Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 226 (2007) 2250–2270
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Efficient level set methods for constructing wavefronts in three spatial dimensions

Li-Tien Cheng

Department of Mathematics, University of California San Diego, La Jolla, CA 92093-0112, United States

Received 22 March 2006; received in revised form 8 March 2007; accepted 10 July 2007
Available online 3 August 2007

Abstract

Wavefront construction in geometrical optics has long faced the twin difficulties of dealing with multi-valued forms and resolution of wavefront surfaces. A recent change in viewpoint, however, has demonstrated that working in phase space on bicharacteristic strips using eulerian methods can bypass both difficulties. The level set method for interface dynamics makes a suitable choice for the eulerian method. Unfortunately, in three-dimensional space, the setting of interest for most practical applications, the advantages of this method are largely offset by a new problem: the high dimension of phase space. In this work, we present new types of level set algorithms that remove this obstacle and demonstrate their abilities to accurately construct wavefronts under high resolution. These results propel the level set method forward significantly as a competitive approach in geometrical optics under realistic conditions.

1. Introduction

In the high frequency regime, wave propagation can be simplified by recasting the wave equation into an eikonal equation for phase and transport equations for amplitude terms in what is known as the geometrical optics approximation. Solution of the phase has a geometric interpretation, involving solution of its level sets, called wavefronts, each of which can be viewed as manifestations of a surface flowing from an initial source through different traveltimes under a time-dependant eikonal equation. The right combinations of viewpoint, framework, and techniques to aid in capturing the generally multi-valued wavefront surfaces under their deformations has been the subject of immense study in applied mathematics.

Two approaches, lagrangian and eulerian, have traditionally dominated the subject of numerical wavefront construction. The algorithms arising from these two frameworks have diametrically opposite characteristics: a lagrangian method easily captures multi-valued behavior but encounters difficulties with the resolution of the surface while an eulerian method automatically resolves the surface but has trouble obtaining multi-valued

E-mail address: lcheng@math.ucsd.edu

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2007.07.019

forms. This trade-off has been the main obstacle to producing suitable numerical solutions. See, e.g. [17,18] for more on lagrangian methods and [2,4] for more on eulerian methods.

Recently, the work of [5] has advanced a different viewpoint that is able to bypass this obstacle. Since higher-dimensional phase space unravels multi-valued wavefronts, it noted that an eulerian method working in this setting can avoid the difficulties of multi-valued forms while controlling the resolution. Numerical experiments using the segment projection method as the eulerian method provided early justification of these claims.

In light of the development, advances, and success of level set approaches in capturing a variety of interfaces under deformations (see, e.g. [11]), the work of [10] continued with the same philosophy but substituted a level set method [12] in place of the segment projection method. Given an initial wavefront, this method represented its form in phase space, called the bicharacteristic strip, as the zeros of a vector-valued level set function. Wavefronts of other traveltimes were generated by evolving this function under the Liouville equation in phase space, which derives from the eikonal equation for wavefronts in spatial space, up to the desired travel time, then extracting the zeros and projecting them down into spatial space. Numerical experiments showed that multi-valued solutions could be captured and properly resolved in a variety of situations involving variable indices of refraction and reflection.

However, all but one of these experiments were conducting in two spatial dimensions. The reason for this is phase space has twice the dimension of spatial space. For two spatial dimensions, phase space is only four-dimensional, where one of the dimensions can be removed without complications. For three spatial dimensions, the natural and more physical environment for wave propagation, phase space is six-dimensional. In this setting, grid-based algorithms must place an emphasis on efficiency or face intolerably long computation times and unacceptably large memory requirements. Thus the algorithms presented in [10] showed level set methods could bypass traditional difficulties in geometrical optics, but were not efficient enough to directly extend for use in three spatial dimensions. The only result given for three spatial dimensions could only use 20 points in each dimension for its grid over a five-dimensional reduced phase space, hardly enough to resolve details in the wavefronts. Subsequent advances in [9] only show extensions to grids with 64 points in each dimension.

In this paper, we modify the level set approach of [10] so that computations are performed only at gridcells near the bicharacteristic strip, the ultimate goal of local level set methods [13]. This effectively reduces the complexity of the approach from the dimension of phase space to approximately that of the bicharacteristic strip itself. The essential technology used is a long-time semi-lagrangian approach (see, e.g. [6,14]) for solving the Liouville equation that provides the flexibility needed to avoid computations on the majority of gridcells (see [7,9] for other semi-lagrangian level set methods). Based on this technology and the level set framework, we consider a theoretically fast multi-resolution algorithm, introduce a practically fast growing algorithm, and further produce advantageous combinations of the two. Section 2 introduces essential concepts and tools used in the algorithms presented in Section 3. Section 4 then investigates the abilities of the algorithms through numerical experiments and compares results with the competitive approach introduced in [9]. Following this, Section 5 summarizes the contributions and content of this paper while 'acknowledgement section' acknowledges supporting factors in the research. Finally, Appendix A separates the main body of the paper from speculation that is nonetheless important for completeness of exposition.

2. Preliminary tools

2.1. Solving the Liouville equation

In this section, we summarize a flexible scheme for capturing the level set function as it flows under the Liouville equation. Let $\Phi(x,p,t)$ denote a four component vector-valued level set function existing in phase space, $\{(x,p) \mid x \in \mathbf{R}^3,\ p \in \mathbf{R}^3\}$. Suppose $\Phi(x,p,t=0)$ is given so that its zeros match the two-dimensional bicharacteristic strip for a given wavefront of zero traveltime. The work of [10] gives details for finding such a level set function when the wavefront is single-valued and stable under representation by a level set function $\tilde{\phi}(x)$ in spatial space, with the strategy consisting of setting the four components of Φ to be

$$(\phi_1, \phi_2, \phi_3) = c\nabla_x\tilde{\phi}/|\nabla_x\tilde{\phi}|,$$
$$\phi_4 = \tilde{\phi},$$

where $c(x)$ denotes the local wave speed in the medium. Here, the first three components are dedicated to phase direction, which is normal to the wavefront, and the last component to spatial location. The equality between the zero level set of the level set function and the bicharacteristic strip is preserved for future traveltimes through the Liouville equation

$$(\phi_i)_t + c\frac{p}{|p|}\cdot\nabla_x\phi_i - |p|\nabla_x c\cdot\nabla_p\phi_i = 0$$

on the components ϕ_i of Φ. Thus to simplify exposition, we use the term bicharacteristic strip when referring to the zero level set of Φ. An algorithm for capturing wavefronts then seeks to solve the Liouville equation for the level set function up to the desired traveltime, extract its zeros, and project them down into spatial space, $\{x|x\in\mathbf{R}^3\}$, to arrive at the wavefront at that traveltime. The bulk of the complexity in this procedure lies in solving the Liouville equation.

According to the method of characteristics, the Liouville equation is a transport equation moving values of Φ unchangingly along integral curves of the velocity field $v = (cp/|p|, |p|\nabla_x c)$ in phase space. The linear nature of the partial differential equation allows for application of a lagrangian strategy for solving for Φ at desired locations in phase space at desired times. Fix time T and position (x_T, p_T) in phase space. To find the point (x_0, p_0) at time 0 that flows to (x_T, p_T) along v after time T, we may solve the ordinary differential equation

$$(x(t), p(t))' = v$$

backwards from time T to time 0, with $x(T) = x_T$, $p(T) = p_T$ given and $x(0) = x_0$, $p(0) = p_0$ the solution. The value of Φ at (x_T, p_T) at time T can then be obtained from the value of the initial Φ at (x_0, p_0) at time 0 under the relationship

$$\Phi(x_T, p_T, T) = \Phi(x_0, p_0, 0).$$

This strategy becomes semi-lagrangian when the points (x_T, p_T) at time T fit into a grid in phase space. See [7] for more on this procedure in geometrical optics.

There are two advantages to solving the Liouville equation in this manner over the traditional choice of a finite differencing approach, as made in [10]. The first arises from the fact that values at different points and times are independently calculated. This means the calculation of a value at one point and time does not have the large domain of dependence of values needed by the stencils of finite differencing. Related to this, the collection of points where values can be calculated using the semi-lagrangian approach need not have a regular structure and can in fact be composed of a random assortment of points. On such grids, finite differencing is not as well-developed and loses many of its desirable abilities. The second advantage is the removal of the Courant–Friedrichs–Lewy (CFL) stability restriction appearing in finite differencing approaches.

These advantages provide the flexibility needed for us to build efficient level set algorithms. The first advantage allows us to calculate values only at gridcells close to the bicharacteristic strip and also to use specialized grids to reduce the dimension of phase space. The second advantage allows us to take larger steps in time, unhindered by stability constraints, for quicker computation.

2.2. Gridcells and bicharacteristic strips

In this section, we seek a method for checking whether a given gridcell intersects a bicharacteristic strip by using values of the level set function generated, for example, from the semi-lagrangian approach described in the previous section. Our purpose for such a checking scheme is to help identify all gridcells of a grid that intersect the bicharacteristic strip of interest so that our constructed algorithms, for efficiency, may compute solely at or around these gridcells. In the absence of simple analytical equations describing the bicharacteristic strip, obtaining a checking scheme that perfectly identifies such gridcells without error is not possible; thus we turn to approximations. Given a grid over the six-dimensional phase space and a numerical method for solving the Liouville equation over this grid, which we take to be the semi-lagrangian approach, let B denote the set of

gridcells intersecting the approximately computed bicharacteristic strip. Fixing a gridcell of the grid, we check whether it is an element of B by making use of values of Φ at judiciously chosen locations, such as the gridpoints of the gridcell, though other choices are also possible when using the semi-lagrangian approach.

Before we list some options for checking schemes, we comment on what makes a good scheme. Since we need to deal with approximations, a checking scheme will encounter false positives and false negatives in its identification of elements of B. The effect of false positives is an overestimate of B, which for our algorithms translates to unwanted additional computations that reduces efficiency. The effect of false negatives can be more severe, possibly changing the topology of the constructed bicharacteristic strip; for example, introducing holes in the shape. Thus we wish to create a checking scheme that has few false positives and no false negatives. Furthermore, since the checking scheme will be used often, at the very least at each element of B, we wish it to be fast. Presented below are a list of options with increasing strictness in avoiding false positives and having no false negatives when gridsizes are small enough. At the end of the section, we briefly comment on ways to ensure gridsizes are adequately small so that false negatives can be avoided.

Our first checking scheme can be found in [3,10] and involves identifying a given gridcell as an element of B if, at the gridpoints, the values of each component of Φ change sign. This implies, by the intermediate value theorem, that the zero level set of each component passes through the gridcell. Such a check is fast, with only signs of values needed; however, in practice, the number of false positives can be quite large in many situations, when the zero level set of each component passes through the gridcell but they do not mutually intersect. Thus we seek different options.

Our second checking scheme is an improvement of the first that is ideal in specific situations. Whereas the first approach can be thought of as trying to determine the nature of the bicharacteristic strip by viewing each of its components individually, our second considers pairwise interactions between components. For any two components, we may determine if the values of one change sign at the points where the zero level set of the other intersects the one-dimensional edges of the gridcell. If there is a sign change, then the intermediate value theorem implies that the zero level sets of the two components intersect. The approach then accepts the gridcell if this condition holds for each pair of components. One-dimensional linear interpolation can be used to determine both the needed points and values from values of Φ at gridpoints, making the checking process fast, though slower than that of the first approach. This method diminishes the number of false positives compared with the first checking scheme, though overestimation of B still occurs when pairwise intersections of zero level sets do not mutually intersect. It is, however, ideal in the two-spatial dimension case, seen so often in the examples of [10], where level set functions with two components are evolved in a three-dimensional reduced phase space. We now seek even more accurate options.

Our third checking scheme is our most accurate, but slowest, one. While the second approach considers pairwise intersections of the zero level sets of the components of the level set function, this one considers the contributions of all components together. This essentially involves extracting a piece of the bicharacteristic strip in the gridcell, if it exists. If it does exist, the gridcell is identified as being an element of B; otherwise, it is deemed not to be in B. Note extraction of the bicharacteristic strip is also a necessary step for plotting the wavefront of interest so two duties can be performed at the same time. This checking scheme option, since it arrives at an accurate, usually second-order, representation of the bicharacteristic strip, is the slowest of the three but will avoid false positives in all but the most degenerate cases.

To extract a piece of the bicharacteristic strip in a gridcell, we may consider each four-dimensional face of the gridcell. In this face, we wish to determine whether the zero level sets of the four components of the level set function intersect and where this happens. Consider linear approximations of the components with normal vectors calculated from central differencing at the center of the gridcell. Because of the equality in the dimension of the face and the number of components of the level set function, the intersection of these approximations in the face satisfies a 4×4 linear system of equations. Solving this linear system with standard solvers, such as Gaussian elimination with complete pivoting, extracts an intersection point which, if it lies in the face, is approximately a point of the bicharacteristic strip. For our checking scheme, the existence of such a point in any four-dimensional face of the gridcell is the criterion for identifying that gridcell as an element of B. For plotting, the point can be projected to spatial space to obtain a point of the wavefront. Small spheres drawn around these points, for example, then form a thicker version of the wavefront that can be easily visualized using surface plotting programs.

This simple extraction method can, on the other hand, be replaced by more sophisticated options such as that found in [8], which returns triangulations of the zero level set surface of vector-valued functions that have arbitrary numbers of components defined in arbitrary dimensions. As a checking scheme, being able to find a triangle of the zero level set inside the gridcell can be the criterion for identifying the gridcell as an element of B. When plotting is concerned, however, these triangulations tend to contain huge numbers of triangles that lead to overly large memory usages in many surface plotting routines.

All the listed checking schemes avoid false negatives when gridsizes are small enough. To ensure that gridsizes are sufficiently small, we may turn to work presented in [16,9] for the purpose of grid refinement when capturing zeros of level set functions. We defer description of this idea to Section A.1 in the appendix since we have not incorporated it in our current algorithms, instead choosing for simplicity to avoid false negatives by taking very fine grids and showing, by viewing plotted wavefronts, that topological problems from false negatives indeed do not affect the topology of the solution.

3. Algorithms

3.1. Multi-resolution algorithm

The first approach for wavefront construction we consider, seen in varying degrees and forms in [7,9], makes use of the tools of the previous section to operate on a multi-resolution grid that refines at the bicharacteristic strip of interest. The algorithm can be summarized as follows:

1. Lay down a coarse grid in phase space and fix a time T.
2. For each gridcell, check if the gridcell intersects the bicharacteristic strip of traveltime T.
3. If the gridcell does intersect the bicharacteristic strip, further check if it is smaller than a desired size. If it does not intersect the bicharacteristic strip, halt further computations on it and return to the second step for another gridcell.
4. If the gridcell is sufficiently small, extract the piece of bicharacteristic strip and wavefront of traveltime T from it and return to the second step for another one. If it is not small enough, refine it and, using recursion, return to the second step for each of the refined gridcells.

Expanding on the details of the steps involved: the grid in the first step can be chosen to be a fixed, regular grid composed of hypercube gridcells based in a computational box assumed to include the bicharacteristic strip up to the time of interest; the check in the second step and extraction in the fourth step follow the discussions of the previous section; and the check in the third step stops the refinement procedure.

We further illuminate the workings of this algorithm with a simple example, following the diagram in Fig. 1. Here, a level set function with one component in $\mathbf{R}^2$ encodes a curve of interest as its zero level set and flows under a fixed velocity field. The first frame shows the initially given curve, in bold. Under the velocity field, the level set function flows to one with zero level set drawn in dashed lines. This curve is the one we wish to capture.

The algorithm starts by laying down a coarse grid over the computational domain, as seen in the second frame. Each gridcell in this coarse grid then has values computed at its gridpoints using the semi-lagrangian method, shown in the third frame. Thus gridpoints are flown backwards along the velocity field and the initial function is evaluated at the resulting locations. The calculated values can then be used by our checking schemes to determine which gridcells intersect the curve. The fourth frame shades in all of these gridcells in the grid. Then, one by one, each of these gridcells is refined, cut through the middle in each dimension to form smaller gridcells. This is shown in the fifth frame for one of the shaded gridcells. The process is repeated on each of these refined gridcells, with the fifth frame further shading in the ones that intersect the curve. If these gridcells reach the smallest the user desires for the refinement, the algorithm is halted, with the sixth frame showing all the gridcells worked on and found to be intersecting the curve. The shaded gridcells then can have their pieces of curve extracted to capture the curve of interest.

Altogether, this produces a theoretically fast and memory efficient algorithm for wavefront construction. To measure the computational complexity, we make two assumptions: first, that the finest level of refinement uses gridcells from a uniform grid with N points in each dimension, and the total number of these gridcells that

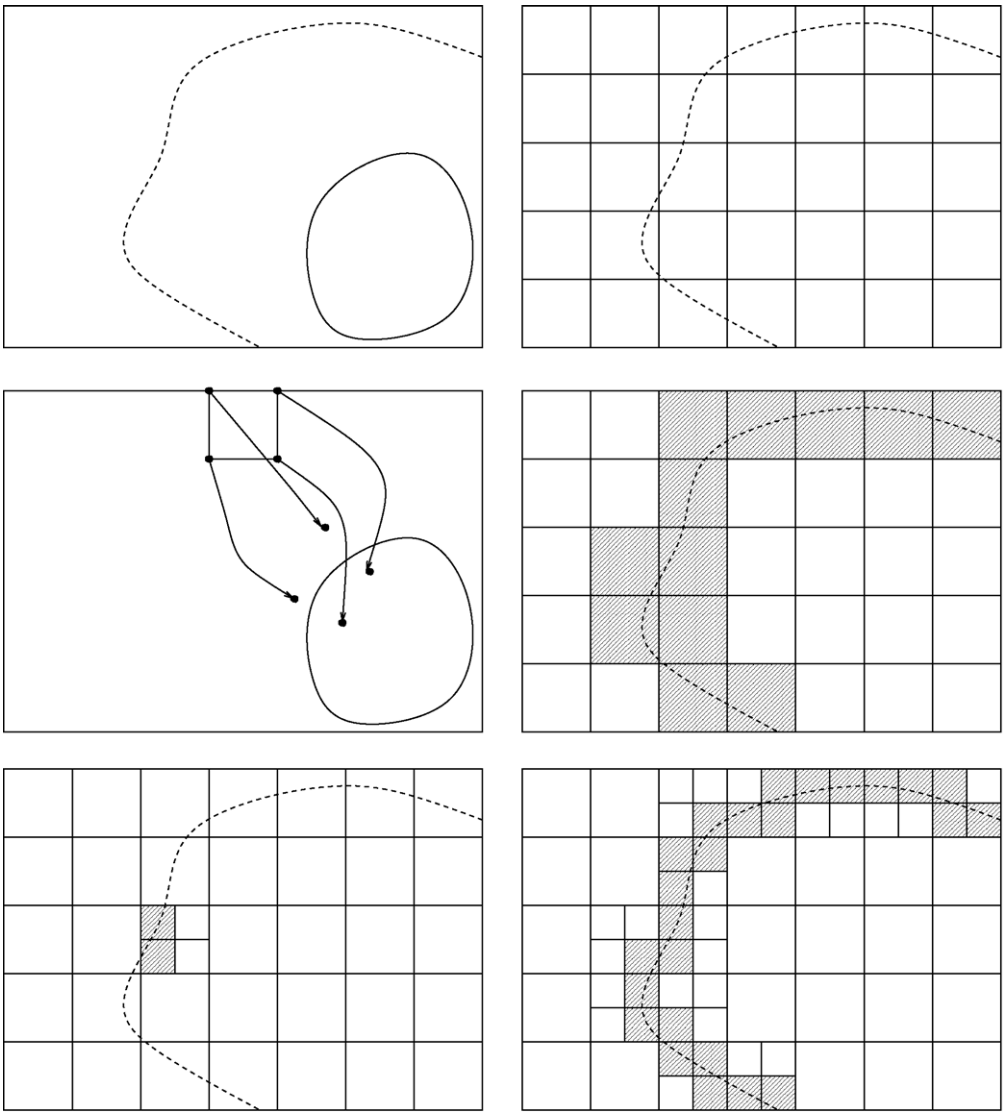

Fig. 1. Multi-resolution algorithm in a simplistic situation.

intersect the two-dimensional bicharacteristic strip is $\mathcal{O}(N^2)$; and second, that the checking scheme found in the third step is perfect in its ability to identify gridcells that intersect the bicharacteristic strip and uses $\mathcal{O}(1)$ computed values of Φ for each gridcell checked. With these assumptions, we can conclude that the number of gridcells touched by the algorithm is $\mathcal{O}(N^2)$, which is two-dimensional in nature. This means that though all quantities live in phase space, the algorithm works mainly around the two-dimensional bicharacteristic strip. The total complexity is dominated by the computation of values of Φ for these gridcells, $\mathcal{O}(MN^2)$, assuming the semi-lagrangian scheme chooses an ordinary differential equation solver that iterates through M time steps. For a further increase in speed, note the algorithm is parallelizable, and different processors can work on different gridcells of the coarse grid without communicating. In addition, memory usage can be cut down to $\mathcal{O}(\log N)$, what is needed in recursion, by discarding gridcells immediately after all computations on them are completed. Obviously, if the piece of wavefront extracted from each gridcell is stored, the memory usage increases to $\mathcal{O}(N^2)$, representing the number of elements in B at the finest grid level.

On one computer, though, and in practical application with moderately sized N, the number of gridcells this algorithm touches can be restrictively large. The two factors influencing this are the initial coarse grid and, to a lesser extent, the tree-based nature of the algorithm. For example, adopting a coarse grid with just 25 points in

each dimension forces the algorithm to work on 64,000,000 gridcells at the coarsest level, leading to slow computation times, as seen with even fewer gridcells in the three-dimensional example considered in [10]. In addition, any grid coarser than this may prevent the output of adequate results due to an increase of false negatives in the checking schemes. An idea for a separate method for avoiding false negatives for arbitrarily coarse grids can be found in [9,16] and is discussed in Section A.1 of Appendix A. This may allow the multi-resolution algorithm to have as its coarsest level a grid composed of just one gridcell. However, it would also increase the number of false positives identified in B, which would negatively affect the algorithm's speed. We, instead, discard this approach, though reincarnate a version of it in Section 3.3, and proceed to the main contribution of our work, achieving faster practical speed with a different approach.

3.2. Growing algorithm

Consider a grid, allowably fine and uniform, placed in phase space. The philosophy of our new approach is to improve speed by confining computations to gridcells intersecting the bicharacteristic strip, i.e., lying in B. The main challenge will be in finding these gridcells. We begin by identifying a single element of B. Let (x_0, p_0) be a point of the initially given bicharacteristic strip. Let (x_T, p_T) be the point after flow along the velocity field of the Liouville equation for time T (i.e., ray tracing). The gridcell this point lies in is an element of B. We call this gridcell the seed.

For more elements, we switch to a different strategy. We can, for example, check all gridcells neighboring the seed to see which of them lie in B and continue this process for the ones that do. However, to minimize the number of gridcells touched, which minimizes the number of times the semi-lagrangian method is used, we modify this approach. By checking which faces of the seed the bicharacteristic strip passes out of, identifying neighboring gridcells sharing those faces as elements of B, and continuing this process for these newly identified elements, we avoid calculating additional values of Φ. Visually, this approach grows gridcells in B uniformly outwards from the initial seed.

The algorithm can be characterized as follows:

1. Lay down a fine grid in phase space and fix a time T.
2. Choose a point of the initial bicharacteristic strip and solve the ordinary differential equation along the velocity field of the Liouville equation to determine its position at time T.
3. Add the gridcell containing that point to the end of a list, which we will call the list of boundary elements of B, and label it as the first generation of gridcells identified.
4. Proceeding through the list in order, from head to tail, take one of its gridcells. Remove elements from the list whose generations are two less than the generation of that gridcell.
5. Extract the piece of bicharacteristic strip and wavefront of traveltime T from the gridcell.
6. Check which faces of the gridcell the bicharacteristic strip exits out of and add the neighboring gridcells that share those faces, if not added before, to the end of the list. These additions are labeled with generation one above that of the gridcell we have chosen from the list. Then return to the fourth step for another gridcell.

This algorithm can be viewed as employing a breadth-first search strategy with constraints on inclusion. Elements of B are identified in an expansion out of the seed, ordered from least to furthest distance away under the taxicab metric. The list defined in the algorithm contains elements at the boundary of this expansion. Thus, geometrically, the bicharacteristic strip is extracted as a surface expanding, or growing, out of the seed.

We note this growth implies that when there are disconnected components of bicharacteristic strips, the algorithm only constructs the specific component connected to the seed. However, if starting with a connected wavefront, the wavefront at future times, in our chosen geometrical optics setting with smooth local wave speed, remains connected. Thus, to capture all connected components of a wavefront of interest, we can simply preprocess the initially given wavefront by finding all its connected components, then seeding each one and applying the growing algorithm for each. In the presence of refraction, which is the case of discontinuous local wave speed c, though, an initially connected wavefront may develop a disconnected form at later time. This does not pose a problem for the multi-resolution algorithm but is an issue for our growing algorithm, which cannot handle this case effectively at the moment.

Expanding on the details of certain steps of the algorithm: the grid of the first step is fine enough to provide the desired resolution but need not physically exist in memory; the second and third steps seed an initial gridcell of B for iteration of the fourth, fifth, and sixth steps; the third step can be modified to add all neighbors of the seed as well, as approximation errors may reject the seed as an element of B; the fourth step culls the list, removing elements in the interior of B and thus no longer needed for the redundancy search in the sixth step; and the check in the sixth step uses the checking schemes of Section 2.2 by viewing the faces of the gridcell as lower-dimensional gridcells.

We further illuminate the workings of this algorithm using the same simplistic setting considered by the multi-resolution algorithm in Section 3.1. The growing algorithm is shown at work in this setting in Fig. 2. The first frame shows the initially given curve, in bold, and the final curve, after flowing under the underlying velocity field, in dashed lines. The algorithm proceeds to lay down a grid, as seen in the next frame. A point is chosen on the initial curve and flown along the velocity field, where it ends up as a point of the desired curve. The gridcell that contains this point is the seed, our first element of B, seen shaded in the third frame. This element is added to a list tagged with a 1, meaning the first generation of identified elements. The purpose of this list is to prevent inclusion of previously identified elements of B. When an element is removed from the list, its generational label will be removed in the picture.

Now working with our only element of the list, we start the process of extracting information and use the semi-lagrangian method, shown in the fourth frame, to obtain values of the corresponding level set function that encodes the desired curve. Thus gridpoints are flown backwards along the velocity field and the initial level set function is evaluated at the resulting locations. These values can then be used to extract the piece of desired curve lying in the gridcell, which can then be outputted.

In the next frame, the values are additionally used in each face of the gridcell to check which ones the desired curve passes out of. The neighbors of those faces are added as elements of B and into our list, with generational labels of 2. Redundancy in the list is technically checked before adding each of these new gridcells to the list, but there will be none at this early stage.

Moving to the sixth frame, the new elements of the list are one by one operated on in the same way: values are obtained at gridpoints, a piece of the desired curve is extracted, and faces are checked to find the next elements to be added to B and our list. Note, since our first generation gridcell is a neighbor of those of second generation, sharing faces that the desired curve passes through, it will be identified in the process. This is where a check through the list will avoid a repeated inclusion. The rest of the new gridcells, since they arise from ones of generational label 2, are given the generational label of 3.

At the end of this step, the gridcell of generation 1 can be removed from the list. This is because the set B only grows at its ends, or boundaries. Thus, the list just needs to be active at these locations and gridcells in the interior of B can be ignored in future redundancy checks. We remove its number in the seventh frame to show this but leave the gridcell shaded to remind us that it is an element of B. The process is then repeated for each element of the list with generation 3, resulting in the gridcells of B labeled with a 4, and, in the last frame, for each element of generation 4, to capture the gridcells of B labeled with a 5. Throughout these steps, redundancies are checked and elements of the list are removed when they no longer contribute to these checks.

The final result, which ends this example, is shown in Fig. 3, where the shaded elements identified by the algorithm are exactly all the gridcells in the grid that intersect the desired curve.

Returning to discussions of the algorithm in the full geometrical optics setting, the number of gridcells the algorithm touches, if we can exactly determine which faces of a gridcell the bicharacteristic strip exits out of, is just the number of gridcells the bicharacteristic strip intersects. Thus, compared with the multi-resolution algorithm that uses the same grid at its finest scale, our growing algorithm computes on fewer gridcells. The list of boundary elements, however, adds to the computational complexity of the approach. The gridcells needed in the list are those neighboring the boundary of the growing set of identified elements of B. This means the elements of the list lie near the one-dimensional boundary of the growing bicharacteristic strip. Thus, using the same notations and under the same assumptions made for the multi-resolution algorithm, we expect the number of elements in the list to be $\mathcal{O}(N)$. A direct consequence of this is that memory storage requirements for the list, and algorithm, when gridcells not in use are discarded, will be $\mathcal{O}(N)$, or $\mathcal{O}(N \log N)$ if the list is additionally stored as a tree. In terms of speed, each redundancy search will have complexity $\mathcal{O}(N)$ with linear searching through the list, or $\mathcal{O}(\log N)$ with tree-based searching. Thus the algorithm's total complexity will receive

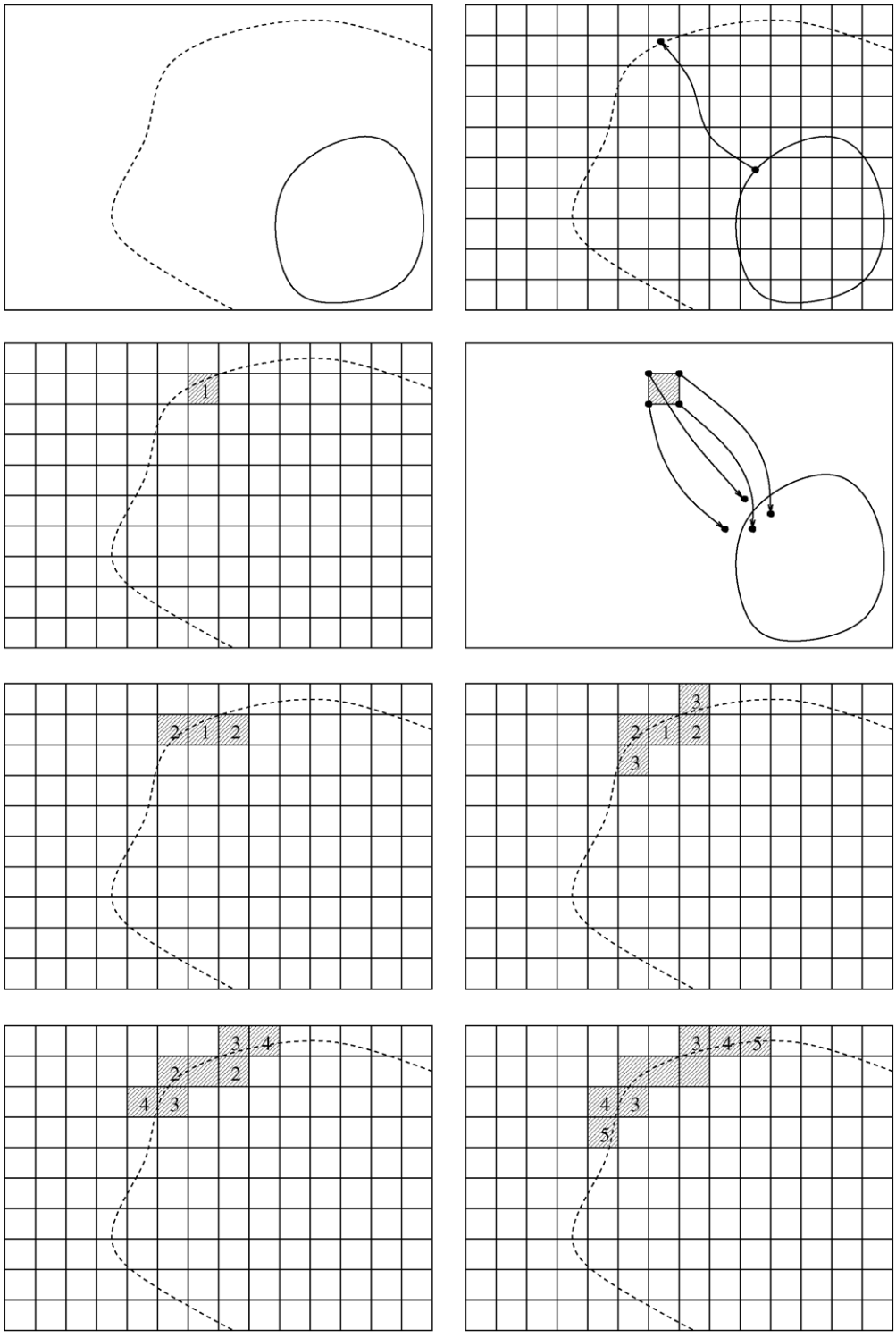

Fig. 2. Growing algorithm in a simplistic situation.

contributions of $\mathcal{O}(MN^2)$ from computing values of Φ at $\mathcal{O}(N^2)$ gridcells and either $\mathcal{O}(N^3)$ or $\mathcal{O}(N^2 \log N)$ from searching. In the case, where M is of the same size as N, the complexity will be the same as for the multi-resolution approach.

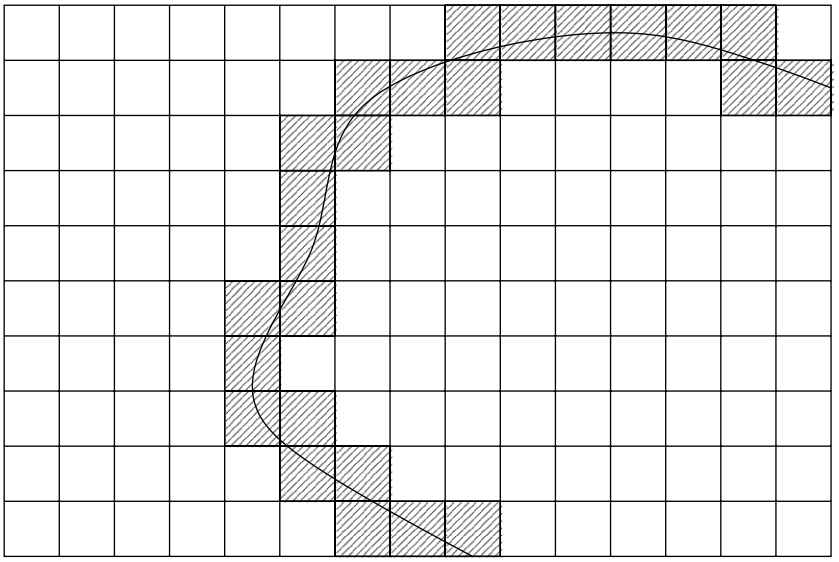

Fig. 3. Final gridcells captured by the growing algorithm in a simplistic situation.

From this analysis, we see that the growing algorithm may have the same complexity as the multi-resolution algorithm for large N; however, it performs much faster for N used in practical situations. We show experimental results of its speed, memory, and constructed wavefronts in Section 4.

3.3. Variations

One variation of our algorithms involves replacing the six-dimensional phase space with the one-dimension lower reduced phase space $\{(x, p/|p|)|x \in \mathbf{R}^3,\ p/|p| \in \mathbf{S}^2\}$. In this setting, the level set function has three components and evolves under the appropriate translation of the Liouville equation. Good parametrizations of the sphere, however, are lacking. The attempts made in [10] were far from optimal due to use of spherical coordinates which are singular at the poles of the sphere. The semi-lagrangian method for solving the Liouville equation allows us to consider other options. Choosing a good triangulation of the sphere, we may work on a grid composed of gridcells that are products of hypercube gridcells of the spatial grid with triangles from the triangulation of the sphere. Both of our algorithms carry through as usual using these new gridcells and the correct Liouville equation.

The advantage of using reduced phase space is an increase in speed due to reduction of dimension from six to five, which lowers the number of gridpoints from 64 to 24 and faces and neighbors from 12 to 9 for each gridcell. Furthermore, the level set function is reduced by one component. One disadvantage is the slight complication arising from the use of triangulations and how to generate, store, and access them. We currently generate the triangulations using Matlab and read in and store the complete triangulation in memory, increasing memory requirements to $\mathcal{O}(N^2)$. Another disadvantage is related to initialization of the level set function. The approach of assigning the components ϕ_1 and ϕ_2 for phase direction and ϕ_3 for spatial location will often lead to the creation of a level set function encoding two bicharacteristic strips: one moving in the desired direction and the other in the opposite direction. This is not a problem for the growing algorithm, which will only pick out the correct one since the two are disconnected; however, the multi-resolution approach needs modification so that additional work in capturing a spurious surface is avoided. Altogether, though, we found that the speeds attained when operating in reduced phase space are only slightly faster and perhaps not worth the added complications.

Other variations involve hybrids of the multi-resolution and growing algorithms. The growing algorithm used first over a coarse grid can identify gridcells that intersect the bicharacteristic strip. The multi-resolution process can then continue on these gridcells, eliminating the need to initially try out all gridcells of the coarse

grid. A description of this can follow the diagrams of Fig. 1 if we start from the third frame and consider the shaded gridcells as ones found to be in B by the growing algorithm for that grid. The remaining frames then go on to show, as before, the refinement procedure of the multi-resolution algorithm.

This hybrid algorithm removes the main obstacle to fast computations for practical N in the multi-resolution algorithm while preserving the same order of speed and memory storage requirements for large N. Numerical results confirming this are shown in Section 4.1. In addition, such a hybrid can apply smaller grids to regions of the bicharacteristic strip that deserve more attention, whether due to a desire for better resolution, accuracy, or to alleviate the symptoms of possible instability in the level set representation. We expand on the latter as it may represent a final hurdle for level set approaches in geometrical optics.

Stability refers to the change in the zero level set under perturbation of the level set function. For a one-component level set function, the method of choice for removing instabilities is called reinitialization (see, e.g. [1,15] for discussions) where a specific form for the level set function, that of signed distance, is chosen and the level set function is replaced with that form every time it deviates from it. Thus reinitialization is usually performed after each time step of a flow on the level set function. For vector-valued level set functions, this is more complicated, though some basic work can be found in [3]. We can follow some of the same principles for our work here, and present a description of some still immature ideas in Section A.2 of Appendix A.

4. Numerical results

4.1. Speed and memory

We begin our section on numerical results by testing the speed and memory usage of our growing and hybrid algorithms. Consider the initial wavefront a point source at y in a medium of local wave speed $c \equiv 1$, given in phase space by the representation

$$(\phi_1, \phi_2, \phi_3) = y,$$
$$\phi_4 = |p| - 1.$$

Wavefronts of other traveltimes form spheres growing out of the initial point source. The constant local wave speed implies the identity

$$\Phi(x, p, t) = \Phi(x - pt, p, 0),$$

since the integral curves of the velocity field for the Liouville equation are straight lines. Use of this identity to obtain values of Φ allows us to determine the contributions of phase space independent of the traveltime. Note, also, this can be thought of as using Euler's method to jump to the solution in one giant step, made possible because there is no CFL condition to satisfy between gridsize and time step.

We study the properties of three approaches: the growing algorithm with linear searching, the growing algorithm with tree-based searching, and a hybrid that follows the growing algorithm with multi-resolution refinement. All three use pairwise intersections of components of the level set function to check for intersections of gridcells with bicharacteristic strips, our second approach listed in Section 2.2.

Our first two algorithms employ the growing algorithm but with different strategies in storing their lists of boundary elements of B. This affects the redundancy searches that are conducted through the lists prior to the adding of new elements. Linear searching proceeds one by one through the list to see if redundancy occurs while tree-based searching stores the list also as a tree for fast searches. Table 1 counts several quantities of importance to speed and memory in growing the spherical wavefront when the computational domain in phase space is $[-1, 1]^6$ and $y = (0.001, 0.001, 0.001)$, chosen not to lie on a gridpoint.

In this table, the first column presents the traveltime of the wavefront constructed and the second the underlying fine grid. The third column counts the number of gridcells touched by the algorithms. This number contributes to speed since each gridcell requires several solves of the Liouville equation for values of the level set function. We make three comments on the numbers in this column. The first is that though we have not adopted the most accurate checking routine for determining intersections of gridcells with bicharacteristic strips, use of the third option described in Section 2.2 will only change the counts by 4% at most for this prob-

Table 1
This table counts quantities of importance to speed and memory in the growing algorithms

Time	Grid	Cells	Order	Searches	Order	List	Order
Speed and memory of growing algorithm							
0	50^6	11,864		9,524,917		543	
0	100^6	47,168	1.9912	73,254,143	2.9431	1,032	0.9264
0	200^6	188,600	1.9994	581,551,384	2.9889	2,043	0.9852
0.5	50^6	26,750		31,793,290		801	
0.5	100^6	107,843	2.0113	256,756,224	3.0136	1,593	0.9919
0.5	200^6	437,505	2.0204	2,117,975,866	3.0442	3,204	1.0081
1	50^6	48,128		76,236,759		1,077	
1	100^6	195,249	2.0204	633,884,315	3.0557	2,199	1.0298
1	200^6	798,201	2.0314	5,311,800,799	3.0669	4,356	0.9862

lem. Other problems, however, may need the slower but more accurate routine. The second comment is that because N is not large enough, the numbers in the column do not share a close relationship with the change in surface area of the bicharacteristic strip at different traveltimes, which can be calculated to be $4\pi(\rho^2 + 1)$ for a spherical wavefront of radius ρ. This relationship seems to hold, however, when N and traveltime are much larger than what is given in the table. The third comment is that the numbers in this column show the power of our method in automatically adapting the number of points in its representation of a bicharacteristic strip under deformation. This is exactly the obstacle faced by ray tracing methods.

Continuing with the table, the fourth column determines the orders associated with these numbers from the use of finer grids and verifies the $\mathcal{O}(N^2)$ complexity corresponding to the two-dimensional nature of touched gridcells. The fifth column adds together the number of elements in the list of boundary elements each time a redundancy check is performed. The size of these numbers relates to the speed, in terms of comparisons, of all searches made through the list. The sixth column determines the order associated with these numbers and verifies the $\mathcal{O}(N^3)$ corresponding to the total number of searches using linear searching. The seventh column shows the maximum size the lists attain throughout the program. This number contributes to the memory usage of the algorithms in storing these quantities. The eighth column determines the order associated with these numbers and verifies the $\mathcal{O}(N)$ complexity corresponding to the one-dimensional nature of the boundary elements of B.

Tables 2 and 3 give further verification of speed and memory estimates based on actual recorded times and memory usages for the growing algorithms on a computer. These numbers, however, are subject to randomness and programming style and so Table 1 may be a better source of information on the speeds and memory usages of the algorithms.

In both these tables, the first column presents the traveltime of the wavefront constructed and the second the underlying fine grid. The third column gives the measured times in seconds, averaged over ten samples, for completion of the algorithms. The fourth column determines the order associated with these numbers. The

Table 2
This table collects runtimes and memory usages of the growing algorithm with linear searching

Time	Grid	Runtime	Order	Memory
Runtime and memory usage with linear searching				
0	50^6	2.304		900
0	100^6	8.944	1.9568	924
0	200^6	37.109	2.0528	964
0.5	50^6	5.291		912
0.5	100^6	22.061	2.0599	944
0.5	200^6	95.512	2.1142	1004
1	50^6	9.466		924
1	100^6	41.286	2.1248	964
1	200^6	178.848	2.1150	1056

Table 3
This table collects runtimes and memory usages of the growing algorithm with tree-based searching

Time	Grid	Runtime	Order	Memory
Runtime and memory usage with tree-based searching				
0	50^6	2.222		992
0	100^6	8.872	1.9974	1112
0	200^6	35.919	2.0174	1364
0.5	50^6	5.372		1164
0.5	100^6	21.749	2.0174	1496
0.5	200^6	92.414	2.0872	2228
1	50^6	9.383		1348
1	100^6	40.110	2.0958	1936
1	200^6	169.852	2.0822	3180

orders of Table 3 agree with the $\mathcal{O}(N^2 \log N)$ complexity of the growing algorithm with tree-based searching; however, those of Table 2 are better than the $\mathcal{O}(N^3)$ complexity predicted for the growing algorithm with linear searching. This hints that only much larger N will produce that theoretical behavior in complexity. The fifth column reveals the amounts of memory used, in kilobytes, by the algorithms. This includes 796 kbyte allocated by the programs in all cases before execution of the first steps of the algorithms. From these tables, we note tree-based searching gives slight benefits in speed for the grids used but has larger memory usage due to overhead for storing tree-structures.

Turning to tests of a hybrid algorithm, we consider a growing algorithm with linear searching strategy performed over a coarse grid with 25 points in each dimension followed by multi-resolution refinement on the resulting identified gridcells. Refinement in this case consists of subdividing hypercube gridcells in half in each dimension. Table 4 counts several quantities of importance to speed and memory in the same spherical wavefront setting.

In this table, the first column presents the traveltime of the wavefront constructed and the second the underlying fine grid. The third column counts the number of gridcells touched by the algorithm, with the growing algorithm contributing 2906 of these from the initial coarse grid. Note these numbers are much larger than those of our growing algorithms studied in Table 1 but also much smaller from the 64,000,000 needed just at the coarsest grid level if the multi-resolution algorithm were applied alone. The fourth column determines the order associated with the information of the third column from the use of finer grids and verifies the $\mathcal{O}(N^2)$ complexity corresponding to touched gridcells. The fifth column counts the number of elements at the finest grid level identified to be in B. These numbers are approximately equal to those identified by the growing algorithms in Table 1, though not exactly equal because the large size of gridcells in the coarse grid causes the growing algorithm to identify slightly fewer elements. This explanation is verified through the use of finer grids at the coarse level which results in matching numbers between the tables. The seventh column determines the

Table 4
This table counts quantities of importance to speed and memory in the hybrid algorithm

Time	Grid	Cells	Order	Final	Order
Speed and memory of hybrid algorithm					
0	50^6	188,890		11,840	
0	100^6	946,650	2.3253	47,144	1.9934
0	200^6	3,963,866	2.0660	188,576	2.0000
0.5	50^6	432,770		26,726	
0.5	100^6	2,143,234	2.3081	107,819	2.0123
0.5	200^6	9,043,650	2.0771	437,469	2.0206
1	50^6	764,400		48,002	
1	100^6	3,836,528	2.3274	194,961	2.0220
1	200^6	16,314,032	2.0882	797,268	2.0319

order associated with the information of the fifth column and verifies the $\mathcal{O}(N^2)$ complexity of identified elements of B.

Table 5 gives further verification of speed and memory estimates based on actual recorded times and memory usages for the hybrid algorithm on a computer. These numbers, again, are subject to randomness and programming style and so Table 4 may be a better source of information on the speed and memory usage of the algorithm.

In this table, the first column presents the traveltime of the wavefront constructed and the second the underlying fine grid. The third column reveals the measured times in seconds, averaged over ten samples, for completion of the algorithm. We see that this algorithm is slower than the growing algorithms studied earlier. The fourth column determines the orders associated with the numbers in the third column and

Table 5
This table collects runtimes and memory storages of the hybrid algorithm

Time	Grid	Runtime	Order	Memory
Runtime and memory usage of hybrid algorithm				
0	50^6	7.849		900
0	100^6	33.018	2.0727	904
0	200^6	132.597	2.0057	904
0.5	50^6	18.046		908
0.5	100^6	80.631	2.1597	908
0.5	200^6	332.910	2.0457	908
1	50^6	29.726		912
1	100^6	140.280	2.2385	916
1	200^6	591.672	2.0765	916

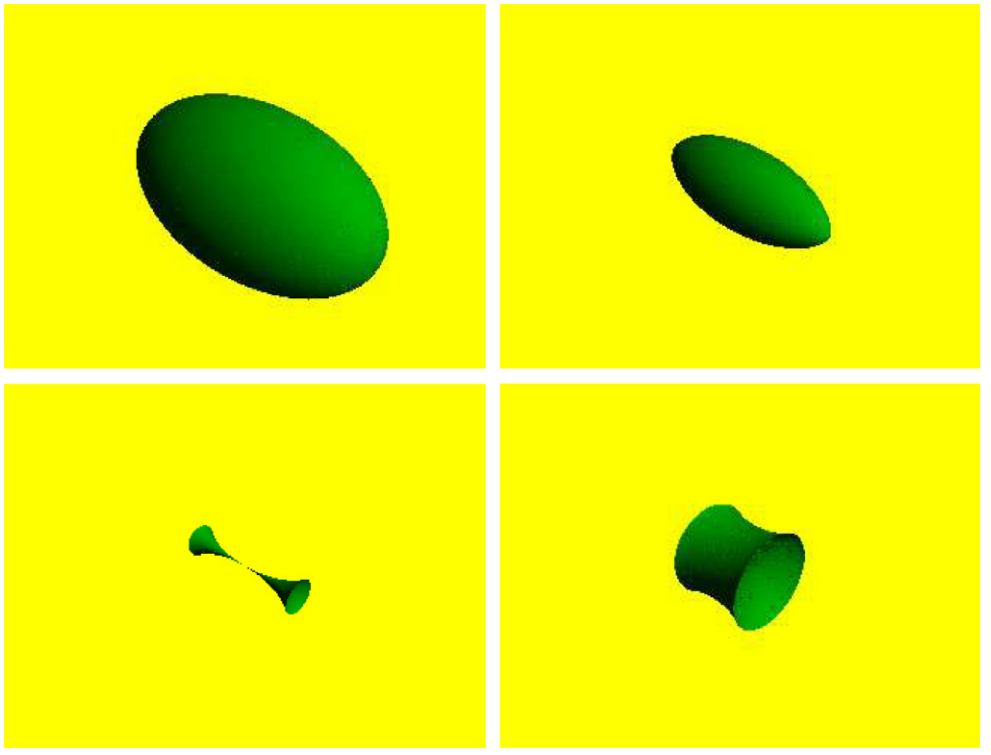

Fig. 4. Ellipsoid with two equal principal radii shrinking into multi-valued forms.

verifies the $\mathcal{O}(N^2)$ complexity of our hybrid algorithm. The fifth column reveals the amounts of memory used, in kilobytes. This includes 800 kbyte allocated in all cases before execution of the first step of the algorithm. Note the low amounts of memory used here throughout the experiments of this table.

In total, we get a sense from these experiments of the speeds and memory usages of our algorithms and the sizes of the grids allowed. Compared with the results of [9], we have been able to run simulations on grids with up to 400^6 cells while those of [9] show results on grids with a maximum of 64^5 cells. Since it may not be fair to compare actual speeds and memory usages due to the randomness of the numbers and the different computers used, we merely say that we suspect our algorithms are better in both respects. Thus we believe we have produced one of the first efficient level set algorithms for large-scale computations in wavefront construction.

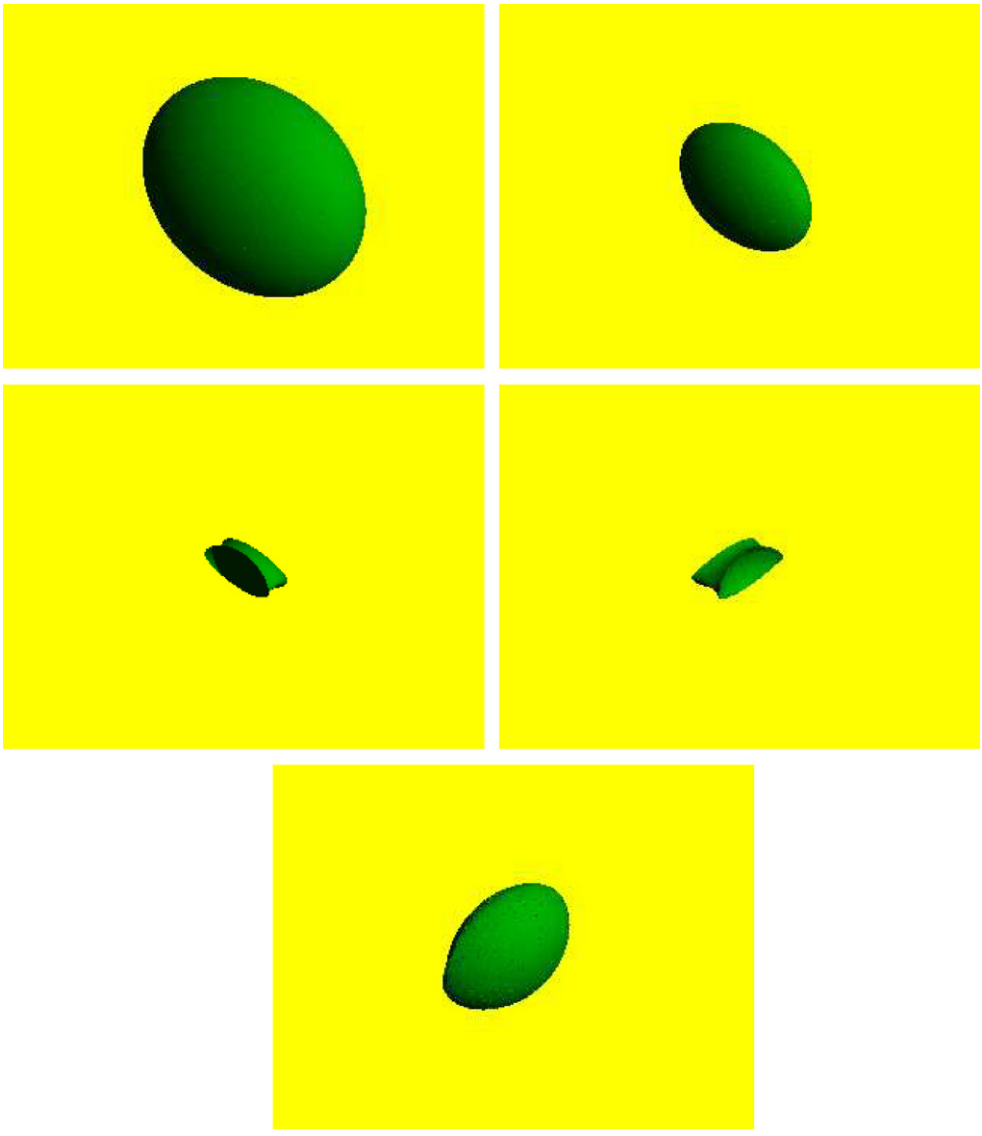

Fig. 5. Ellipsoid with different principal radii shrinking into multi-valued forms.

4.2. Wavefronts

We now present plots of some of the wavefronts constructed by our algorithms in three dimensions. Starting with constant local wave speed, $c \equiv 1$, Fig. 4 shows a shrinking ellipse where two of the principal radii are the same, constructed by the growing algorithm with linear searching and a 100^6 point grid. The results correspond to surfaces of rotation of shrinking ellipses in two-dimensional space, such as those shown in [10]. From the figure, we see that not only are surface details resolved but multi-valued characteristics are correctly displayed. Fig. 5 shows a shrinking ellipse where the principal radii are all different, constructed using the same algorithm. Again, resolution and multi-valued forms are satisfactorily handled by our approach. These two figures are in fact plotted by triangulating the bicharacteristic strip. When working with gridcells from the algorithm's list of recently identified elements, we may take a gridcell's center, that of a neighbor further toward the tail end of the list, and that of the neighbor's neighbor even further down and project them to the bicharacteristic strip to form a triangle lying on the surface. The number of triangles, though smaller than that returned by application of [8], is large but for these two cases within a range that can be handled by standard surface plotters.

To project the center of the gridcell onto the bicharacteristic strip, we may apply central differencing there to approximate derivatives of each component from values of Φ at gridpoints. These can then be used to form linear approximations of each component. Assigning our point to be the one at the intersection of the hyperplane zero level sets of minimum distance from the center, computable using lagrange multipliers on this constrained quadratic minimization problem to form a linear system of equations, gives us our projection to the bicharacteristic strip. Note drawing spheres around such points for each gridcell in B leads to another plotting scheme for the bicharacteristic strip. This is, in fact, the approach we use, for simplicity, for our remaining figures.

Our next series of results keep a constant local wave speed, $c \equiv 1$, but introduce complexities through the presence of reflecting solids in the environment. The algorithm incorporates two modifications to treat this problem. The first involves changing the semi-lagrangian scheme at the boundaries of solids so that a point

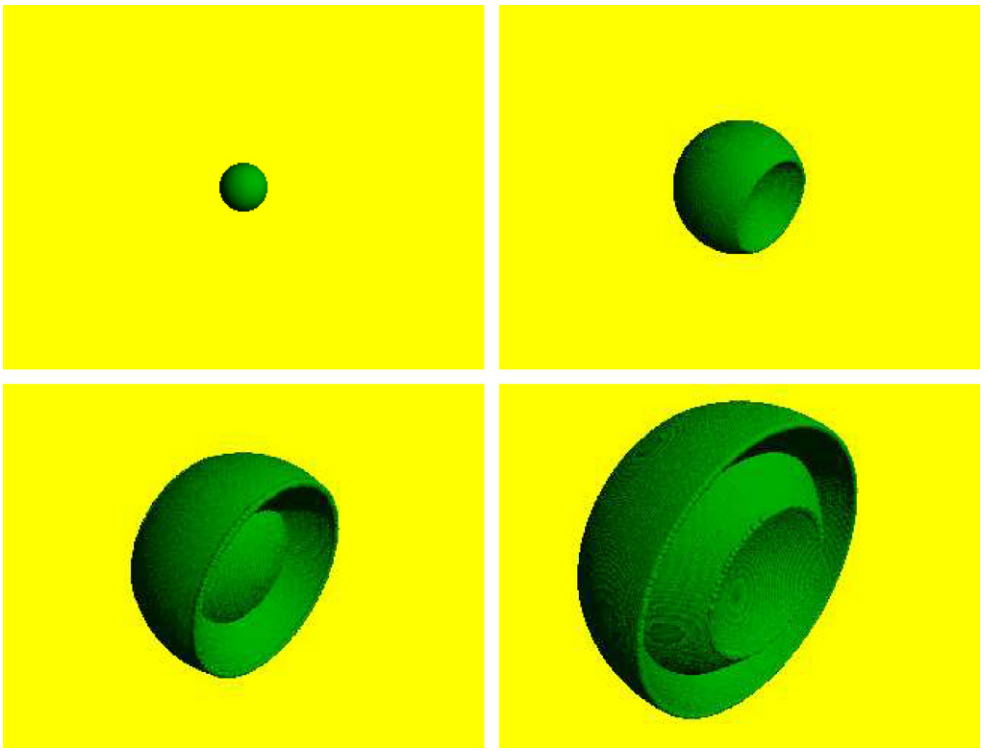

Fig. 6. Growing sphere trapped between and reflecting off two walls.

flowing along the velocity field of the Liouville equation, when it finds itself entering a solid, will be lifted to a direction exiting the solid that respects reflection conditions. The discontinuity of this flow at these boundaries, however, will generate disconnected bicharacteristic strips that forms the wavefront. Thus the second modification involves the use of multiple initial seed points in the growing algorithm, one for each disconnected piece of reflected bicharacteristic strip. In this work, we assume that these seed points are given. This, we note, is an unreasonable assumption for the application of wavefront construction with reflection. Tackling this application, however, is not our goal here. Instead, the adding of reflection, even when seed points need to be given, is an interesting case that results in complicated wavefront shapes and requires multiple applications of our algorithm. This thus serves as an effective test of our algorithm. A full study of reflection, however, is one of our goals for future work.

In Fig. 6, the reflecting boundaries are two walls and the wavefront grows from a point source located in the middle. The wavefront reflects off one wall, travels to the other, and again reflects off that one. In Fig. 7, the reflecting solid is an ellipsoid and the initial wavefront is a shrinking sphere surrounding it. The wavefront first reflects off the elongated ends of the ellipsoid. At later time, when all points have reflected off the ellipsoid, the wavefront can be seen growing away from the reflecting shape. In both examples, the multi-valued characteristics of the wavefronts are captured and the growing surfaces are properly resolved.

Our last series of results introduce complexity with a variable local wave speed,

$$c(x) = 0.5\sin(2\pi x_1)\sin(2\pi x_2)\sin(2\pi x_3) + 1.$$

Starting with a spherical wavefront of radius 0.4, Fig. 8 shows the behavior of an initially shrinking sphere under our growing algorithm with linear searching. Runge–Kutta of third-order is used to solve the ordinary differential equations of the semi-lagrangian scheme and second-order extraction, our third approach of Section 2.2, is employed for a more accurate test of the intersection of gridcells with bicharacteristic strips. Throughout the evolution shown in the figure, the wavefronts remain well-resolved and their multi-valued characteristics are properly captured by the algorithm, as verified when compared to ray tracing results. Be-

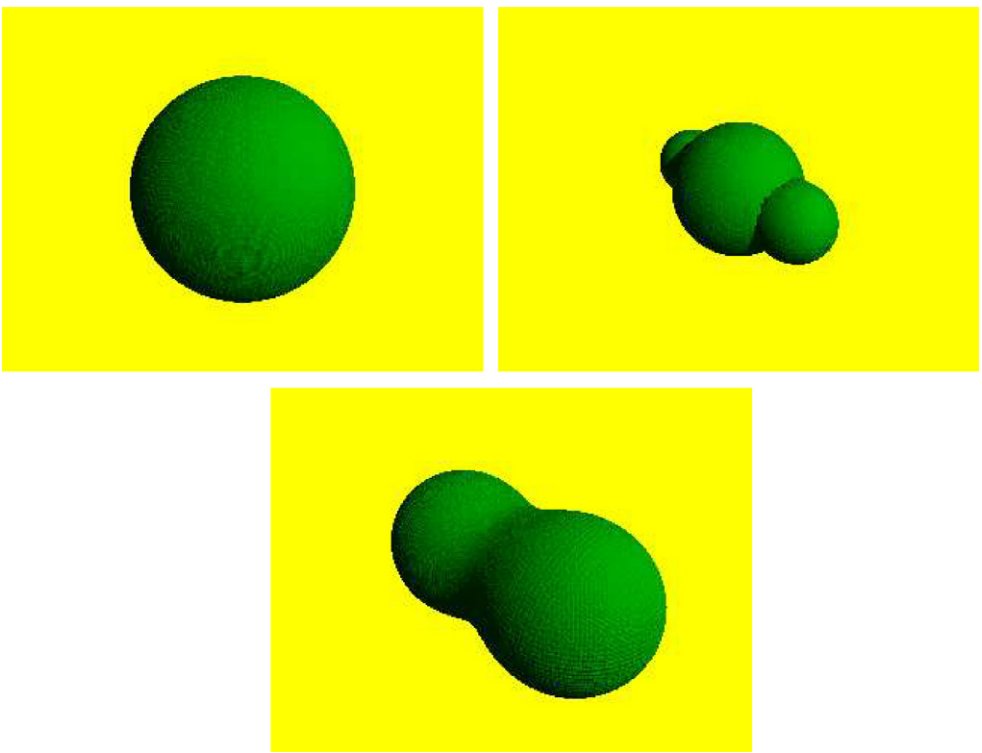

Fig. 7. Shrinking sphere reflecting off an ellipsoid in its interior.

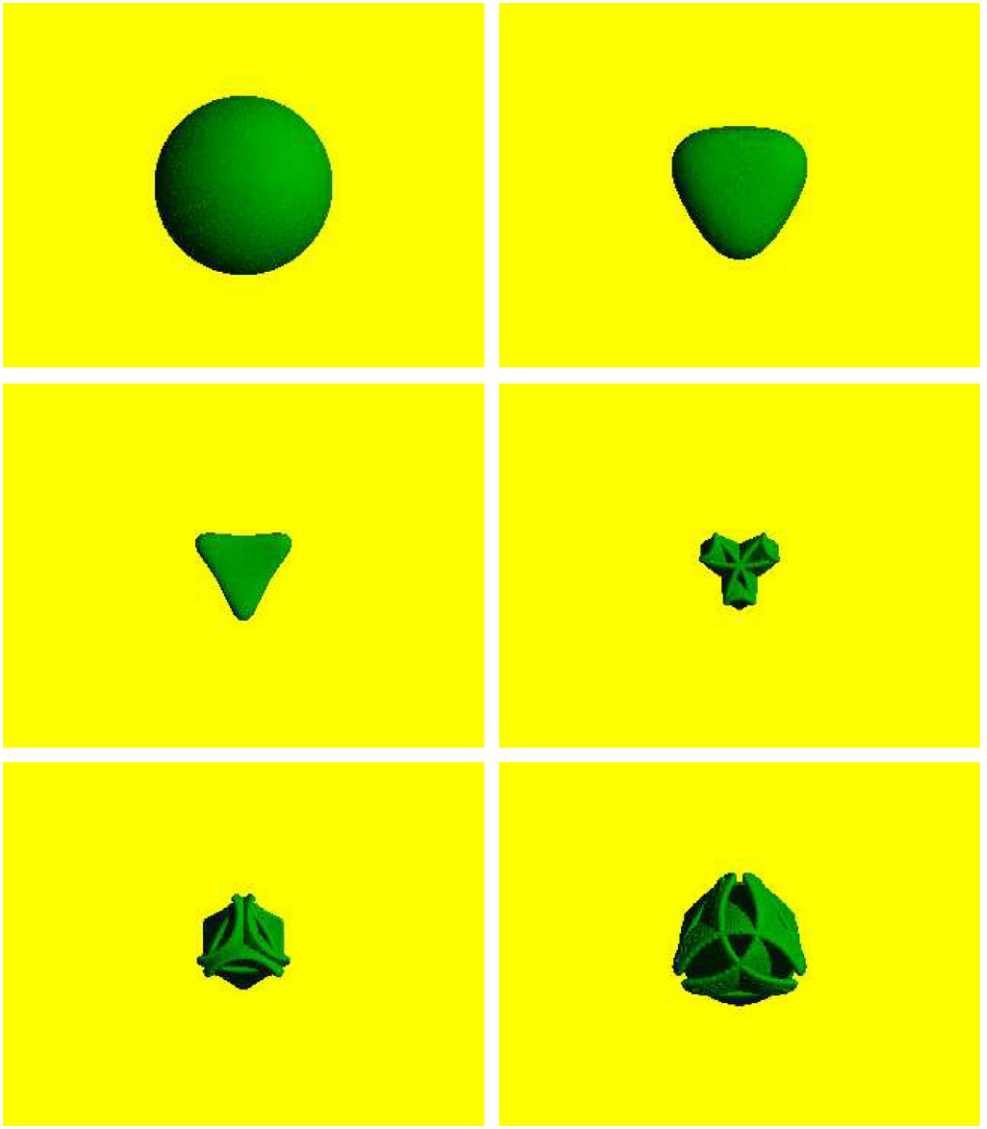

Fig. 8. Shrinking sphere in medium with variable local wave velocity.

cause the surfaces in the figures come from the plotting of spheres around points on the wavefront, corners on the multi-valued surface may not appear as sharp as they in reality should be.

5. Conclusion

We introduced in this work a multi-resolution algorithm and a growing algorithm for efficient construction of wavefronts in three spatial dimensions. Both of these algorithms take advantage of the flexibility afforded by a long-time semi-lagrangian method for solving the Liouville equation to reduce the number of gridcells that require calculation. The multi-resolution algorithm has good speed and memory characteristics while the growing algorithm is faster in practice. Hybrids of the two improve on certain aspects of both approaches and may be able to alleviate the effects of unstable level set representations. With three spatial dimensions now very much a reality, investigating better methods for removing instabilities is the last major hurdle for the level set method in wavefront construction.

Acknowledgments

The author thanks Jianliang Qian, Stanley Osher, Chohong Min, and Yen-Hsi Richard Tsai for enlightening discussions and encouragement throughout this work. Research was supported in part by a Sloan Fellowship and NSF Grant #0511766.

Appendix A

A.1. Avoiding false negatives

The techniques found in [9,16] can be used to ensure gridcells are small enough to avoid false negatives in checking schemes. These techniques, using knowledge of the Lipschitz constant of the level set function along with values at gridpoints, can be used to determine whether it is possible for the function to have zeros in a gridcell. This aided in grid refinement in those papers, as it works even on coarse grids, and gridcells were refined as long as there was a chance that there were zeros in it. We can use it similarly for our purposes by adding it to our checking schemes. This means we can identify a gridcell as an element of B if it is identified by both schemes, i.e., identified by the checking scheme and also positive for possibly having a zero in the gridcell with the Lipschitz constant scheme. In addition, we reject the gridcell if it is rejected by both schemes. If the two do not agree, the conclusion is that the gridcell is not sufficiently small, perhaps leading to a false negative. Such gridcells are then refined.

In order for this to work, the Lipschitz constant for a function flowing under a velocity field v needs to be calculated in terms of the Lipschitz constant of the initial form. We present this in the case where the functions, and velocity fields, are smooth. Let $u(x,t)$ be a function flowing for time t under the transport equation

$$u_t + v \cdot \nabla u = 0$$

with initial condition $u(x,0) = u_0(x)$. Suppose $|\nabla u_0| \leqslant L$, so that L is the Lipschitz constant for u_0. We wish to find a bound on $|\nabla u|$ at time t. Taking a gradient on both sides of the transport equation, we get

$$\nabla u_t + \nabla v \cdot \nabla u + \nabla^2 u \cdot v = 0,$$

where $(\nabla v)_{ij} = (v_j)_{x_i}$. This equation shows how the quantity ∇u changes under the flow. Taking a dot product of this result with $2\nabla u$ and rearranging terms leads to the equation

$$(|\nabla u|^2)_t + v \cdot \nabla(|\nabla u|^2) = -2\nabla u \cdot \nabla v \cdot \nabla u.$$

Note, however, that $\nabla u \cdot \nabla v \cdot \nabla u = \nabla u \cdot (\nabla v)^{\mathrm{t}} \cdot \nabla u$, and so

$$-2\nabla u \cdot \nabla v \cdot \nabla u = -(\nabla u \cdot (\nabla v + (\nabla v)^{\mathrm{t}}) \cdot \nabla u),$$

which is a symmetric matrix. Thus, we can use properties of symmetric matrices involving their eigenvalues, such as the bound

$$\lambda_{\min}|\nabla u|^2 \leqslant -(\nabla u \cdot (\nabla v + (\nabla v)^{\mathrm{t}}) \cdot \nabla u) \leqslant \lambda_{\max}|\nabla u|^2,$$

where $\lambda_{\max} \in \mathbf{R}$ is the maximum eigenvalue of $-(\nabla v + (\nabla v)^{\mathrm{t}})$ and $\lambda_{\min} \in \mathbf{R}$ the minimum eigenvalue. This implies

$$(|\nabla u|^2)_t + v \cdot \nabla(|\nabla u|^2) \leqslant \lambda_{\max}|\nabla u|^2.$$

Now letting $w(y,t) = w(x - vt, t) = |\nabla u(x,t)|^2$, we have

$$w_t(y,t) \leqslant \lambda_{\max} w(y,t).$$

Thus, by Gronwall's inequality, we can conclude that

$$w(y,t) \leqslant \mathrm{e}^{\lambda_{\max} t} w_0(y),$$

where $w_0(y) = w(y, 0)$. Translating back to u, we get

$$|\nabla u|^2 \leqslant e^{\lambda_{\max} t} |\nabla u_0|^2,$$

and so

$$|\nabla u| \leqslant e^{\lambda_{\max} t/2} L.$$

Thus u, at time t, has Lipschitz constant $e^{\lambda_{\max} t/2} L$.

A.2. Avoiding instabilities

A level set representation is unstable if the zero level set changes significantly under a small perturbation of the level set function. For our vector-valued level set functions, this occurs, for example, when any of the components develops small derivatives at the bicharacteristic strip. Furthermore, if the zero level set of one component is almost parallel to that of another, a small perturbation may move their intersection significantly. Both cases make it difficult to obtain accurate and reliable information from the level set function needed, for example, when extracting and plotting the bicharacteristic strip. Better resolution at unstable locations, naturally, alleviates some of the effects of instabilities, but does not fix the issue.

Borrowing from the technique of reinitialization for one-component level set functions, our current idea for removing instabilities involves running our algorithms in intervals in time where, at the end of each interval, Φ is iterated to or replaced by a more stable form (see [3] for more on reinitialization of vector-valued level set functions). The added operations, however, will decrease the speed of our algorithms and increase the programming complexity. The computational complexity when reinitialization is performed at each time step will be $\mathcal{O}(MN^2)$ for the multi-resolution algorithm, $\mathcal{O}(MN^3)$ for the growing algorithm with linear searching, and $\mathcal{O}(MN^2 \log N)$ for the growing algorithm with tree-based searching. Thus, this method reduces the speed of our growing algorithm to possibly unacceptable levels, but is the best we have at the moment. In-depth studies of the instabilities that arise and approaches for their removal must be left to future research.

References

[1] D. Adalsteinsson, J.A. Sethian, A fast level set method for propagating interfaces, J. Comput. Phys. 118 (1995) 269–277.
[2] J.-D. Benamou, An introduction to eulerian geometrical optics (1992–2002), J. Sci. Comput. 19 (2003) 63–93.
[3] P. Burchard, L.-T. Cheng, B. Merriman, S. Osher, Motion of curves in three spatial dimensions using a level set approach, J. Comput. Phys. 170 (2) (2001) 720–741.
[4] B. Engquist, O. Runborg, Computational high frequency wave propagation, in: Acta Numerica, Cambridge University Press, Cambridge, UK, 2003, pp. 1–86.
[5] B. Engquist, O. Runborg, A.-K. Tornberg, High frequency wave propagation by the segment projection method, J. Comput. Phys. 178 (2) (2002) 373–390.
[6] M. Falcone, R. Ferretti, Semi-lagrangian Schemes for Hamilton–Jacobi equations, discrete representation formulae and Godunov methods, J. Comput. Phys. 175 (2002) 559–575.
[7] S. Leung, J. Qian, S. Osher, A level set method for three-dimensional paraxial geometrical optics with multiple point sources, Commun. Math. Sci. 2 (4) (2004) 657–686.
[8] C. Min, Simplicial isosurfacing in arbitrary dimension and codimension, J. Comput. Phys. 190 (1) (2003) 295–310.
[9] C. Min, Local level set method in high dimension and codimension, J. Comput. Phys. 200 (1) (2004) 368–382.
[10] S. Osher, L.-T. Cheng, M. Kang, H. Shim, Y.-H. Tsai, Geometric optics in a phase space based level set and eulerian framework, J. Comput. Phys. 179 (2) (2002) 622–648.
[11] S. Osher, R. Fedkiw, Level set methods: an overview and some recent results, J. Comput. Phys. 169 (2001) 463–502.
[12] S. Osher, J.A. Sethian, Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 169 (1) (1988) 12–49.
[13] D. Peng, B. Merriman, S. Osher, H.K. Zhao, M. Kang, A PDE-based fast local level set method, J. Comput. Phys. 155 (2) (1999) 410–438.
[14] J. Strain, Semi-Lagrangian methods for level set equations, J. Comput. Phys. 151 (1999) 498–533.
[15] M. Sussman, P. Smereka, S. Osher, A level set method for computing solutions to incompressible two-phase flow, J. Comput. Phys. 114 (1994) 146–159.
[16] Y.-H. Tsai, L.-T. Cheng, P. Burchard, S. Osher, G. Sapiro, Dynamic visibility in an implicit framework, CAM Report 02-06, UCLA, 2002.

[17] V. Vinje, E. Iversen, K. Astebol, H. Gjøystdal, Estimation of multivalued arrivals in 3D models using wavefront construction – Part I, Geophys. Prospecting 44 (1996) 819–842.
[18] V. Vinje, E. Iversen, K. Astebol, H. Gjøystdal, Part II: Tracing and Interpolation, Geophys. Prospecting 44 (1996) 843–858.