University of California
Los Angeles

# A Computational Framework Tracking a Moving Interface in Arbitrary Dimension and Codimension

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Mathematics

by

**Chohong Min**

2004

The dissertation of Chohong Min is approved.

Stefano Soatto

Russel Caflisch

Chris Anderson

Stanley Osher, Committee Chair

University of California, Los Angeles

2004

*To my wife, Meehwa Oh,*
*who has supported me with love*

# Table of Contents

# List of Figures

# List of Tables

# Acknowledgments

I was so fortunate to have wonderful colleagues as Chiu-Yen Kao, Thomas Cecil, Yen-Hsi Tsai and YonSeo Kim. Discussions with them often helped me to resolve a clutter of ideas. Prof. Mark Green has been of great help whenever I have met problems related to the algebraic geometry. I am indebted to Prof. David Ebert and Chris Weigle for their amazing kindness. They reviewed my isosurfacing paper very carefully and suggested a better way.

Mostly, I would like to thank my advisor, Prof. Stanley Osher for his enourmous encouragement and excellent guidance.

# Vita

| | |
|---|---|
| 1972 | Born in Anyang, Korea. |
| 1990–1997 | B.S. (Mathematics), Korea Advanced Institute of Scicence and Technology. |
| 1994–1996 | Communication Engineer, Korea Air Force. |
| 1998–1999 | Technical Staff, Samsung Data System, Korea. |
| 1999–2000 | Technical Staff, ObjectSoft, Korea. |
| 1999–2000 | Technical Director, VentureBrain, Korea. |
| 2001–2002 | Teaching Assistant, Mathematics Department, UCLA. |
| 2002–present | Research Assistant, Mathematics Department, UCLA. |

# Publications

Chohong Min, *Simplicial Isosurfacing in Arbitrary Dimension and Codimension* Journal of Computational Physics 190:295-310 (2003)

Chohong Min, *Local Level Set Method in High Dimension and Codimension* Journal of Computational Physics (accepted in Apr 14, 2004)

Abstract of the Dissertation

# A Computational Framework Tracking a Moving Interface in Arbitrary Dimension and Codimension

by

**Chohong Min**
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2004
Professor Stanley Osher, Chair

The level set method was originally introduced to track a moving interface of codimension one, and recently generalized to arbitrary codimension. The generalized level set method represents an interface as the level set of a vector valued function, and invoked two computational problems. One is to construct and visualize the level set of a vector valued function. The other is to efficiently advance the vector valued function in high dimension

The purpose of this article is to introduce a unified theory and efficient numerical algorithms solving the two computational problems. The numerical methods constitutes a computational framework for tracking a moving interface in arbitrary dimension and codimension

# CHAPTER 1

# Introduction

Recently the level set method was utilized to track the propagation of high frequency waves. By embedding the wave front into the phase space, the method enabled the tracking of a wave propagation with superpositions or with complicated geometries, just by solving a linear advection equation. However it invoked two computational problems of isosurfacing and local level set method.

The subsequent two chapters introduce a unified theory and an efficient algorithm of isosurfacing and local level set method. They were written not only for the wave propagation but for a general setting that can track a moving interface in arbitrary dimension and codimension.

## 1.1 High Frequency Wave Propagation

The numerical simulation of high frequency has been an important topic in many areas that include geometric optics [OCK02, ERT02], seismic wave [QCO03] and semi-classical Schrodinger equation [CLO03, ER03]. Since the superposition is typical in wave propagations, the governing equations of the wave phenomenon take the form of linear partial differential equation. Although numerical methods for linear partial differential equations have been fully developed, it is still challenging to simulate the linear equation directly. According to the Nyquist's theorem [Nyq28], the sampling frequency should be at least two times bigger than the wave frequency to numerically resolve the wave properly, which forces sampling $\frac{2}{7} \cdot 10^7$ points in 1 *meter* to resolve the red light of $700\,nm$ wavelength.

To reduce this enormous computational cost, the linear equations are asymptotically approximated to a Hamilton-Jacobi type equation ;

$$S_t + H(\mathbf{x}, \nabla S) = 0$$

For the wave equation, $u_{tt} = \Delta u$ and the Schrodinger equation, $i\psi_t = \Delta\psi + V\psi$, their corresponding Hamiltonians, $H(\mathbf{x}, \mathbf{p})$ are $|\mathbf{p}|$ and $V(\mathbf{x}) + \frac{|\mathbf{p}|^2}{2}$, respectively [Whi74, CLO03]. Since two colliding wave fronts cross ignoring each other, the wavefront propagation is not the viscosity solution, but the characteristic so-

lution of the Hamilton-Jacobi equation. The characteristic solution satisfies the following ordinary differential equation ;

$$\begin{cases} \dot{\mathbf{x}} &= \nabla_{\mathbf{p}} H \\ \dot{\mathbf{p}} &= -\nabla_{\mathbf{x}} H \end{cases}$$

Osher et al suggested a new representation of the wavefront $\Gamma(t)$ in $\mathbb{R}^n$ as the zero level set of a vector valued function $\vec{\phi}(t) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n+1}$ defined in the phase space $\mathbb{R}^n \times \mathbb{R}^n$ [OCK02]. Since the wavefront moves along the characteristic curve in the phase space, the level set function $\vec{\phi}(t)$ satisfies the Liouville equation ;

$$\vec{\phi}_t + \nabla_{\mathbf{p}} H \cdot \nabla_{\mathbf{x}} \vec{\phi} - \nabla_{\mathbf{x}} H \cdot \nabla_{\mathbf{p}} \vec{\phi} = 0$$

This representation allows us to track the wave propagation possibly with superpositions or with complicated geometries, just by solving the linear Liouville equation. Good results of this method have been reported in the simulations of geometric optics [OCK02], seismic wave [QCO03] and semi-classical Schrodinger equation [CLO03]. Actually this representation of the wavefront propagation can be put to a more general setting that will be discussed in the next Section.

## 1.2 Generalized Level Set Method

The level set method presented by Osher and Sethian [OS88] tracks a moving interface of codimension one by representing the interface as the level set of a *scalar* valued function. Ambrosio and Soner theoretically extended the method to arbitrary codimension, and still used a scalar valued function to represent an interface [AS96]. It was pointed out that the representation of a scalar valued function is numerically unstable for locating isosurface, and the instability was fixed by employing a vector valued function [BCM99, OCK02].

In this generalized level set method, an isosurface $\Gamma(t) \subset \mathbb{R}^n$ of codimension $m$ is represented as the zero level set of a vector valued function $\vec{\phi}(t) : \mathbb{R}^n \to \mathbb{R}^m$. If the interface $\Gamma(t)$ moves with a velocity $\vec{V} : \mathbb{R}^n \to \mathbb{R}^n$, the method tracks the interface by solving the following advection equation.

$$\vec{\phi}_t + \left(\vec{V} \cdot \nabla\right) \vec{\phi} = 0$$

This generalized level set method invokes two main computational problems; how to construct and visualize the level set of $\vec{\phi} : \mathbb{R}^n \to \mathbb{R}^m$, and how to efficiently advance the level set function $\vec{\phi}(t)$ in high dimension. When the codimension $m$ is one, these problems are throughly studied under the names of isosurfacing [LC87,

BK96, BWC00] and local level set method [PMZ99, AS95, Str99b] respectively. For the higher codimension, we have just begun to see their developments.

The following two chapters introduce unified theories and very efficient algorithms of isosurfacing and local level set method in high codimension. These two methods constitutes a computational framework tracking a moving interface in arbitrary dimension and codimension.

# CHAPTER 2

# Simplicial Isosurfacing

The level set method has been a very successful tool capturing a moving interface. The final step of the method is to construct and visualize the isosurface of a discrete function $\phi : \{0, \cdots, N\}^n \to \mathbb{R}^m$. There have existed many practical isosurfacing algorithms when $n = 3, m = 1$ or $n = 2, m = 1$. Recently we have begun to see the development of isosurfacing algorithms for higher dimensions and codimensions. This paper introduces a unifed theory and an efficient isosurfacing algorithm that works in arbitrary number of dimensions and codimensions.

The isosurface $\Gamma$ of a discrete function $\phi$ is defined as the isosurface of its simplicial interpolant $\hat{\phi} : [0, N]^n \to \mathbb{R}^m$. With this simplicial definition, $\Gamma$ is geometrically a piecewise intersection of a simplex and $m$ hyperplanes. $\Gamma$ is constructed as the union of simplices. When $n = m + 1$ or $m + 2$, $\Gamma$ is projected down into $\mathbb{R}^3$ and can be visualized. For surface visualizations, a simple formula is presented calculating the oriented normal vector field of the projection of $\Gamma$ into $\mathbb{R}^3$, which gives light shadings.

## 2.1 Introduction

The purpose of this paper is to construct and visualize the isosurface of a discrete function in arbitrary dimension and codimension. A discrete function $\phi : \mathbb{Z}^n \to \mathbb{R}^m$ is meant to be a vector valued function defined on a uniform grid in $\mathbb{R}^n$, which is identified as $\mathbb{Z}^n$ by translation and scaling. Geometric objects such as curves and surfaces have been successfully represented as the isosurface of a discrete function. This implicit representation enables us to easily handle a moving geometric object, because updating a discrete function may have the same effect as moving the geometric object. This is the idea of the *level set method* presented by Osher and Sethian in 1989 [OS88]. They approximated an interface in $\mathbb{R}^n$ moving with curvature dependent speed by simply updating a discrete function $\phi : \mathbb{Z}^n \to \mathbb{R}$. In a recent application of the level set method to geometric optics [OCK02], wave fronts in $\mathbb{R}^3$ were lifted into $\mathbb{R}^5$ to remove singularities and represented as the isosurface of $\phi : \mathbb{Z}^5 \to \mathbb{R}^3$. Ambrosio and Soner [AS96] extended the theory of the level set method to arbitrary dimension. Based on

the extended theory, Lorigo and Faugeras et al [LFG00] used the isosurface of $\phi : \mathbb{Z}^3 \to \mathbb{R}^2$ on the segmentation of blood vessels. Of all the successful methods above, the final step is to construct and visualize the isosurface of a discrete function $\phi : \mathbb{Z}^n \to \mathbb{Z}^m$, which is the purpose of this paper.

Before discussing the construction of the isosurface of a discrete function, we must give the definition of the isosurface. Since a discrete function is defined only on grid points, an interpolation should be given to define the isosurface of a discrete function as the isosurface of its interpolant. Historically there have developed two types of interpolations in parallel; one is piecewise interpolation on cubes and the other is on simplices. Lorenson and Cline in 1987 presented a cube-based isosurfacing, *marching cubes* which works in $\mathbb{R}^3$ [LC87]. Bhaniramka, Wenger and Crawfis extended the marching cubes to arbitrary dimension [BWC00]. On the other hand, Carneiro, Silva and Kaufman suggested a simplex-based isosurfacing algorithm, *tetra cubes* which works in $\mathbb{R}^3$ [BK96]. Weigle and Banks proposed a simplex-based isosurfacing algorithm that works in arbitrary dimension and codimension [WB96].

The two types of isosurfacing algorithms have very different properties. Simplicial interpolation is naturally defined as the unique linear interpolant on a simplex, but cubic interpolation on a cube is often ambiguous and so is the definition of an isosurface. This is because a cube has $2^n$ vertices, while a simplex has $(n+1)$ vertices in $\mathbb{R}^n$. For this reason, Montani and Scorpigno modified the marching cubes algorithm to remove the ambiguity of cubic interpolation [MSS94] and Bhaniramka et al in their paper [BWC00] extended their modifications to arbitrary dimension. Given a discrete function on a uniform grid, an isosurfacing algorithm based on cubic interpolation iteratively applies to each grid cell. But for a simplicial isosurfacing to be used, each grid cell should be decomposed into simplices. It is not known yet what is the minimum number of simplices for the decomposition of a cube [HA96]. Most optimal decomposition algorithms split a cube into $O(n!)$ simplices [Hai91, Sal82]. So the simplicial isosurfacing should iterate $O(n!)$ times more than cubic isosurfacings. If we compare programming complexities, we see that simplicial isosurfacing is much simpler than cubic isosurfacings. In $\mathbb{R}^3$, 2 types of isosurfaces exist on a 3-simplex, but 14 types on a 3-cube [LC87]. In $\mathbb{R}^4$, 2 types of isosurfaces exist on a 4-simplex, but 222 types on a 4-cube [BWC00]. In $\mathbb{R}^n$, $\left\lceil \frac{n+1}{2} \right\rceil$ types of isosurfaces exist on a $n$-simplex, but it is hard even to classify the types of isosurfaces on an $n$-cube [BWC00].

Weigle and Banks proposed a simplicial isosurfacing algorithm that works in arbitrary dimension and codimension [WB96]. We followed their framework to define the isosurface of a discrete function as the isosurface of its simplicial interpolant. As they pointed out in their paper [WB96], the isosurface $\Gamma$ of a discrete function $\phi : \mathbb{Z}^n \to \mathbb{R}^m$ is then a piecewise intersection of a simplex and

$m$ hyperplanes. We introduce a new construction algorithm of $\Gamma$ that numerically costs $O(1)$ in time and space for each simplex. Our algorithm is explicit, while that of Weigle and Banks is recursive.

In Section 2.2, the simplicial interpolation algorithm is briefly reviewed. With the simplicial definition, the isosurface of $\phi : \mathbb{Z}^n \to \mathbb{R}^m$ is a piecewise intersection of a simplex and $m$ hyperplanes in $\mathbb{R}^n$. In Section 2.3, we give the triangulation tables of the intersection of a simplex and a hyperplane. A counting theorem is stated to prove the optimality of the triangulation tables. From the triangulation tables, an intersection of a simplex and any number of hyperplanes can be constructed as the union of simplices. In Section 2.4, our main isosurfacing algorithm is presented. To visualize it, the isosurface in high dimension is often projected down to $\mathbb{R}^3$. In Section 2.5, we discuss a visualization of the projected isosurface. In Section 2.6, several numerical examples are given. In Section 2.7, we summarize our algorithms and discuss future work.

## 2.2 Simplicial Interpolation

We introduce some definitions in computational geometry to review the simplicial interpolation algorithm [AS85, Kuh60]. An *affine set* is a translation of a vector space. A *hyperplane* is an $(n-1)$-dimensional affine set in $\mathbb{R}^n$. A set of points is called *affinely independent*, if it is a translation of a linearly independent set. An *$m$-simplex* is the convex hull of $(m+1)$ affinely independent points, which are called *vertices* of the $m$-simplex. An $l$-simplex is called a *face* of a $m$-simplex $S$, if its vertices are vertices of $S$. A *triangulation* $T$ of $D \subset \mathbb{R}^n$ is a finite collection of $m$-simplices such that $\cup_{\sigma \in T} \sigma = D$ and $\sigma_1 \cap \sigma_2$ is empty or a common face of $\sigma_1$ and $\sigma_2$, if $\sigma_1, \sigma_2 \in T$. An *affine map* is the composition of a linear map and a translation. An *$n$-cube* is the direct product of $n$ intervals. A *polytope* is the convex hull of a set of finite points. Two sets are *affinely isomorphic*, if there exists a bijective affine map between them.

The simplicial interpolation is an operator $\wedge$ from discrete function space $C(\mathbb{Z}^n : \mathbb{R}^m)$ to the continuous function space $C(\mathbb{R}^n : \mathbb{R}^m)$. By a discrete function $\phi \in C(\mathbb{Z}^n : \mathbb{R}^m)$, we mean a vector valued function defined on a uniform grid, which is identified with $\mathbb{Z}^n$ by scaling and translation. To define the simplicial interpolation $\wedge$, an $n$-cube decomposition algorithm should be given. We choose the canonical Kuhn triangulation algorithm to decompose an n-cube into simplices.

### 2.2.1 Kuhn Triangulation

Let us define $S_n$ as the set of all permutations of $\{1, \cdots, n\}$, i.e. bijective maps from $\{1, \cdots, n\}$ onto $\{1, \cdots, n\}$. Let us define the standard $n$-cube $\bar{C} = [0,1]^n$. Given a permutation $J \in S_n$, we define a set $\bar{C}_J$ as

$$\bar{C}_J = \left\{ x \in \bar{C} \mid 1 \geq x_{J(1)} \geq \cdots \geq x_{J(n)} \geq 0 \right\}$$

Then $\bar{C}_J$ is an n-simplex with the following vertices

$$\begin{aligned} \bar{v}_0 &= (0, \cdots, 0) \\ \bar{v}_1 &= \bar{v}_0 + e_{J(1)} \\ &\vdots \\ \bar{v}_n &= \bar{v}_{n-1} + e_{J(n)} \end{aligned}$$

$e_k$ is the $k$-th canonical base of $\mathbb{R}^n$. $\bar{C} = \cup_{J \in S_n} \bar{C}_J$, because for any $x \in \bar{C}$, there is a permutation $J$ such that $x_{J(1)} \geq \cdots \geq x_{J(n)}$. Since $|S_n| = n!$, $\bar{C}$ is the union of $n!$ simplices.

A general $n$-cube $C = [a_1, b_1] \times \cdots \times [a_n, b_n]$ is affinely isomorphic to $\bar{C}$ under an affine map $f : \bar{C} \to C$ such that $f(x) = \left( \frac{x_1 - a_1}{b_1 - a_1}, \cdots, \frac{x_n - a_n}{b_n - a_n} \right)$. The decomposition $\bar{C} = \cup_{J \in S_n} \bar{C}_J$ is preserved to be a decomposition of C under the affine map $f$; $C = f^{-1}\left(\bar{C}\right) = \cup_{J \in S_n} f^{-1}\left(\bar{C}_J\right)$. Let us denote the set $f^{-1}\left(\bar{C}_J\right)$ as $C_J$. Then $C_J$ is an $n$-simplex with the following vertices.

$$\begin{aligned} v_0 &= (a_1, \cdots, a_n) \\ v_1 &= v_0 + (b_{J(1)} - a_{J(1)}) \cdot e_{J(1)} \\ &\vdots \\ v_n &= v_{n-1} + (b_{J(n)} - a_{J(n)}) \cdot e_{J(n)} \end{aligned}$$

Given a uniform grid on $\mathbb{R}^n$, we identify the grid as $\mathbb{Z}^n$ by translation and scaling. We define a grid cell $C^a = \{x \in \mathbb{R}^n \mid a_i \leq x_i \leq a_i + 1\}$ for each $a \in \mathbb{Z}^n$, which is an $n$-cube. Each grid cell $C^a$ is decomposed into $n!$ simplices $C^a_J$. Then we have the following decomposition of $\mathbb{R}^n$ into $n$-simplices;

$$\mathbb{R}^n = \cup_{a \in \mathbb{Z}^n} C^a = \cup_{a \in \mathbb{Z}^n} \cup_{J \in S_n} C^a_J$$

Furthermore, a set of $n$-simplices $\{C^a_J \mid a \in \mathbb{Z}^n, J \in S_n\}$ is a triangulation of $\mathbb{R}^n$ and called the *Kuhn triangulation* [AS85].

### 2.2.2 Interpolation Procedure

Given a discrete function $\phi : \mathbb{Z}^n \to \mathbb{R}^m$, its simplicial interpolant $\hat{\phi} : \mathbb{R}^n \to \mathbb{R}^m$ is piecewise defined on each simplex $C^a_J$ in the Kuhn triangulation. In a

simplex $C_J^a = conv[v_0, \cdots, v_n]$, $\hat{\phi} \mid_{C_J^a}$ is defined as the linear interpolant of $\phi$ at $\{v_0, \cdots, v_n\}$. With the barycentric coordinates $\lambda = (\lambda_0, \cdots, \lambda_n)$ s.t. $\lambda_i \geq 0, \forall i$ and $\sum_{i=0}^n \lambda_i = 1$, $\hat{\phi}$ is defined as

$$\hat{\phi}\left(\sum_{i=0}^n \lambda_i v_i\right) = \sum_{i=0}^n \lambda_i \phi(v_i)$$

With the standard coordinates, there is a practical procedure to evaluate $\hat{\phi}$, as described by Kuhn [Kuh60]. We assume that the uniform grid is $\mathbb{Z}^n$. The following procedure can be applied to a general uniform grid by translation and scaling. Given $x \in \mathbb{R}^n$, $a \in \mathbb{Z}^n$ is defined as $a_i = [x_i]$ $\forall i$, and $y \in [0,1]^n$ as $y = x - a$. Let $J$ be a permutation of $\{1, \cdots, n\}$ such that $y_{J(1)} \geq \cdots \geq y_{J(n)}$, then

$$\begin{array}{rcrcl}
\hat{\phi}(x) & = & \left(1 - y_{J(1)}\right) & \cdot & \phi\left(a\right) \\
& + & \left(y_{J(1)} - y_{J(2)}\right) & \cdot & \phi\left(a + e_{J(1)}\right) \\
& \vdots & & & \\
& + & \left(y_{J(n-1)} - y_{J(n)}\right) & \cdot & \phi\left(a + e_{J(1)} + \cdots + e_{J(n-1)}\right) \\
& + & \left(y_{J(n)}\right) & \cdot & \phi\left(a + e_{J(1)} + \cdots + e_{J(n-1)} + e_{J(n)}\right)
\end{array}$$

If $\phi$ is scalar valued, $\nabla\hat{\phi}$ on the simplex $C_J^a$ is easily calculated as

$$\begin{array}{rcl}
\nabla\hat{\phi}_{J(1)} & = & \frac{\phi(v_1) - \phi(v_0)}{b_{J(1)} - a_{J(1)}} \\
\nabla\hat{\phi}_{J(2)} & = & \frac{\phi(v_2) - \phi(v_1)}{b_{J(2)} - a_{J(2)}} \\
& \vdots & \\
\nabla\hat{\phi}_{J(n)} & = & \frac{\phi(v_n) - \phi(v_{n-1})}{b_{J(n)} - a_{J(n)}}
\end{array} \tag{2.1}$$

Now, we show that the simplicial interpolation operator $\wedge$ is an operator from discrete function space $C(\mathbb{Z}^n : \mathbb{R}^m)$ into continuous function space $C(\mathbb{R}^n : \mathbb{R}^m)$.

**Theorem 2.2.1** *$\hat{\phi} : \mathbb{R}^n \to \mathbb{R}^m$ is a continuous function.*

**Proof** $\hat{\phi}$ is continuous on the interior of $C_J^a$, because it is a polynomial. On $C_J^a \cap C_{J'}^{a'}$, $\hat{\phi}$ has two possible definitions $\hat{\phi} \mid_{C_J^a}$ and $\hat{\phi} \mid_{C_{J'}^{a'}}$. Since $\{C_J^a\}$ is a triangulation, $C_J^a \cap C_{J'}^{a'}$ is a common face of $C_J^a$ and $C_{J'}^{a'}$. Every face of a simplex is also a simplex of lower dimension. Since there is only one linear interpolant on a simplex, the two definitions must be the same. $\square$

Now, we define the isosurface $\Gamma$ of a discrete function $\phi : \mathbb{Z}^n \to \mathbb{R}^m$ as the isosurface of its simplicial interpolant $\hat{\phi} : \mathbb{R}^n \to \mathbb{R}^m$. We assume that $\Gamma$ is the zero isosurface of $\hat{\phi}$ without losing generality. By this simplicial definition,

$$\begin{aligned}\Gamma &= \left\{x \in \mathbb{R}^n \mid \hat{\phi}(x) = 0\right\} \\ &= \cup_{a\in\mathbb{Z}^n} \cup_{J\in S_n} \left\{x \in C_J^a \mid \hat{\phi}(x) = 0\right\} \\ &= \cup_{a\in\mathbb{Z}^n} \cup_{J\in S_n} \left[C_J^a \cap \left\{\hat{\phi}_1 \mid_{C_J^a} = 0\right\} \cap \cdots \cap \left\{\hat{\phi}_m \mid_{C_J^a} = 0\right\}\right]\end{aligned}$$

Since each component $\hat{\phi}_i$ of $\hat{\phi}$ is a first order polynomial on each $C_J^a$, the set $\left\{\hat{\phi}_i \mid_{C_J^a} = 0\right\}$ is geometrically a hyperplane, and $\Gamma$ is a piecewise intersection of a simplex and $m$ hyperplanes. $\Gamma$ will be piecewise constructed as the union of $(n-m)$-simplices in §4. Before constructing the $\Gamma$, the next section introduces a triangulation algorithm of the intersection of a simplex and a hyperplane.

## 2.3 Intersection of a Simplex and a Hyperplane

Let an $n$-simplex $S = conv\,[v_1, \cdots, v_{n+1}]$ and a hyperplane $H = \{x \in \mathbb{R}^n \mid \psi(x) = 0\}$ with a first order polynomial $\psi : \mathbb{R}^n \to \mathbb{R}$ be given . Let us first assume that $H$ does not include the vertices of $S$, i.e. $\psi(v_i) \neq 0$, $\forall i$. This assumption will be removed later in this section. Then $S \cap H$ is a polytope with vertices $v_{ij}$, where

$$v_{ij} \doteqdot \frac{\psi(v_i)}{\psi(v_i) - \psi(v_j)} \cdot v_j - \frac{\psi(v_j)}{\psi(v_i) - \psi(v_j)} \cdot v_i, \text{ if } \psi(v_i) < 0, \psi(v_j) > 0 \qquad (2.2)$$

$v_{ij}$ is the interpolation point between $v_i$ and $v_j$ such that $\psi(v_{ij}) = 0$. The intersection $S \cap H$ is said to be type $(p, q)$, if $\psi(v_i) < 0$ for $p$ vertices and $\psi(v_j) > 0$ for $q$ vertices. It is worth noting that all sections of the same type are isomorphic to each other [Som58]. Hence a triangulation of one specific intersection of type $(p, q)$ can be applied to the triangulation of any intersection of type $(p, q)$. An intersection of type $(p, q)$ is naturally isomorphic to an intersection of type $(q, p)$, because $\{\psi = 0\} = \{-\psi = 0\}$. Since $H$ is assumed not to pass through any vertex of $S$, $p + q = n + 1$ and there exist $\left[\frac{n+1}{2}\right]$ types of intersection between $S$ and $H$.

In this section, triangulation tables of the intersection are presented up to dimension five. Triangulation tables of two and three dimensions are easily generated by the following Figures 2.1 and 2.2. Triangulation tables of four and five dimensions are generated by applying the Delaunay triangulation [Zie95] to a specific section of each type. Higher dimensional tables can be also generated in such a way. In the Tables 2.1, 2.2, 2.3 and 2.4, $v_1, \cdots, v_{n+1}$ are the vertices of

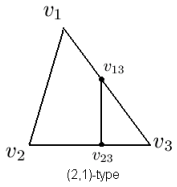


Figure 2.1: Intersection of a 2-simplex and a hyperplane

| $v_1$ | $v_2$ | $v_3$ | $w_1$ | $w_2$ |
|---|---|---|---|---|
| $-$ | $-$ | $+$ | $v_{13}$ | $v_{23}$ |

Table 2.1: A triangulation of the intersection of a 2-simplex and a hyperplane

a simplex $S$ and $w_1, \cdots, w_n$ are the vertices of a simplex in the triangulation of the intersection of $S$ and a hyperplane $H$.

We show that the preceding tables are optimal in a sense that the number of simplices in each triangulation can not be reduced. First we quote a theorem in Haiman's paper [Hai91], then our statement follows as a corollary.

**Theorem 2.3.1** *Let $\Delta_k$ be a k-simplex and $\Delta_l$ a l-simplex. Every triangulation of $\Delta_k \times \Delta_l$ uses exactly $\frac{(k+l)!}{k!l!}$ simplices.*

**Proof** Since every k-simplex is affinely isomorphic to any other, let us pick the standard k-simplex with vertices $e_1, e_2, \cdots, e_{k+1} \in \mathbb{R}^{k+1}$. Every simplex of a triangulation of $\Delta_k \times \Delta_l$ has the volume $\frac{1}{k!l!}$. Since the volume of $\Delta_k \times \Delta_l$ is $\frac{1}{(k+l)!}$, every triangulation of $\Delta_k \times \Delta_l$ uses exactly $\frac{(k+l)!}{k!l!}$ simplices. see [Hai91] for details. □

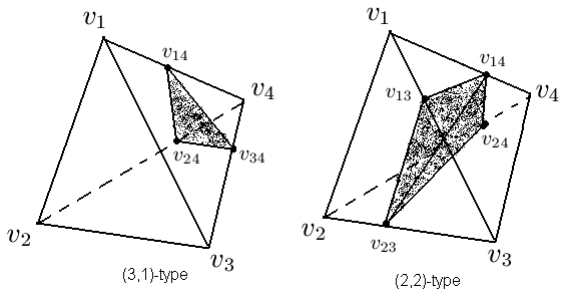


Figure 2.2: Intersections of a 3-simplex and a hyperplane

| $v_1$ | $v_2$ | $v_3$ | $v_4$ | $w_1$ | $w_2$ | $w_3$ |
|---|---|---|---|---|---|---|
| $-$ | $-$ | $-$ | $+$ | $v_{14}$ | $v_{24}$ | $v_{34}$ |
| $-$ | $-$ | $+$ | $+$ | $v_{13}$ | $v_{23}$ | $v_{14}$ |
| | | | | $v_{14}$ | $v_{23}$ | $v_{24}$ |

Table 2.2: A triangulation of the intersection of a 3-simplex and a hyperplane

| $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $w_1$ | $w_2$ | $w_3$ | $w_4$ |
|---|---|---|---|---|---|---|---|---|
| $-$ | $-$ | $-$ | $-$ | $+$ | $v_{15}$ | $v_{25}$ | $v_{35}$ | $v_{45}$ |
| $-$ | $-$ | $-$ | $+$ | $+$ | $v_{14}$ | $v_{15}$ | $v_{34}$ | $v_{24}$ |
| | | | | | $v_{15}$ | $v_{24}$ | $v_{25}$ | $v_{34}$ |
| | | | | | $v_{15}$ | $v_{25}$ | $v_{35}$ | $v_{34}$ |

Table 2.3: A triangulation of the intersection of a 4-simplex and a hyperplane

| $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_5$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $-$ | $-$ | $-$ | $-$ | $-$ | $+$ | $v_{16}$ | $v_{26}$ | $v_{36}$ | $v_{56}$ | $v_{46}$ |
| $-$ | $-$ | $-$ | $-$ | $+$ | $+$ | $v_{15}$ | $v_{16}$ | $v_{25}$ | $v_{45}$ | $v_{35}$ |
| | | | | | | $v_{16}$ | $v_{25}$ | $v_{26}$ | $v_{35}$ | $v_{45}$ |
| | | | | | | $v_{16}$ | $v_{26}$ | $v_{35}$ | $v_{45}$ | $v_{36}$ |
| | | | | | | $v_{16}$ | $v_{26}$ | $v_{36}$ | $v_{45}$ | $v_{46}$ |
| $-$ | $-$ | $-$ | $+$ | $+$ | $+$ | $v_{14}$ | $v_{15}$ | $v_{16}$ | $v_{36}$ | $v_{25}$ |
| | | | | | | $v_{14}$ | $v_{15}$ | $v_{25}$ | $v_{36}$ | $v_{35}$ |
| | | | | | | $v_{14}$ | $v_{16}$ | $v_{25}$ | $v_{26}$ | $v_{36}$ |
| | | | | | | $v_{14}$ | $v_{24}$ | $v_{25}$ | $v_{34}$ | $v_{26}$ |
| | | | | | | $v_{14}$ | $v_{25}$ | $v_{26}$ | $v_{36}$ | $v_{34}$ |
| | | | | | | $v_{14}$ | $v_{25}$ | $v_{34}$ | $v_{36}$ | $v_{35}$ |

Table 2.4: A triangulation of the intersection of a 5-simplex and a hyperplane

**Corollary 2.3.2** *Every triangulation of $(p,q)$-type section uses exactly $\frac{(p+q-2)!}{(p-1)!(q-1)!}$ simplices.*

**Proof** Since all sections of the same type are isomorphic to each other, we can choose a specific section of $(p,q)$-type. Let a simplex $S = conv\,[v_1, \cdots, v_{p+q}]$ and a hyperplane $H = \{\psi(x) = 0\}$ be given such that

$$\begin{aligned} \psi(v_1) = \cdots = \psi(v_p) &= -1 \\ \psi(v_{p+1}) = \cdots = \psi(v_{p+q}) &= 1 \end{aligned}$$

With the barycentric coordinates $x\,(\lambda_1, \cdots, \lambda_{p+q}) = \sum_{k=1}^{p+q} \lambda_k v_k$,

$$\begin{aligned} \psi\,(x(\lambda)) = \psi\left(\textstyle\sum_{k=1}^{p+q} \lambda_k v_k\right) &= \textstyle\sum_{k=1}^{p+q} \lambda_k \psi(v_k) \\ &= \textstyle\sum_{j=p+1}^{p+q} \lambda_j - \sum_{i=1}^{p} \lambda_i \end{aligned}$$

$H$,$S$, and $H \cap S$ are algebraically expressed with the barycentric coordinates.

$$\begin{aligned} x(\lambda) \in H &\Leftrightarrow \textstyle\sum_{i=1}^{p} \lambda_i = \sum_{j=p+1}^{p+q} \lambda_j \\ x(\lambda) \in S &\Leftrightarrow \textstyle\sum_{k=1}^{p+q} \lambda_k = 1,\ \lambda_k \geq 0 \\ x(\lambda) \in S \cap H &\Leftrightarrow \textstyle\sum_{i=1}^{p} \lambda_i = \sum_{i=p+1}^{p+q} \lambda_j = \frac{1}{2},\ \lambda_i, \lambda_j \geq 0 \end{aligned}$$

Since coordinates $\lambda_i$ and $\lambda_j$ are decoupled in the expression of $S \cap H$,

$$\begin{aligned} S \cap H &\simeq \left\{(\lambda_1, \cdots, \lambda_p) \mid \textstyle\sum_{i=1}^{p} \lambda_i = \frac{1}{2},\ \lambda_i \geq 0\right\} \\ &\times \left\{(\lambda_{p+1}, \cdots, \lambda_{p+q}) \mid \textstyle\sum_{j=p+1}^{p+q} \lambda_j = \frac{1}{2},\ \lambda_j \geq 0\right\} \end{aligned}$$

Hence $S \cap H$ is isomorphic to $\Delta_{p-1} \times \Delta_{q-1}$. By the theorem above, every triangulation of $S \cap H$ should use $\frac{(p+q-2)!}{(p-1)!(q-1)!}$ simplices.□

By the triangulation tables, we can construct the intersection of an $n$-simplex and a hyperplane as the union of $(n-1)$-simplices. Here is the construction procedure. Given an $n$-simplex $S = conv\,[v_1, \cdots, v_{n+1}]$ and a hyperplane $H = \{\psi = 0\}$, signums of $\psi(v_1), \cdots, \psi(v_{n+1})$ are counted. If there are more positive signums than negative, $\psi$ is inverted ; $\psi = -\psi$. Let $p$ be the number of negative signums and $q$ be the number of positive, then $S \cap H$ is an intersection of type $(p,q)$. The vertices of $S$ are reordered such that $\psi(v_1), \cdots, \psi(v_p) < 0$ and $\psi(v_{p+1}), \cdots, \psi(v_{n+1}) > 0$. Interpolation points $v_{ij}$ (2.2) are calculated for $i = 1, \cdots, p$ and $j = p+1, \cdots, n+1$. Referencing the triangulation table of $(p,q)$-type, $S \cap H$ is constructed as the union of some simplices of one lower dimension.

$$S \cap H = S^1 \cup \cdots \cup S^l \tag{2.3}$$

Numerically only vertices of simplices are stored. We note that this triangulation procedure is numerically finite in space and time, because the number of simplices is bounded by the tables. Also the procedure is numerically stable, since the only numerical calculation is to interpolate inner points $v_{ij}$(2.2) between $v_i$ and $v_j$. Practically if $H$ passes through a vertex $v_i$ of $S$, we perturb $\psi$ by $\psi(v_i) = \epsilon$, where $0 < \epsilon \ll 1$.

## 2.4 Isosurfacing

We define the isosurface $\Gamma$ of a discrete function $\phi : \mathbb{Z}^n \to \mathbb{R}^m$ as the isosurface of its simplicial interpolant $\hat{\phi} : \mathbb{R}^n \to \mathbb{R}^m$. We may assume that $\Gamma$ is the zero isosurface of $\hat{\phi}$ without losing generality. By this simplicial definition,

$$\begin{aligned}
\Gamma &= \left\{ x \in \mathbb{R}^n \mid \hat{\phi}(x) = 0 \right\} \\
&= \cup_{a \in \mathbb{Z}^n} \cup_{J \in S_n} \left\{ x \in C_J^a \mid \hat{\phi}(x) = 0 \right\} \\
&= \cup_{a \in \mathbb{Z}^n} \cup_{J \in S_n} \left[ C_J^a \cap \left\{ \hat{\phi}_1 \mid_{C_J^a} = 0 \right\} \cap \cdots \cap \left\{ \hat{\phi}_m \mid_{C_J^a} = 0 \right\} \right]
\end{aligned}$$

Since each component $\hat{\phi}_i : \mathbb{R}^n \to \mathbb{R}$ of $\hat{\phi}$ is a first order polynomial on each simplex $C_J^a$, the set $\left\{ \hat{\phi}_i \mid_{C_J^a} = 0 \right\}$ is geometrically a hyperplane, and $\Gamma$ is the intersection of a simplex and $m$ hyperplanes. In §3, an algorithm was presented to construct the intersection of a simplex and a hyperplane as the union of simplices of one lower dimension. Weigle and Banks pointed out that the intersection of a simplex and several hyperplanes can be constructed as the union of simplices by successively applying the algorithm [WB96]. On a simplex $C_J^a$, let us say $H^1 = \left\{ \hat{\phi}_1 \mid_{C_J^a} = 0 \right\}, \cdots, H^m = \left\{ \hat{\phi}_m \mid_{C_J^a} = 0 \right\}$. Then $C_J^a \cap H^1 \cap \cdots \cap H^m$ is constructed by the following procedure.

$$\begin{aligned}
C_J^a \cap H^1 &= \cup_{j=1}^{n_1} S^{1,j} \\
C_J^a \cap H^1 \cap H^2 &= \cup_{j=1}^{n_1} [S^{1,j} \cap H^2] = \cup_{j=1}^{n_2} S^{2,j} \\
&\vdots \\
C_J^a \cap H^1 \cap \cdots \cap H^m &= \cup_{j=1}^{n_{m-1}} [S^{m-1,j} \cap H^m] = \cup_{j=1}^{n_m} S^{m,j}
\end{aligned}$$

We get a set of $(n-m)$-simplices, $T_{a,J} = \{S^{m,j} \mid j = 1, \cdots, n_m\}$ such that $C_J^a \cap \Gamma = C_J^a \cap H^1 \cap \cdots \cap H^m = \cup_{\sigma \in T_{a,J}} \sigma$. We note that it is numerically finite in time and space to construct the set $T_{a,J}$. For example, the intersection of a 5-simplex and 3 hyperplanes is constructed as the union of up to $6 \times 3 \times 2$ number of 2-simplices, where $6, 3, 2$ are maximum number of simplices in the triangulation tables of $5, 4, 3$ dimensions. That means $n_1 \leq 6$, $n_2 \leq 6 \times 3$, and $n_3 \leq 6 \times 3 \times 2$.

Iterating all the simplices $C_J^a$ in the Kuhn triangulation, $\Gamma$ is constructed as the union of $(n-m)$-simplices;

$$\begin{aligned} \Gamma &= \cup_{a\in\mathbb{Z}^n} \cup_{J\in S_n} [\Gamma \cap C_J^a] \\ &= \cup_{a\in\mathbb{Z}^n} \cup_{J\in S_n} \cup_{\sigma\in T_{a,J}} \sigma \end{aligned} \tag{2.4}$$

## 2.5 Visualization

In Section 2.4, an algorithm was introduced to construct the isosurface $\Gamma$ of a discrete function $\phi : \mathbb{Z}^n \to \mathbb{R}^m$ for any dimension $n$ and any codimension $m$. $\Gamma$ is an $(n-m)$-dimensional manifold in its smooth region. To visualize it, $\Gamma$ is often projected down into $\mathbb{R}^3$ under a projection map $P : \mathbb{R}^n \to \mathbb{R}^3$. Since a projection is an affine map, it maps a simplex to a simplex. From the isosurfacing algorithm(2.4), $P(\Gamma)$ is the union of projected $(n-m)$-simplices;

$$P(\Gamma) = \cup_{a\in\mathbb{Z}^n} \cup_{J\in S_n} \cup_{\sigma\in T_{a,J}} P(\sigma) \tag{2.5}$$

When $n = m+1$ or $m+2$, $P(\Gamma)$ is the union of line segments or triangles, which can be visualized by usual graphic libraries such as OpenGL. But, for a surface visualization in $\mathbb{R}^3$, the normal vector field is needed for shadowings. Now we present a simple formula calculating the normal vector field of $P(\Gamma)$.

In a general framework, let $P : \mathbb{R}^n \to \mathbb{R}^{n-m+2}$ be a projection such that $P(x_1, \cdots, x_n) = (x_1, \cdots, x_{n-m+2})$. In $\mathbb{R}^n$, $\nabla\hat{\phi}_1, \cdots, \nabla\hat{\phi}_m$ form the normal vector space of $\Gamma$ in smooth region. Let us define a vector field $\vec{n} : \mathbb{R}^n \to \mathbb{R}^n$ as

$$\vec{n} = \begin{vmatrix} \nabla\hat{\phi}_1 & \nabla\hat{\phi}_1 \cdot e_{n-m+2} & \cdots & \nabla\hat{\phi}_1 \cdot e_n \\ \vdots & \vdots & & \vdots \\ \nabla\hat{\phi}_m & \nabla\hat{\phi}_m \cdot e_{n-m+2} & \cdots & \nabla\hat{\phi}_m \cdot e_n \end{vmatrix} \tag{2.6}$$

$\nabla\hat{\phi}_j$ is calculated by the divided difference formula (2.1) and the determinant is expanded with cofactors.

$$\begin{aligned} \vec{n} \quad = \quad & \nabla\hat{\phi}_1 \cdot \begin{vmatrix} \nabla\hat{\phi}_2 \cdot e_{n-m+2} & \cdots & \nabla\hat{\phi}_2 \cdot e_n \\ \vdots & & \vdots \\ \nabla\hat{\phi}_m \cdot e_{n-m+2} & \cdots & \nabla\hat{\phi}_m \cdot e_n \end{vmatrix} \\ & \vdots \\ +(-1)^m \quad & \nabla\hat{\phi}_m \cdot \begin{vmatrix} \nabla\hat{\phi}_1 \cdot e_{n-m+2} & \cdots & \nabla\hat{\phi}_1 \cdot e_n \\ \vdots & & \vdots \\ \nabla\hat{\phi}_{m-1} \cdot e_{n-m+2} & \cdots & \nabla\hat{\phi}_{m-1} \cdot e_n \end{vmatrix} \end{aligned}$$

Since $\vec{n}$ is a linear combination of $\nabla\hat{\phi}_1, \cdots, \nabla\hat{\phi}_m$, it is a normal vector field of $\Gamma$. By the property of determinant, $\vec{n} \cdot e_j = 0$ if $j = n-m+2, \cdots, n$. Given a tangential vector $T$ of $\Gamma$ at $x \in \Gamma$, $0 = \vec{n}(x) \cdot T = \vec{n}(x) \cdot P(T)$. Since the projection $\pi$ is surjective, it is also surjective in tangential vector space. Therefore $\vec{n}$ is orthogonal to any tangent vector of $\pi(\Gamma)$ and it is the normal vector field of $P(\Gamma)$ in $\mathbb{R}^{n-m+1}$. For example, when $\phi : \mathbb{Z}^5 \to \mathbb{R}^3$ and $P : \mathbb{R}^5 \to \mathbb{R}^3$, the normal vector $\vec{n}$ is given by

$$\begin{aligned}
\vec{n} &= \nabla\hat{\phi}_1 \left( \left(\nabla\hat{\phi}_2 \cdot e_4\right)\left(\nabla\hat{\phi}_3 \cdot e_5\right) - \left(\nabla\hat{\phi}_3 \cdot e_4\right)\left(\nabla\hat{\phi}_2 \cdot e_5\right) \right) \\
&- \nabla\hat{\phi}_2 \left( \left(\nabla\hat{\phi}_1 \cdot e_4\right)\left(\nabla\hat{\phi}_3 \cdot e_5\right) - \left(\nabla\hat{\phi}_3 \cdot e_4\right)\left(\nabla\hat{\phi}_1 \cdot e_5\right) \right) \\
&+ \nabla\hat{\phi}_3 \left( \left(\nabla\hat{\phi}_1 \cdot e_4\right)\left(\nabla\hat{\phi}_2 \cdot e_5\right) - \left(\nabla\hat{\phi}_2 \cdot e_4\right)\left(\nabla\hat{\phi}_1 \cdot e_5\right) \right)
\end{aligned}$$

## 2.6 Numerical Examples

Every example was implemented in C++ and run on a PC with 2.2GHz CPU and 512MB memory. All isosurfaces were constructed by the isosurfacing algorithm in Section 2.4. 1-simplices or 2-simplices in the construction were visualized by the OpenGL graphic library. For 2-simplices, the normal vector field in Section 2.5 was used for shadowing.

### 2.6.1 A Singularity Resolves in $\mathbb{R}^2$

Let a curve $\Gamma \subset \mathbb{R}^2$ be given by $r = 1 + sin(7\theta)$ in the polar coordinates. $\Gamma$ is a 7-leafed curve and singular at the origin, where 14 curves meet. In this case, we can resolve the singularity by adding a phase variable $\theta$, as describe by Osher et al [OCK02] and Engquist et al [ERT02]. With a new coordinate system $(x, y, \theta)$, we define $\Gamma' \subset \mathbb{R}^3$ as the solution of the following equations

$$\begin{cases} x &= (1 + \sin(7\theta)) \cdot \cos(\theta) \\ y &= (1 + \sin(7\theta)) \cdot \sin(\theta) \end{cases}$$

Then $\Gamma = \pi(\Gamma')$, where $\pi(x, y, \theta) = (x, y)$. In this example, $\Gamma$ and $\Gamma'$ are approximated as the isosurfaces of discrete functions $\psi : \{1, \cdots, 128\}^2 \to \mathbb{R}$ and $\phi : \{1, \cdots, 128\}^3 \to \mathbb{R}^2$ respectively, where $a_i = -1 + 2 \cdot \frac{i}{128}$, $\theta_k = 2\pi \cdot \frac{k}{128}$ and

$$\psi(i, j) = \sqrt{a_i^2 + a_j^2} - 1 - \sin\left(7 \cdot \tan^{-1}\left(\frac{a_j}{a_i}\right)\right)$$

$$\phi(i, j, k) = \begin{pmatrix} a_i &- &(1 + \sin(7\theta_k)) \cdot \cos(\theta_k) \\ a_j &- &(1 + \sin(7\theta_k)) \cdot \sin(\theta_k) \end{pmatrix}$$

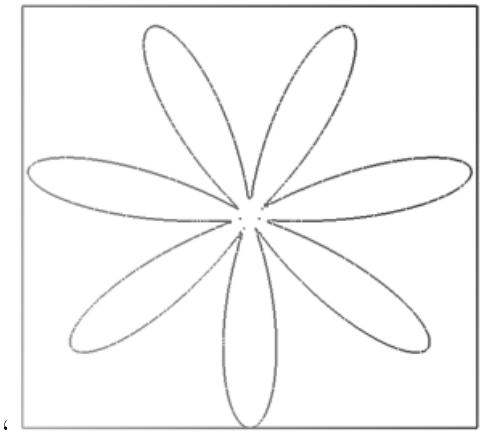

Figure 2.3: Approximation of the singular curve $\Gamma \subset \mathbb{R}^2$

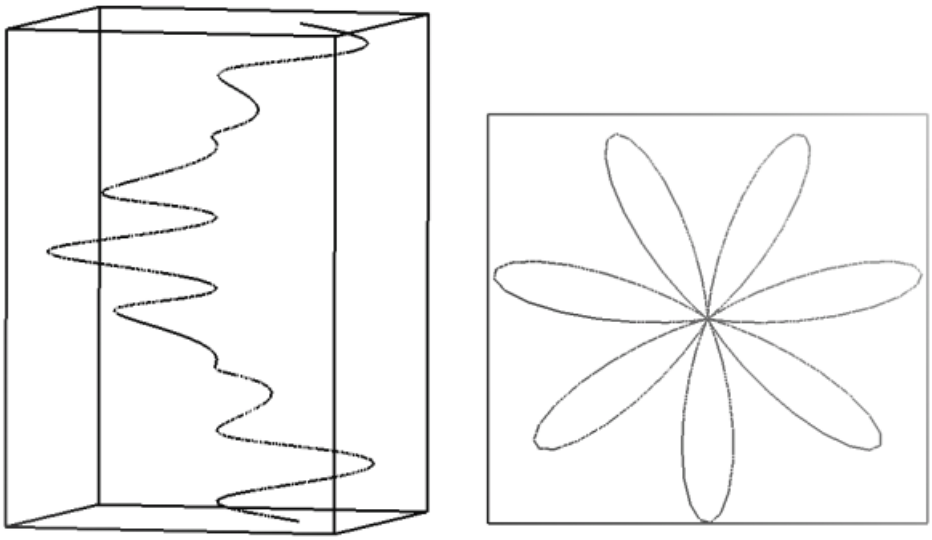

Figure 2.4: Approximations of the nonsingular curve $\Gamma' \subset \mathbb{R}^3$ and $\pi(\Gamma') \subset \mathbb{R}^2$

### 2.6.2 A Singularity Resolves in $\mathbb{R}^3$

Let a surface $\Gamma \subset \mathbb{R}^3$ be given by $r(\theta, \varphi) = 1+\cos(2\varphi)$ in the spherical coordinates $(r, \theta, \varphi)$. $\Gamma$ is a surface of revolution obtained by rotating 2-leaf curve around the z-axis and it is singular at the origin. In this case, we can resolve the singularity by adding phase variables $\theta$ and $\varphi$, as describe by Osher et al [OCK02]. This resolution is a direct generalization of the technique used in section 2.6.1. With

a new coordinate system $(x, y, z, \theta, \varphi)$, we define $\Gamma' \subset \mathbb{R}^5$ as the solution of equations

$$\begin{cases} x &= (1 + \cos(2\varphi)) \cdot \cos(\theta) \cdot \sin(\varphi) \\ y &= (1 + \cos(2\varphi)) \cdot \sin(\theta) \cdot \sin(\varphi) \\ z &= (1 + \cos(2\varphi)) \cdot \cos(\varphi) \end{cases}$$

Then $\Gamma = \pi(\Gamma')$, where $\pi(x, y, z, \theta, \varphi) = (x, y, z)$. In this example, $\Gamma$ and $\Gamma'$ are approximated as the isosurfaces of discrete functions $\psi : \{1, \cdots, 32\}^3 \to \mathbb{R}$ and $\phi : \{1, \cdots, 32\}^5 \to \mathbb{R}^2$ respectively, where $a_i = -2 + \frac{4}{32}i$, $\theta_m = 2\pi \cdot \frac{m}{32}$, $\varphi_n = \pi \cdot \frac{n}{32}$ and

$$\psi(i, j, k) = \sqrt{a_i^2 + a_j^2 + a_k^2} - 1 - \cos\left(2\tan^{-1}\left(\frac{a_j}{a_i}\right)\right)$$

$$\phi(i, j, k, m, n) = \begin{pmatrix} a_i &- (1 + \cos(2\varphi_n)) \cdot \cos(\theta_m) \cdot \sin(\varphi_n) \\ a_j &- (1 + \cos(2\varphi_n)) \cdot \sin(\theta_m) \cdot \sin(\varphi_n) \\ a_k &- (1 + \cos(2\varphi_n)) \cdot \cos(\varphi_n) \end{pmatrix}$$

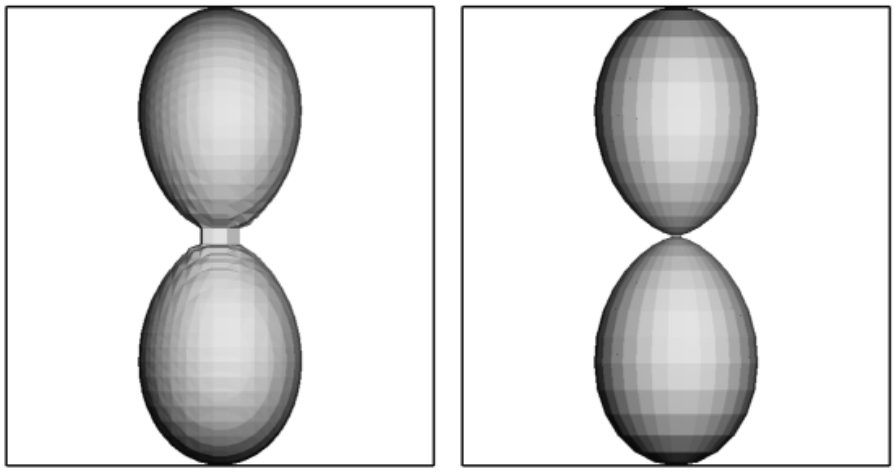

Figure 2.5: Approximations of $\Gamma \subset \mathbb{R}^3$ and $\pi(\Gamma') \subset \mathbb{R}^3$

### 2.6.3 Algebraic Curves in $\mathbb{C}^2$

Let an algebraic curve $\Gamma \subset \mathbb{C}^2$ is defined by $\{(z_1, z_2) \in \mathbb{C}^2 \mid f(z_1, z_2) = 0\}$ where $f : \mathbb{C}^2 \to \mathbb{C}$ is a polynomial. In their paper [WB96], Weigle and Banks pointed out that $\Gamma$ can be represented by the isosurface of $\bar{f} : \mathbb{R}^4 \to \mathbb{R}^2$, where

$$\bar{f}(x, y, p, q) = \begin{pmatrix} Re\left[f(x + iy, p + iq)\right] \\ Im\left[f(x + iy, p + iq)\right] \end{pmatrix}$$

In this example, $f$ is chosen as $f(z_1, z_2) = z_1^2 + z_2^2 - 1$ and $\Gamma \subset \mathbb{R}^4$ is approximated as the isosurface of a sampled discrete function from $\bar{f}$ in $32^4$ uniform grid. The approximation of $\Gamma \subset \mathbb{R}^4$ is then projected down into $(x, y, p)$ and $(x, y, q)$ coordinates.

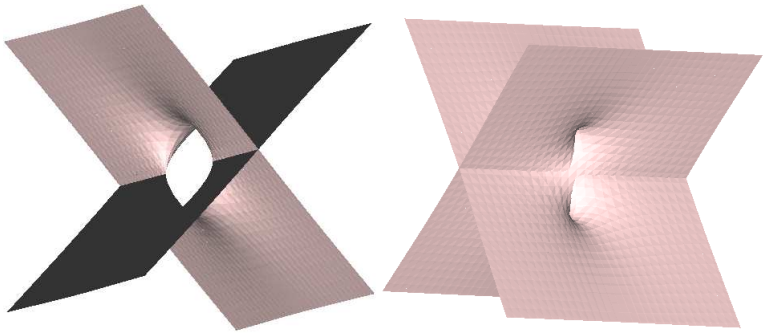

Figure 2.6: Projections of an algebraic curve in $\mathbb{C}^2$

## 2.7 Conclusion

We have presented an isosurfacing algorithm that works in an arbitrary number of dimensions and codimensions. For a discrete function $\phi : \{0, \cdots, N\}^n \to \mathbb{R}^m$, the isosurface $\Gamma$ of $\phi$ is defined as the isosurface of its simplicial interpolant $\hat{\phi} : [0, \cdots, N]^n \to \mathbb{R}^m$. Geometrically $\Gamma$ is a piecewise intersection of a simplex and $m$ hyperplanes. Triangulation tables are given of the intersection of a simplex and a hyperplane. A counting theorem was stated to prove the optimality of the tables. Referencing the tables, $\Gamma$ is constructed as the union of simplices. The construction costs $O(1)$ for each grid cell.

When $n = m + 1$ or $m + 2$, $\Gamma \subset \mathbb{R}^n$ is projected down into $\mathbb{R}^3$ and can be visualized. A simple formula is introduced calculating the normal vector field of the projected surface, when $n = m + 2$.

# CHAPTER 3

# Local Level Set Method

A new method is presented for numerically capturing a moving interface of arbitrary dimension and codimension. The method is named the '*local level set method*', since it localizes the *level set method* near the interface to significantly reduce the computational expense of the level set method.

Following the framework of the level set method, an interface is implicitly represented as the zero level set of a vector valued function. A spatial tree structure is used to locally sample the vector valued function near the interface. Using a Lipschitz stable interpolation and a semi-Lagrangian scheme, our method is stable under both the maximum norm and the Lipschitz semi norm. Due to this stability, the method does not need to reinitialize a level set function. Several numerical examples with high codimension are successfully tested.

## 3.1 Introduction

The *level set method* in [OS88] has been a successful tool for simulating a moving interface of codimension one, because of its simplicity and efficiency. Its successful applications include multiphase flows [SSO94, FAX99], surface reconstructions [ZOF01, ZOM00], and image processing [ROF92, CV01]. One drawback of the level set method is its high computational expense because it expands the domain of computation from the interface to a grid in one higher dimension. To reduce the cost, two main approaches have been employed. One approach is to restrict the domain of a uniform grid near the interface [PMZ99, AS95]. The other approach uses a multi-resolution grid to enable high resolution only near the interface using a spatial tree structure, a so called *quadtree* in $\mathbb{R}^2$ and an *octree* in $\mathbb{R}^3$ [Str99b, DMR01].

The level set method of codimension one [OS88] was theoretically extended to simulate a moving interface of arbitrary codimension [AS96], where an interface is implicitly represented as the zero level set of a *scalar* valued function, and applied e.g., to a medical active contouring of codimension two [LFG00]. However, the theoretical extension to higher codimension is unstable for locating isosurfaces. The instability was fixed in [BCM99] by representing the isosurface as the zero

level set of a *vector* valued function, and successfully applied to a moving curve in $\mathbb{R}^3$ with geometry dependent speed [BCM99] and recently to capturing a one dimensional wavefronts in $\mathbb{R}^3$ and two dimensional wavefronts in $\mathbb{R}^5$ [OCK02].

A drawback of the level set method in high codimension is its inefficiency by its expanding the domain of computation from the interface to a grid in higher dimension. It is the purpose of this paper to introduce a numerical method overcoming this drawback, while keeping the simplicity and versatility of the level set method. Our method originates from [Str99b] in using a spatial tree structure and a semi-Lagrangian scheme, and is a generalization of [Str99b], because it can deal with a moving interface of arbitrary codimension and does not require the reinitialization procedure every step.

The key concept of our method is to put more grid points near the interface and less grid points away from the interface, and implemented by using a tree structure that enables a multi-resolution grid. A grid cell keeps being split if it is near to the interface. The splitting condition can be easily formulated using the Lipschitz constant of the level set function and the magnitude of the function values, as shown in [TCO03, Str99b]. Another key concept is to make all the building blocks in our method Lipschitz stable, because the sampling process in our method heavily relies on the Lipschitz constant of a vector valued function.

A sampling algorithm is presented in Section 3.2 that adaptively samples a vector valued function. In Section 3.3, an interpolation algorithm reconstructs a Lipschitz continuous function from the sampled function. In Section 3.4, an evolution algorithm of a level set function is introduced that is Godunov-type, i.e., a combination of the sampling algorithm in Section 3.2, a approximate solution operator, and the interpolation algorithm in Section 3.3. A practical implementation of our method is given in Section 3.5. Our method is tested with several examples in Section 3.6.

## 3.2 Adaptive Sampling

Let a vector valued function $\vec{\phi} : \mathbb{R}^d \to \mathbb{R}^c$ be given with its level set $\Gamma \subset \mathbb{R}^d$. Since the level set method eventually deals with $\Gamma$, not with $\vec{\phi}$, it is desirable to adopt a multi-resolution grid to enable a fine grid near $\Gamma$ and a coarse grid away from $\Gamma$. For this purpose, we employ a spatial tree structure, called a Quadtree in $\mathbb{R}^2$ and a Octree in $\mathbb{R}^3$, which has been a very successful tool in many areas in which multi-resolution is needed [Sam90b, Sam90a].

If the interface $\Gamma(t) \subset \mathbb{R}^d$ moves with velocity $\vec{V} : \mathbb{R}^d \to \mathbb{R}^d$, the level set method implicitly tracks the interface as the zero level set of $\vec{\phi}(t) : \mathbb{R}^d \to \mathbb{R}^c$ by solving a convection equation;

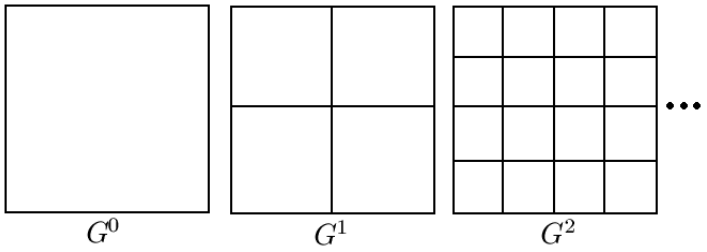


Figure 3.1: Nested grids in $\mathbb{R}^2$

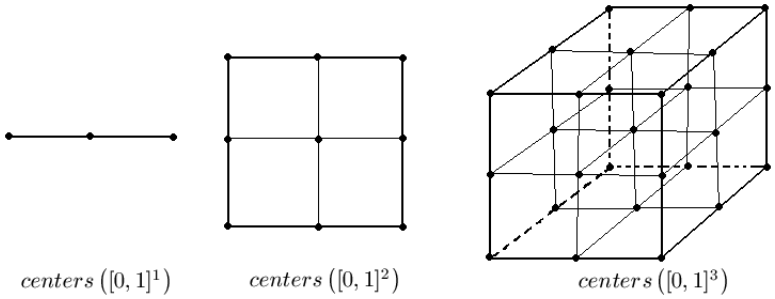


Figure 3.2: Center points of a unit cube

$$\vec{\phi}_t + \left(\vec{V} \cdot \nabla\right) \vec{\phi} = 0$$

Since the solution of this equation stays Lipschitz continuous if the initial data is, it is valid to assume that $\vec{\phi} : \mathbb{R}^d \to \mathbb{R}^c$ is Lipschitz continuous, that is

$$Lip(\vec{\phi}) := \sup_{x \neq y \in \mathbb{R}^d} \frac{\left\| \vec{\phi}(x) - \vec{\phi}(y) \right\|_\infty}{\|x - y\|_2} < \infty$$

Let a hierarchical grid $\left\{G^l\right\}_{l=0,\cdots,l_{max}}$ be given in $\mathbb{R}^d$ such that $G^l$ has $2^{l \cdot d}$ number of grid cells of equal size, and $G^{l+1}$ is a refinement of $G^l$. Figure 3.1 illustrates a nested grid in $\mathbb{R}^2$. Given a grid cell $C \in G^l$, let us denote its parent cell in $G^{l-1}$ by prnt$(C)$, the set of its children in $G^{l+1}$ by children$(C)$, its size by size$(C)$, and the set of the center points of all the faces of C by centers$(C)$. For a grid cell $C = \prod_{i=1,\cdots,d}[a_i, b_i]$, centers $(C) = \prod_i \left\{a_i, \frac{a_i+b_i}{2}, b_i\right\}$ and $size(C) = \sqrt{\sum_i (b_i - a_i)^2}$. Figure 3.2 illustrates the set of center points of a unit cube $[0,1]^d$ when $d = 1, 2, 3$.

In the hierarchical grid $\left\{G^l\right\}_{l=0,\cdots,l_{max}}$, we need to determine which cells are

to be sampled. One rule of the determination is to guarantee that a parent cell is sampled whenever its child is sampled. This rule implies that there is always an hierarchical chain of grid cells from the root cell $G^0$ to a sampled cell, and make the domain $D$ form a tree structure, which allows fast access to its elements. The other rule is to ensure high resolution of sampling near the interface $\Gamma$ and low resolution away from $\Gamma$. Satisfying these two rules, a very useful sampling algorithm, known as Whitney decompositions, appeared in [Str99b] such that *"recursively split and sample any cell whose edge length exceeds its minimum distance to* $\Gamma$*"*.

For a general Lipschitz continuous function $\vec{\phi} : \mathbb{R}^d \to \mathbb{R}^c$, the Whitney decompositions can be generalized as

*Recursively split and sample any cell satisfying* $\eta$.

The splitting condition $\eta$ is given by

$$\eta(C) : \min_{v \in centers(C)} \left\| \vec{\phi}(v) \right\|_\infty \leq Lip\left(\vec{\phi}\right) \cdot \frac{size(C)}{4}$$

Algorithm 1 is a concrete formulation of our adaptive sampling algorithm. The algorithm starts with the root grid cell $G^0$, and recursively constructs a subdivision $\mathcal{S} = \{C_i\}$ of $G^0$.

**Algorithm 1** subdivision of $G^0$ according to $\vec{\phi} : \mathbb{R}^d \to \mathbb{R}^c$

| | |
|---|---|
| Input: | $G^0$ and $\vec{\phi} : \mathbb{R}^d \to \mathbb{R}^c$ |
| 1. | $C = G^0$ and $\mathcal{S} = \emptyset$ |
| 2. | $\mathcal{S} = \mathcal{S} \cup \{C\}$ |
| 3. | if $C \notin G^{l_{max}}$ and $\eta(C)$ is *true* |
| 4. | $\mathcal{S} = \mathcal{S} - \{C\}$ |
| 5. | for all $C' \in children(C)$ |
| 6. | go to 2 with $C = C'$ |
| Output: | $\mathcal{S}$ |

From the subdivision $\mathcal{S} = \{C_i\}$ of the root cell $G^0$, a set $D \subset \mathbb{R}^d$ is defined as

$$D = \cup_i \text{centers}(C_i)$$

A sampled function $\mathcal{R}\vec{\phi}$ of $\vec{\phi}$ is accordingly defined as a restriction of $\vec{\phi} : \mathbb{R}^d \to \mathbb{R}^c$ to $D$, i.e. $\mathcal{R}\vec{\phi} \doteq \vec{\phi} \mid_D : D \to \mathbb{R}^c$. The following is a proposition explaining the properties of the sampling algorithm.

**Proposition 3.2.1** *every grid cell intersecting* $\Gamma$ *is contained in the subdivision* $\mathcal{S}$.

**Proof** Given a grid cell $C$ such that $C \cap \Gamma \neq \emptyset$, let $w \in C \cap \Gamma$. Then there exist a $v \in \text{centers}(C)$ such that $\|v - w\|_2 \leq \frac{\text{size}(C)}{4}$, since the maximum distance in a cube is its diagonal size. Then,

$$\begin{aligned} \left\|\vec{\phi}(v)\right\|_\infty = \left\|\vec{\phi}(v) - \vec{\phi}(w)\right\|_\infty &\leq Lip(\vec{\phi}) \cdot \|v - w\|_2 \\ &\leq Lip(\vec{\phi}) \cdot \tfrac{size(C)}{4} \end{aligned}$$

Therefore $\eta(C)$ is true. Since $C \subset \text{prnt}(C)$, $\text{prnt}(C) \cap \Gamma \neq \emptyset$ and $\eta\left(\text{prnt}(C)\right)$ is true. Also every ancestor of $C$ satisfies the splitting condition $\eta$. Since the algorithm 1 recursively constructs from an ancestor to its successor satisfying $\eta$, $C$ is included in the subdivision $\mathcal{S}$. $\square$

By the above Proposition, the highest resolution is guaranteed near $\Gamma$. If every component $\phi_i$ of $\vec{\phi}$ is a signed distance function to its zero level set $\left\{x \in \mathbb{R}^d \mid \phi_i(x) = 0\right\}$, then the splitting condition is equivalent to a statement that the distance from $C$ to $\Gamma$ is smaller than one fourth of the size of $C$. Therefore the memory size in a grid $G^l$ is about the order of $\Gamma$ which is $(d-c)$ dimensional. Let $N := 2^{l_{max}}$, then $\left|D \cap G^l\right| = O(N^{d-c})$ for $0 \leq l \leq l_{max}$. Combining all grids from $l = 0, \cdots, l_{max}$, the total memory size, $|D|$ will be $O(N^{d-c} \cdot log(N))$. Compared to the memory size $O(N^d)$ of the uniform sampling, algorithm 1 significantly saves memory, while maintaining its highest resolution near the interfaces by Proposition 3.2.1.

## 3.3 Interpolation

In Section 3.2, we discussed a sampling procedure from continuous functions to discrete functions. Here we discuss the reverse, an interpolation procedure from discrete to continuous. On a uniformly sampled function, piecewise polynomial interpolation has been one of the best ways to achieve both accuracy and efficiency. However its direct use on a multi-resolution grid would lead to Lipschitz instability. Figure 3.3 shows a case where the piecewise multi-linear interpolation invokes a Lipschitz instability on a multi-resolution grid in $\mathbb{R}^2$.

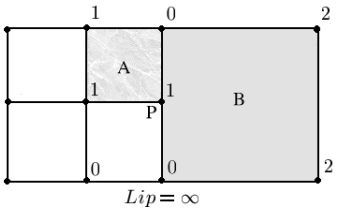


Figure 3.3: The piecewise multi-linear interpolation invokes discontinuity at $P$. The interpolation values at $P$ from grid cells $A$ and $B$ are 1 and 0 respectively.

In Section 3.4, we shall introduce a semi-Lagrangian scheme that is a combination of the adaptive sampling in Section 3.2, approximate solution operator, and the interpolation in this Section. The whole scheme needs to be Lipschitz stable. For this purpose, we present a Lipschitz stable interpolation on a multi-resolution grid.

### 3.3.1 Triangulation

Let $\vec{\psi} : D \to \mathbb{R}^c$ be an adaptively sampled function with a subdivision $\mathcal{S} = \{C_i\}$ of $G^0$ generated by Algorithm 1. To enable a Lipschitz stable interpolation of $\vec{\psi}$, the subdivision $\mathcal{S}$ needs to be refined into a triangulation. For this purpose, we employ a well-known refining algorithm, *pulling* [Lee97]. A pulling of a point $P$ on a subdivision $\mathcal{S} = \{C_i\}$ of $G^0$ results in a subdivision $\mathcal{T}$ of $G^0$ that is obtained by modifying each element of $\mathcal{S}$ as follows,

$$\begin{cases} P \notin C_i \text{ then } C_i \in T \\ P \in C_i \text{ then for every facet } F \text{ of } C_i \text{ not containing } P,\ conv(P, F) \in T \end{cases}$$

Let us give the center points of every $k$-dimensional face of $C_i$ the label $k$, $0 \leq k \leq d$. By sequentially pulling all the center points in order of non-increasing label, the cubic subdivision $\{C_i\}$ is refined to a triangulation, $\mathcal{T} = \{S_j\}$, which is a general triangulation procedure known as a *complete barycentric subdivision* [Lee97, Bay88]. Figure 3.4 shows a cases in $\mathbb{R}^2$. Since the domain $D$ is the union of all center points of each $C_i$, $\mathcal{T} = \{S_j\}$ is a triangulation of the root cell $G^0$with vertices in $D$.

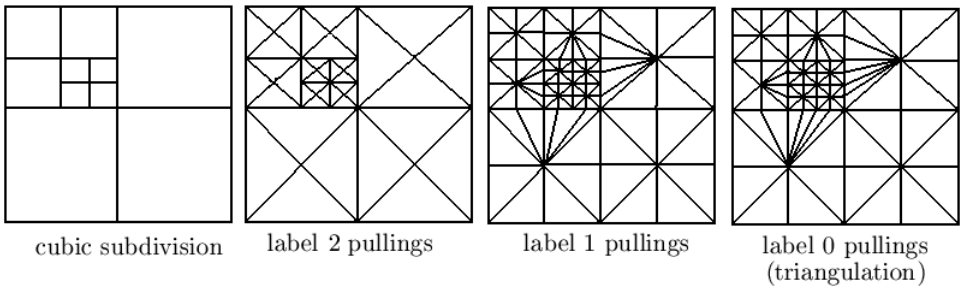


Figure 3.4: Triangulation of a cubic subdivision by pullings in $\mathbb{R}^2$

### 3.3.2 Simplicial Interpolation

In Section 3.3.1, the domain of $\vec{\psi} : D \to \mathbb{R}^c$ was triangulated into simplices $\{T_j\}$. Based on this triangulation, we define an interpolation, or a prolongation $\mathcal{P}\vec{\psi} : \mathbb{R}^d \to \mathbb{R}^c$ as follows,

$$\mathcal{P}\vec{\psi}(x) = \begin{cases} \mathcal{P}\vec{\psi}\,|_{T_j}\,(x) & , x \in T_j \\ \mathcal{P}\vec{\psi}(x^*) & , x \neq G^0 \end{cases}$$

Here $x^*$ denotes the nearest-point projection of $x$ into $G^0$, i.e., the nearest point in $G^0$ to $x$, which is well defined since $G^0$ is convex. Figure 3.5 shows some cases of projections in $\mathbb{R}^2$. $\mathcal{P}\vec{\psi}\,|_{T_j}$ is defined as the unique linear interpolant of $\vec{\psi}$ on the simplex $T_j$. The interpolation $\mathcal{P}$ is Lipschitz stable by the following proposition.

Here we define the Lipschitz constant of a discrete function $\vec{\psi} : D \to \mathbb{R}^c$ as

$$Lip(\vec{\psi}) = \max_{x \neq y \in D} \frac{\left\| \vec{\psi}(x) - \vec{\psi}(y) \right\|_\infty}{\|x - y\|_2}$$

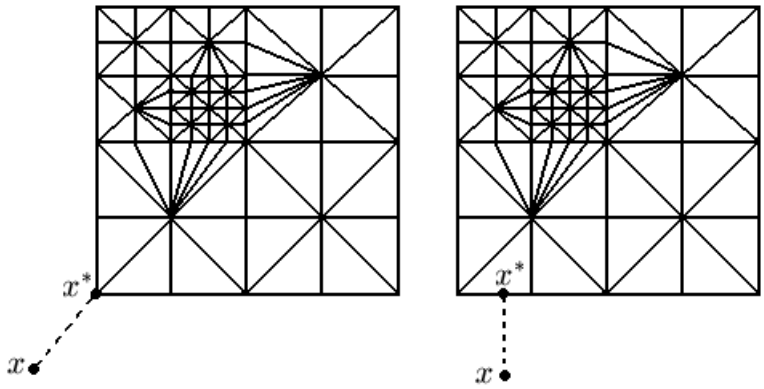


Figure 3.5: Projection of $x$ outside into $x^*$ inside the domain

**Proposition 3.3.1** *The interpolation $\mathcal{P}$ is Lipschitz stable, such that $Lip(\mathcal{P}\vec{\psi}) = Lip(\vec{\psi})$*

**Proof** Since $\mathcal{P}\vec{\psi}$ is a linear interpolant on each simplex $T_j$, $Lip\left(\mathcal{P}\vec{\psi}\mid_{T_j}\right) = Lip\left(\vec{\psi}\mid_{T_j}\right) \leq Lip(\vec{\psi})$. On $T_j \cap T_{j'}$, $\mathcal{P}\vec{\psi}$ has two possible definitions $\mathcal{P}\vec{\psi}\mid_{T_j}$ and $\mathcal{P}\vec{\psi}\mid_{T_{j'}}$. Because $\{T_j\}$ is a triangulation, $T_j \cap T_{j'}$ is also a simplex. Since there exits only one linear interpolant on a simplex, the two definitions must be the same. Therefore $\mathcal{P}\vec{\psi}$ is continuous on $G^0$. Since $Lip\left(\mathcal{P}\vec{\psi}\mid_{T_j}\right) \leq Lip(\vec{\psi})$ and $\mathcal{P}\vec{\psi}$ is continuous on $G^0$,

$$Lip\left(\mathcal{P}\vec{\psi}\mid_{G^0}\right) \leq Lip(\vec{\psi})$$

For any $x, y \in \mathbb{R}^d$, $\|x^* - y^*\|_2 \leq \|x - y\|_2$. Therefore,

$$\left\|\mathcal{P}\vec{\psi}(x) - \mathcal{P}\vec{\psi}(y)\right\|_\infty = \left\|\mathcal{P}\vec{\psi}(x^*) - \mathcal{P}\vec{\psi}(y^*)\right\|_\infty \leq Lip\left(\mathcal{P}\vec{\psi}\mid_{G^0}\right) \cdot \|x - y\|_2$$

So, $Lip(\mathcal{P}\vec{\psi}) \leq Lip(\vec{\psi})$. Since the domain of $\vec{\psi}$ is a subset of the domain of $P\vec{\psi}$, $Lip(\mathcal{P}\vec{\psi}) = Lip(\vec{\psi})$.□

## 3.4 Evolution

In this Section, we present an algorithm numerically capturing an interface in high dimension and codimension. Let us assume that an interface $\Gamma \subset \mathbb{R}^d$ of

codimension $c$ is moving with velocity $\vec{V} : \mathbb{R}^d \to \mathbb{R}^d$. It can be implicitly represented as the zero level set of $\vec{\phi}(x,t) : \mathbb{R}^d \times \mathbb{R}^+ \to \mathbb{R}^c$, where each component of the $\vec{\phi}$ satisfies the so-called level set equation

$$\frac{\partial}{\partial t}\phi_i + \vec{V} \cdot \nabla \phi_i = 0$$

for $i = 1, \ldots, c$. In a uniform grid, the level set equation is usually discretized with higher order accurate ENO or WENO schemes in space and the Runge-Kutta methods in time [HEO87, LOC96], but in a multi-resolution grid, neighboring points may not exist, which makes it hard to apply conventional finite difference schemes. Because of that, a semi-Lagrangian scheme has been used for discretizing the level set equation on a multi-resolution grid when the codimension is one [Str99b]. We extend the techniques of [Str99b] to higher dimension and codimension.

Given $\vec{\phi}^n : D^n \to \mathbb{R}^c$, the next level set function $\vec{\phi}^{n+1} : D^{n+1} \to \mathbb{R}^c$ is defined as

$$\vec{\phi}^{n+1} = (\mathcal{R} \circ \mathcal{S} \circ \mathcal{P})\vec{\phi}^n$$

Here, $\mathcal{P}$ is the interpolation operator in Section 3.3, $S$ is an approximate solution operator, and $\mathcal{R}$ is the adaptive sampling operator in Section 3.2. Via the characteristic curve of the level set equation, $\mathcal{S}$ is equivalent to an ODE solver to the backward characteristic ODE, $\dot{x} = -\vec{V}(x,t)$. We choose the Euler scheme for an ODE solver, $\mathcal{S}$, then we have

$$(\mathcal{S}\mathcal{P}\vec{\phi}^n)(x) = (\mathcal{P}\vec{\phi}^n)(x - \Delta t \vec{V}(x, t^n))$$

To adaptively sample $\vec{\phi}^{n+1}$ from $\mathcal{S}\mathcal{P}\vec{\phi}^n : \mathbb{R}^d \to \mathbb{R}^m$, we need an estimate on the Lipschitz constant of $\mathcal{S}\mathcal{P}\vec{\phi}^n$, which is given in the following proposition.

**Proposition 3.4.1**

$$Lip(\mathcal{S}\mathcal{P}\vec{\phi}^n) \leq Lip(\vec{\phi}^n) \cdot \left(1 + \Delta t \cdot \sup_{x \in \mathbb{R}^d} \left\| \nabla \vec{V}(x, t^n) \right\|_2 \right)$$

**Proof** By the definitions of $S$ and $P$,

$$\begin{aligned}
\left\| \mathcal{S}\mathcal{P}\vec{\phi}^n(x) - \mathcal{S}\mathcal{P}\vec{\phi}^n(y) \right\|_\infty &= \left\| P\vec{\phi}^n \left( x - \Delta t \vec{V}(x, t^n) \right) - P\vec{\phi}^n \left( y - \Delta t \vec{V}(y, t^n) \right) \right\|_\infty \\
&\leq Lip \left( P\vec{\phi}^n \right) \cdot \left\| x - y + \Delta t \left( \vec{V}(x, t^n) - \vec{V}(y, t^n) \right) \right\|_2 \\
&\leq Lip \left( \vec{\phi}^n \right) \cdot \left( \|x - y\|_2 + \Delta t \cdot \left\| \vec{V}(x, t^n) - \vec{V}(y, t^n) \right\|_2 \right)
\end{aligned}$$

Applying the mean value theorem to the vector valued function $\vec{V}$, we have

$$\left\|\vec{V}(x,t^n)-\vec{V}(y,t^n)\right\|_2 \le \sup_{z\in\mathbb{R}^d}\left\|\nabla\vec{V}(z,t^n)\right\|_2 \cdot \|x-y\|_2$$

Here, $\nabla\vec{V}$ denotes the deformation matrix of $\vec{V}$, and $\left\|\nabla\vec{V}\right\|_2$ is the matrix 2−norm that is the maximal singular value of the matrix $\nabla\vec{V}$. The proposition follows from these two inequalities. □

## 3.5 Implementation

We discuss an implementation of our method that consists of an adaptive sampling in Section 3.2, a Lipschitz stable interpolation in Section 3.3, and an evolution algorithm in Section 3.4. Since the evolution is a combination of sampling and interpolation, it is enough to discuss implementations of sampling and interpolation.

### 3.5.1 Implementation of Sampling

Since the sampling algorithm 1 recursively subdivides the coarsest cube $G^0 \in \mathbb{R}^d$, a $2^d$ branched tree is a natural choice for the data structure of the sampling. Possible implementations of this tree structure are thoroughly discussed in [Sam90b, Sam90a]. Among them, *region based trees* and *matrix based trees* are the most appropriate for our purpose. The region based tree is fast for interpolation but requires more memory for construction, while the matrix based tree is slow for interpolation but requires less memory for construction. Each point in a region based tree may be multiply defined, possibly $2^d$ times, so it is hard to modify the tree, but, each point in matrix-based tree is singly defined, so it becomes easy to modify. Our algorithms, which consists of sampling and interpolation, do not need any modification, but only need new creations of adaptively sampled functions. For these reasons, we take a region-based tree as a choice of the tree implementation.

Unlike the uniform sampling, we can not predict the memory size of the adaptive sampling before doing it. Therefore the programming languages with dynamic memory allocations are preferred, such as C++ or Java. We have chosen C++ for better performance. An implementation of region-based tree is given in the following, when the adaptive sampling is performed to a vector valued function $\vec{\phi}:\mathbb{R}^d\to\mathbb{R}^c$ with $d=5$ and $c=3$.

```
struct Node {
```

```
    int i1,i2,i3,i4,i5;
    int size;
};


struct Leaf : public Node {
    float vs1[3][3][3][3][3];
    float vs2[3][3][3][3][3];
    float vs3[3][3][3][3][3];
};


struct Branch : public Node {
    Node* children[2][2][2][2][2];
};
```

The quantity node represents a cube, $\left\{x \in \mathbb{R}^d \mid i_j \leq x_j \leq i_j + size,\ j = 1, \cdots, d\right\}$, and has two possible types, Leaf and Branch. If a node does need to be split, it takes a type of Branch that has $2^d$ number of children. If not, it takes a type of Leaf, and its center points are sampled and stored. Since $\text{centers}(C) = [0, \frac{1}{2}, 1]^d$ for a cube $C = [0, 1]^d$, it would sample $3^d \cdot c$ number of center points for each leaf.

### 3.5.2 Barycentric Interpolation on a Cube

Any cube $C \subset \mathbb{R}^d$ is affinely isomorphic to $[-1, 1]^d$ under translations and scalings. So, we need only to discuss an interpolation on $[-1, 1]^d$, and the general case will follow by the affine isomorphism. Since $\text{centers}([-1, 1]^d) = \{-1, 0, 1\}^d$, let us assume a function $\vec{\psi} : \{-1, 0, 1\}^d \to \mathbb{R}^c$.

Although the barycentric interpolation is defined as a piecewise linear interpolant on the triangulation of the domain in Section 3.3, we do not explicitly triangulate a cube, but employ the following very efficient algorithm.

Given $x \in [-1, 1]^d$, the coordinates of $x$ are sorted with a permutation $J$ of $\{1, \cdots, d\}$ such that

$$1 \geq \left|x_{J(1)}\right| \geq \cdots \geq \left|x_{J(d)}\right| \geq 0$$

For a notational convenience, let us set $x_{J(0)} = 1$ and $x_{J(d+1)} = 0$ to have $\left|x_{J(0)}\right| \geq \cdots \geq \left|x_{J(d+1)}\right|$. Define $P_0 = \vec{0}$ and

$$P_i = P_{i-1} + \text{sgn}\left[x_{J(i)}\right] \vec{e}_{J(i)}, \text{ for } i = 1, \cdots, d$$

Here $\vec{e}_i$ denotes the canonical $i^{th}$ unit vector, and $\mathrm{sgn}(x)$ refers to the signum of $x$, i.e. $H(x) - H(-x)$ with the Heaviside function $H$. Then we have the following barycentric decomposition of $x$.

$$\begin{aligned} x &= \sum_{i=1}^{d} x_i \vec{e}_i \\ &= \sum_{i=1}^{d} x_{J(i)} \vec{e}_{J(i)} \\ &= \sum_{i=1}^{d} \left|x_{J(i)}\right| \cdot \mathrm{sgn}\left[x_{J(i)}\right] \vec{e}_{J(i)} \\ &= \sum_{i=1}^{d} \left|x_{J(i)}\right| \cdot [P_i - P_{i-1}] \\ &= \sum_{i=0}^{d} \left(\left|x_{J(i)}\right| - \left|x_{J(i+1)}\right|\right) P_i \end{aligned}$$

Since the barycentric interpolant $\mathcal{P}\vec{\psi} : [-1,1]^d \to \mathbb{R}^c$ is a linear interpolant on a simplex $conv\,[P_0, \cdots, P_d]$, we have the following formula.

$$\mathcal{P}\vec{\psi}(x) = \sum_{i=0}^{d} \left[\left|x_{J(i)}\right| - \left|x_{J(i+1)}\right|\right] \vec{\psi}\,(P_i)$$

Note that $\vec{\psi}\,(P_i)$ is well defined, since $P_i \in \{-1, 0, 1\}^d$ for all $i = 0, \cdots, d$.

### 3.5.3 Barycentric Interpolation on a Multi-Resolution Grid

Let an adaptively sampled function $\vec{\psi} : D \to \mathbb{R}^c$ with $D = \cup_i \mathrm{centers}(C_i)$ be given. This Section discusses an implementation of its barycentric interpolation $\mathcal{P}\vec{\psi} : \mathbb{R}^d \to \mathbb{R}^c$. Since $\mathcal{P}\vec{\psi}(x)$ for $x \notin G^0$ is defined as $\mathcal{P}\vec{\psi}(x^*)$ with $x^* \in G^0$, it is enough to discuss an interpolation procedure of $\mathcal{P}\vec{\psi}(x)$ with $x \in G^0$.

$\mathcal{P}\vec{\psi} : \mathbb{R}^d \to \mathbb{R}^c$ is defined as the piecewise linear interpolation on the triangulation of the domain $D$ of $\vec{\psi}$. Since the triangulation is expensive to implement, it is desired to circumvent the triangulation as done in the previous Section. The complete barycentric subdivision, which is the algorithm triangulating $D$, sequentially refines the subdivision $\{C_i\}$ of $G^0$ by pulling all the center points of $\{C_i\}$ of label $d, \cdots, 0$. As a result, a cube may be subdivided by any neighborhood sharing a face of dimension $0, \cdots, d-1$. Since so many ($2^{2d}$ in the worst case) neighboring cubes are involved for the interpolation on a cube, the interpolation procedure becomes expensive if we access all the neighboring cubes.

If we stop the pullings of the complete barycentric subdivision at the label $d-1$, it may not be a triangulation in general and therefore invoke discontinuities, but this provides a very efficient algorithm to implement and does not degrade numerical results too much according to our experiments in Section 3.6. In Figure 3.4, the fourth picture indicates the complete barycentric subdivision, and the third one indicates the partial barycentric subdivision that stops pullings at the

label $d-1$. From these reasons, let us employ this partial barycentric subdivision to obtain the following efficient algorithm.

Given $x \in G^0$, it is not difficult to access the smallest cube $C$ containing $x$ in $\{C_i\}$, because the subdivision $\{C_i\}$ forms a tree structure that enables a fast access. Let us denote by $P$ the center of $C$, by $Q$ the intersection of the boundary of $C$ and the line connecting $P$ and $x$, and by $C'$ the neighborhood of $C$ sharing the $(d-1)$ dimensional face with $C$ and containing $Q$. Figure 3.6 illustrates a case in $\mathbb{R}^2$. If $C$ is smaller than $C'$, then $C$ needs to be subdivided by some center points of $C'$. Otherwise, it would be enough to use the barycentric interpolation only on $C$, which is already discussed in the previous Section. Here follows a concrete formulation of the interpolation procedure.

Let the cube $C$ be given with coordinates; $C = \{y \in \mathbb{R}^d | a_i \leq y_i \leq b_i\}$. Its center point $P$ is given by $P_i = \frac{a_i+b_i}{2}$ for $i = 1, \cdots, d$. Let us define $\lambda \in \mathbb{R}$ as

$$\lambda = \min_{1 \leq i \leq d} \frac{b_i - P_i}{|x_i - P_i|}$$

Then $Q$, the intersection point between $\partial C$ and the line connecting $P$ and $x$ is given by

$$Q = P + \lambda(x - P)$$

Let $C'$ be the smallest cube in $\{C_i\}$ containing $Q$, then

$$\mathcal{P}\vec{\psi}(x) = \begin{cases} \left(\mathcal{P}\vec{\psi}\right)|_C(x) & \text{if size}(C) \leq \text{size}(C') \\ \frac{1}{\lambda} \cdot \left(\mathcal{P}\vec{\psi}\right)|_C(P) + \left(1 - \frac{1}{\lambda}\right) \cdot \left(\mathcal{P}\vec{\psi}\right)|_{C'}(Q) & \text{if size}(C) > \text{size}(C') \end{cases}$$

Barycentric interpolations on a cube, $\mathcal{P}\vec{\psi}\,|_C$ and $\mathcal{P}\vec{\psi}\,|_{C'}$ were given in Section 3.5.2.

## 3.6 Numerical Examples

Every example was programmed in C++ and run on a PC with 2.2GHz CPU and 2GB memory. Since the adaptive sampling in Section 3.2 is uniform near the interface by Proposition 3.2.1, an isosurfacing algorithm [Min03] on a uniform grid was used to isosurface and visualize the level set of adaptively sampled functions.

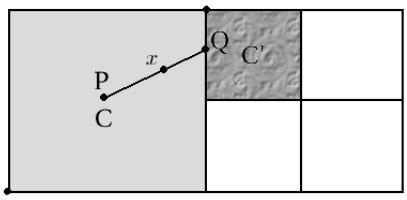


Figure 3.6: Weak Barycentric Interpolation on a Dyadic Grid

| grid | $L^1$error | rate |
|---|---|---|
| $128^2$ | 0.0286 | - |
| $256^2$ | 0.0197 | 0.54 |
| $512^2$ | 0.0132 | 0.58 |
| $1024^2$ | 0.00928 | 0.51 |

Table 3.1: Accuracy test

### 3.6.1 Accuracy Test

Our method employed the piecewise linear interpolation in space and the Euler method in time, which would result in a first-order accurate method in a uniform grid. Since a multi-resolution grid is used for the space discretization in our method, the course parts of the grid negatively affects the fine parts, which will weaken the first order accuracy. Here, we numerically test the accuracy of our method.

As a test example, a circle of center $\left(-\frac{1}{3}, -\frac{1}{3}\right)$ and of radius $\frac{1}{3}$ is advected with a constant velocity $\vec{V} = (1, 1)$ until $T = 0.625$. Since the deformation matrix $\nabla\vec{V}$ is the zero matrix, the Lipschitz constant stayed the initial Lipschitz constant for the whole time steps. Table 3.1 shows that the convergence rate of our method is about a half.

### 3.6.2 Wave Reflections in $\mathbb{R}^2$

It is well known that a high frequency wave behaves like a particle, which can be nicely represented as the Eikonal equation,

$$u_t + c(x) \cdot \|\nabla u\| = 0$$

However, the classical viscosity solution of this equation does not allow super-

positions that are natural in the wave phenomenon. To overcome this deficiency, Osher et al [OCK02] substituted the Eikonal equation with its characteristic equation that captures the wave fronts in phase space, which is

$$\vec{\phi}_t + (\vec{V} \cdot \nabla)\vec{\phi} = 0$$

with velocity

$$\vec{V}(x, y, \theta) = (\cos\theta, \sin\theta, 0)$$

As a test example, we take an initial wave front as a circle of center at $(0, 0)$ and of a radius $\frac{1}{2}$ that is implicitly represented as the zero level set of

$$\vec{\phi}_0(x, y, \theta) = \left(x - \frac{1}{2}\cos\theta, y - \frac{1}{2}\sin\theta\right)$$

We take a domain of $[-1, 1]^2 \times [-\pi, \pi]$ whose boundary behaves a mirror, or a reflector of the wave. To incorporate the reflection, we modify the interpolation procedure in Section 3.3. Given $\vec{\phi} : D \to \mathbb{R}^2$ with $(x, y, \theta) \notin [-1, 1]^2 \times [-\pi, \pi]$, $\mathcal{P}\vec{\phi}(x, y, z)$ is defined as $\mathcal{P}\vec{\phi}(x^*, y^*, \theta^*)$ with $(x^*, y^*, \theta^*) \in [-1, 1]^2 \times [-\pi, \pi]$ such that

$$x^* = \begin{cases} 2 - x, & x > 1 \\ -2 - x, & x < -1 \\ x, & |x| \le 1 \end{cases} \qquad y^* = \begin{cases} 2 - y, & y > 1 \\ -2 - y, & y < -1 \\ y, & |y| \le 1 \end{cases}$$

$$\theta^* = \begin{cases} -\theta, & |x| \le 1 \text{ and } |y| > 1 \\ \pi - \theta, & |x| > 1 \text{ and } |y| \le 1 \\ \pi + \theta, & |x| > 1 \text{ and } |y| > 1 \\ \theta, & |x| \le 1 \text{ and } |y| \le 1 \end{cases}$$

The time step $\Delta t$ was chosen as $\Delta t = \frac{\sqrt{\Delta x^2 + \Delta y^2 + \Delta\theta^2}}{2}$. The Lipschitz constants of $\vec{\phi}(x, y, \theta, t^n)$ was increased by $L^{n+1} = L^n \cdot (1 + \Delta t)$ with $L^0 = \frac{\sqrt{5}}{2}$. The evolution was simulated until $T = 1.5$.

| | Local Level Set Method | | | | Level Set Method | |
|---|---|---|---|---|---|---|
| grid | space(MB) | rate | time(sec) | rate | space(MB) | rate |
| $128^3$ | 61.4 | - | 250 | - | 33.6 | - |
| $256^3$ | 138 | 2.2 | 1160 | 4.6 | 268 | 8.0 |
| $512^3$ | 290 | 2.1 | 5045 | 4.3 | 2147 | 8.0 |
| $1024^3$ | 594 | 2.0 | 21068 | 4.2 | 17179 | 8.0 |

Table 3.2: Reflections in $\mathbb{R}^2$

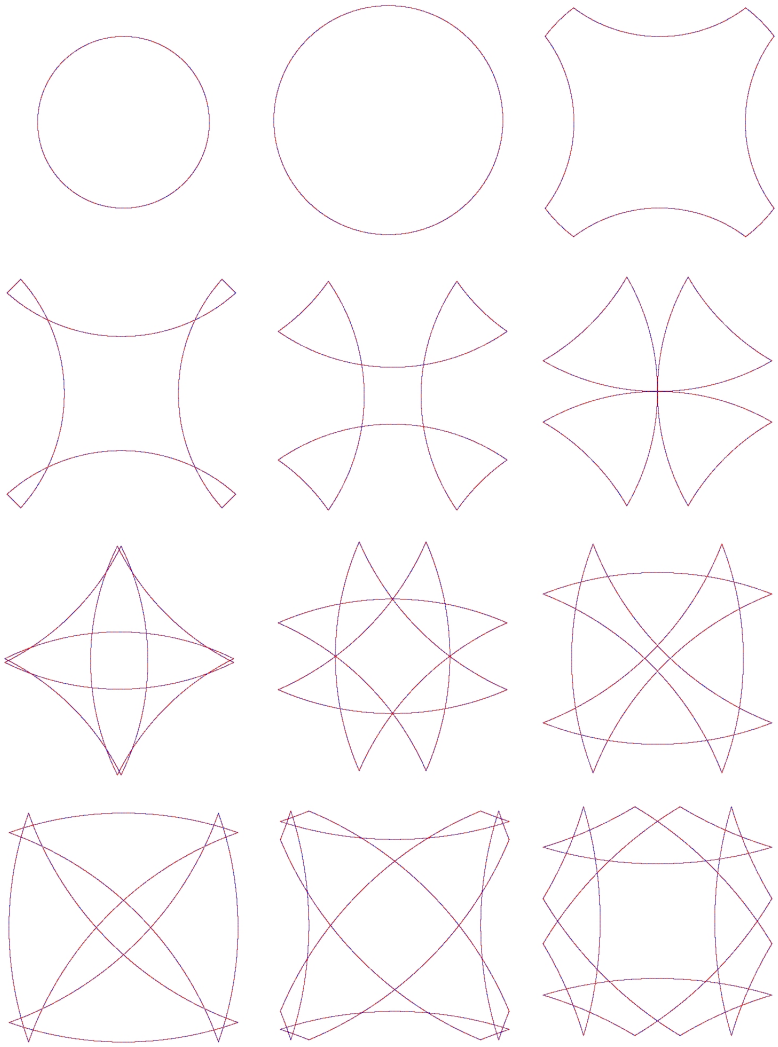

Figure 3.7: Reflections in $\mathbb{R}^2$ simulated on a $1024^3$ grid

### 3.6.3 Wave Reflections in $\mathbb{R}^3$

This example is an extension of the previous example to $\mathbb{R}^3$. Osher et al [OCK02] substituted the Eikonal equation with its characteristic equation that captures the wave fronts in phase space, which is

$$\vec{\phi}_t + (\vec{V} \cdot \nabla)\vec{\phi} = 0$$

with velocity

$$\vec{V}(x, y, z, \theta, \psi) = (\cos\theta \sin\psi, \sin\theta \sin\psi, \cos\psi, 0, 0)$$

As a test example, we take an initial wave front as a sphere of center at $(0, 0, 0)$ and of a radius $\frac{1}{2}$ that is implicitly represented as the zero level set of

$$\vec{\phi}_0(x, y, z, \theta, \psi) = \left(x - \frac{1}{2}\cos\theta \sin\psi, y - \frac{1}{2}\sin\theta \sin\psi, z - \frac{1}{2}\cos\psi\right)$$

We take a domain of $[-1, 1]^3 \times [-\pi, \pi] \times [0, \pi]$ whose spatial boundary, $\partial[-1, 1]^3$ behaves a mirror, or a reflector of the wave. To incorporate the reflection, we modify the interpolation procedure in Section 3.3. Given $\vec{\phi} : D \to \mathbb{R}^3$ with $(x, y, z, \theta, \psi) \notin [-1, 1]^3 \times [-\pi, \pi] \times [0, \pi]$, $\mathcal{P}\vec{\phi}(x, y, z, \theta, \psi)$ is defined as $\mathcal{P}\vec{\phi}(x^*, y^*, z^*, \theta^*, \psi^*)$ with $(x^*, y^*, z^*, \theta^*, \psi^*) \in [-1, 1]^3 \times [-\pi, \pi] \times [0, \pi]$ such that

$$a^* = \begin{cases} 2 - a, & a > 1 \\ -2 - a, & a < -1 \\ a, & |a| \le 1 \end{cases} \quad \text{for } a = x, y, z$$

$$\theta^* = \begin{cases} -\theta, & |x| \le 1 \text{ and } |y| > 1 \\ \pi - \theta, & |x| > 1 \text{ and } |y| \le 1 \\ \pi + \theta, & |x| > 1 \text{ and } |y| > 1 \\ \theta, & |x| \le 1 \text{ and } |y| \le 1 \end{cases} \qquad \psi^* = \begin{cases} \pi - \psi, & |z| > 1 \\ \psi, & |z| \le 1 \end{cases}$$

Time step $\Delta t$ was chosen as $\Delta t = 0.5 \cdot \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2 + \Delta\theta^2 + \Delta\psi^2}$. The Lipschitz constants of $\vec{\phi}(x, y, z, \theta, \psi, t^n)$ was increased by $L^{n+1} = L^n \cdot (1 + \Delta t)$ with $L^0 = \frac{\sqrt{5}}{2}$. The evolution was simulated until $T = 0.15$.

| | Local Level Set Method | | | | Level Set Method | |
|---|---|---|---|---|---|---|
| grid | space(MB) | rate | time(sec) | rate | space(MB) | rate |
| $8^5$ | 3.0 | - | 3 | - | 0.4 | - |
| $16^5$ | 53 | 17.7 | 52 | 17.4 | 12.6 | 32.0 |
| $32^5$ | 284 | 5.4 | 240 | 4.0 | 403 | 32.0 |
| $64^5$ | 1092 | 4.2 | 4257 | 17.7 | 12885 | 32.0 |

Table 3.3: Reflections in $\mathbb{R}^3$

## 3.7 Conclusion

We have introduced a new method that can track an interface in arbitrary dimension and codimension. By localizing the level set method near the interface, our new method significantly reduced the high computational expense of the level set method, while keeping the simplicity and efficiency of the level set method.

Our method is stable under both the maximum norm and the Lipschitz seminorm. Due to its Lipschitz stability, the method does not need any reinitialization of level set function. However, without reinitializations, the Lipschitz constant of a level set function might keep increasing by the estimate in Section 3.4, which makes the method a bit less efficient. Numerical results show that our method is a half order accurate.

In the future, the author expects to employ the reinitialization algorithms in [OCK02, Str99a] to the tree structure in our method to improve the memory efficiency, and intends to improve the accuracy of our method.

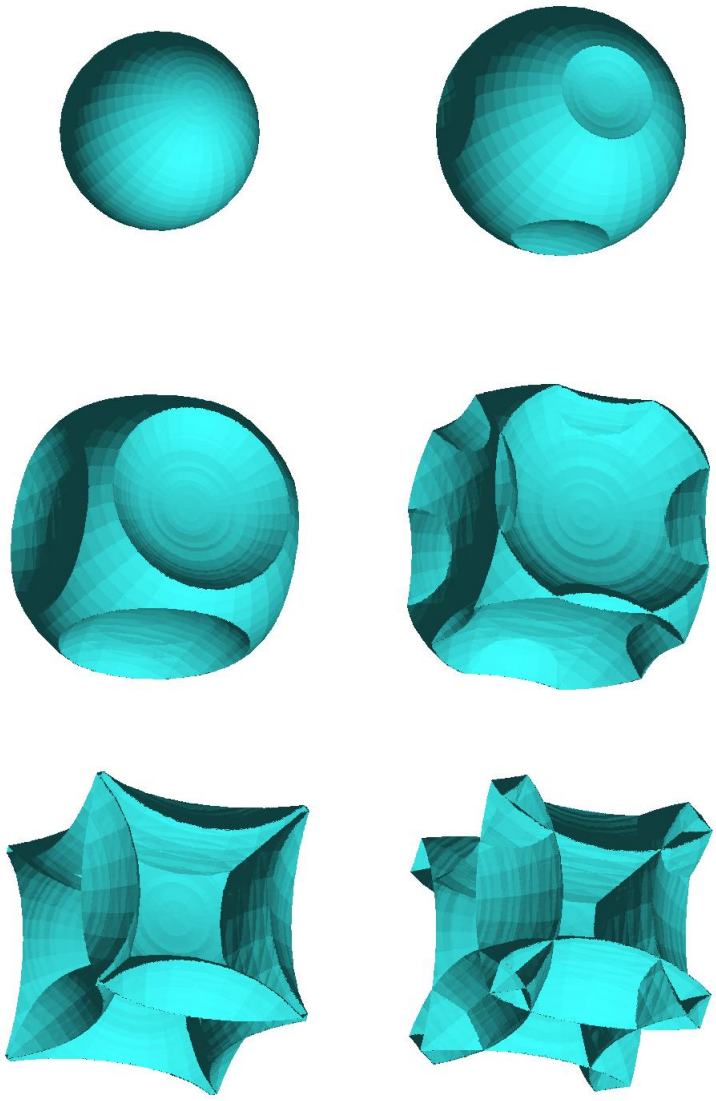

Figure 3.8: Reflections in $\mathbb{R}^3$ simulated on a $32^5$ grid until 1.0 sec

# References


[AS85] E. L. Allgower and P. H. Schmidt. "An algorithm for piecewise-linear approximation of an implicitly defined manifold." *SIAM J. Numer. Anal.*, **22**:322–346, 1985.

[AS95] D. Adalsteinsson and J. Sethian. "A fast level set method for propagating interfaces." *J. Comput. Phys.*, **118**:269–277, 1995.

[AS96] L. Ambrosio and M. Soner. "Level set approach to mean curvature flow in arbitrary codimension." *J. Diff. Geom.*, **43**:693–737, 1996.

[Bay88] M. M. Bayer. "Barycentric subdivisions." *Pacific J. Math.*, **135**:1–16, 1988.

[BCM99] P. Butchard, L.-T. Cheng, B. Merriman, and S. Osher. "Motion of curves in three spatial dimensions using a level set approach." *J. Comput. Phys.*, **170**:720–741, 1999.

[BK96] C. Silva B. P. Carneiro and A. E. Kaufman. "Tetra-cubes: an algorithm to generate 3d isosurfaces based upon tetrahedra." *Anais do IX SIBGRAPI*, pp. 205–210, 1996.

[BWC00] P. Bhaniramka, R. Wenger, and R. Crawfis. "Isosurfacing in higher dimension." *Proceedings of visualization*, pp. 267–273, 2000.

[CLO03] L.-T. Cheng, H. Liu, and S. Osher. "Computational high-frequency wave propagation using the level set method, with applications to the semi-classical limit of schrodinger equations." *Comm. Math. Sci.*, **1**:593–621, 2003.

[CV01] T. Chan and L. Vese. "Active contours without edges." *IEEE Trans. on Image Processing*, **10**:266–277, 2001.

[DMR01] M. Droske, B. Meyer, M. Rumpf, and C. Schaller. "An adaptive level set method for medical image segmentation." *Lecture Notes in Computer Science IPMI*, pp. 416–422, 2001.

[ER03] B. Engquist and O. Runborg. "Computational high frequency wave propagation." *Acta Numerica*, pp. 181–266, 2003.

[ERT02] B. Engquist, O. Runborg, and A.-K. Tornberg. "High frequency wave propagation by the segment projection method." *J. Comput. Phys.*, **178**:373–390, 2002.

[FAX99] R. Fedkiw, T. Aslam, and S. Xu. "The ghost fluid method for deflagration and detonation discontinuities." *J. Compt. Phys.*, **154**:393–427, 1999.

[HA96] R. B. Hughes and M. R. Anderson. "Simplexity of the cube." *Discrete Math.*, **158**:99–150, 1996.

[Hai91] M. Haiman. "A simple and relatively efficient triangulation of the n-cube." *Discrete Comput. Geom.*, **6**:287–289, 1991.

[HEO87] A. Harten, B. Engquist, S. Osher, and S. Chakravarthy. "Uniformly high-order accurate essentially non-oscillatory schemes III." *J. Comput. Phys.*, **71**:231–303, 1987.

[Kuh60] H. W. Kuhn. "Some topological lemmas in topology." *IBM J. Res. Develop.*, **4**:518–524, 1960.

[LC87] W. E. Lorenson and H. E. Cline. "Maching cubes: a high resolution 3d surface reconstruction algorithm." *Comp. Graphics.*, **21**:163–169, 1987.

[Lee97] C. W. Lee. "Subdivisions and triangulations of polytopes." *Handbook of Discrete and Computational Geometry, CRC Press LLC*, 1997.

[LFG00] L. Lorigo, O. Faugeras, W. Grimson, R. Keriven, R. Kikinis, A. Nabavi, and C. Westin. "Codimension-two geodesic active contours for the segmentation of tubular structures." *Conf. on Computer Vision and Pattern Recognition, IEEE 2000*, pp. 444–451, 2000.

[LOC96] X.-D. Liu, S. Osher, and T. Chan. "Weighted essentially non-oscillatory schemes." *J. Comput. Phys.*, **126**:202–212, 1996.

[Min03] C. Min. "Simplicial isosurfacing in arbitrary dimension and codimension." *J. Comput. Phys.*, **190**:295–310, 2003.

[MSS94] C. Montani, R. Scateni, and R. Scopigno. "A modified look-up table for implicit disambiguation of Marching Cubes." *Visual Computer*, **10**:353–355, 1994.

[Nyq28] H. Nyquist. "Certain topics in telegraph transmission theory." *AIEE Trans.*, **47**:617–644, 1928.

[OCK02] S. Osher, L.-T. Cheng, M. Kang, H. Shim, and Y.-H. Tsai. "Geometric optics in a phase space based level set and eulerian framework." *J. Comput. Phys.*, **179**:622–648, 2002.

[OS88] S. Osher and J. Sethian. "Fronts propagating with curvature dependent speed: algorithms based on hamilton-jacobi formulations." *J. Comput. Phys.*, **79**:12–49, 1988.

[PMZ99] D. Peng, B. Merriman, H. Zhao, S. Osher, and M. Kang. "A pde based fast local level set method." *J. Comput. Phys.*, **155**:410–438, 1999.

[QCO03] J. Qian, L.-T. Cheng, and S. Osher. "A level set based eulerian approach for anisotropic wave propagations." *Wave Motion*, **37**:365–379, 2003.

[ROF92] L. Rudin, S. Osher, and E. Fatemi. "Nonlinear total variation based noise removal algorithms." *Physica D*, **60**:259–268, 1992.

[Sal82] J. F. Sallee. "A triangulation of the n-cube." *Discr. Math.*, **40**:81–86, 1982.

[Sam90a] H. Samet. "Applications of spatial data structures: computer graphics, image processing and gis." *Addison-Wesley*, 1990.

[Sam90b] H. Samet. "The design and analysis of spatial data structures." *Addison-Wesley*, 1990.

[Som58] D. M. Y. Sommerville. "An introduction to the geometry of n-dimensionals." *Dover, New York*, 1958.

[SSO94] M. Sussman, P. Smereka, and S. Osher. "A level set approach to computing solutions to incompressible two-phase flow." *J. Compt. Phys.*, **114**:146–159, 1994.

[Str99a] J. Strain. "Fast tree based redistancing for level set computations." *J. Comput. Phys.*, **152**:664–686, 1999.

[Str99b] J. Strain. "Tree methods for moving interfaces." *J. Comput. Phys.*, **151**:616–648, 1999.

[TCO03] Y.-H. Tsai, L.-T. Cheng, S. Osher, P. Burchard, and G. Sapiro. "Visibility and its dynamics in a pde based implicit framework." *UCLA CAM Report 03-22*, 2003.

[WB96] C. Weigle and D. C. Banks. "Complex-valued contour meshing." *IEEE Visualization*, **96**:173–180, 1996.

[Whi74] G. B. Whitham. "Linear and nonlinear waves." *Wiley*, 1974.

[Zie95] G. M. Ziegler. "Lectures on polytopes." *Springer-Verlag*, 1995.

[ZOF01] H. K. Zhao, S. Osher, and R. Fedkiw. "Fast surface reconstruction using the level set method." *1st IEEE Workshop on Variational and Level Set Methods, 8th ICCV, Vancouver*, pp. 194–202, 2001.

[ZOM00] H. K. Zhao, S. Osher, B. Merriman, and M. Kang. "Implicit and non-parametric shape reconstruction from unorganized data using a variational level set method." *Comput. Vision and Image Understanding*, **80**:295–31, 2000.