ARTICLE IN PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics xxx (2005) xxx–xxx

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

Simplex free adaptive tree fast sweeping and evolution methods for solving level set equations in arbitrary dimension

Thomas C. Cecil [a,*], Stanley J. Osher [b], Jianliang Qian [b,1]

[a] *ICES, University of Texas, Austin, 1 University Station, C0200, Austin, TX 78712, United States*

[b] *Department of Mathematics, University of California, Los Angeles, 405 Hilgard Avenue, Los Angeles, CA 90095-1555, United States*

Received 9 May 2005; received in revised form 17 August 2005; accepted 18 August 2005

Abstract

We introduce simplex free adaptive tree numerical methods for solving static and time-dependent Hamilton–Jacobi equations arising in level set problems in arbitrary dimension. The data structure upon which our method is built in a generalized n-dimensional binary tree, but it does not require the complicated splitting of cubes into simplices (aka generalized n-dimensional triangles or hypertetrahedrons) that current tree-based methods require. It has enough simplicity that minor variants of standard numerical Hamiltonians developed for uniform grids can be applied, yielding consistent, monotone, convergent schemes. Combined with the fast sweeping strategy, the resulting tree-based methods are highly efficient and accurate. Thus, without changing more than a few lines of code when changing dimension, we have obtained results for calculations in up to $n = 7$ dimensions.

Keywords: Numerical methods; Binary trees; Adaptive methods; Level sets; Hamilton–Jacobi; Fast sweeping

1. Introduction

In this paper, we present a simplex free adaptive tree numerical method for solving static and time-dependent Hamilton–Jacobi partial differential equations (H–J PDEs) arising in level set problems in arbitrary dimension. The method's adaptivity increases resolution near the interface being studied, and simplifies previous successful tree-based implementations, allowing for its extension to arbitrary dimension without an increase in the complexity of function reconstruction, which is a necessary part of finding spatial derivatives needed in solving the PDEs.

Applications in higher dimensions requiring adaptive meshes to resolve fine details arise in numerous fields. In [23,14,8,30] multi-valued solutions to H–J equations were found by replacing a single-valued solution with

* Corresponding author. Tel.: +1 512 2327761.

E-mail addresses: tcecil@ices.utexas.edu (T.C. Cecil), sjo@math.ucla.edu (S.J. Osher), qian@math.ucla.edu, qian@math.wichita.edu (J. Qian).

[1] Present address: Department of Mathematics and Statistics, Wichita State University, Wichita, KS 67260-0033, United States.

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2005.08.020

the level set (or intersection of level sets) of a higher-dimensional function. This idea was also used in [5,7] to study interfaces with codimension > 1. In [28], the incompressible Euler equations were studied. In [33,34], a level set formulation was used to solve problems arising in mathematical finance, where high-dimensional issues are routinely encountered. Even for codimension-1 problems in 3D, there is still a desire to find implementable adaptive methods to resolve fine details, such as in the segmentation of the human brain, or other applications involving highly curved surfaces such as Wulff crystals. See [24,32] for a wide range of physical problems to which level set methods are applied.

Since the introduction of level set methods for interface tracking [22], there has been work done in an attempt to reduce the component of the computational portion of the method subject to the most criticism: the necessity of extra dimensions. Within a few years following [22] narrow band methods were proposed that reduced the computational complexity by resolving the level set function only near the interface being tracked [1,39,26]. These methods were able to use the well established, convergent, finite difference schemes available to uniform grids.

However, these narrow band methods did not reduce the storage requirements, limiting them to the same resolutions which uniform grids were restricted. Following this, tree-based methods were introduced, allowing for adaptivity of the mesh near the interface, while not sacrificing too much complexity [35,20,10,18]. The tree data structure used in these methods was well understood by the computer science community, and thus data storage and retrieval were able to be carried out in an efficient manner. However, the non-uniformity of the mesh required new schemes to be developed for the PDEs to be solved. In some cases semi-Lagrangian CIR [9] schemes were used for time-dependent level set equations. These schemes have some drawbacks, though. Firstly, they are only provably convergent for hyperbolic problems, and many level set PDEs involve mean curvature or are otherwise parabolic in nature. Secondly, they require a backtracking along characteristics and an interpolation at an arbitrary point within the domain. This interpolation is a delicate process that requires the division of the domain into simplices, which can become complicated in higher dimensions [20]. In [18], CIR was used for advection of values stored at cell corners, and cell centered data was stored for the pressure equation in Navier–Stokes, where a one-point (constant within each cell) interpolation technique was used to avoid apparent complexities, and to preserve the symmetry of the discretization. In addition, for the eikonal equation used to maintain the signed distance property of the level set function, [18] used some special treatment at T-junctions in the context of the fast marching spirit [38].

There have been other local level set methods [19,36,4,13,3] proposed which range from variants of AMR to using tubes of uniformly spaced grid points near the interface. Some of the methods approach the complexity of [26], eliminating the need to store the unused grid points away from the interface of interest. They also allow for the standard finite difference schemes to be used as the grid is uniform near the interface. However, with these gains comes additional complexity in implementation, and it should be noted that the successive improvements and acceptance of the tree-based methods in various applications are testaments to their facility and usefulness.

In this paper, we introduce a tree-based method that retains the advantages of the previous tree-based algorithms, such as having a well studied and understood data structure, while avoiding the drawbacks of having inconsistent schemes requiring n-dimensional simplices and interpolation. Thus we are able to use the standard numerical Hamiltonians derived for uniform grids (modified slightly) which result in consistent, monotone, convergent numerical methods. Combined with the fast sweeping strategy [41,37,15,29], the resulting tree-based methods are highly efficient and accurate.

The paper consists of a brief overview of the tree data structure, followed by a discussion of the numerical schemes for static H–J equations, and then time-dependent H–J equations. Finally, numerical results are given for codimension-1, codimension-2, and codimension-n problems.

2. Tree data structure

In this section, we describe the tree data structure used. We use a generalized binary tree (e.g., quadtree in 2D, octree in 3D, etc.) data structure, details of which can be found in numerous computer science texts [17,31,12]. We describe the portions of the implementation that are specific to our problem of solving a PDE in a bounded spatial domain.

We assume a computational domain, $\Omega = [0,1]^n$. At the kth level of the tree, each node, c, represents a hypercube cell with sides of length $dx_c = 1/2^k$, and center x_c. We assume that the level set function value, ϕ, is stored at the centers of mass of the nodes at the finest level of the tree, also known as the leaves. An alternate storage location could be the vertices of the cells. We choose the centers because of the simplicity of implementation (e.g., there are no storage points belonging to multiple cells), and the fact that local interpolations can be done in a dimension independent way.

When refinement of a cell is done, the cell is split into 2^n subcells with side lengths $1/2^{k+1}$. We do not allow any cells with side length ratio >2 or <0.5 to be neighbors. This restriction results in what is known as a balanced tree, see [21] for some results concerning tree balancing. This balancing can be obtained by following the criterion of refining any cell whose distance to the interface, Γ, is less than a constant times its edge length [35]. In practice, if we are using a single level set function ϕ, e.g. for codimension-1 problems, if ϕ is a signed distance function then we can set $\rho \geqslant (1 + \sqrt{n}/2)$ and refine if $|\phi(x_c)| < \rho dx_c$. For problems where the intersection of multiple level set functions, $\{\phi_j\}$, represents Γ, where the level sets of the ϕ_j are mutually orthogonal and each ϕ_j is a distance function measured along the level sets of the other $\phi_{i\neq j}$, we can compare $\|\phi\|_{l_2}$ to ρdx_c.

3. Static H–J equations

Here we introduce a fast sweeping implementation for solving certain static Hamilton–Jacobi equations such as the eikonal equation $|\nabla\phi| = f$ or in general $H(\nabla\phi) = f$ [37,40,41,16]. These types of equations are commonly used when reinitializing the level set function to be a signed distance function during a dynamic evolution, or in other weighted distance calculations that arise in numerous physical problems.

In order to avoid n-triangulations that can lead to very complicated local Hamiltonian solvers [29], we follow the same ideology that many other adaptive and tree-based methods follow: study a small number of local node configurations of the grid, and then appropriately scale them so that the various operations of interpolation, refinement, etc. can be applied in the same way anywhere in the domain.

3.1. Local numerical Hamiltonian solver

The first part of the fast sweeping method is the local numerical Hamiltonian solver. We will design monotone, consistent solvers that do not require n-triangulations.

3.1.1. Upwind Hamiltonian

The upwind Hamiltonians approximating the eikonal equation $|\nabla u| = f$ with boundary data given on Γ are based on using the Godunov Hamiltonian (GH):

$$\hat{H}\left(D_-^{x_1}u(y), D_+^{x_1}u(y), \ldots, D_-^{x_n}u(y), D_+^{x_n}u(y)\right) = \sqrt{\sum_{i=1}^{n} \max\left\{\left(D_-^{x_i}u(y)\right)^+, \left(D_+^{x_i}u(y)\right)^-\right\}^2}, \tag{1}$$

where y is a grid point at which we wish to update the numerical solution u, or using the Osher–Sethian Hamiltonian (OSH):

$$\hat{H}\left(D_-^{x_1}u(y), D_+^{x_1}u(y), \ldots, D_-^{x_n}u(y), D_+^{x_n}u(y)\right) = \sqrt{\sum_{i=1}^{n} \left[\left(D_-^{x_i}u(y)\right)^+\right]^2 + \left[\left(D_+^{x_i}u(y)\right)^-\right]^2}. \tag{2}$$

Godunov numerical Hamiltonians were evaluated in [2,25]. For nonlinear H–J equations whose Hamiltonians differ significantly from that of the eikonal equation the resulting expressions become quite complicated, involving many "if" statements.

When the grid is uniform along the ith axis, the standard 2-point finite difference can be used for, e.g. $D_-^{x_i}u(y)$, by taking $\frac{u(y)-u(y-\delta e_i)}{\delta}$, where δ is the local spacing between nodes in the ith direction.

Admittedly, the grid is not uniform everywhere, so when we try to compute quantities such as $D_-^{x_i}u(y)$ we will not have a uniform definition throughout all of the domain. However, because the grid refinement is done predictably (by this we mean that there are only a small number of local configurations of the grid, up to scaling), we can quickly find the value at an offset point in the $-e_i$ direction, $u(y - \delta e_i)$, needed in $D_-^{x_i}u(y)$.

We assume that the tree is constructed so that in each cell S, u_S is defined at the center, y_S, of S, and each cell is an n-cube with equal side lengths given by δ_S. This uniform restriction can be relaxed, but for expositional purposes it will not be. Also, we note the restriction that the ratio of the δ of neighboring cells is either 2, 1 or 1/2.

In any dimension, n, there are only three possible local configurations that need examining, when attempting to find $D^{x_i}_{-}u(y)$:

1. The cell B that is adjacent to cell $A \ni y$ in the $-e_i$ direction is exactly the same size as A, thus $u(y - \delta e_i) = u_B$, $\delta = \delta_A = \delta_B$. This is standard 2-point finite differencing.
2. The cell B that is adjacent to cell $A \ni y$ in the $-e_i$ direction is smaller than A, i.e. $\delta_B = \delta_A/2$. In this case, we have a situation illustrated in Fig. 1 in 2D. In n dimensions, we take $u(y - \delta e_i) = \{$the average of the 2^n adjacent neighboring cells whose faces with outward normal e_i are touching the face of A that has outward normal $-e_i\}$. As the centers of all these points are coplanar, this is just the linearly interpolated value at the point P that is the intersection of the plane containing these neighboring cell centers and the ray given in parametric form by $r(\tau) = y - \tau e_i$, $\tau \geqslant 0$.
3. The cell B that is adjacent to cell $A \ni y$ in the $-e_i$ direction is larger than A, i.e. $\delta_B = 2\delta_A$. In this case, y_B does not lie along $r(\tau)$. However, because of the structure of the grid points in the tree, there is a neighboring cell, C, of A such that $\overline{y_B y_C}$ intersects $r(\tau)$. This cell C is the diagonal neighbor of A in the direction

$$(\mathrm{sgn}(y_{A,1} - y_{B,1}), \ldots, \mathrm{sgn}(y_{A,i-1} - y_{B,i-1}), -\mathrm{sgn}(y_{A,i} - y_{B,i}), \mathrm{sgn}(y_{A,i+1} - y_{B,i+1}), \ldots, \mathrm{sgn}(y_{A,n} - y_{B,n})),$$

where sgn is the signum function.

In this case, there are two possibilities for C: *Case 1*. C is either the same size as A; *Case 2*. C is the same size as B. See Figs. 2 and 3 for diagrams of these cases in 2D and 3D.

The interpolated value at the intersection point P is

$$u(P) = \frac{1}{|\overline{y_B y_C}|}\left(u(y_B)|\overline{y_B P}| + u(y_C)|\overline{y_C P}|\right) = u(y_B)w_B + u(y_C)w_C. \tag{3}$$

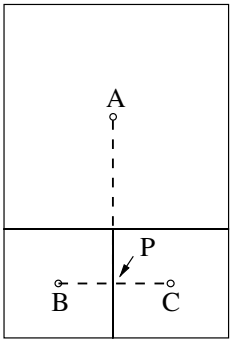

Fig. 1. Case where neighboring cells are smaller than A.

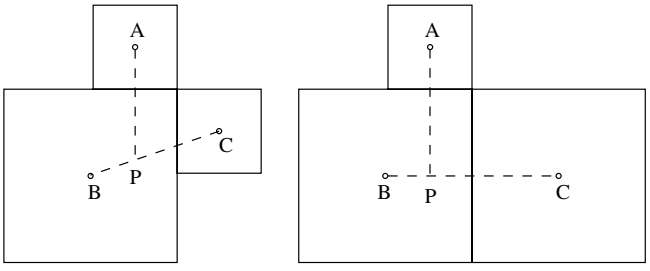

Fig. 2. Cases where neighboring cell B is larger than A in 2D. Left: Case 1, right: Case 2.

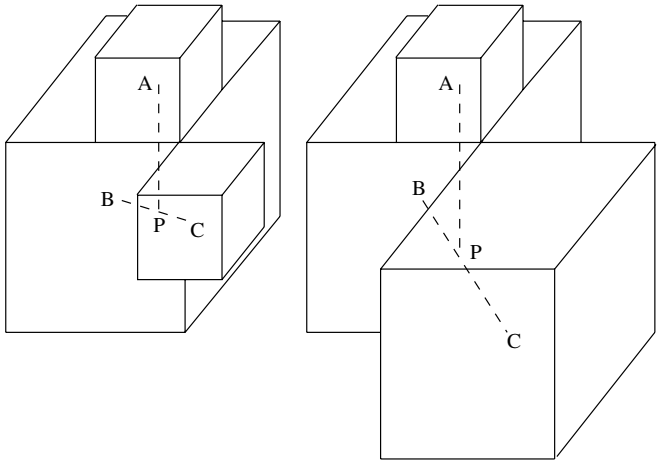

Fig. 3. Cases where neighboring cell B is larger than A in 3D. Left: Case 1. Right: Case 2.

Also, we have $\delta = w_B|y_{B,i} - y_{A,i}| + w_C|y_{C,i} - y_{A,i}|$. In Case 1, we find $w_B = 2/3$, $w_C = 1/3$, and in Case 2 we find $w_B = 3/4$, $w_C = 1/4$. These are the weights for any dimension n, which is very appealing in that we do not have to resort to complicated n-triangulations.

We define $\delta_{i,-} = \delta$ in this particular case because $y_{B,i} < y_{A,i}$. In the case, where $y_{B,i} > y_{A,i}$, $\delta_{i,+}$ is defined as the distance between P and A.

Once one has found all the $D^{x_i}_{\pm}u(y),\ \forall i$ one can solve the quadratic equation arising from the local numerical Hamiltonian for $u(y)$, and update the solution with this found value. Because the $\delta_{i,\pm}$ in each particular $D^{x_i}_{\pm}u(y)$ could be different, the Godunov solver introduced in [41] with its simple *min* and *if* statements is not applicable. However, the procedure for finding the correct solution of

$$\left[\frac{(x-a_1)^+}{h_1}\right]^2 + \cdots + \left[\frac{(x-a_m)^+}{h_m}\right]^2 = f \tag{4}$$

can be used. We present the case for OSH, as GH is more complicated (but feasible).

1. Let the a_j, $j = 1, \ldots, 2n$ be the points from the finite differences
   $$\{a_j\} = \{u(y_A \pm e_i\delta_{i,\pm})\}, \quad i = 1, \ldots, n,$$
   ordered from least to greatest, and the h_j be the corresponding offset distances from y_A (the h_j will not be in any particular order). We set $a_{2n+1} = \infty$.
2. Set $m = 1$;
3. Solve $\sum_{j=1}^{m}[\frac{(x-a_j)^+}{h_j}]^2 = f$ to get a solution $\hat{x}$.
4. Check to see if $\hat{x} \leqslant a_{m+1}$. If so, then we are done and we set $u(y) = \hat{x}$. If not, then we set $m \to m + 1$, and go to step 3, unless $m = 2n$, then we are done.

Note. The implementation of this solution algorithm is independent of n, except for the number of terms in the sum.

The algorithm for GH is more complicated as GH includes *max* functions that OSH does not. The only drawback of OSH is its slightly larger error near sonic shocks, but if we refine the grid such that it is locally uniform at sonic shocks, then GH could be used there (and anywhere else on the grid that is locally uniform), as it can be solved by the method presented in [41].

Note that monotonicity is satisfied because of the positive weights, w, multiplying the $u(z)$, $z \neq y$ in each finite difference. Also, because of the linear interpolations used, the scheme is consistent. Thus, the scheme is convergent.

3.1.2. Lax–Friedrichs Hamiltonian

Here, we present a Lax–Friedrichs Hamiltonian (LFH) which does not require nonlinear inversions when it is being solved. This extends its applicability to a wide range problems including those with non-convex Hamiltonians. This is a generalization of the Hamiltonian presented in [15].

The numerical Hamiltonian for $H(\nabla u) = f$ with boundary data given on Γ is as follows:

$$\begin{aligned}\hat{H}\left(D_-^{x_1}u(y), D_+^{x_1}u(y), \ldots, D_-^{x_n}u(y), D_+^{x_n}u(y)\right) \\ = H\left(w_1^- D_-^{x_1}u(y) + w_1^+ D_+^{x_1}u(y), \ldots, w_n^- D_-^{x_n}u(y) + w_n^+ D_+^{x_n}u(y)\right) - \sum_{i=1}^{n} \sigma_i \left(D_+^{x_i}u(y) - D_-^{x_i}u(y)\right),\end{aligned} \tag{5}$$

where

$$w_i^+ = \frac{\delta_{i,+}}{\delta_{i,+} + \delta_{i,-}}, \quad w_i^- = \frac{\delta_{i,-}}{\delta_{i,+} + \delta_{i,-}}. \tag{6}$$

Note that $w_i^+ + w_i^- = 1$ so the scheme is consistent.

To determine the size of σ_i, we note that monotonicity requires that

$$\partial\hat{H}/\partial p_i^+ \leqslant 0, \quad \partial\hat{H}/\partial p_i^- \geqslant 0.$$

This leads to the requirement

$$\sigma_i \geqslant |H_{p_i}| \max(w_i^+, w_i^-).$$

Within the fast sweeping framework, in order to advance the solution a single iteration, one writes (5) in the form

$$\hat{H} = L(u(\Omega \setminus y)) + cu(y),$$

and then the advancement can be written as

$$u^{n+1}(y) = \frac{f(y) - L(u^n(\Omega \setminus y))}{c}.$$

The weights w are composed precisely so that the coefficient c can be calculated in a linear way purely from the artificial diffusion term.

Implementation of this LFH is simpler than GH or OSH because it does not require the inversion of H. Thus it does not require any *if* statements and is thus faster per iteration. However, it does require more iterations to converge.

3.2. Sweeping directions

The second part of the fast sweeping method is to determine the directions of the sweep. In [29], a strategy using reference points was introduced, with an initial sequential ordering of the nodes with respect to their distances from these points. This could work for our method, but the built-in structure of the tree allows for another method of sweeping.

The tree specific sweeping method is as follows:

1. Each ordering of sweeping is defined by an ordering of the vertices of the n-cube. In n-dimensions, there are 2^n possible sweeps that are found by taking all combinations of sweeping from low to high, or high to low in each dimension. So in 2D if the vertices of a square are as in Fig. 4 the possible orderings are

   $\{A, B, C, D\}$,
   $\{D, C, B, A\}$,
   $\{C, D, A, B\}$,
   $\{B, A, D, C\}$.

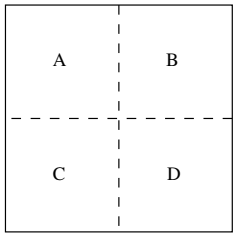

Fig. 4. Sample child ordering in 2D. The outer perimeter is the boundary of the parent cell. Each lettered interior square corresponds to a child pointer to one of the 4 smaller squares.

2. For each of the orderings in step 1, call a preorder traversal [17] of the grid starting at the root node, based on a particular ordering of the children given from the previous step. When a leaf is reached, update the solution using the local Hamiltonian solver.

This means that if we choose the child ordering $\{A, B, C, D\}$ then we call a preorder traversal with node A as the starting node, followed by a preorder traversal with node B as the starting node, etc. Once the traversal has gotten to the leaf of the tree (i.e. a node with no children) we update u. This recursive type of tree visitation is standard and can be found in any thorough book on computer algorithms and binary trees [17,31].

Fig. 5 shows a sample ordering of the nodes when all nodes are at a uniform depth in the tree. In this particular case, the sweeping algorithm will give exactly the same result as the standard sweep ordering [41] for a uniform grid of this size with a fixed node at the origin with $u(0,0) = 0$. This is because for each node visited in this sweep, all nodes to the south and west of it have already been updated in the sweep.

However, when the grid is not uniform we have found that extra sweeps are needed as $D^{x_i}_{-}u(y)$ may depend on nodes that are farther north or east than y, as is the case when the stencil choices in Fig. 2 would be used. We make some comments on this. Firstly, since the sweeping strategy accesses all nodes systematically, the tree-based methods will converge eventually, no matter how many sweeps it needs. Secondly, since there are many more possible information flowing directions on a non-uniform grid as demonstrated in [29], and the tree specific sweeping method designed here will only allow a finite number of such information flowing directions to be treated simultaneously, it is reasonable for the tree specific methods to use extra sweeps to converge. Thirdly, it is possible to design optimal tree specific sweeping methods by reordering the nodes so that all the possible information flowing directions can be covered with a finite number of tree-based orderings.

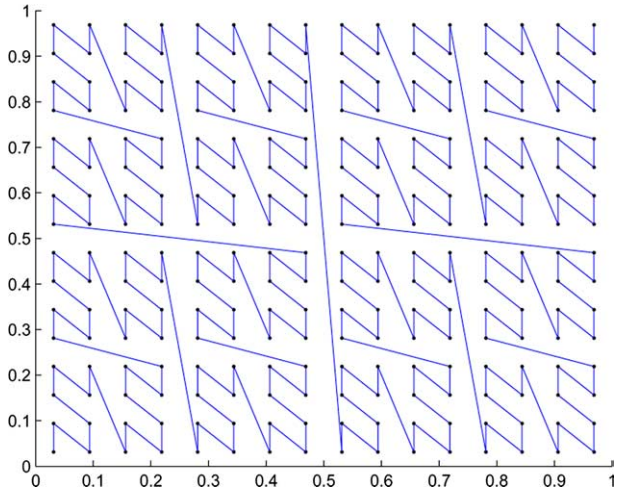

Fig. 5. Nodes that are leaves of a tree on a uniform 16×16 grid, and the path connecting sequential nodes in the sweep from bottom left to top right.

3.3. Solution procedure

The complete solution procedure is as follows:

1. Assume that we are given a grid upon which the solution ϕ will be solved. Find all points within a small O(dx) distance of Γ and find an approximate solution ϕ^n at these points. This can be done by interpolating the known boundary Γ. Set all other points to a large value, e.g. $\phi^n = 10^{10}$, which is larger than the exact solution on the grid.
2. Sweep through the domain $\{x_j\}$ using a preorder traversal specified by one of the 2^n orderings of child pointers, solving for $\phi^*(x_j)$ at each grid point using Gauss–Siedel iteration by inverting GH, OSH or LFH. Take $\phi^{n+1} = \min(\phi^*(x_j), \phi^n(x_j))$.
3. Check if $\|\phi^n - \phi^{n+1}\|_\infty < \epsilon$, where ϵ is a fixed tolerance to indicate convergence has been reached. If convergence has not been reached go to step 2.

4. Time-dependent level set equations

This section concerns solving time-dependent H–J equations as well as higher order parabolic equations such as mean curvature motion that arise frequently in level set problems. We will introduce the way in which: (1) the numerical Hamiltonian is constructed and (2) new values are chosen on the grid in a monotone way after a refinement/coarsening has taken place.

4.1. Numerical Hamiltonian

Examining the constructions for the functions at the points $y \pm \delta e_i$ from the section on static H–J equations we can see that 2-point upwind differences can be calculated easily for any fine grid point y. For example,

$$D_-^{x_i} u(y) = \frac{u(y) - (u(y_B) w_B + u(y_C) w_C)}{\delta}, \tag{7}$$

using the notation from (3). Then standard monotone numerical Hamiltonians such as GH, OSH and LFH can be used with Runge–Kutta (R–K) timestepping with the CFL condition determined by the finest cell size being used. For higher order WENO type reconstructions, a bit more work would be involved, but they are certainly possible to construct and we would still avoid the need to use high dimensional triangulations. Central differences and higher order derivatives can be calculated by using the weightings, w, that were introduced in Section 3.1.2 for Lax–Friedrichs Hamiltonians after calculating the first-order upwind differences at the grid points.

4.2. Interpolation after refinement/coarsening

We would like the interpolation processes to be consistent and monotone as well so that our entire scheme including adaptation is stable. Also, we would like to avoid having to work with any high dimensional simplices. It turns out that as was the case with the upwind differencing, we have only a few different cases that can be applied to all dimensions in the same way.

4.2.1. Coarsening

After a coarsening the value at the center of the node of size $1/2^k$ that just became a leaf is set to be the average of the 2^n function values at the nodes with side lengths $1/2^{k+1}$ that were its children. Coarsening is done if all the siblings of a cell c are at the same level as c and have not been marked for refinement, and if the resulting averaged coarsened value, $\phi^{\text{coarsened}}$, does not violate the refinement condition (i.e. we do not have $|\phi^{\text{coarsened}}| < 2\rho dx_c$).

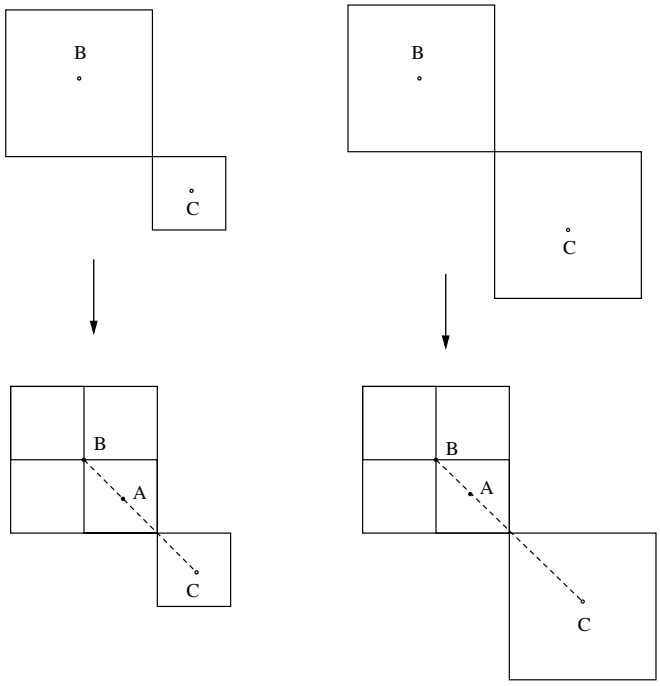

Fig. 6. Possible refinement cases in 2D. (A) Node where interpolation is evaluated. (B) Node at center of coarse cell that was divided. (C) Diagonal neighbor used in interpolation. Left: Case 1. Right: Case 2.

4.2.2. Refinement

Examining Fig. 6, we can see that after refinement we can use 2-point linear interpolation to find the new value at point A with weights: in case 1, $w_B = 2/3$, $w_C = 1/3$; and in case 2, $w_B = 3/4$, $w_C = 1/4$. These are the weights for any dimension n, where the diagonal neighbor, C, is the adjacent cell in the direction

$$(\mathrm{sgn}(y_{A,1} - y_{B,1}), \ldots, \mathrm{sgn}(y_{A,n} - y_{B,n})).$$

Both the coarsening and refinement interpolations are monotone and consistent for linear functions ϕ.

4.3. Solution procedure

The complete solution procedure for the time-dependent problem is as follows:

1. Assume that we are given a grid upon which the solution ϕ will be determined, and we are given initial conditions ϕ^n. If ϕ^n is not a signed distance function then use the fast sweeping method with OSH or GH to reinitialize the solution.
2. For each time step, for every grid point, advance the time-dependent H–J equation forward one time step.
3. Reinitialize the solution using fast sweeping.
4. Refine the solution where necessary.
5. Coarsen the solution where necessary.
6. Go to step 2.

It should be noted that the reinitialization/refinement/coarsening procedures do not need to be carried out every time step, but they should be done at least once every few time steps.

5. Numerical results

In this section, numerical results are presented for both static and time-dependent problems for codimension-1, codimension-2, and codimension-n problems. We refrain from studying the reduced memory

requirements that the tree method gains over uniform discretizations as these are well presented in [20]. There it is shown that for interface evolutions with codimension > 1 one finds storage requirements increasing at a rate proportional to the dimension of the interface, not the dimension of the computational domain (i.e. for a dimension d interface the storage increases by a factor of approximately 2^d when the globally minimal dx is halved, no matter how large the codimension of the interface is). Time reduction is also indicated in [20], but the reduction is less significant (increasing by approximately 2^{2d} when the minimal dx is halved) than for memory because of the added complexity of accessing neighboring cells, as well as the extra time needed in adapting the mesh. We focus on presenting convergence rate estimates and show error results in dimensions up to $n = 7$. For the time-dependent problems, we use forward Euler time advancement. The n-dimensional midpoint quadrature rule is used to calculate the L_p errors throughout. For the time-dependent problems, we set the refinement radius ρ as described in Section 2 to be $\rho = 1.5\sqrt{n}$.

In Table 1, we show errors and node counts for a codimension-n problem of solving the eikonal equation

$$|\nabla\phi| = 1, \tag{8}$$

with a boundary point at $x_b = (0.5, 0.5, \ldots, 0.5)$ with value $u(x_b) = 0$. In this table, the number of leaves represents the number of points where the function value is stored, while the number of uniform points represents what this number would be if a uniform grid with the finest dx listed was used. We note that the adaptivity for this problem is based on knowledge of the distance to the fixed node at the center of the domain, which is essentially knowledge of the solution to (8). Thus a better adaptivity routine needs to be formulated, which will be the subject of future research.

We use the preorder traversal fast sweeping method with the OSH numerical Hamiltonian. The error is measured on the finest 3 levels of leaves, except in 7D, where it is measured on the finest 2 levels of leaves.

An interesting point to note is that when a stopping criterion is used such as stopping when the change in the error is $<10^{-12}$ we find that the number of sweeps needed to achieve this criterion does not increase linearly in n, but rather stagnates. For example in 5D, we need only 30 sweeps, in 6D we need 35, and in 7D only 32. Perhaps this is some effect particular to our specific example, but perhaps not.

In Table 2, we show errors and convergence rate estimates for the time-dependent H–J equation

$$\phi_t + |\nabla\phi| = 0, \tag{9}$$

with initial condition given by a hypersphere of radius 0.2 centered at the point $x_b = (0.5, 0.5, \ldots, 0.5)$. The normal motion moves the hypersphere inwards towards x_b with constant normal velocity $= 1$. The error is measured by determining the volume of the hypersphere at the final time, T, (which is taken to be near 0.1) and then measuring the error in the implied radius versus the exact radius $= 0.2 - T$. This is basically an L_1 error estimate.

In Table 3, we show errors and convergence rate estimates for time-dependent motion by mean curvature

$$\phi_t + \kappa|\nabla\phi| = 0, \tag{10}$$

where

$$\kappa \equiv -\left[\sum_{i=1}^{n} \phi_{x_i x_i}\left(\sum_{\substack{j=1 \\ j\neq i}}^{n} \phi_{x_j}^2\right) - \sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} \phi_{x_i x_j}\phi_{x_i}\phi_{x_j}\right] \Bigg/ |\nabla\phi|^3$$

Table 1
Errors for eikonal equation

Finest dx	L_1 error	L_∞ error	N leaves of tree	N uniform points	n
1/256	6.046E − 05	7.193E − 03	304	65,536	2
1/256	1.293E − 05	1.126E − 02	2472	16,777,216	3
1/256	3.056E − 06	1.615E − 02	26,896	4.295E + 09	4
1/256	7.209E − 07	2.027E − 02	309,536	1.100E + 12	5
1/128	2.018E − 05	4.802E − 02	2,822,464	4.398E + 12	6
1/16	1.607E − 01	2.307E − 01	9,379,840	2.684E + 08	7

Table 2
Convergence rate estimate for constant motion in normal direction

Finest dx	Error	Rate	Dimension
1/32	9.605E − 03	–	3
1/64	4.376E − 03	1.134	3
1/128	2.110E − 03	1.052	3
1/256	1.263E − 03	0.741	3
1/32	8.201E − 03	–	4
1/64	4.147E − 03	0.984	4
1/128	2.138E − 03	0.956	4
1/16	3.220E − 02	–	5
1/32	1.112E − 02	1.535	5
1/16	1.743E − 02	–	6

Table 3
Convergence rate estimate for mean curvature motion

Finest dx	Error	Rate	Dimension
1/32	1.821E − 03	–	2
1/64	6.011E − 04	1.599	2
1/128	1.581E − 04	1.927	2
1/256	6.457E − 06	4.614	2
1/32	1.786E − 03	–	3
1/64	3.217E − 04	2.473	3
1/128	1.951E − 05	4.044	3
1/16	2.114E − 02	–	4
1/32	1.482E − 02	0.513	4
1/64	7.155E − 03	1.051	4

is the scaled mean curvature of the set $\Gamma = \{x|\phi(x)\} = 0$. We initialize Γ as a hypersphere of radius 0.2 centered at the point $x_b = (0.5, 0.5, \ldots, 0.5)$. The first partial derivatives of κ are found using centered differences as suggested above with the weights w indicated in the section on the LFH, and the second derivatives are found using 3-point centered stencils.

The mean curvature motion moves the hypersphere inwards towards x_b with normal velocity $= (n-1)/r(t)$, where $r(t)$ is the radius of the hypersphere at time t. The exact solution is a hypersphere with $r(t) = \sqrt{0.2^2 - 2(n-1)t}$, centered at x_b. The final time varies from dimension to dimension but we take $O(1/dx_{\text{global min}})$ number of time steps for the coarsest grid in each dimension's convergence study. The error is measured in the same way as it was for the normal motion case.

In Table 4, we show error convergence rates for a codimension-2 problem of advection of a closed curve in 3D [5]. We advance

$$(\phi_j)_t + V \cdot \nabla \phi_j = 0, \quad V = (1,1,1), \tag{11}$$

for $j = 1,2$, where the initial condition is given by a circle of radius 0.2 centered at $x_b = (0.25, 0.25, 0.25)$. This circle is defined by the intersection of the 0 level sets of 2 level set functions, ϕ_1, ϕ_2. The final time is $T = 0.3125$. The exact solution is a circle centered at $(0.5625, 0.5625, 0.5625)$ with radius $= 0.2$. For this problem, it is necessary to set the ϕ_j to be orthogonal to each other every few time steps. This is done in a fast sweeping way as presented in [6]. The error is measured at $t = T$ by sampling the circle that is the exact solution with 500 points, $\{y\}$, and estimating the L_1 error of $\sqrt{\phi_1(y)^2 + \phi_2(y)^2}$ on the circle.

Here, we include a short discussion of the gain in time and memory for the codimension-2 problem. In Fig. 7, we show the cell centers used when $t = 0$ and the finest $dx = 1/128$, along with the circle that is being tracked. For this figure, the total number of nodes used in the entire tree is $N = 34{,}929$, while the total number of leaves is $L = 30{,}563$. Thus the number of pointers in the tree is $9N - 8L = 69{,}857$. The total number of nodes for a uniform grid would be $N_{\text{unif}} = (2^7)^3 = 2{,}097{,}152$. For a tree storing only the function values

Table 4
Convergence rate estimate for motion of a closed curve in 3D

Finest dx	Error	Rate
1/16	4.520E − 02	–
1/32	2.544E − 02	0.829
1/64	1.372E − 02	0.891
1/128	7.261E − 03	0.918

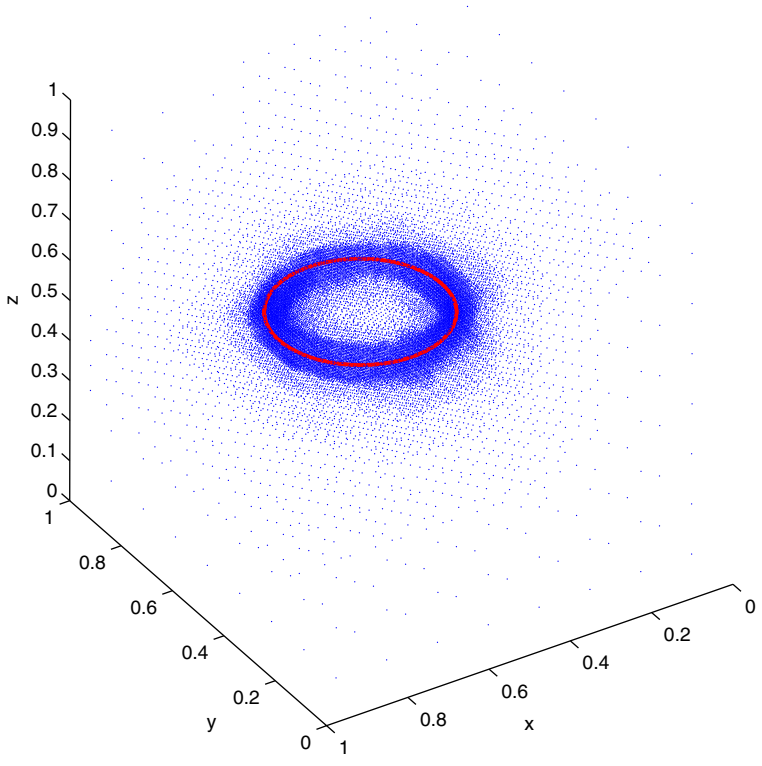

Fig. 7. Cell centers (blue) and codimension-2 circular interface (red) at $t = 0$ when the finest $dx = 1/128$. (For interpretation of the references in color in this figure legend, the reader is referred to the web version of this article.)

and pointers to child and parent nodes the memory reduction is apparent, and even if physical locations were to be stored at the nodes there would still be a significant reduction. In terms of time if we assume that the amount of time needed for accessing a neighbor on a uniform grid is K, then we can bound the time needed for accessing a neighbor in the tree by $K\log_{2^n} N_{\text{unif}}$ (although most neighbor accesses are done in an amount of time closer to K because of the moving pointer technique [20]). Thus the time needed to get a neighbor at every node (proportional to the time needed to advance one timestep) would be less than $7KL \approx 2K \times 10^5$ for the tree, versus approximately $N_{\text{unif}}K \approx 2K \times 10^6$ for the uniform grid. The overall time would need to include the time needed to refine the grid, which should be similar to the that needed by the advancement step, thus yielding a faster method by at least a factor of 5 if adaptation is done every time step.

In Fig. 8, we show contour plots of Wulff shapes [27] arising from solving

$$\left(\sqrt{p^2+q^2+r^2}+2|r|\right)\left(1+3\sqrt{\frac{(p^2+q^2)^{3/2}-(3p^2q-q^3)}{2(p^2+q^2)^{3/2}}}\right)=1, \tag{12}$$

where $(p,q,r) = (\phi_x, \phi_y, \phi_z)$. This equation is equivalent to solving

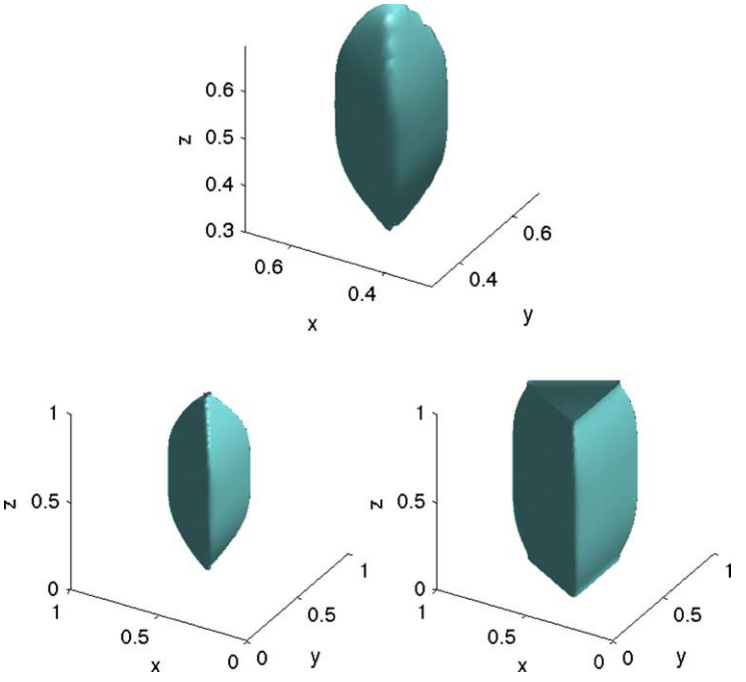

Fig. 8. Wulff crystal with surface tension $\gamma(\nabla\phi/|\nabla\phi|) = (1 + 2|\sin\theta_1|)(1 + |\sin(1.5(\theta_2 + 0.5\pi))|)$. Contours on top, lower left, lower right at $\phi = 0.08, 0.14, 0.2$.

$$\gamma\left(\frac{\nabla\phi}{|\nabla\phi|}\right)|\nabla\phi| = (1 + 2|\sin\theta_1|)(1 + |\sin(1.5(\theta_2 + 0.5\pi))|)|\nabla\phi| = 1,$$

where γ is known as the surface tension in materials science, and is a function of the spherical coordinates (θ_1, θ_2). For this and the next example, we use fast sweeping with LFH. The diffusion terms satisfy $\sigma = (9/4, 9/4, 27/8)$. We fix a point in the center of the domain with the value 0. The grid is adapted similarly to how it would be for a usual level set evolution, but with the coarsest resolution being $dx = 1/64$, and the finest resolution, $dx = 1/1024$. Neumann BCs $\phi_\eta = 0$ are used. The total number of iterations needed for the maximum change in the solution in subsequent iterations to be reduced to $<10^{-6}$ in this example is 836. For visualization purposes, the final solution is interpolated to a uniform 64^3 grid for this and the next two figures.

In Fig. 9, we show contour plots of Wulff shapes arising from solving

$$\left(\sqrt{p^2+q^2+r^2} + \sqrt{2\sqrt{p^2+q^2+r^2}|\sqrt{3}|r| - \sqrt{p^2+q^2}|}\right)\left(1 + \sqrt{\frac{(p^2+q^2)^{5/2} - (-5p^4q + 10p^2q^3 - q^5)}{2(p^2+q^2)^{5/2}}}\right) = 1. \tag{13}$$

This equation is equivalent to solving

$$\gamma\left(\frac{\nabla\phi}{|\nabla\phi|}\right)|\nabla\phi| = (1 + 2\sqrt{\sin(|\theta_1| - 0.5\pi)})(1 + |\sin(2.5(\theta_2 + 0.5\pi))|)|\nabla\phi| = 1.$$

The diffusion terms satisfy $\sigma = (7/4, 7/4, 2)$. The total number of iterations needed for the maximum change in the solution in subsequent iterations to be reduced to $<10^{-6}$ is 429.

For these last 2 examples, the adaptation strategy is based on each point's l_2 distance to the fixed point, which is not the optimal refinement strategy for minimizing the global error in most norms. Thus there remains work to do in determining better adaptation strategies. However, the point of these examples is to show the convergence of the fast sweeping method using the LFH in a number of sweeps comparable to the number found in [15] for the same examples. In [15] 2032 sweeps are needed for Fig. 8, and 432 sweeps are needed for Fig. 9. See [15] for more details on these Wulff crystal problems.

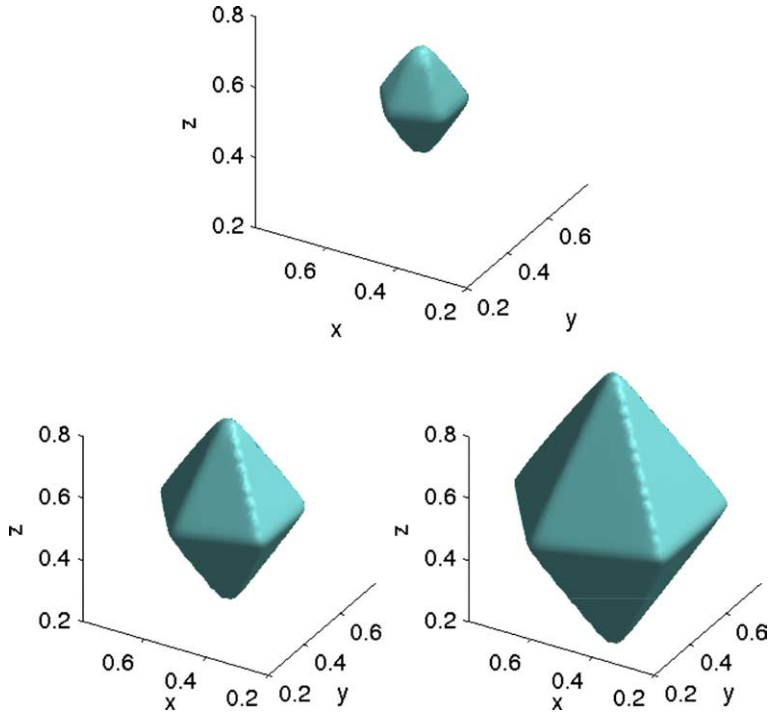

Fig. 9. Wulff crystal with surface tension $\gamma(\frac{\nabla\phi}{|\nabla\phi|}) = (1 + 2\sqrt{\sin(|\theta_1| - 0.5\pi)})(1 + |\sin(2.5(\theta_2 + 0.5\pi))|)$. Contours on top, lower left, lower right at $\phi = 0.08$, 0.14, 0.2.

In Fig. 10, we show the evolution of a crystal whose asymptote is a Wulff shape that is a cube. The PDE being solved is

$$\phi_t + |\phi_x| + |\phi_y| + |\phi_z| = 0, \tag{14}$$

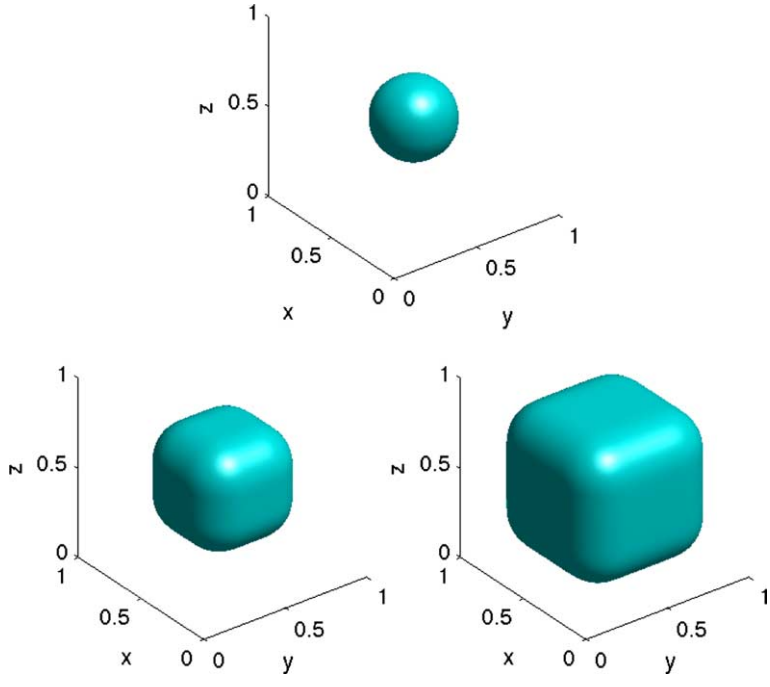

Fig. 10. Wulff shape evolution of $\phi_t + |\phi_x| + |\phi_y| + |\phi_z| = 0$ at times $t = 0$ (top), $t = 0.098$ (lower left), and $t = 0.195$ (lower right).

with initial conditions of a sphere with radius = 0.2. This is an example where the monotone Hamiltonians (OSH is used here) have an advantage over the simple characteristic following CIR scheme.

6. Conclusion

We have introduced a tree-based adaptive method for solving level set equations which has the advantage of being simplex free. Its implementation changes very little from dimension to dimension, allowing use by even the most hyperspatially challenged practitioner. It allows for well studied monotone numerical Hamiltonians to be used, avoiding the complications that often arise when constructing monotone numerical Hamiltonians on unstructured grids. The method has been applied to codimension-m, $1 \leqslant m \leqslant n$, linear and nonlinear, first- and second-order, static and time-dependent H–J problems in up to 7D.

Future work will include improving the adaptive procedure for static problems, based on user defined global errors. Also, WENO type methods will be explored to increase the accuracy. Although we do not show numerical examples here, the method could be extended to higher (>2) order nonlinear PDEs such as Willmore flows [11]. Finally, it should be noted that although the methods presented are done so for H–J problems, they could be applied to other problems requiring adaptive meshes and function interpolation and reconstruction.

Acknowledgments

The authors were partially supported by an ONR MURI Grant N00014-02-1-0720. The third author was partially supported by NSF DMS-0542174.

References

[1] D. Adalsteinsson, J.A. Sethian, A fast level set method for propagating interfaces, J. Comput. Phys. 118 (2) (1995) 269–277.
[2] M. Bardi, S. Osher, The nonconvex multidimensional Riemann problem for Hamilton–Jacobi equations, SIAM J. Math. Anal. 22 (2) (1991) 344–351.
[3] D. Breen, R. Fedkiw, K. Museth, S. Osher, G. Sapiro, R. Whitaker, Level sets and pde methods for computer graphics, in: ACM SIGGRAPH '04 COURSE 27, 2004.
[4] R. Bridson, Computational aspects of dynamic surfaces. Ph.D. thesis, Stanford Univ., 2003.
[5] 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.
[6] T. Cecil, D. Marthaler, A variational approach to path planning in three dimensions using level set methods, UCLA CAM Report (04-74) (2004).
[7] L.-T. Cheng, P. Burchard, B. Merriman, S. Osher, Motion of curves constrained on surfaces using a level-set approach, J. Comput. Phys. 175 (2) (2002) 604–644.
[8] L.-T. Cheng, H. Liu, S. Osher, Computational high-frequency wave propagation using the level set method, with applications to the semi-classical limit of Schrödinger equations, Commun. Math. Sci. 1 (3) (2003) 593–621.
[9] R. Courant, E. Isaacson, M. Rees, On the solution of nonlinear hyperbolic differential equations by finite differences, Comm. Pure Appl. Math. 5 (1952) 243–255.
[10] M. Droske, B. Meyer, M. Rumpf, C. Schaller, An adaptive level set method for medical image segmentation, Lect. Notes Comput. Sci. (2001) 412–422.
[11] M. Droske, M. Rumpf, A level set formulation for Willmore flow, Interf. Free Bound. 6 (3) (2004) 361–378.
[12] S.F. Frisken, R.N. Perry, Simple and efficient traversal methods for quadtrees and octrees, J. Graphics Tools 7 (3) (2002) 1–11.
[13] B. Houston, M. Wiebe, C. Batty, RLE sparse level sets, in: Proceedings of the SIGGRAPH 2004 Conference on Sketches & Applications, ACM Press, 2004, p. 1.
[14] S. Jin, S. Osher, A level set method for the computation of multivalued solutions to quasi-linear hyperbolic PDEs and Hamilton–Jacobi equations, Commun. Math. Sci. 1 (3) (2003) 575–591.
[15] C.-Y. Kao, S. Osher, J. Qian, Lax–Friedrichs sweeping scheme for static Hamilton–Jacobi equations, J. Comput. Phys. 196 (1) (2004) 367–391.
[16] C.-Y. Kao, S. Osher, Y.-H, Tsai. Fast sweeping methods for static Hamilton–Jacobi equations, UCLA CAM Report, (03-75), (2003).
[17] D.E. Knuth, The Art of Computer Programming, third ed.Fundamental Algorithms, vol. 1, Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1997.
[18] F. Losasso, F. Gibou, R. Fedkiw, Simulating water and smoke with an octree data structure, ACM Trans. Graph. 23 (3) (2004) 457–462.
[19] R.B. Milne, An adaptive level set method. Ph.D. Thesis, Berkeley National Lab., Phys. Division, Math. Dept., 2003.

[20] Chohong Min, Local level set method in high dimension and codimension, J. Comput. Phys. 200 (1) (2004) 368–382.
[21] D. Moore, The cost of balancing generalized quadtrees, in: SMA '95: Proceedings of the third ACM Symposium on Solid Modeling and Applications, ACM Press, New York, NY, USA, 1995, pp. 305–312.
[22] S.J. Osher, J.A. Sethian, Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1988) 12–49.
[23] 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.
[24] S. Osher, P. Ronald Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Oxford Univ. Press, 2002.
[25] S. Osher, C.-W. Shu, High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations, SIAM J. Numer. Anal. 28 (4) (1991) 907–922.
[26] D. Peng, B. Merriman, S. Osher, H. Zhao, M. Kang, A PDE-based fast local level set method, J. Comput. Phys. 155 (2) (1999) 410–438.
[27] D. Peng, S. Osher, B. Merriman, H.-K. Zhao, The geometry of Wulff crystal shapes and its relations with Riemann problems, in: Nonlinear Partial Differential Equations (Evanston, IL, 1998), Contemp. Math., vol. 238, American Mathematical Society, Providence, RI, 1999, pp. 251–303.
[28] S. Popinet, Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries, J. Comput. Phys. 190 (2) (2003) 572–600.
[29] J. Qian, Y.-T. Zhang, H. Zhao, Fast sweeping methods for eikonal equations on triangulated domains, UCLA CAM Report (05-07), 2005.
[30] J. Qian, L.-T. Cheng, S. Osher, A level set-based Eulerian approach for anisotropic wave propagation, Wave Motion 37 (4) (2003) 365–379.
[31] Hanan Samet, Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS, Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA, 1990.
[32] J.A. Sethian, Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science, in: Level Set Methods and Fast Marching Methods, second ed., Cambridge Monographs on Applied and Computational Mathematics, vol. 3, Cambridge University Press, Cambridge, 1999.
[33] H. Mete Soner, Nizar Touzi, Superreplication under gamma constraints, SIAM J. Control Optim. 39 (1) (2000) 73–96, electronic.
[34] H. Mete Soner, N. Touzi, A stochastic representation for the level set equations, Comm. Partial Differential Equations 27 (9–10) (2002) 2031–2053.
[35] J. Strain, Tree methods for moving interfaces, J. Comput. Phys. 151 (2) (1999) 616–648.
[36] M. Sussman, A.S. Almgren, J.B. Bell, P. Colella, L.H. Howell, M.L. Welcome, An adaptive level set approach for incompressible two-phase flows, J. Comput. Phys. 148 (1) (1999) 81–124.
[37] Y.-H.R. Tsai, L.-T. Cheng, S. Osher, H.-K. Zhao, Fast sweeping algorithms for a class of Hamilton–Jacobi equations, SIAM J. Numer. Anal. 41 (2) (2003) 673–694, electronic.
[38] J.N. Tsitsiklis, Efficient algorithms for globally optimal trajectories, IEEE Trans. Automat. Control 40 (9) (1995) 1528–1538.
[39] R.T. Whitaker, A level-set approach to 3d reconstruction from range data, Int. J. Comput. Vision 29 (3) (1998) 203–231.
[40] H.K. Zhao, S. Osher, B. Merriman, M. Kang, Implicit and non-parametric shape reconstruction from unorganized points using variational level set method, Comp. Vis. Image Under. 80 (2000) 295–319.
[41] Hongkai Zhao, A fast sweeping method for eikonal equations, Math. Comp. 74 (250) (2005) 603–627 (electronic).