corresponding to the nearest surface (the variables are single valued; only one set is defined at each node, although a number of nodes can have variables for the same surface): We denote a surface position and velocity stored at node l as y_l and u_l, regardless of which surface they refer to. Each time step (n), each node (say l_0) surveys the nearest neighbors regarding the stored surface coordinates y_l, where l denotes the neighboring nodes: $l_0 \pm 1$. The (unsigned) distance between the *actual* coordinate of the l_0 node, x_{l_0}, and the surface coordinates stored on the neighboring nodes, y_l, is then computed,

$$d_l = \left| x_{l_0} - y_l \right|.$$

The node with the smallest d_l (say $l = m$) is chosen and the attributes of the surface stored at that node (including y_m and u_m) are transferred to node l_0. The surface position is updated at each node in this way. After this transfer (in the simplest version of the method), a new value of y_l is computed by integration in time at each node (at least near each surface) for the next time step using the newly transferred velocity, u_m (see Fig. 2).

If the index of refraction, and hence velocity, depends on position, the new velocity is interpolated on the nodes $(j, j+1)$ surrounding the surface: $x_j < y_j < x_{j+1}$. New values of velocity are then propagated to the surrounding nodes.

Although, of course, the attributes need only be "broadcast" to those grid nodes near each surface, they can, instead, be continued throughout the grid. Then, the attributes are smoothly varying (possibly constant) at nodes surrounding each surface with discontinuities between surfaces. It is important to mention that the Lagrangian labels (k) are not used and the surfaces can be treated as indistinguishable, as in other Eulerian methods.

3.2. *Implementation*

In the following we denote "actual" surface attributes, which, as above, are defined as separate Lagrangian variables and used only for reference, with a tilde. We let the surfaces move with coordinate $\tilde{y}_k(t)$ and velocity $\tilde{u}_k$, which can depend on other parameters. For example,

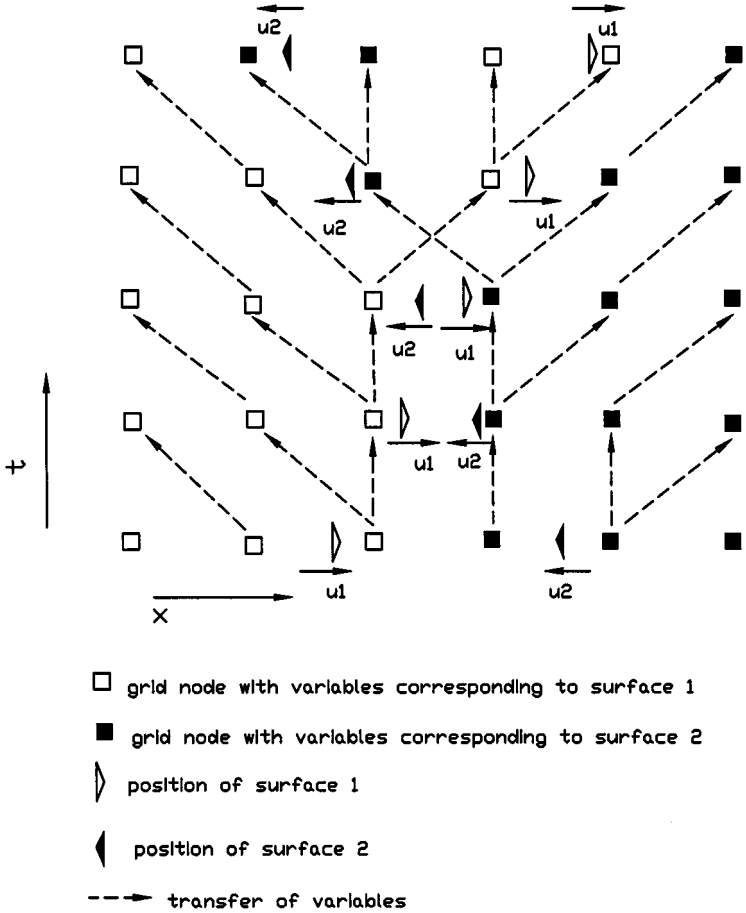

FIG. 2. Illustration of the method in 1-D.

if each has a fixed velocity then it obviously moves from time t to $t + \Delta t$ according to

$$\tilde{y}_k(t + \Delta t) = \tilde{y}_k(t) + \tilde{u}_k \cdot \Delta t.$$

Other parameters could define amplitude, polarization, etc. This is, of course, trivial to treat in a Lagrangian way.

To transform from the Lagrangian to our Eulerian representation we define surface attributes stored at each node, j, as y_j, u_j, etc., without a tilde. The position of a nearby surface is then, for example, y_j. Then, at each time step (n) and each grid node (j), we perform 4 operations:

1. Compute 3 distances from node j,

$$d_j^{+1} = x_j - y_{j+1}^n$$
$$d_j^0 = x_j - y_j^n$$
$$d_j^{-1} = x_j - y_{j-1}^n.$$

Here x_j is the (fixed) coordinate of node j and y_l is the value of the moving surface coordinate stored at node $l = j, j \pm 1$ (see Fig. 2).

2. Find σ such that

$$|d_j^\sigma| = \min\left(|d_j^{+1}|, |d_j^0|, |d_j^{-1}|\right),$$

i.e., find the neighboring point whose stored y value represents the surface closest to node j.

3. Transfer surface attributes (if $\sigma \neq 0$) to node j,

$$\hat{y}_j^n = y_{j+\sigma}^n, \qquad \hat{u}_j^n = u_{j+\sigma}^n.$$

4. Compute new values of y_j at time step $n+1$, i.e., to first order,

$$y_j^{n+1} = \hat{y}_j^n + \hat{u}_j^n \cdot \Delta t.$$

This algorithm will allow surfaces to move "through" each other without interacting unless more than two happen to meet at the same node. Only then will information be lost. The point is that we want to compute multidimensional non-interacting surfaces and this is a good prototype. In multidimensions for problems involving surfaces, a large number of surrounding nodes usually has attributes corresponding to each surface and if information is lost at any particular node at one time step due to the convergence of more than two surfaces, it is quickly replaced in the next few time steps by transfer from nearby nodes (unless the surfaces are exactly parallel), as will be seen in Sections 4 and 5.

In the 1-D problem described in this section, the motion of each surface depends only on its own attributes and not on a varying external field such as index of refraction. Then, new coordinates for any surface can be computed at each node based on stored attributes since it moves at a constant speed. This is analogous to the free-space propagation of multidimensional surfaces that we are interested in.

To clarify the basic ideas of the Dynamic Surface Extension method, we consider a specific case of two surfaces moving with opposite (constant) velocities, such that they pass through each other near the middle of the grid. Each surface then only has two attributes: coordinate $\tilde{y}_k(t)$ and velocity $\tilde{u}_k$, $k = 1, 2$ and each grid node (labled j) has values of y_j^n and u_j^n at each time step n. The basic idea is that the values on each node correspond to those of the nearest surface.

The problem is initialized such that the correct values of $\tilde{y}_k(t)$ and velocity $\tilde{u}_k$ are set on only those nodes at the cells containing the initial values, i.e., j_1, j_1+1 and j_2, j_2+1 such that

$$x_{j_1} < \tilde{y}_1(0) < x_{j_1+1},$$
$$x_{j_2} < \tilde{y}_2(0) < x_{j_2+1},$$

where x_j is the coordinate of node j.

Initially, large values of y_j are defined for all other nodes, corresponding to very distant surfaces, and u_j is set to 0. As n increases, the actual surfaces move. What we see on the grid is a set of defining field "waves" moving, initially at the speed of one cell per time step, as $\tilde{u}_1$ or $\tilde{u}_2$ replaces the previous values of u_j which were 0, and $\tilde{y}_1$ or $\tilde{y}_2$ replaces the original large values. This sequence is presented in Fig. 3 for y_j where symbols on the bottom of the plots represent the actual surface positions, ($\tilde{y}$). The waves can be seen to stop progressing when they meet near the middle and reach the ends because each node then has values corresponding to the nearest surface. The u_j values (not shown) remain constant between the waves and, as can be seen, the y_j values move uniformly up or down corresponding to the changing values of $\tilde{y}_1$ and $\tilde{y}_2$. As expected, there is finally a discontinuity in u_j and y_j near the middle after the defining field waves have met. (These computational "waves"

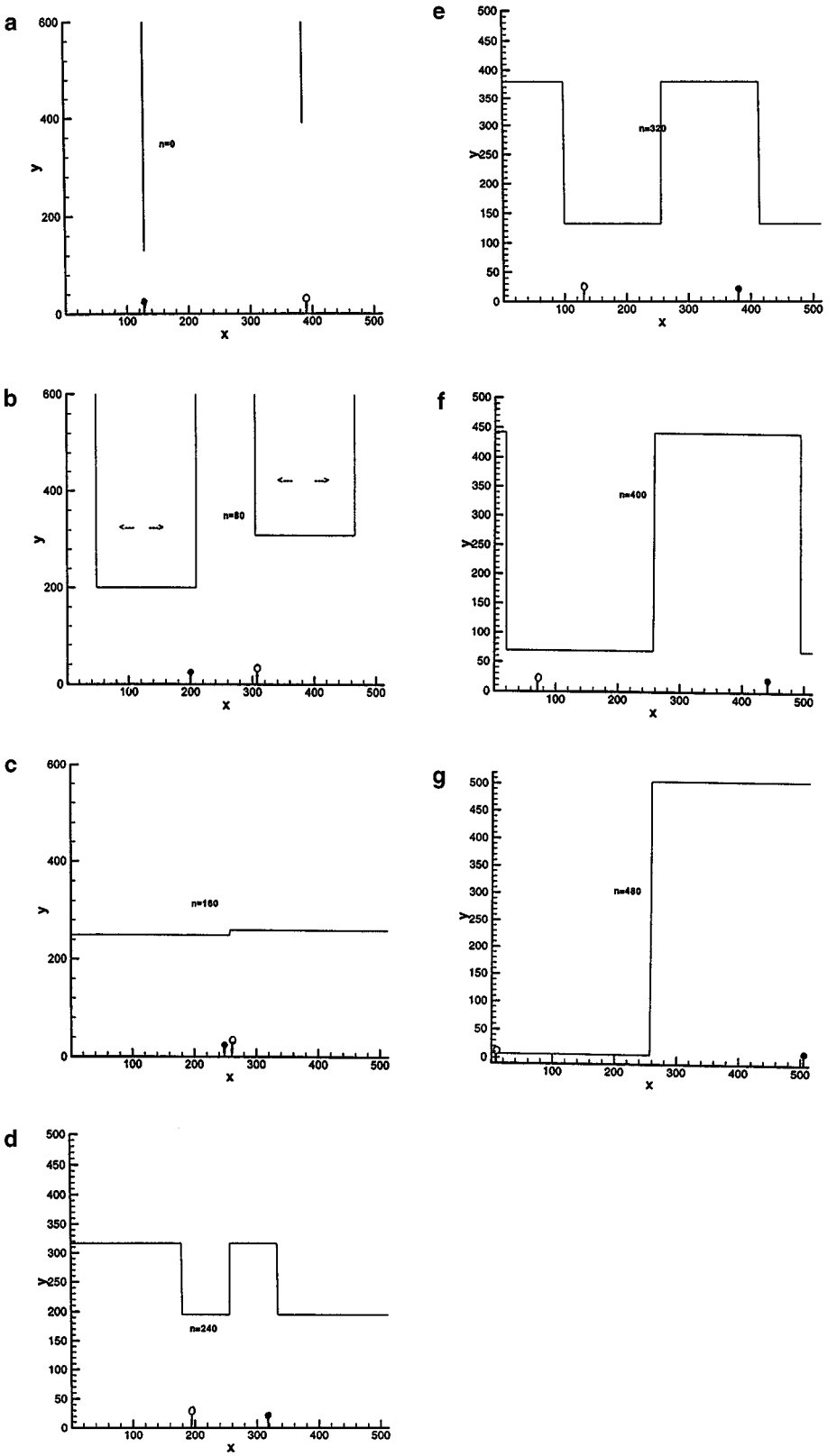

FIG. 3. Propagation of position information (two surfaces in 1-D).

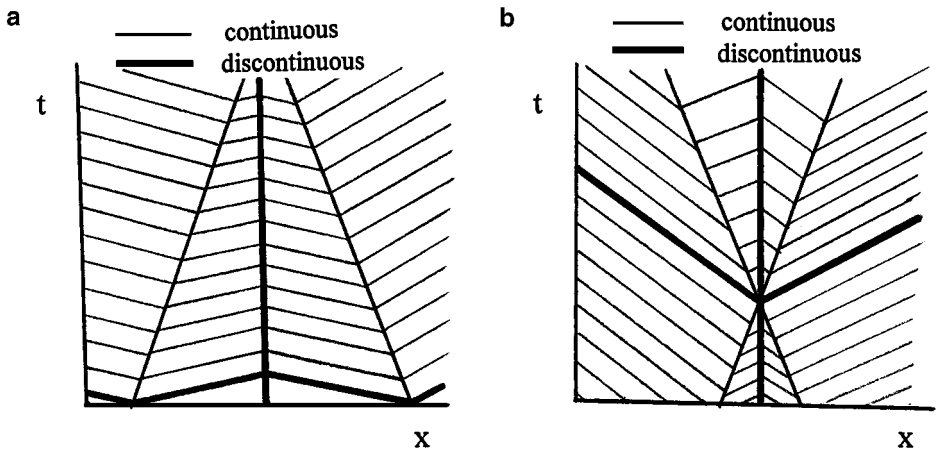

FIG. 4. Characteristics for information propagation (two surfaces in 1-D).

are, of course, waves of the defining fields and not the physical surfaces that we wish to compute.) Of course, as mentioned, we only need to compute these values at nodes within a few cells of the actual surface positions. This could obviously be done to save computing time. We continue the propagation to avoid introducing another length scale and to better illustrate the basic features of the method.

When the surfaces move through each other, new defining field "waves" are created. This is because before the intersection surface 1 was closest to the left grid nodes and surface 2 closest to the right nodes. After the intersection this situation reversed and the u_j and y_j values started to change again at the rate of one cell per time step (see Fig. 2).

A representation of the flow of information can be seen in Fig. 4 where characteristics are sketched before and after the intersection. These characteristics have the slope corresponding to one cell per time step. The paths of the surfaces are also shown.

There are two obvious alternatives to the way the transfers are implemented: First, as mentioned, they do not have to be continued across the grid but need only be performed for a few cells around each surface. Second, multiple transfers can be done between surface time steps (movements of the surfaces, or integrations of the y_j values), instead of simultaneously with the surface motion. Then, the surface motion would not have to be restricted to less than one cell per time step. The method that we describe here has a more physical interpretation of surfaces sending "signals" that travel at a fixed speed while they are moving (at a slower speed).

4. THE 2-D IMPLEMENTATION

4.1. *Propagating 2-D Wave Fronts: Multiple Reflections from Flat Surfaces*

As in the 1-D example described above (and as in the 3-D example described in Section 5), the sequence of computations for propagation of the surfaces involves the following steps at each node (l) and each time step:

(1.1) Poll neighboring nodes (l'). Use defining field variables stored at each l' to compute distance from node l to the surface corresponding to variables at each l' (different nodes can have information corresponding to different surfaces).

(1.2) Find l' whose stored variables give a minimum distance to l.

(1.3) Transfer defining field variables from that l' to node l.

(2) Integrate the equations of motion for the surface whose variables are now stored in node l.

The actual integration depends, of course, on the particular variables used to define the surfaces and will be described below.

All of the multidimensional cases computed in this paper perform the "transfer" operations ((1.1)–(1.3) defined above) two times per time step and the integration (2), once. Also, for the cases computed, the integration time step moved the surface the same fraction (ν) of a grid cell for each case. Results were not sensitive to values of $\nu < 0.9$ and a value of 0.67 was chosen.

As explained in Section 2, the defining fields of a surface in 2-D, in our representation, vary smoothly *along* the surface and are constant along rays normal to it (at least for points close to the surface and closer than to any other surface) (see Fig. 1). This variation, although smooth, does not allow us to use the simple direct transfer method described in Section 2 for constant velocity propagating "surfaces" in 1-D. There, the defining fields were constant throughout the "cloud" of nodes surrounding each surface. There are several other differences, to be described below.

First, unless the defining fields are constant along each surface, a stable propagation of the defining fields requires, and is easily accomplished, using an "upwind" bias in the direction of propagation (increasing distance along each ray) as described in the inset of Fig. 1 rather than a centered scheme. Accordingly, not all the surrounding nodes are surveyed for possible transfer, as in Section 3, but only those "upwind" nodes whose dot products

$$D = \left(\mathbf{x}_l - \mathbf{x}_{l_0}\right) \cdot \left(\mathbf{x}_l - \mathbf{y}_l\right) \tag{1}$$

are positive.

Here, $\mathbf{x}_{l_0}$ denotes the vector position of a node whose defining field values are to be updated by transfer from neighboring nodes (labeled $l = 1, 2, \ldots$ at positions $\mathbf{x}_l$) and $\mathbf{y}_l$ is the vector position of the intersection of a normal ray from the surface which goes through node l at (fixed) position $\mathbf{x}_l$. (Close to a smooth surface, $\mathbf{y}_l$ is the closest point on the surface to $\mathbf{x}_l$.) The coordinates $\mathbf{y}_l$ can either be stored directly in node l or computed from other defining field variables stored there. In the 2-D and 3-D examples presented below the radius of curvature and coordinates of the center of curvature at the closest point on the surface are used as defining field variables. This is particularly efficient for constant velocity of propagation, for reasons described below. These variables are then used, together with the coordinates of the node, to compute the coordinates of the closest point on the surface and distance to the surface. The minimum distance test described in Section 3 but with the above restriction of Eq. (1) is then used to determine the node ($l = m$) which has minimum $|\mathbf{x}_{l_0} - \mathbf{y}_l|$ and a simple transfer of the stored variables is effected, as described in Section 3.

Second, a weighted sum of values should be used to account for the tangential variation of the stored variables, since neighboring grid points will not, in general, lie along the same ray and a simple transfer may not, by itself, be accurate (see Fig. 1). There are many ways to account for this variation.

In Ref. [1] we described a method where derivatives of the defining fields (y_l, u_l, etc.) are computed. The variation in the tangential direction (along the surface) between $\mathbf{x}_l$ and $\mathbf{x}_{l_0}$ could then be accounted for by using these derivatives as a first term in a Taylor expansion.

This, or a similar interpolation scheme would be required if, for example, the coordinates, $\mathbf{y}_l$, of the intersection of each ray belonging to each node were used *directly* as part of the defining field, rather than the variables described below.

In the multidimensional cases treated in this paper, the following variables are used as defining fields propagated between nodes, and used to compute the $\mathbf{y}_l$ values. These do not require the computation of derivatives or accurate interpolation, at least for the problems that we are treating. These variables are the coordinates, $\mathbf{y}_l^c$, of the local center of curvature of the surface at the intersection point, $\mathbf{y}_l$, and the local radius of curvature at that point, R_l. These are defined to be constant along each ray but, of course, also vary along the surface (from ray-to-ray) and hence, vary between nodes. If these variables are used, then only a transfer, as described above, followed by a centered smoothing is necessary to accurately compute the surface propagation. The reason that we do not need to interpolate is that, even though these new center of curvature variables change as we move along the surface, each set of values describes a larger region of the surface, to a given accuracy, than the $\mathbf{y}_l$. Hence, values from neighboring nodes, even though different, can also be used to describe the same segment of the surface.

Our choice of defining field variables appears to be effective for the problems that we are treating: Constant speed propagation with no refraction and reflection only from flat or convex surfaces. Our point, however, is to emphasize the *general* DSE idea and not a particular implementation. In a subsequent paper [13], Steven Ruuth *et al.* have used the above $\mathbf{y}_l$ variables with high order interpolation and have added the possibility of treating reflection from concave surfaces and "swallow-tail" phenomena.

For the smoothing operation, a test of the normalized vectors of surrounding points,

$$\min(B_{ll'}) > b, \qquad B_{ll'} = \left(\frac{\mathbf{x}_l - \mathbf{y}_l}{|\mathbf{x}_l - \mathbf{y}_l|}\right) \cdot \left(\frac{\mathbf{x}_{l'} - \mathbf{y}_{l'}}{|\mathbf{x}_{l'} - \mathbf{y}_{l'}|}\right)$$

must be satisfied or the smoothing is not done. This avoids smoothing across discontinuities, where nodes would have defining field variables corresponding to different surfaces. For the cases treated here, b was 0.9. If different surfaces are almost parallel, a different test may have to be used. The formation of a cusp in the surface after reflection from a curved surface results in such a situation and, as described below, further study is required there.

Using the new variables, the distance from a point ($\mathbf{x}_l$) at node l along the ray to the surface is then

$$d_l = \left|\mathbf{x}_l - \mathbf{y}_l^c\right| - R_l$$

and the intersection point of the ray with the surface is then

$$\mathbf{y}_l = \mathbf{y}_l^c + \left(\frac{\mathbf{x}_l - \mathbf{y}_l^c}{\left|\mathbf{x}_l - \mathbf{y}_l^c\right|}\right) \cdot R_l,$$

where $\mathbf{y}_l^c$ and R_l are the values of $\mathbf{y}^c$ and R stored at node l. If the medium is homogeneous (as considered here), as the surface propagates $\mathbf{y}_l^c$ remains constant for a given surface and R_l increases linearly with time.

In addition to propagation, reflection must be treated. This turns out to be simple in the geometric optics limit, at least if the radius of curvature of the reflecting surface is not too small and cusps or other singularities do not form. (These problems will be treated in

subsequent studies—our main purpose here is to describe the basic ideas of the method.) A set of image points is used at the reflecting boundary with reflected values of $\mathbf{y}_l^c$ and a value of R_l, derived from the current values at the point just inside the reflecting boundary.

Results of this procedure are presented in Fig. 5 for an initially elliptic surface propagating outward and reflecting from straight walls. The initial elliptic surface is described by

$$0.5^2\left(x^{(1)}\right)^2 + \left(x^{(2)}\right)^2 - 45.^2 = 0.$$

The local radius of curvature is then

$$R = \left[0.5^4\left(x^{(1)}\right)^2 + \left(x^{(2)}\right)^2\right]^{3/2}/22.5^2$$

and the coordinates of the center of curvature $y^{c(1)}$, $y^{c(2)}$, are given by

$$y^{c(1)} = x^{(1)} - x^{(1)}\left[0.5^4\left(x^{(1)}\right)^2 + \left(x^{(2)}\right)^2\right)\Big]^2/45.^2$$
$$y^{c(2)} = x^{(2)} - x^{(2)}\left[0.5^4\left(x^{(1)}\right)^2 + \left(x^{(2)}\right)^2\right)\Big]^2/22.5^2.$$

Initially, these values of R and $\mathbf{y}^c$ were set at those grid points (l) within one grid cell of the elliptic surface. Away from the surface, R was set to a large value, 10^7.

At each time step, as described above, correct values of R replaced the initial large values and the correct values of $\mathbf{y}^c$ transferred. Then, the values of R were incremented to account for the normal motion of the curve. Conditions were set at the four boundaries of each time step such that boundary grid nodes had the same value of R and the tangential component of $\mathbf{y}^c$, and a reflected value of the normal component, as the neighboring interior nodes.

In the first example, a 256×256 cell grid was used. Distance contours are plotted in Fig. 5, where the distance, $|\mathbf{x}_l - \mathbf{y}_l|$, is less than $1/2$ grid cell width (h), for a sequence of results in increments of 200 time steps. To assess the accuracy of the method, some markers are also shown (represented by small circles) computed using a separate ray tracing method which is exact. These markers, of course, are not used in the method but are only plotted in the figure to illustrate the accuracy of the method.

It can be seen that although there are some small temporary gaps in the surfaces after an intersection, they soon fill in, as anticipated in Section 3. Also, by comparison with the markers, it can be seen that the surface position is correct to plottable accuracy.

To further describe the method, we show how the defining field values move with the propagating surfaces: The normalized vectors

$$\frac{\mathbf{x}_l - \mathbf{y}_l}{|\mathbf{x}_l - \mathbf{y}_l|}$$

are plotted in Fig. 6 together with the wave fronts for two different stages of the computation. Although, of course, this information is only needed within a few cells of each surface, we continue it across the entire grid to illustrate the flow of information. This flow is the important feature that allows surfaces to pass through each other.

4.2. *Grid Dependence Study*

A study was made using exactly the same procedure for the above problem but for a sequence of computational grids: 256×256, 128×128, 64×64, and 32×32. For each

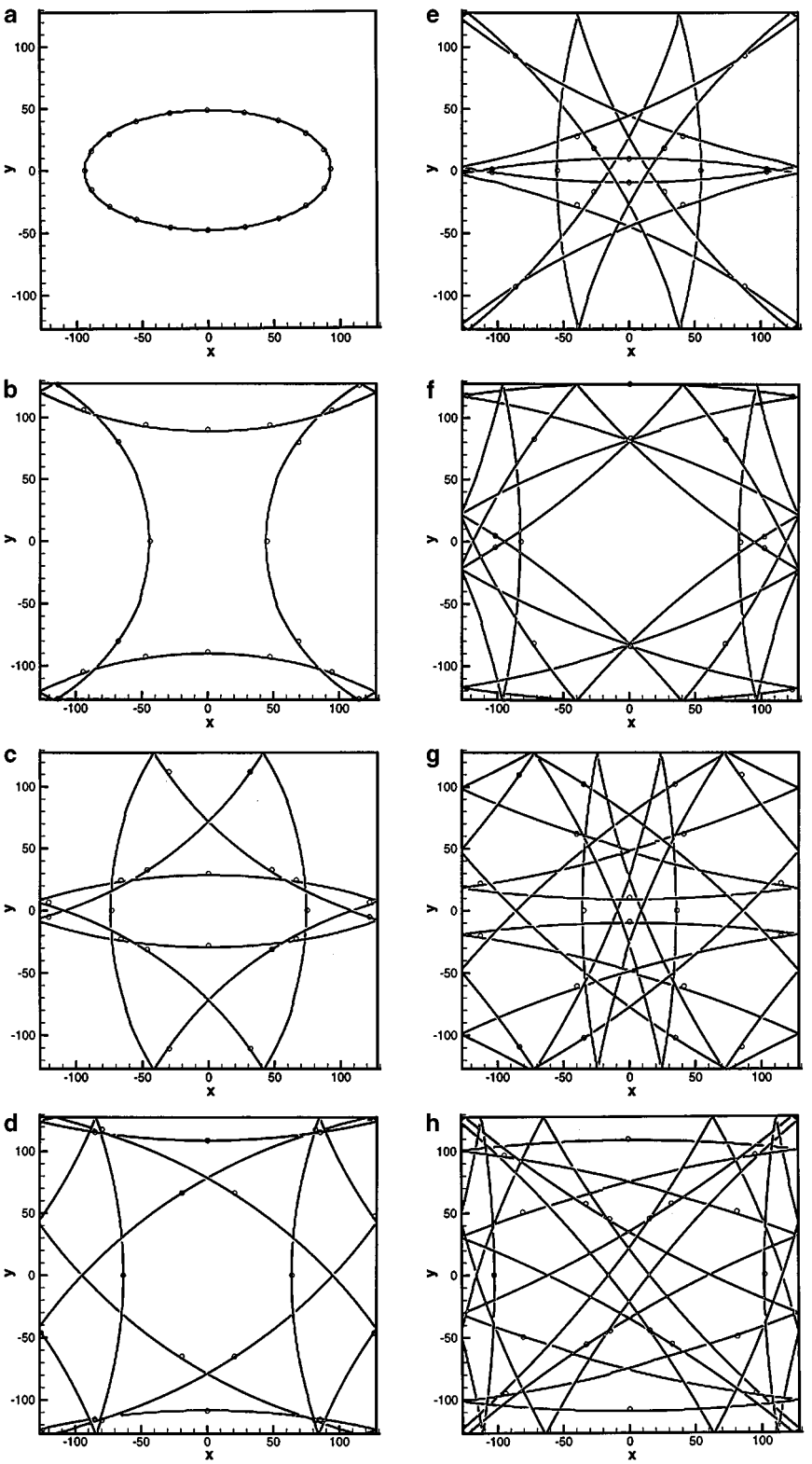

FIG. 5. Normally propagating surface with reflection from walls in 2-D (—, DSE solution; ○, Lagrangian solution).

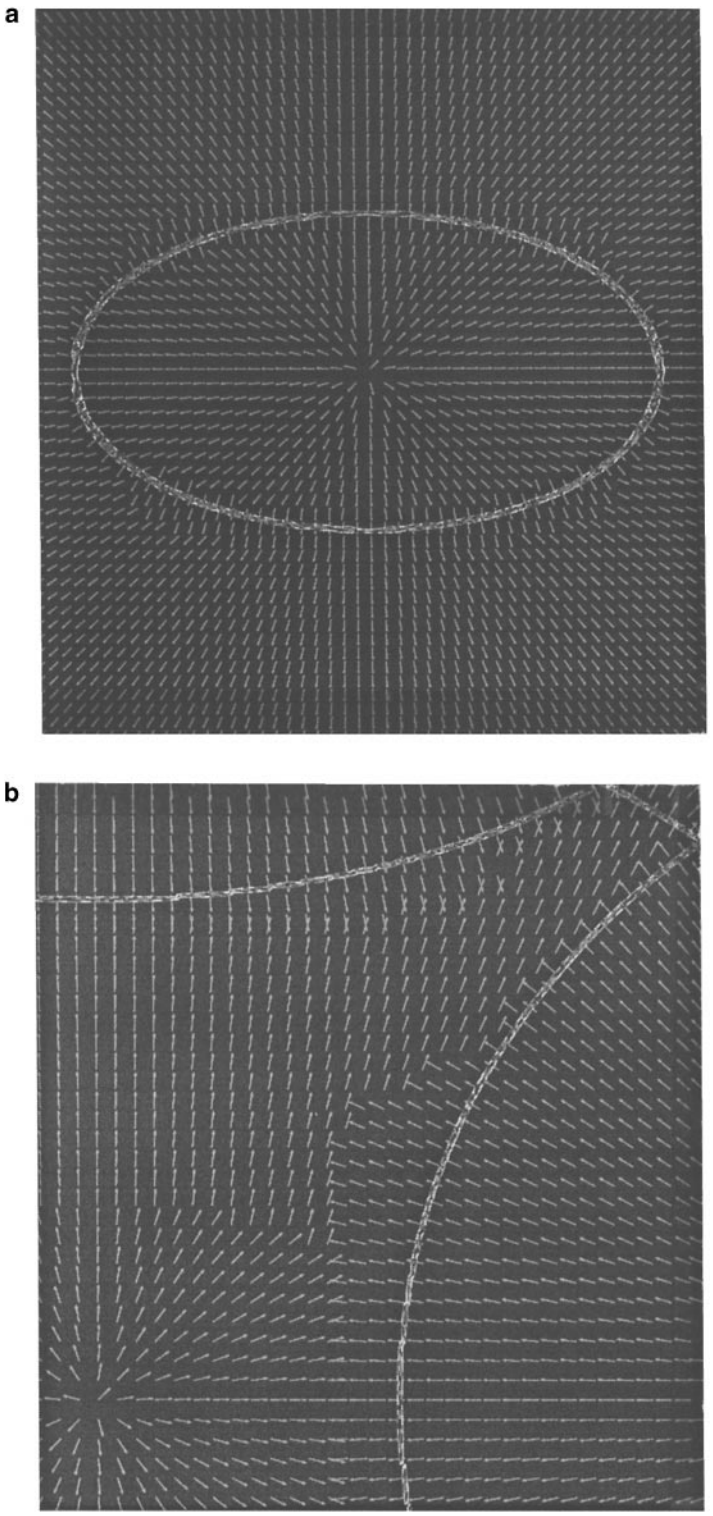

FIG. 6. Normal vectors in 2-D reflecting surface problem showing propagation of defining fields.

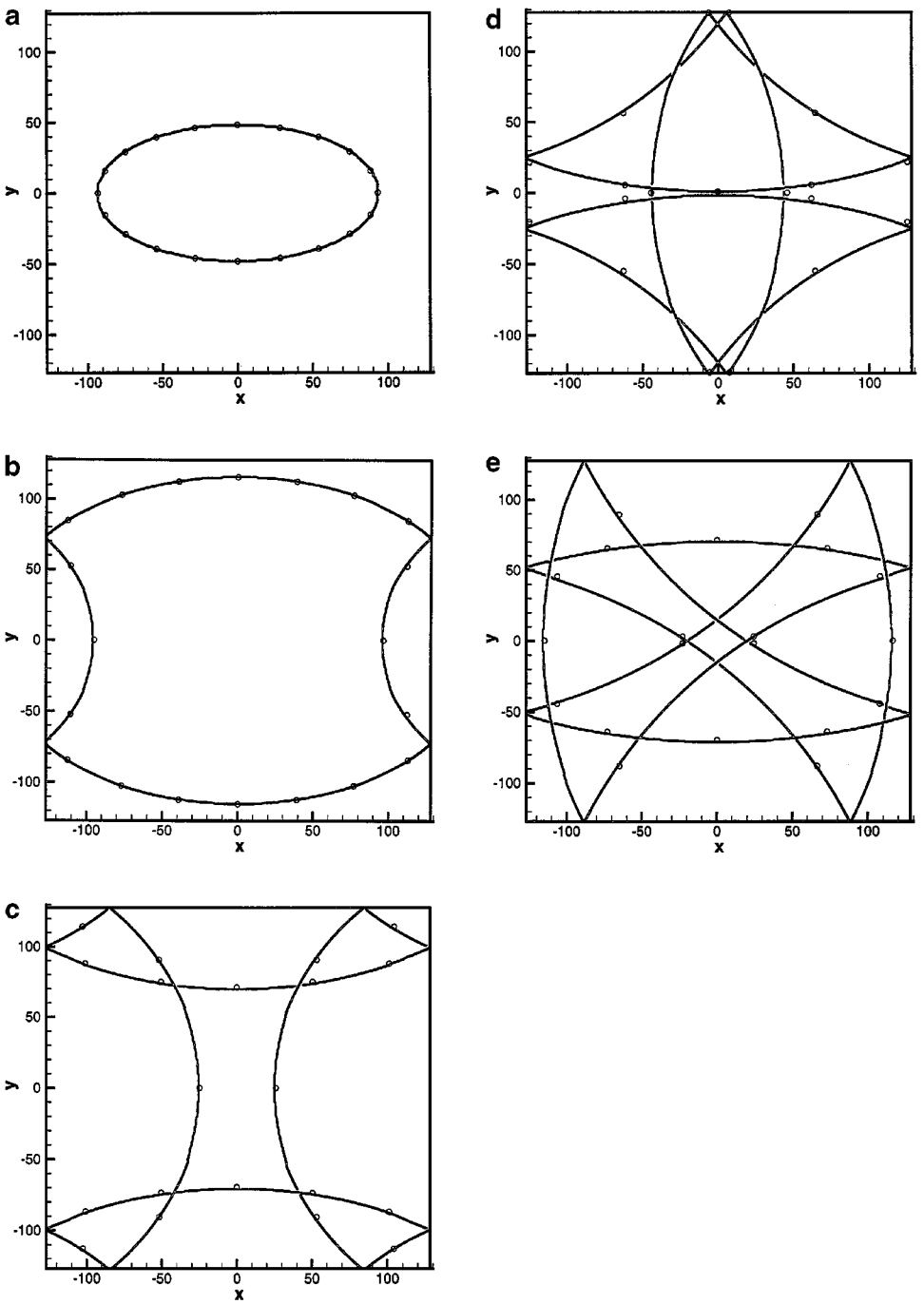

FIG. 7. DSE solution, 256×256, 128×128, 64×64, and 32×32 grids.

grid, results are displayed at intervals of 120, 60, 30, and 15 time steps, respectively (corresponding to the same actual time interval). Results are plotted in Fig. 7 for these cases, where distance contours are plotted for values less than $1/2$ grid cell.

It can be seen, qualitatively, how the solution degrades with decreasing resolution. It is difficult to quantify the order of accuracy, since there are a number of dimensionless ratios

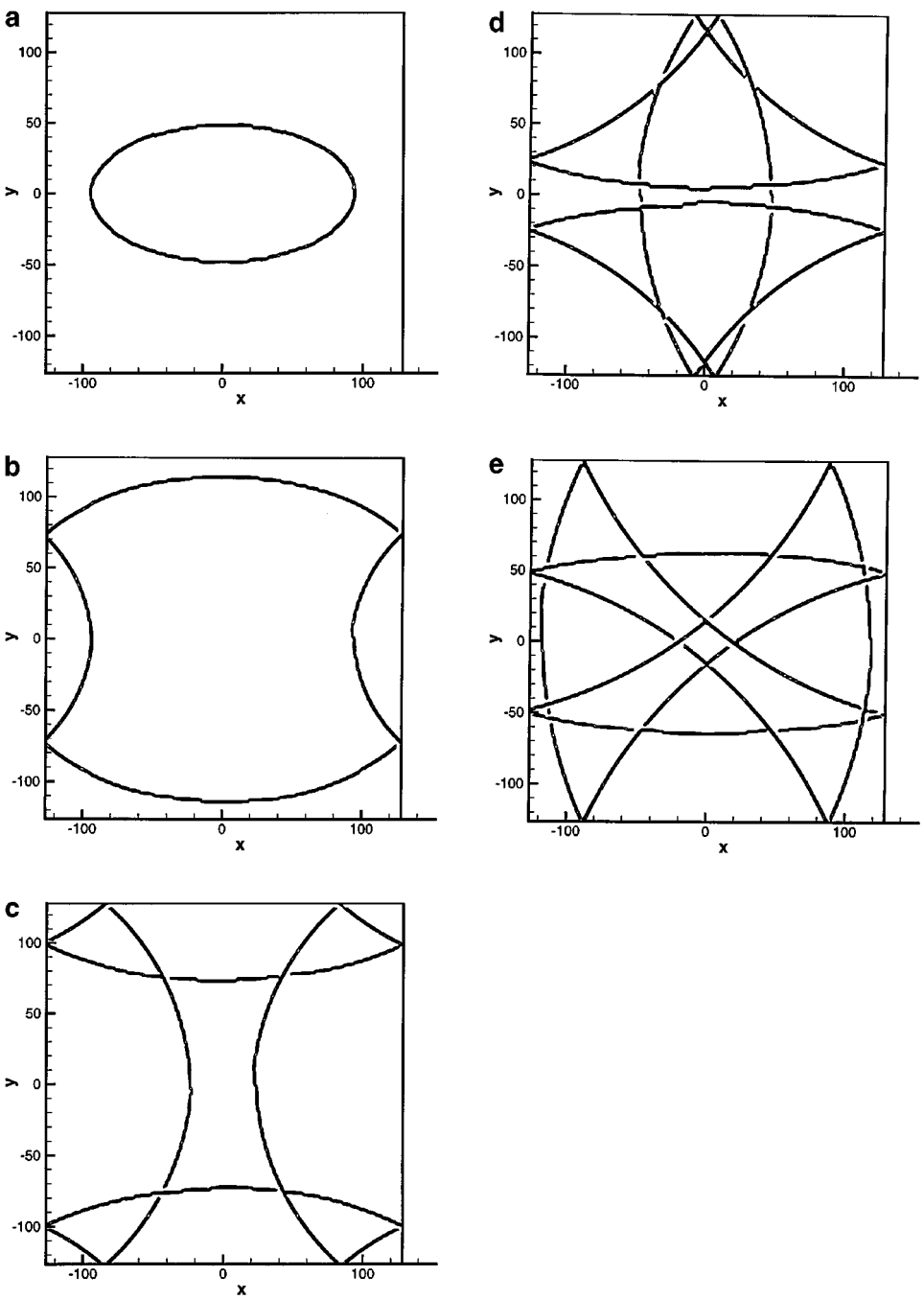

FIG. 7—*Continued*

that can be formed depending on, for example, cell size, radius of curvature, rate of variation of this curvature, distance between surfaces, etc. Also, the issues are different than if we were trying to discretize a (continuum) partial differential equation, such as an Eikonal equation. The solutions can be seen to form gaps when there are a number of other nearby surfaces. However, except for this gap formation the solutions can be seen to be essentially

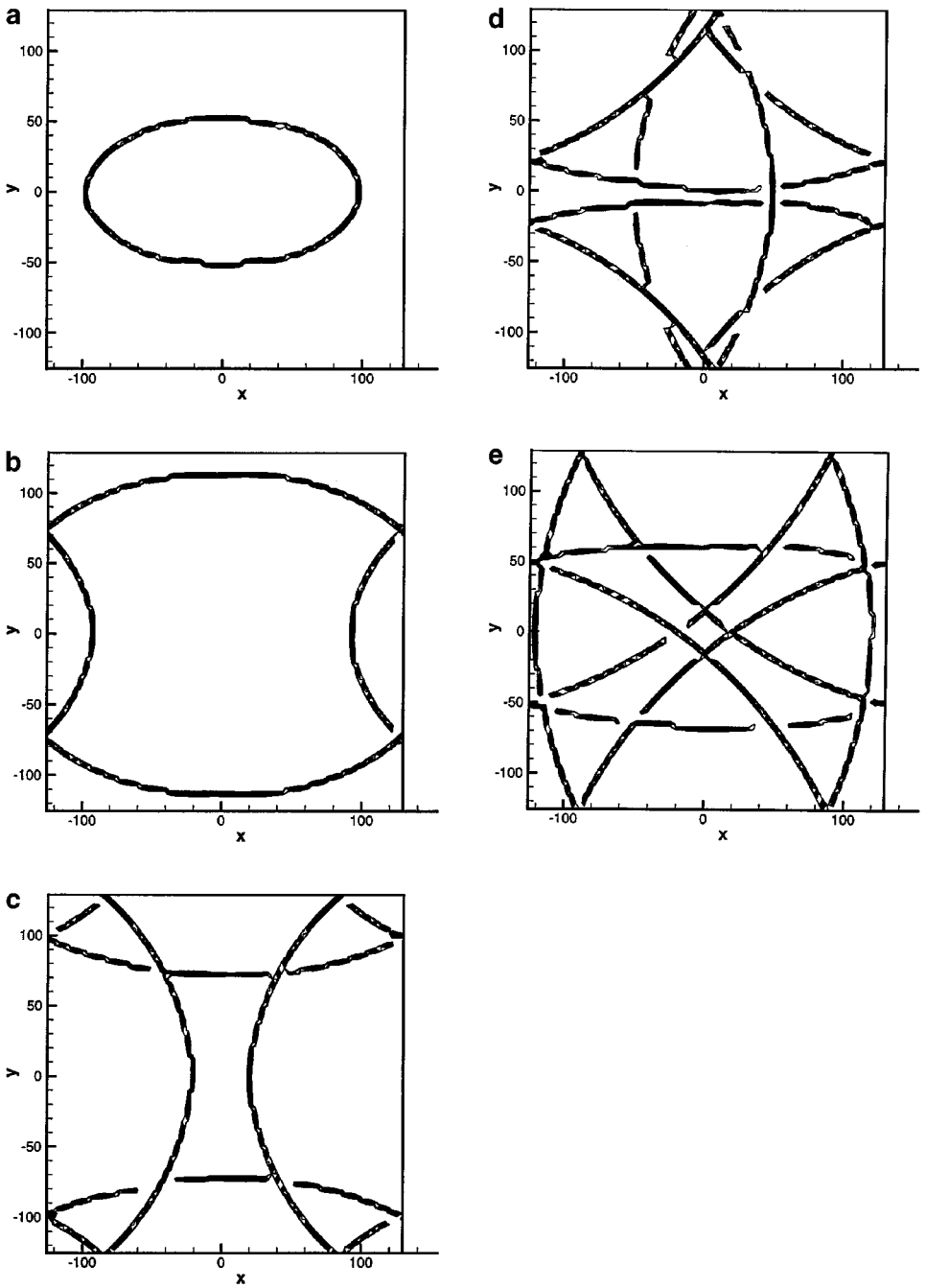

FIG. 7—*Continued*

the same to within a grid cell thickness (h). For a single, non-intersecting surface, these gaps would not form.

An error measure for a single propagating surface can easily be defined. Each grid node, at each time step, contains a local definition of the surface parameterized by radius of curvature, R^c, and center of curvature, $\mathbf{y}^c$. If we compute a set of marker trajectories, x_k^m, y_k^m, which

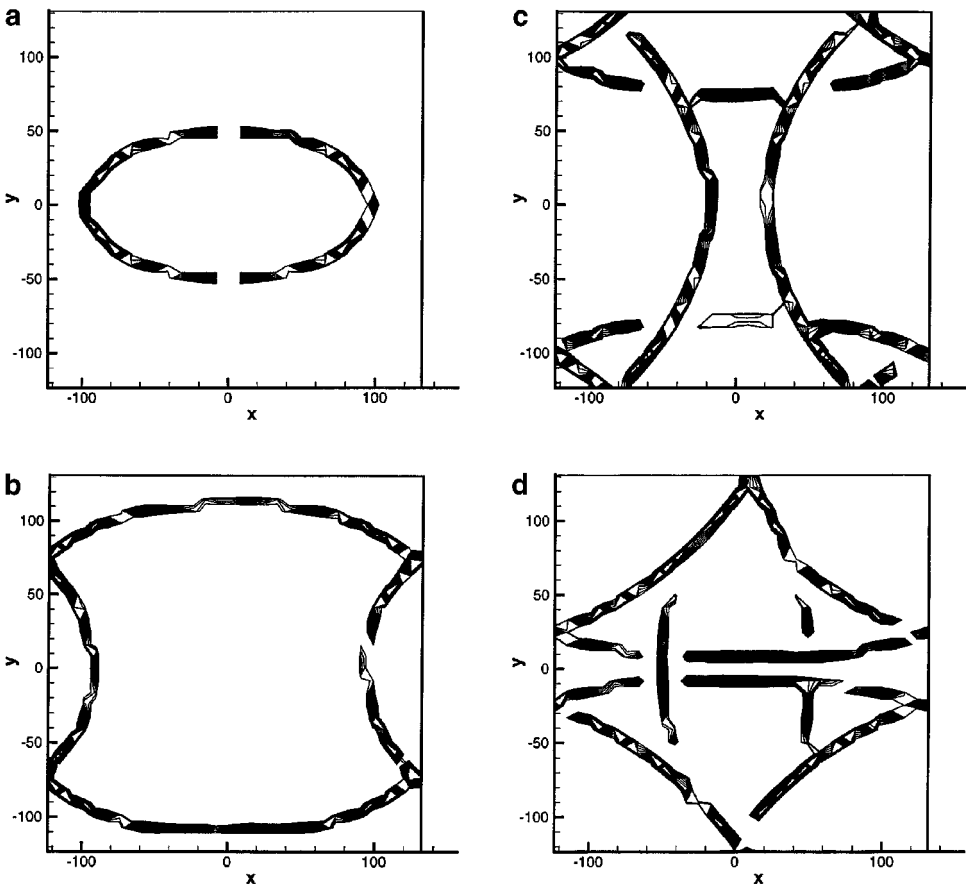

FIG. 7—*Continued*

exactly follow the surface, then, at any time step we can measure the distance of these markers to the locally defined surface on the grid,

$$e_k = \left[\left(x_k^m - x_k^c\right)^2 + \left(y_k^m - y_k^c\right)^2\right]^{\frac{1}{2}} - R_k^c,$$

where x_k^c, y_k^c, and R_k^c are the node values of $\mathbf{y}^c$ and R^c from the 4 nodes surrounding the marker, interpolated onto the marker location, x_k^m, y_k^m. Then an L_1 norm can be computed for N markers,

$$e^1 = \frac{1}{N}\sum_1^N |e_k|.$$

In Table I, we display values of e^1/h for each grid, initially and after the surface has traveled a distance $D = 32$, where D is given in the same units as the initial ellipse semi-major and semi-minor axes (90, 45). Due to the symmetry we only treat one quadrant and use 1000 markers, and we use a standard bilinear interpolation scheme.

It can be seen that, over the range of values computed, the error does not grow with distance traveled and is essentially first order in grid cell size.

TABLE I
Error Estimate for Computed Propagating Surface

Grid	32×32	64×64	128×128	256×256
$e^1/h;\ D = 0.$	0.148	0.171	0.164	0.140
$e^1/h;\ D = 32.$	0.145	0.123	0.119	0.150

4.3. *Propagating 2-D Wave Fronts: Reflection from Cylindrical Surface*

An initially cylindrical outgoing wave was made to impinge on a reflecting cylinder, both circular. Essentially the same reflection law (and grid) was used as above in Subsection 4.2 except for the local radius of curvature, R', of the reflected surface. This was determined by

$$R' = \left(R^{-1} + R_s^{-1}\right)^{-1},$$

where R is the local radius of curvature of the incident surface and R_s is the local radius of curvature of the reflecting surface. For the case treated in Subsection 4.2, involving flat reflecting surfaces, this formula reduces to $R' = R$ as used there.

An additional change involved "grazing" rays that impinge on the cylinder at a small angle, where the incident surface is approximately normal to the reflecting surface. The reflection law was not accurate there and had to be modified. The result is displayed in Fig. 8, for a sequence of results in increments of 60 time steps, using the same distance contours as in Fig. 5. Some markers computed using ray tracing (which is exact) are also displayed as small circles, so that the accuracy of our method can be assessed. Again, these markers are not, of course, used in the computation.

Most of the reflection is as expected: the only significant numerical error appears near each cusp. Each cusp should be at the end of the line extending from it but numerical error causes it to collapse to a line there. We are currently developing methods to detect a cusp and prevent this growth in that region. A further topic involves using the extension into the "shadow" region to approximate short wavelength diffraction corrections. As in Subsection 4.2, it can be seen that, except for the cusp region, the method yields results which are exact to plottable accuracy.

5. WAVE FRONTS IN 3-D WITH REFLECTION FROM FLAT SURFACES

Essentially the same method used in Subsections 4.2 and 4.3 was used in 3-D, but on a $128 \times 128 \times 128$ grid. This proved to be a straightforward extension of the 2-D method. The results are displayed in Fig. 9 for a sequence of time steps, in increments that best displayed the surface features (50, 20, 30, 30, 30, 30, 20), where surfaces are displayed corresponding to small values ($< h/2$) of distance (the figure for the last time step is cut-off for clarity).

The results are as expected from the 2-D study of Subsection 4.2. Also, a 2-D section of the results (in the symmetry plane) was examined and found to be within plottable accuracy of the corresponding 2-D results (except for the additional reflected waves present in 3-D). These, in turn, agreed with the exact ray tracing results.

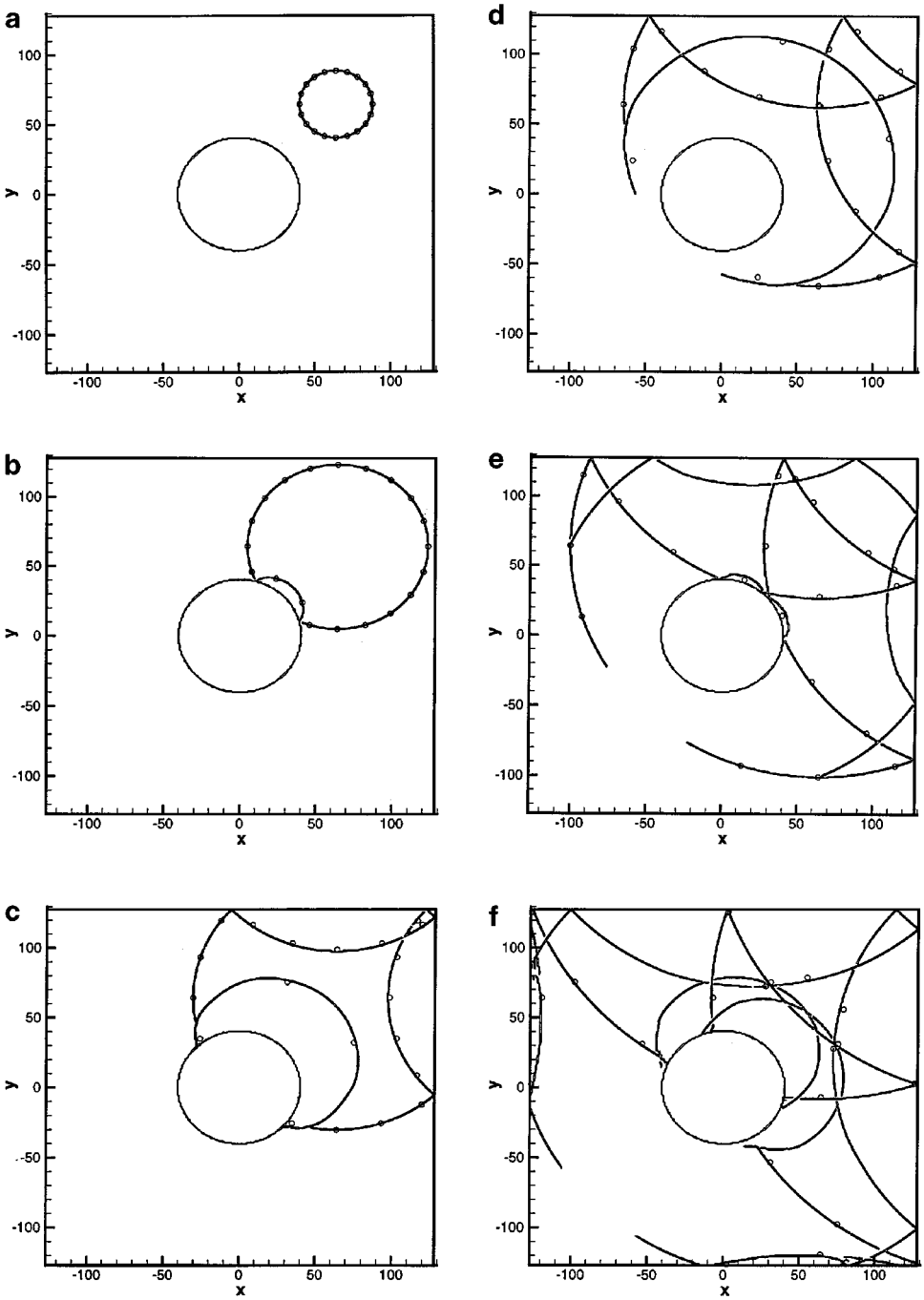

FIG. 8. Normally propagating surface with reflection from cylinder in 2-D (—, DSE solution; ○, Lagrangian solution).

6. EXTENSION AND PROBLEMS

The Dynamic Surface Extension representation may be useful for other surfaces such as shocks and contact discontinuities and for representing filaments and particles. A description of its use in a molecular dynamics simulation is given in Ref. [14].

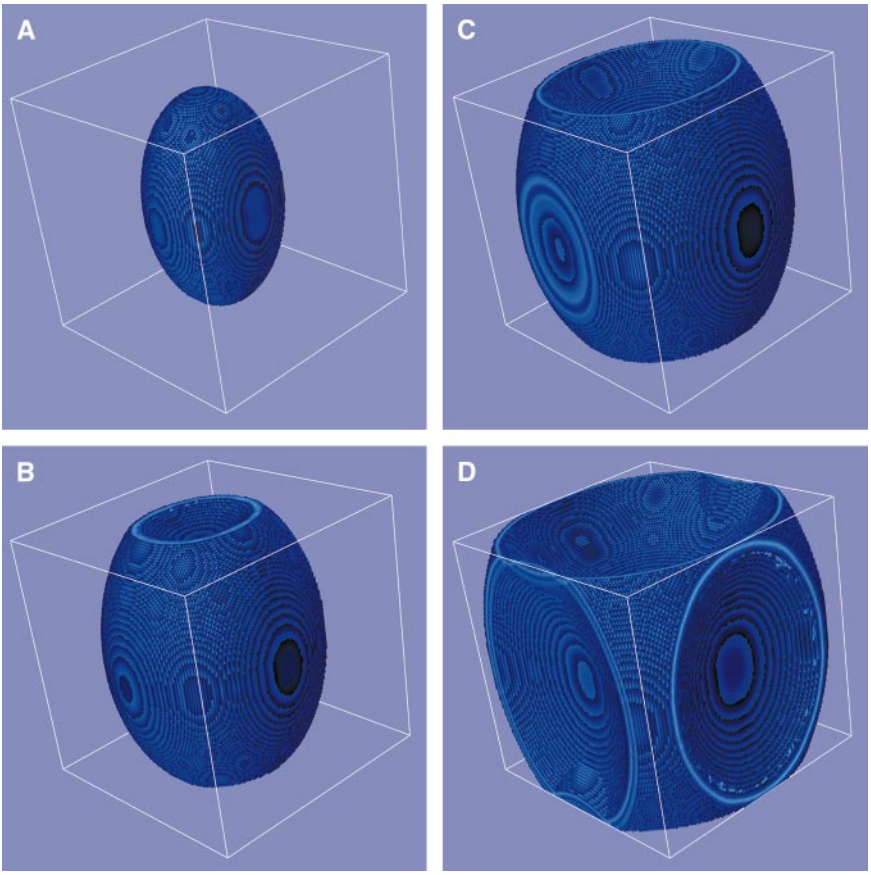

FIG. 9. Normally propagating surface with reflection from walls in 3-D, DSE solution.

Interpolation formulas for surface propagation and accuracy issues should be carefully addressed and the behaviour of scattered waves near the "shadow" zone should be investigated. Besides resolving the spurious collapse near cusps, ways to simulate short wavelength diffraction should be investigated.

Finally, as in Ref. [3], the propagating pulse can be used to compute the Eikonal phase function and amplitude, including reflections. This can be used to solve the Helmholtz equation in the high frequency limit and to obtain (multivalued) Green's functions.

7. CONCLUSION

The new Dynamic Surface Extension representation described in this paper may prove useful, especially in electromagnetic and acoustic propagation and scattering problems in the geometrical optics limit for the time dependent wave equation and the Helmholtz equation.

For special problems involving free space with no inhomogeneities or reflecting surfaces there exist closed form Green's functions that can be used to treat these problems. For the general case, however, the only alternative until now has been ray tracing. The method introduced in this paper may represent an alternative that is completely Eulerian and does not have the drawbacks of Lagrangian methods.

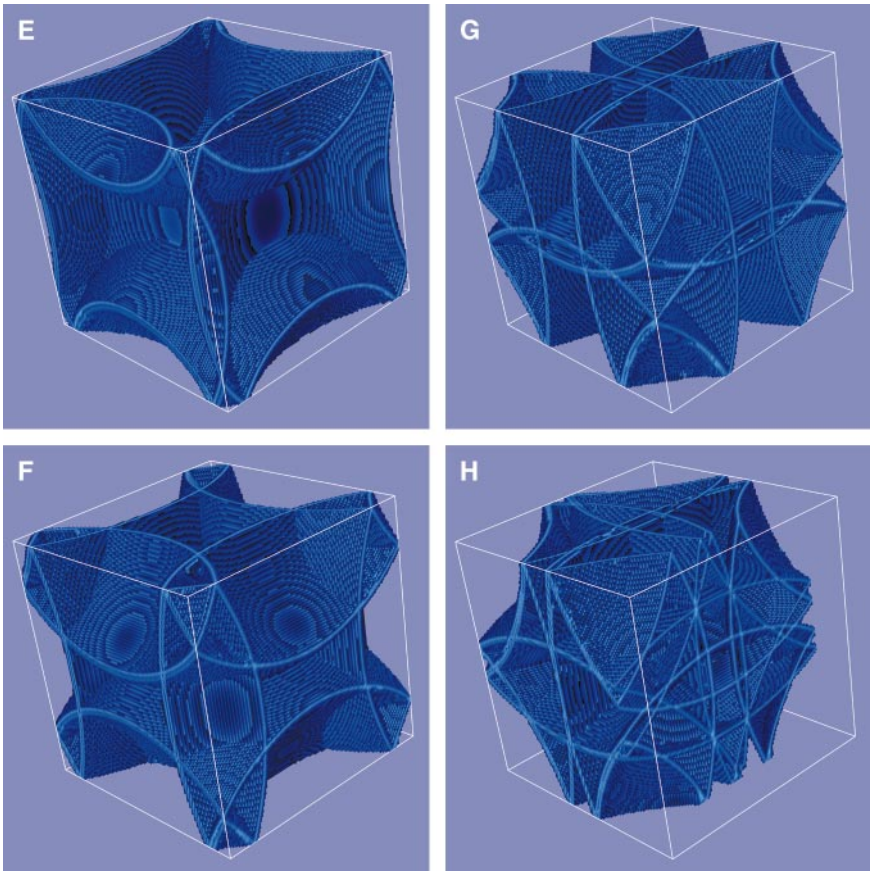

FIG. 9—*Continued*

Further, the Dynamic Surface Extension method treats segments of a pulse, or in a different application, moving particles, without labeling them, as opposed to Lagrangian methods which assign particular markers to them. This "indistinguishable" feature is more in the spirit of a local field method and may have other advantages, where features can be created or destroyed.

Finally, the continuum limit of the Dynamic Surface Extension representation may be interesting. The simplest way to implement the method, as described in the paper, results in a (node to node) propagation speed of the defining fields that is not isotropic, due to the effects of the lattice on the transfers. A propagation law that is isotropic can also easily be implemented. The continuum (small h) limit then would correspond to a physical model where particles, curves, or surfaces act as sources of the defining fields which propagate away according to a local partial differential equation at a constant speed, and where the speed of the particles, curves, or surfaces is limited to a value less than this speed. These possibilities should result in a partial differential equation representation of the method. This will be a subject of future research.

ACKNOWLEDGMENTS

Development of the basic methods leading to this study was funded mainly by the Army Research Office. Subsequent implementation of the method was funded by the U.S. Air Force. The first author acknowledges the

hospitality of Stanley Osher and many interesting discussions with him, Barry Merriman, Steve Ruuth, and others at UCLA. The authors acknowledge the helpful comments of the anonymous reviewers.

REFERENCES

1. J. Steinhoff and M. Fan, Eulerian computation of evolving surfaces, curves and discontinuous fields, UTSI preprint, January 1998.
2. C. K. W. Tam, Computational aeroacoustics: Issues and methods, *AIAA J.* **33**, 10 (1995).
3. J. Steinhoff, Y. Wenren, D. Underhill, and E. Puskas, Computation of short acoustic pulses, in *Proceedings 6th International Symposium on CFD, September 1995*.
4. J. Steinhoff, E. Puskas, S. Babu, Y. Wenren, and D. Underhill, *Computation of Thin Features over Long Distances Using Solitary Waves*, AIAA Paper 97-1976.
5. R. C. Strawn, R. Biswas, and M. Garceau, Unstructured adaptive mesh computations of rotorcraft high-speed impulsive noise, *J. Aircraft* **32**, 4 (1995).
6. J. Benamou, Big ray tracing: Multivalued travel time field computation using viscosity solutions of the Eikonal equation, *J. Comput. Phys.* **128**, 2 (1996).
7. S. Osher and J. A. Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations, *J. Comput. Phys.* **79**, 12 (1988).
8. E. Fatemi, B. Engquist, and S. Osher, Numerical solution of the high frequency asymptotic expansion for the scalar wave equation, *J. Comput. Phys.* **120** (1995).
9. D. Adalsteinsson and J. A. Sethian, A fast level set method for propagating interfaces, *J. Comput. Phys.* **118** (1995).
10. J. A. Sethian, A fast marching level set method for monotonically advancing fronts, *Proc. Nat. Acad. Sci.* **9** (1996).
11. M. G. Brown, Numerical considerations in ray tracing and ray expansions of the acoustic wavefield, *J. Acoust. Soc. Am.* (1984).
12. J. E. Ffowcs Williams and D. L. Hawkings, Sound generated by turbulence and surfaces in arbitrary motion, *Philos. Trans. R. Soc. London A* **264** (1969).
13. S. J. Ruuth, B. Merriman, and S. Osher, A fixed grid method for capturing the motion of interfaces arising in geometrical optics and related PDEs, UCLA preprint, April 1999.
14. J. Steinhoff, M. Fan, and L. Wang, A new Eulerian method for the computation of propagating particles and surfaces, UTSI preprint, August 1998.